Generating Mock Skeletons for Lightweight Web Service Testing

A thesis presented in partial fulfilment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science
at Massey University, Manawatū,
New Zealand.

Thilini Bhagya, Randunu Pathirannehelage

2020

Abstract

Modern application development allows applications to be composed using lightweight HTTP services. Testing such an application requires the availability of services that the application makes requests to. However, continued access to dependent services during testing may be restrained, making adequate testing a significant and non-trivial engineering challenge. The concept of Service Virtualisation is gaining popularity for testing such applications in isolation. It is a practise to simulate the behaviour of dependent services by synthesising responses using semantic models inferred from recorded traffic. Replacing services with their respective mocks is, therefore, useful to address their absence and move on application testing.

In reality, however, it is unlikely that fully automated service virtualisation solutions can produce highly accurate proxies. Therefore, we recommend using service virtualisation to infer some attributes of HTTP service responses. We further acknowledge that engineers often want to fine-tune this. This requires algorithms to produce readily interpretable and customisable results. We assume that if service virtualisation is based on simple logical rules, engineers would have the potential to understand and customise rules. In this regard, Symbolic Machine Learning approaches can be investigated because of the high provenance of their results.

Accordingly, this thesis examines the appropriateness of symbolic machine learning algorithms to automatically synthesise HTTP services' mock skeletons from network traffic recordings. We consider four commonly used symbolic techniques: the C4.5 decision tree algorithm, the RIPPER and PART rule learners, and the OCEL description logic learning algorithm. The experiments are performed employing network traffic datasets extracted from a few different successful, large-scale HTTP services. The experimental design further focuses on the generation of reproducible results.

The chosen algorithms demonstrate the suitability of training highly accurate and human-readable semantic models for predicting the key aspects of HTTP service responses, such as the status and response headers. Having human-readable logics would make interpretation of the response properties simpler. These mock skeletons can then be easily customised to create mocks that can generate service responses suitable for testing.

Acknowledgements

This thesis would not have been possible without the inspiration and support of a number of wonderful individuals. My thanks and appreciation to all of them for being part of this journey and making this a success. Especially, I would like to express my deepest gratitude to my supervisors, Prof. Hans Guesgen and A/Prof. Jens Dietrich. Without their constant enthusiasm, encouragement, support, and optimism, this thesis would not have been achievable. Their immense knowledge and plentiful experience have always inspired me in my academic research and in my daily life, helping me to grow both professionally and personally. I would also like to thank the staff and my colleagues in the Computer Science cluster, Massey University, Palmerston North for their valuable discussions and feedback on my research, and, of course, their kind cooperation and friendship. A special appreciation must also go to my family. They have given unwavering support and encouragement to complete my PhD successfully.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter presents the background and main contributions of this research. A brief introduction to the research is given in Section 1.1 and Section 1.2 discusses a motivating example. Section 1.3 includes an explanation of the aims and objectives of this research, followed by Section 1.4 which briefly details the research methodology adopted. Finally, Section 1.5 outlines the remaining chapters of the thesis.

1.1 Overview

Service-Oriented Computing (SOC) is a popular approach to facilitate the development of large, modular applications using diverse technologies. There is a range of technologies that have been used in SOC, starting with early attempts to establish standards around the Web Services Description Language (WSDL) [1] and Simple Object Access Protocol (SOAP) [2] protocols. In recent years, more lightweight HTTP-based services (i.e., *RESTful services* [3]) have become the mainstream.

When using HTTP services, different parts of the application cooperate by sending and responding to HTTP requests, typically in order to access and manipulate resources. The ubiquitousness of HTTP means that clients and servers can be easily implemented in a wide range of languages and deployed on many platforms. While this is useful in itself to architect and design large applications, this approach is now increasingly used to facilitate the development of product ecosystems around successful services. Examples include the Application Programming Interfaces (APIs) that can be used to access the services of Google[1], Twitter[2], Amazon[3], and Flickr[4].

This has created new challenges for both the engineering and the research community. Of particular interest is how to assure the accurate functioning of service-oriented applications before live deployment. Adequately testing with dependent services is not always possible due to limitations in the observability of service code, lack of control, and the costly access [4]. Therefore, a number of ap-

1. `https://developers.google.com/apis-explorer` [accessed 02 Aug. 2020]
2. `https://developer.twitter.com/en.html` [accessed 02 Aug. 2020]
3. `https://developer.amazon.com/documentation` [accessed 02 Aug. 2020]
4. `https://flickr.com/services/api` [accessed 02 Aug. 2020]

```
LinkedList mockedList = mock(LinkedList.class);
when(mockedList.get(0)).thenReturn("first");
```

Figure 1.1: A Sample Stubbing Method Call

plicable approaches have been investigated over the years to test such applications independent of the services which they depend on.

A most common practise is *mocking*, i.e., to *manually* define behavioural responses from scratch based on underlying service syntax and semantics (i.e., mock objects [5]). A challenge of mocking is that it requires a detailed understanding of the service semantics. Service Virtualisation (SV) [6, 7] tries to address this problem by *automatically* constructing virtual models of services suitable for testing by inferring service semantics from network traffic recordings. For example, SV will try to emulate the behaviour of a dependent service by generating anticipated responses using the inferred semantic model. Yet, existing approaches to SV are error-prone. Some [8–11] require detailed knowledge of the service structure and system protocol for the response generation to be effective. Others [12–20] specify interaction behaviour rather conforming to the contemporary context of services. Besides, comprehensibility (i.e., results present in such a way that it may be inspected and interpreted by humans) is one of the major features missing.

SV often draws on Artificial Intelligence (AI) techniques as inference mechanisms. Amongst them, Symbolic Machine Learning (SML) algorithms have gained popularity due to the provenance of their results, which means that humans can interpret the outcome of the results with relative ease as the system produces human-readable explanations (through representations like decision trees and logical rules). This is in stark contrast to many sub-symbolic AI techniques (e.g., neural networks and deep learning algorithms) that lack provenance, and are black-box by nature. In the context of symbolic learning, attribute-based learning (i.e., decision tree learning and rule learning) and inductive logic programming/description logic learning are the typical approaches [21]. While these techniques are well-established, they have gained attention relatively recently in Explainable-AI [22].

The approach proposed in this thesis can be seen as a hybrid method: we propose to use SV to infer some attributes of HTTP service responses, but acknowledge that engineers will often want to fine-tune this to create mocked services. This requires the SV algorithm to generate results that can be understood and customised by engineers. Inherently interpretable AI systems [23] usually provide clear, human-readable results in the form of rules. Therefore, we expect that if the SV is based upon inference rules, the engineers used to writing mock tests will find it easy to customise those rules. To emphasise this point, consider the snippet of Java code in Figure 1.1 written using the popular Mockito framework [24]. In the second line, the functionality of a linked list is *stubbed* for the purpose of testing. Interestingly, no actual `LinkedList` is required at this stage. The process of stubbing basically uses a simple logical rule, expressed using a domain-specific language provided by the mock framework. By using explainable SML techniques, we aim at a solution that provides a sweet spot between highly accurate automation and customisability. This takes into account that completely automated techniques are unlikely to provide sufficient accuracy to mock complex services. Consider for instance a service providing financial transactions: while it is certainly feasible to infer rules modelling the response codes of accessing account information, based on authentication headers, URIs, resource ids and state inferred from transaction history, it is much more complex to model unexpected server behaviour (such as a server returning a response with status code `500 Internal Server Error`) and in general, the flakiness (non-deterministic behaviour) associated with distributed and concurrent systems, or the content of data returned by the server (such as inferring the structure of PDF documents synthesised by the server, used for account statements). Similarly, flakiness issues in testing with services (tests provide varying outcomes even though there are no changes in the source code or execution environment) often impede the implementation of fully automated mocking techniques. Test flakiness has become a significant and acknowledged issue in modern service-oriented application development, i.e., the recent study [25] shows that a non-negligible percentage of flaky tests has been observed in Microsoft (4.6% of individual tests were identified as flaky when monitoring five projects over a one-month period).

Accordingly, this research is initiated in order to understand the suitability of symbolic learning techniques in predicting some of the features of HTTP service responses directly from the recorded interaction traces. Considering that a single HTTP response has multiple features that are optimal to predict, from the machine learning perspective, this is a multi-class or multi-target classification problem. And we name the resulting set of output predictions *mock skeleton*. These model predictions with explanatory power will allow engineers to precisely comprehend the semantics of the response properties, making it easier to edit or refine them to create mocks that can generate responses for given requests.

1.2 Motivating Example

Assume, for example, a simplified scenario of developing an e-commerce platform, which is our Application-Under-Test (AUT). The application is primarily to be integrated with certain external HTTP/REST services in its production environment to realise certain functional requirements (such as with a payment service offered by a third-party vendor in order to incorporate online payment capability). The behaviour of the application depends on the responses it receives from these dependent services, and engineers have to test how the application behaves based on the responses prior to live deployment (they want to know what happens if particular responses are returned, including various HTTP statuses as well as response bodies and headers).

As the application depends on services not operated in-house, it cannot be fully tested without accessing external service providers. Especially, if the application development is iterative, each development version must be repeatedly tested against the same service. However, these services are not readily available and controllable for testing purposes, e.g., certain services have restrictions and/or costs associated with their invocations. Frequent testing at a high rate of usage is thus quite challenging. If engineers proceed without completing the required tests, there would be errors that are far more difficult and expensive to fix later on.

Mocking is a promising approach to bypassing the absence of dependent services in testing, i.e., aims to simulate the behaviour of real services on which the application depends. The most general practise is to manually define the responses that a real service would produce using the available knowledge of the underlying interaction protocol and service behaviour. However, each such response is only capable of fulfilling a certain development need at a certain point in time. Engineers, therefore, need to write and rewrite a variety of interaction patterns to perform tests throughout the entire development cycle. It takes a significant amount of time and is often prone to errors. In particular, the lack of precise knowledge of the service hinders the accuracy of the synthesised responses. SV is another practise. It automatically produces responses using machine learning models derived from traffic recordings (a collection of requests sent by the AUT to the live service and the responses sent back from the service). Once a virtual service model is built, testing can be carried out free of charge as often as desired over multiple development cycles. But, with current SV frameworks, it is not feasible to generate accurate approximations of the real responses of HTTP-based services (especially as they simulate responses rather adhering to the service state). It is also obvious that fully automated solutions cannot render highly precise mocks in practical settings as there are response properties which cannot be predicted using automated methods (such as the date and time at which the response was originated). On top of that, the solutions lack transparency, which means that the engineers using them cannot understand how the responses are formed and what factors have been taken into account in the response generation and whether the responses are rational. They are usually unable to decide if the mocked service is a suitable representation of the actual service, which may potentially result in lower acceptance and satisfaction. In addition to this, engineers often want to inspect and adjust the responses generated, e.g., to return responses representing various failure scenarios that might be difficult to reproduce using the real service, and when service evolution occurs (if there are certain changes in the new version of the service than the one created mocks with). Presenting responses to engineers in human-readable format can facilitate comprehension of the semantics of the service (this helps them to understand why the system has delivered particular outcomes), thereby increasing trust and reliance on the system and making it easier to modify them accordingly (providing clear-cut means for customisation).

A novel SV technique is, therefore, required that is capable of automatically infer some attributes of HTTP service responses (i.e., mock skeletons) while maintaining a high level of accuracy with human comprehension. SML approaches can be explored in this regard due to the inherent interpretability of their results.

1.3 Aims and Objectives

The aim of this study is two-fold. The primary aim is to examine the appropriateness of SML techniques for automatically generating HTTP services' mock skeletons that are both accurate and customisable. The following research objectives would facilitate the achievement of the main study aim.

- The first objective is to investigate the potential of SML techniques to automatically synthesise some of the attributes of HTTP service responses without explicit knowledge of the service. It focuses on examining whether the algorithms can infer HTTP response properties directly from network traffic recordings. We conduct experiments on algorithms targeting at learning different response properties employing network traffic datasets extracted from a few different successful, large-scale HTTP services (i.e., GitHub, Twitter, Google, and Slack).

- The second objective is to investigate the potential of SML techniques to automatically synthesise some of the attributes of HTTP service responses in an accurate manner. Typically, valid HTTP responses adhere to the protocol structure and the current state of the service. This objective, therefore, focuses on examining whether the algorithms are capable of automatically inferring protocol- and state-compliant response properties. We conduct experiments on algorithms targeting at learning response properties by extracting knowledge about the protocol structure and the service status from recorded interactions. We evaluate the validity of the results produced using metrics such as predictive accuracy, precision, and recall.

- The third objective is to investigate the potential of SML techniques to automatically synthesise some of the attributes of HTTP service responses in a format that is easy to customise. Generally, presenting logical rules in a human-readable format will make it easier to comprehend and modify them. This objective, therefore, focuses on investigating whether the algorithms are capable of automatically making human-readable logical inferences for response properties. During the experiments, we evaluate the comprehensibility of the results produced with metrics such as model size.

In order to achieve the aforementioned research objectives, a set of SML algorithms must be selected. We are particularly interested in employing a leading algorithm (which is extensively used to support computing research) in each category of symbolic learning techniques (i.e., decision tree learning, rule learning, and description logic learning) to make the research findings unbiased. As well, considering that the real-world HTTP traffic (request/response pairs) that has to be processed typically contains both categorical and numerical data, the algorithms to be employed should be capable of handling both data types. These criteria result in the selection of four algorithms from different approaches: the C4.5 [26] (decision tree learning), the RIPPER [27] and PART [28] (rule learning), and the OCEL [29] (description logic learning). The C4.5 algorithm is a landmark algorithm in decision tree learning that has been most widely used in practice to date, and both the RIPPER and PART algorithms are considered as state-of-the-art in rule induction [30,31]. The OCEL is the prevailing scheme for description logic learning [32]. Apart from being recognised as the prominent approaches in use today for their respective categories, these algorithms have proved their ability to work directly with nominal and numeric attributes in concept learning.

The secondary aim of this study is to provide research results that are easier to reproduce (i.e., to be able to obtain the same results of the study using the same data and the same methods used). Recently, there has been a greater attention on the reproducibility of results obtained from empirical studies [33, 34], with some disciplines now considering reproducibility as an expectation to be published in conferences and journals (including ACM SIGPLAN conferences). An integral aspect to facilitate reproducibility is the use of standard datasets, along with the provision of executing scripts or artefact. Follows that, we perform experiments using carefully sourced and/or constructed datasets (all datasets are extracted from well-defined processes, also holding characteristics that reflect the up-to-date use of HTTP services in general). We develop a set of scripts to automate the execution of all the experiments and provide clear instructions about how to re-run all the experiments and to collect all the experimental results. Both datasets and implementations reported in this research are also made publicly available. Moreover, we provide a pre-configured VirtualBox image that replicates our experimentation environment to ensure that all the experiments can be reproduced with little effort in any computer that has VirtualBox installed and meets our minimum system requirements. It should be noted that the results of existing SV studies are difficult/impossible to reproduce (there is no open access to either datasets or scripts) where this particular aim advances over those approaches.

1.4 Research Method

The whole research study can be roughly divided into four key phases: **Inception**, **Elaboration**, **Construction**, and **Transition**. It is structured following the main idea of the Rational Unified Process (RUP) [35] approach, which represents a very robust and disciplined pathway to various software engineering projects. Some of these steps can be iterative or even mutually influence each other when practically conducting this research.

1. Inception

The inception phase is focused on understanding the nature of the problem to be addressed in this study. In this regard, the latest research approaches in service-oriented application testing in relation to service simulation are examined.

2. Elaboration

A more extensive review of the literature is performed during the elaboration phase. Experiments are also designed to assess the suitability of SML algorithms to learn some attributes of HTTP service responses from recorded traffic. In this case, HTTP datasets are constructed from network traffic and a broader range of SML techniques are explored.

3. Construction

The main focus of the construction phase is to run the experiments with selected algorithms across all datasets. Later, the inferred semantic models are evaluated, the results are interpreted, and conclusions are drawn.

4. Transition

The primary research findings are published as conference or journal papers during the transition phase. Furthermore, the implementation code and the experimental datasets are made publicly available to permit reproduction of the research results reported in this thesis.

1.5 Thesis Overview

Chapter 2 (Preliminaries)

The chapter covers some background knowledge which helps with understanding the rest of this thesis. It briefly introduces the concept of services and the evolution of SOC. This is followed by an introduction to lightweight HTTP-based services, that explicitly utilise the REST constraints. Then, the definitions and advances in application testing are presented, along with the approaches to test service-oriented applications, including SV. It also summarises the history of relevant AI techniques and then presents a description of the concept learning algorithms used in this research. The chapter ends with an overview of the HTTP protocol.

Chapter 3 (Systematic Literature Review)

The chapter discusses the related work to this research. It reports systematic review results on existing approaches in service-oriented applications testing. The chapter then proceeds into a discussion on studies in the field of SV. By covering major issues in existing approaches, we raise the need for creating accurate and provenance semantic models of HTTP services in the absence of services' knowledge. It further covers the different approaches to services testing and the latest work on AI-driven strategies to software testing, which are important references to our research.

Chapter 4 (Datasets Acquisition)

The chapter outlines the network traffic datasets used for the experiments. Four standard datasets are derived from real HTTP services (i.e., GitHub, Twitter, Google, and Slack) referring to the second study aim which is to facilitate reproducibility of results. The chapter discusses use cases for such datasets and extracts a set of requirements from those use cases. Then, it presents the design, and the methods and tool used to construct the datasets. We conclude our contribution by providing some selective metrics that characterise each dataset and basic instructions on how to obtain and use the datasets.

Chapter 5 (Experimental Methodology)

The chapter gives a detailed description of the basic procedure used during the experiments. The process includes four major steps. In the first step, basic cleaning is done to remove the contents that impede raw datasets from being processed and/or parsed correctly. In the second step, the preprocessed datasets are converted and filtered to best expose to chosen SML algorithms (i.e., C4.5, RIPPER, PART, and OCEL). In the third step, the algorithms are applied to train classification models from training data to predict different attributes associated with response properties. Finally, the predictive ability of the models is assessed by cross validation. The predictive accuracy, precision, and recall are measured to evaluate the validity of classification models. The size of the tree or the number of rules or the length of the class expression produced is considered to assess the readability. Additionally, the chapter provides instructions on how to reproduce the results.

Chapter 6 (Experimental Results)

The chapter presents the results of this study and discusses with reference to the main research intent, which is to examine the suitability of SML algorithms to automatically synthesise mock skeletons of HTTP services directly from network traffic recordings. It mainly explores experimental results based on datasets. For each dataset, the results of algorithms are analysed with respect to the target attributes associated with the key HTTP response features, including the status, response headers, and response body. It addresses the possible threats to the validity of the experimental results at the end.

Chapter 7 (Conclusions and Future Work)

The chapter sums up our contributions. It briefly concludes the results and findings of this research. At last, it presents an overview of the prospects for future work.

Chapter 2

Preliminaries

This chapter presents some of the background to the research presented in this thesis.

2.1 Services

The concept of services is widely adopted in modern heterogeneous distributed computing. It has evolved from object-oriented and component computing to microservices. REST is currently the most common form of service implementation. As such, Section 2.1.1 examines the evolution of services in detail, Section 2.1.2 defines the term *service*, Section 2.1.3 outlines the basic interactions between services, and Section 2.1.4 explains the basic principles of REST and explores services based on REST and HTTP.

2.1.1 History of Service-Oriented Computing

Back in the early 1980s, the Remote Procedure Call (RPC) [36] technique marked the first major step toward systems distributed computing. The idea behind RPC was to invoke procedures (functions) on remote servers in the same way as local calls (in order to allow transparent access). It was a powerful facilitator for the development of large, modular applications. The concept was initially formed by Sun Microsystems during the implementation of the Network File System (NFS) [37]. RPC was also used as the foundation of Apollo's Network Computing System (NCS) [38]. Distributed Computing Environment (DCE) [39] was established at the end of the 1980s by the Open Software Foundation (OSF) as an attempt to standardise these various RPC implementations. DCE also incorporated standards for security, naming, and time management to build and run applications. However, it did not gain massive support from the industry for political reasons [40].

The advent of object-oriented programming led to the next major shift in the techniques used to develop distributed applications. During the time, distributed object solutions were widely implemented. In 1989, the Object Management Group (OMG)[1] was founded by a group of major platform vendors (e.g., Microsoft, IBM, AT&T, and Sun Microsystems) to create standards for distributed object computing. As the first effort, OMG released the Common Object Request Broker Archi-

1. `https://omg.org` [accessed 02 Aug. 2020]

tecture (CORBA) [41], a language-independent and architecture-neutral platform for building distributed object-oriented applications. Objects in CORBA architecture were usually supported by an Object Request Broker (ORB), which managed interactions between remote objects transparently. CORBA quickly became more popular and was used to develop a number of distributed applications. Around the early 1990s, Microsoft released its own Distributed Component Object Model (DCOM) [42] to compete with OMG CORBA. DCOM was closely linked to the CORBA model but designed exclusively for the integration of Windows-based applications [43]. Simultaneously, Sun Microsystems introduced the Remote Method Invocation (RMI) model [44]. RMI was similar to CORBA and DCOM but worked only with objects written in Sun's Java programming language. All of these technologies were quite successful in integrating homogeneous applications within a local area network. Gradually, at that time, the industry was confronted with a competitive battle between the standards [45].

Advances in component-based technology allowed distributed computing to expand further. A component is a reusable modular unit subject to third-party composition with a contractually defined interface [46]. Starting in the mid-1990s, several platform vendors built component-based development models. This originated with the Common Business Object Facility (CBOF) [47] in 1996 but, due to political infighting, this effort was eventually abandoned. Subsequently, Sun Microsystems released Enterprise Java Beans (EJB) [48]. EJB was built on top of Java programming language and was therefore not interoperable with other languages. As a language-neutral superset of Sun's EJB, OMG introduced the CORBA Component Model (CCM) [49] at the end of 1990s.

The next phase of the evolution of distributed computing technologies came with the emergence of the Web. Starting in the mid-1990s, the Web became popular and key to commercial success for most enterprises. The Hypertext Transfer Protocol (HTTP) was the primary transport protocol of the Web (see Section 2.4 for more detail on HTTP). At that time, there was a growing demand for application interoperability across networks. However, most of the existing distributed application frameworks could not be accessed via the Internet (did not comply with Web standards) [45]. During the late 1990s, Extensible Markup Languages (XML) [50] became another new industry standard with an application-independent ability to represent data. Microsoft, therefore, created an entirely new technology for method invocation in conjunction with XML. The World Wide Web Consortium (W3C)[2] standardised it as Simple Object Access Protocol (SOAP) [2] in 1999. It was extensively used by the major platform vendors at the time for incorporating applications across networks. This eventually led to the advent of the notion of Web Services primarily to support distributed computing for the integration of highly heterogeneous systems over the Internet [51]. In 2000, IBM, Microsoft, and Ariba developed a new set of standards called Web Services Description Language (WSDL) [1] and Universal Definition, Discovery and Integration (UDDI) [52] to enable Web services to be described, written and used in a language, platform and location-independent manner. As a result, Web services were developed and deployed around 2005 mainly together with SOAP, WSDL, and UDDI technologies [53].

2. `https://w3.org` [accessed 02 Aug. 2020]

Service-Oriented Architecture (SOA) has become the next major support model for the growth of distributed client-server applications in parallel with these advancements. The concept of SOA emerged at some point in the early 1980s, but the recent interest in the architecture was spurred, especially after the invention of Web services [54]. SOA is a set of software design principles for building and deploying business functions as distinct services, enabling them to be easily accessed and reused through well-defined, documented interfaces and messaging protocols, and to be rapidly consolidated at runtime to function as a whole [40]. In particular, SOA relies on the concept of Services (Section 2.1.2 points out what a service is). A service in the SOA is designed in such a way that various services can be interconnected through an enterprise service bus with minimal effort to carry out business transactions. By that time Web services (services delivered over the Web) became the preferred way to realise SOA. Notably, from 2005 to 2010, SOA and Web services were collectively applied in most distributed application development projects [55]. Implementing SOA with Web services enabled interoperable deployments. Over time, however, the industry noted that SOAP-based Web services were often complicated, although it offered very rich functionalities, and gradually started shifting to alternative technologies to build distributed computing solutions.

In the meantime, the advent of REST (REpresentational State Transfer) introduced a simple, lightweight, and scalable service implementation paradigm. REST is a set of architectural constraints derived from the doctoral dissertation by Roy Fielding in 2000 [3]. It was strongly inspired by the philosophy of the Web and reused the features of HTTP as much as possible (see Section 2.1.4 below). By around 2010, RESTful implementations via HTTP began gaining popularity by enabling heterogeneous services to interact best on the Web, even some leading software firms, such as Google and Amazon, had started to adopt REST to implement their Web services (i.e., at that point, Amazon had both SOAP and REST implementations to its Web services, but the REST accounted for 85% of their use[3]). The JavaScript Object Notation (JSON) [56] message format has also become a popular alternative to XML due to its more lightweight structure and object-oriented notations (see Section 2.4.2 for further details on XML and JSON).

The Open Service Gateway Initiative (OSGi) framework [57] also emerged as another viable approach for developing and deploying services. It was introduced as a modularisation platform for Java by the OSGi alliance[4] in 1999. OSGi provides primitives and supports the construction of modular Java applications, based on the composition of small, reusable, and self-contained modules called bundles [58]. OSGi leverages the SOA support to allow interactions among bundles. OSGi bundles interact with each other through services, which are Java objects accessible via public Java interfaces. Popular industry applications such as Eclipse[5], Glassfish[6], WebLogic[7], and WebSphere[8] have widely adopted this approach.

3. `https://aws.amazon.com/blogs/aws/rest_vs_soap` [accessed Aug. 02 2020]
4. `https://osgi.org` [accessed 02 Aug. 2020]
5. `https://eclipse.org` [accessed 02 Aug. 2020]
6. `https://glassfish.java.net` [accessed 02 Aug. 2020]
7. `https://oracle.com/middleware/technologies/weblogic.html` [accessed 02 Aug. 2020]
8. `https://ibm.com/cloud/websphere-application-platform` [accessed 02 Aug. 2020]

Microservices is the latest trend in the field of distributed computing. Around 2011, ideas related to microservices originated from best practises in the industry, but only recently, academic researchers started to study this approach. Microservices architecture is an evolution of the traditional SOA [59]. The services are more fine-grained and functionally independent from each other and communicate through a simple API [60]. Lightweight HTTP-based Web services (or RESTful services which use HTTP as their underlying protocol) have become the standard for most microservices-based application development. In line with this paradigm, large-scale ecosystems are also being developed around successful software products to provide easy access to their resources and services, which include Google and Twitter.

2.1.2 Definition of Service

Services are the basic building blocks in Service-Oriented Computing (SOC). Still, in the context of SOC, there is no widely accepted formal definition for a service; the different standards bodies and the researchers have their own interpretations with varying levels of detail. Therefore, it would be wise to revisit some of them in order to come up with a meaningful definition appropriate to guide this study.

The W3C provides a general definition of a service as an abstract entity able to perform a function [61]. The Organization for the Advancement of Structured Information Standards (OASIS)[9] defines a service as a mechanism for allowing access to one or more capabilities where access is given through a defined interface and is exercised in accordance with the constraints and policies laid down in the service description [62], but in this respect it is difficult to distinguish services from objects and components.

Some of the most widely accepted service definitions exist in academic literature. For example, a service is described in [63] as a coarse-grained, discoverable software object that acts as a single instance and interacts with applications and other resources through a loosely coupled, message-based communication model. The idea is based on SOA principles but is given without reference to the self-contained nature, which is a popular feature of the service-oriented paradigm. In [64], a service is described as a well-defined, self-contained entity that does not depend on the context or status of other services.

Therefore, given the nature of this research and in accordance with existing descriptions, the suitable definitions which can be extracted is that a service is: *a self-contained (does not require state information from other services), loosely-coupled (hides implementation from user and is easily accessible over a network), modular (serves a distinct business function) software unit that communicates with applications and other services via messages.*

2.1.3 Service-Oriented Interaction Schema

There are two major roles in any service-oriented application: *service providers* and *service consumers*. The service providers offer a range of resources and activities for service consumers to use. In order to advertise and promote services, the providers

9. `https://oasis-open.org` [accessed 02 Aug. 2020]

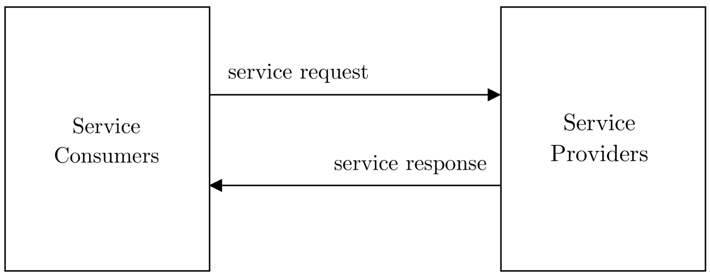


Figure 2.1: The Interaction Between Service Consumers and Service Providers

publish them in a registry along with service contracts (defines the nature of the service, how to access it, the service specifications, and the service charges). The service consumers could locate the services from the registry and build the necessary client components to bind and use the services.

The focus of the research in this thesis is on the end-use of services where service consumers interact directly with service providers (service consumers send requests to service providers and, in return, receive responses from those service providers). This relationship is depicted in Figure 2.1. Each request and response exchanged between a consumer and a provider is considered as a *transaction*.

Communication between the parties is controlled by an application-layer protocol. Therefore, each request and response must comply with the syntax and semantics of the protocol. Every message should include two different types of information: protocol structure information that specifies the type and format of the message, and payload information that includes service-specific attribute values and metadata. The information returned with the response is usually dependent on the values of the request and the internal service state (previous transactions history can affect the current transaction). For a more detailed example, see Section 2.4.3.

It is also very common for service-oriented applications to act as both the service provider and consumer by offering a service responding to consumers requests as well as making requests for other services. In this thesis, however, we only consider service-oriented applications from the point of view of the consumers.

2.1.4 RESTful Services

The term REST stands for REpresentational State Transfer and was defined by Roy Fielding (one of the co-authors of the HTTP specification) in his PhD thesis back in 2000 [3]. REST is not the software architecture itself, but rather a set of architectural constraints to generate software services that are faster, more efficient, and easier to scale. These constraints can be summarised as follows:

- Client-server architecture

  It is important to follow a client-server architecture. The client has to be decoupled from the server allowing one another to evolve independently.

- Statelessness

Interaction between the client and the server must be stateless in such a way that each client request must be dealt with independently, which is not related to any prior request and must include all the information needed to understand and function. The server does not have to keep the status information between requests. Here, statelessness refers in particular to the state of the application (the position of the client within the interaction and varies from client to client) whereas the servers still maintain the state of the resource (data properties on the server and is the same for all clients). Statelessness improves the scalability (each request can be processed by a separate server) and the visibility (any request can be understood).

- Cacheability

  The response the server sends to the request should explicitly indicate whether or not the data is cacheable. Cacheable responses can then be reused by clients later. Better controlled caching decreases the load on the server (as it eliminates some client-server interactions) and improves efficiency, scalability, and performance.

- Layered system

  The architecture should consist of hierarchical layers. Each layer of intermediaries should be inserted seamlessly and must act independently and only interact with immediately adjacent layers. Layering allows for greater flexibility.

- Uniform interface

  There should be a clearly defined interface between the client and the server. The four guiding principles of the uniform interface are:

  – Identification of resources
    A resource must be identified by using the URI standard[10].
  – Manipulation of resources through representations
    The resource representation and the resource itself are two distinct things. A resource could have multiple representations. The server must present a resource representation to manipulate the resource according to the needs and capabilities of the client.
  – Self-descriptive messages
    Each request and response message should contain sufficient information to describe how to process the message.
  – Hypermedia as the engine of application state
    Each response should contain appropriate hypermedia controls (such as hyperlinks) to inform the client of possible next steps to be taken to change the resource (state of the resource).

A service that exhibits all defined constraints is known as *RESTful*. More often, RESTful services are implemented using HTTP as the REST constraints works seamlessly with the HTTP protocol (REST can be deployed via any protocol).

10. `https://w3.org/Addressing` [accessed 02 Aug. 2020]

HTTP inherits the client-server architecture where clients make HTTP requests and servers listen to and respond to HTTP requests. It also is designed in a stateless manner: each HTTP request must contain all the information necessary for the server to understand it (for example, HTTP requests should always carry headers like Host, even if they remain the same all the time). HTTP explicit caching headers (e.g., `Cache-Control`, `ETag`) address on cacheability. In addition, HTTP meets the layered system constraint by allowing the direct insertion of intermediate layers (such as proxies and firewalls). The uniform interface constraint is generally implemented through a combination of different HTTP constructs. For example, resources are uniquely identified by HTTP URIs. Clients are provided with different representations through HTTP content negotiation (the client and the server agree on the best mutually understandable media type among the alternatives available). Typically, the HTTP client specifies its preferences with the HTTP `Accept` header, the HTTP server indicates its choice with the `Content-Type` header. Well-defined syntax and semantics of HTTP also permit self-descriptive messages. Each message within an HTTP interaction is expressed explicitly based on agreed media types. HTTP also includes a limited number of standardised methods (i.e., `POST`, `GET`, `PUT/PATCH`, and `DELETE`) which, in practise, often correspond and are mapped to create, read, update, and delete (CRUD) database operations where intermediaries could easily interpret messages. In addition, HTTP allows the response to include hyperlinks to show what steps a client should take, and to clarify how such steps should be taken.

HTTP is a robust protocol which can support REST constraints directly. Section 2.4 provides a more detailed explanation of the HTTP constructs. However, most of the Web services that are currently branded as RESTful are simply HTTP-based services where the design is tied to the HTTP protocol standards (do not implement all aspect of REST), but if these services explicitly enforce REST constraints using HTTP constructs as previously mentioned, it particular allows simple, lightweight, and fast HTTP-based services. This approach is becoming increasingly popular in the development of distributed client-server applications. In this study, we consider these lightweight HTTP-based services.

2.2 Application Testing

Application testing is necessary to produce highly reliable applications. Throughout the history of application development, there have been many advances in testing. These aspects are discussed briefly in Section 2.2.1. Section 2.2.2 then outlines the service virtualisation technique proposed to provide a testing environment suitable for service-oriented applications.

2.2.1 History of Application Testing

Application testing was really basic in the early days of computing, and the main focus was on finding errors in the final software product. Testing was usually carried out once at the end of the application development effort and performed manually [65]. As application complexity dramatically increased since the early 1980s, testing both functional and non-functional attributes become important throughout the development cycle. Typically, the units were first tested, and the integration tests were then carried out, and the fully integrated system was then tested. In some cases, the acceptance test was performed after the system test.

Most of the current functional and non-functional testing concepts were initially proposed before the 1980s [66], including stress testing to evaluate how the system performs beyond its normal operating limits, mutation testing to identify programmes error by modifying small parts of the code, regression testing to ensure that the correctness of the programme is not adversely affected by adding new functionality or removing faults, load testing to evaluate the application can handle predefined load conditions, etc.

Beginning in the mid-1980s, automated testing tools started to appear in the industry to automate the manual nature of testing [65]. These tools were initially fairly simple and backed by the recording and playback approach; recording manual tests and playing them back. In 1989, Mercury Interactive released the first version of LoadRunner[11] to accelerate the performance testing. Since then, automated test technologies have evolved and developed, providing rich scripting languages and reporting facilities, thus reducing the need for human involvement and taking less time. There are now several commercial as well as open-source software platforms available to support various aspects of application testing.

At the beginning of the 1990s, the development of object-oriented applications became popular in the software development market. At that time, the majority of testing activities focused on the applicability of traditional testing techniques to object oriented applications [67]. Testing for such a programme, however, was more difficult than for traditional systems because objects might communicate with each other with unpredictable variations and invocations (flakiness) [68]. As a result, many testing techniques have been discovered to support the development of object-oriented applications, such as state-based testing [69] to assess interactions of methods in each class, use-case testing [70] to test each scenario in every use case, class-based testing [71] to evaluate each class, class derivative, interaction and aggregation, and thread-based testing [72] to test the integration of all classes necessary to perform a single application case.

In addition, testing evolved in the early 1990s from using unstructured approaches to a process-driven methodology [66]. The test process started with test preparation, and then test cases were designed, tests were developed, maintained and executed. In the mid-1990s, component-based technology emerged as the main application development strategy. Testing in this paradigm also introduced new challenges [73]. The unavailability of external and dependent components for testing was the main difficulty encountered by most engineers, and the general approach was to test a particular component directly in isolation. This was usually done by replacing the dependencies with stubs that are lightweight versions of the system components [74].

By the late 1990s, Test-Driven Development (TDD) [75] was practised in the software development industry. Though it had been around for some time, it has been re-discovered as one of the main features of Extreme Programming (XP) [76]. It is a software building technique which guides iterative development by first

11. `https://microfocus.com/products/loadrunner-professional` [accessed 02 Aug. 2020]

writing the test and then implementing system functionality to pass the test. TDD was first practised in the Smalltalk[12] environment by designing the SUnit[13] unit testing framework to support the TDD process. Since then, for almost any programming language, a series of tools known as XUnit began to appear that essentially applied the same approach to unit testing. Among them, the most successful unit testing framework was JUnit[14], which brought the automated unit testing to Java.

Eventually, the concept of *mock objects* emerged from the XP community as a TDD support technique [77]. A mock object was a much more advanced version of stub in which the notion of isolation was supported [5]. From then on, the industry saw an increase in the use of mock objects and frameworks to support mocking. EasyMock [78] was created in 2001 as a Java-based mock library but originally allowed only mock interfaces. The extension that allowed class mocking appeared in 2003. There is now a wide range of mocking frameworks in many different languages, e.g., Moq [79] for C# and Mocker [80] for Python.

By around 2000, SOC gained increased attention in the application development industry. Services were primarily considered as test units, and functional testing was conducted in accordance with service specifications [81]. When services evolved and moved towards the SOA, different testing strategies emerged with respect to evaluating SOA capabilities (i.e., discovery, binding, and publishing). Testing service compositions and versioning also saw substantial demand during this time.

Developing applications with external services has caused difficulties in testing, as adequate testing with external dependent services has not always been feasible. Initially, engineers practised mocking to emulate dependent services. There, the behavioural responses of each method call were specified explicitly in order to bypass their absence at a particular test run. Mocking, however, posed a greater hurdle in the service-oriented paradigm, primarily due to the lack of a detailed understanding of service syntax and semantics as well as being too time-consuming. Service Virtualisation (SV) [6, 7] has originated from the industry as an alternative to mock objects. SV is a cost- and time-effective method for automatically generating service responses from recorded traffic. Section 2.2.2 discusses further details about SV and introduces important SV steps. In the early days, SV practises used the record and replay approach, but currently use AI techniques to infer the service's semantic model to be used to generate service responses. SV is still a new research topic in application testing; in particular, academia has begun to study this approach only around the mid-2010s.

Numerous automated testing tools for service-oriented application testing have also been implemented. For example, JMeter [82] was developed as a load-generating performance testing tool by the Apache Software Foundation (ASF) in 2001, the first version of SoapUI [83] was released in 2005 for Web service functional testing, Parasoft released its SV solution [9] in 2002 as the first step towards SV.

A shift in service-orientation towards microservices has been taking place over the last few years. As the concept of microservices is relatively new, there are very

12. `https://squeak.org` [accessed 02 Aug. 2020]
13. `http://sunit.sourceforge.net` [accessed 02 Aug. 2020]
14. `http://junit.org` [accessed 02 Aug. 2020]

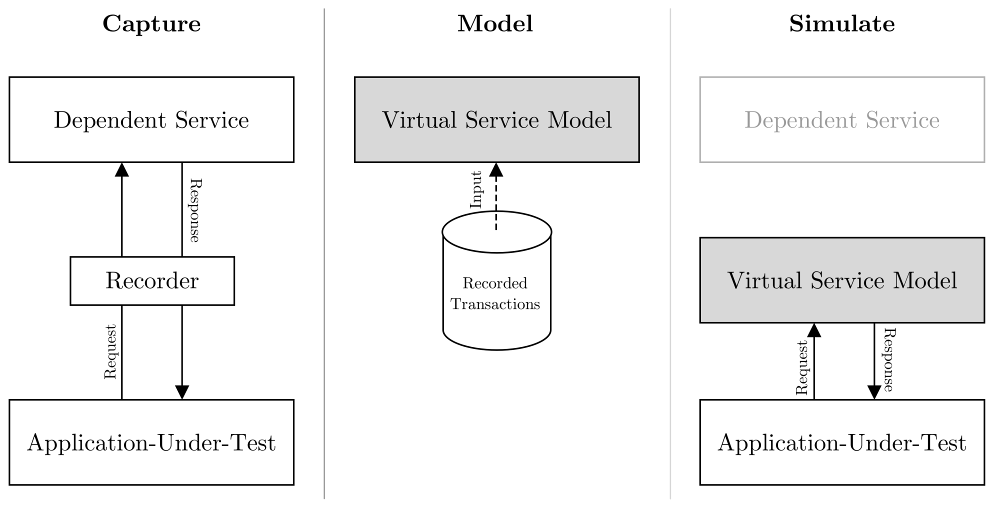


Figure 2.2: The Virtual Service Creation Process

few studies in the field of microservice testing. Since microservices are expanding SOA practises, industry practitioners use most of the current service-oriented application testing methods, such as SV.

2.2.2 Service Virtualisation

As SOC opens the possibility of service reuse in different contexts, today most applications are not built from the ground. Instead, third-party services are used extensively to realise certain functional behaviours that an application needs. Access to external service dependencies is, therefore, important when performing application testing (e.g. integration tests, functional tests, and end-to-end testing). But this raises a greater challenge as external services may be:

- lack of control over many aspects of the test situation
- difficult to configure within a test environment
- only available in a limited capacity or at inconvenient times for testing
- costly to use
- restricted in the observability of service specification and code

Service Virtualisation (SV) eliminates such dependency constraints in application testing. It is a practise to create virtual service models that are capable of simulating the interactive behaviour of a dependent service in such a way that it responds and reacts, in the same way, the actual service [6, 7].

Typically, a semantic model of the service is derived from recorded message-level interactions between the Application-Under-Test (AUT) and the real dependent service. Those recorded transactions usually follow the protocol specification and contain information about request-response patterns, including parameter values and possible temporal properties. The inference usually takes place through supervised machine learning (learning from input-output pairs). SV uses the inferred models to communicate with the AUT by producing responses to incoming requests and thus tries to simulate the behaviour of constrained services. As shown in Figure 2.2, the basic process of SV can be simplified into three key stages [6].

1. Capture: record live interactions between an AUT (in terms of request-response messages) and the service on which it relies
2. Model: construct a virtual service model using the collected traffic
3. Simulate: deploy the model in the development environment as a stand-in for the real service that generates responses to incoming requests.

SV enables frequent and comprehensive testing, even in the absence of actual services in the application architecture. This helps in lowering testing costs, increasing productivity, and delivering application of higher quality in a shorter time frame. It could, therefore, be beneficial to use SV to test applications that rely heavily on external, more lightweight HTTP services. We suggest using SV in this study to infer some of the response attributes of HTTP services.

2.3 Symbolic Machine Learning Techniques

Artificial intelligence enables machines to learn from experience and/or data. It has many approaches, but most examples of artificial intelligence used today (from chess computer programmes to self-driving cars) rely heavily on machine learning techniques. Amongst them, deep learning is currently the most widely used machine learning technique. Despite its high efficiency, the most important weakness in deep learning is the lack of interpretability of the models generated, which makes it difficult to handle descriptive learning tasks. In the contrast, the models created using symbolic machine learning techniques produce inherently interpretable models (also referred to as explainable models) that directly incorporate interpretability into the model structure and are thus are self-explanatory. In the context of symbolic learning, attribute-based learning and description logic learning are typical approaches. Thereby, Section 2.3.1 explores the evolution of artificial intelligence techniques with an emphasis on machine learning. Sections 2.3.2 and 2.3.3 describe common symbolic techniques: attribute-based learning (i.e., decision tree and rule-induction) and description logic learning (has been developed as an extension of inductive logic programming) where few selective algorithms are discussed in each category, i.e., C4.5, RIPPER, PART, and OCEL (note that the exact implementation of these algorithms has not been studied in depth since it is beyond the scope of this thesis).

2.3.1 History of Machine Learning

Machine Learning (ML) is a sub-field of the Artificial Intelligence (AI) discipline, which aims to design and develop algorithms and techniques that enable models to be automatically learned from sample data (i.e., training data/examples) that can be used to make predictions or decisions. The origin of ML dates back almost to the beginning of AI [84]. Since then, numerous ML algorithms have been developed, ranging from declarative, symbolic forms to neural network training and numerical, statistical methods.

There are two major categories of ML: supervised and unsupervised. The supervised ML is based on the labelled training data (example input-output pairs). Regression (algorithm returns a numerical target for each example) and Classification (algorithm attempts to classify each example by selecting between different classes) are two types of supervised techniques. Unsupervised ML does not require such initial labelling.

Symbolic ML (SML) has become one of the most impactful research areas within the AI in recent years. It is an ML approach that involves training models in a declarative form capable of human comprehension. In 1952, Arthur Samuel at IBM [85] created one of the first learning programmes. It was a programme to play Checkers, capable of studying movements and learning patterns that would offer better moves for the latter stages and was a truly exciting development back then.

Sub-Symbolic ML is yet another widely accepted field of AI that is *black-box* by nature, which basically means that the trained model can not be inspected to interpret the output in terms of the features of its input [86]. This contrasts with symbolic AI techniques where the trained model has human-readable explanations. The Neural Network (NN) is a prominent example of sub-symbolic learning strategies that emerged from the interest of academic and industrial researchers in processing data similar to that of the human brain. The general idea is to train layers of functions as much as neurons in the human brain to convert data input into the desired output. The first major step towards NN was proposed by Frank Rosenblatt, named Perceptron, in 1957 [87]. The perceptron was based on a single-layer NN which was able to learn classification models for grouping data into one of the two classes by adjusting the weights of the connection. It was successful in classifying certain patterns, but unfortunately it was rejected as a general learning technique because there were fundamental limitations in what functions could be presented, such as the inability to properly measure even a simple function like XOR [88]. In the late 1960s, heuristic-based pattern recognition algorithms were also developed, including Nearest Neighbours [89]. It was a huge advancement in ML at that time since approximation and heuristics-based techniques allowed computationally intensive problems to be solved.

The 1970s era was known as AI Winter, as funding and interest in AI and ML research fell substantially. Not much work was done in the ML area until the 1980s. However, in the early 1980s, researchers showed renewed interest in NN. In 1981, Multi-Layer Perceptron (MLP) [90] was discovered with Backpropagation (BP) learning algorithm in order to overcome Perceptron's inherent limitations. By using hidden layers of neurons, the MLP could be used for a wide range of complex functions. Several researchers in the early 1960s developed the BP algorithm as a general method of optimisation for performing automatic differentiation of complex nested functions. But it was not applied to NN until the early 1980s. Today, BP is the key element in NN architectures.

The revival of SML began with the introduction of attribute-based learning approaches, more specifically with the development of algorithms for Decision Tree (DT) and Rule induction. The DT algorithm, known as Iterative Dichotomiser 3 (ID3) [91], developed in the early 1980s was one of the first significant developments. DT algorithms typically construct classification models in the form of a tree structure based on the provided attributes. The induction of DT became quite popular at that time as the models provided a clear representation of patterns in the training data. After ID3, several different learning algorithms have been developed by the ML research community, such as C4.5 [26], Random Forest [92], ADTree [93], and CART [94]. Section 2.3.2 further describes the concept of the DT.

In parallel with these, the Quasi-Optimal (AQ) family of algorithms [95, 96] emerged as one of the early approaches to rule learning. The aim of rule induction is to produce classification models as symbolic decision rules (essentially are equivalent to propositional logic) where humans can interpret the outcome with relative ease. This approach has been widely adapted with promising benefits, and several other rule learners have emerged such as CN2 [97], OneR [98], DecisionTable [99], RIPPER [27], and PART [28]. Section 2.3.2 is a brief description of the rule induction. Still, DT and rule learning methods are the prominent topics in AI due to their easy comprehensibility.

In the early 1990s, there was a growing demand for ML techniques which have the capability to represent complex relationships among instances. This resulted in the introduction of a number of algorithms that learn at first-order predicate logic levels, such as FOIL [100] and Golem [101]. This led to the creation of a new ML area called Inductive Logic Programming (ILP) [102, 103]. ILP is a type of SML that uses logic programming. The distinctive feature of ILP is that it can use background knowledge in the learning process. It aims at finding a hypothesis (a set of rules) that covers all positive examples and none of the negatives while taking into account the background knowledge. Since then, a number of ILP algorithms (such as Aleph [104]) have evolved that can induce theories from examples and background knowledge, and use computational logic as a representation mechanism. Using an expressive representation language, ILP algorithms gained an advantage over other learning approaches. Yet, ILP continues to be one of the ongoing research fields of ML.

During the same timeframe, the ML work shifted from knowledge-driven to more data-driven learning, mainly due to the intersection of Computer Science and Statistics. One of the most significant findings during that time was the Support Vector Machines (SVMs) [105] proposed in 1995. Similarly, Bayesian Networks [106] emerged as another approach to statistical learning.

ML work from 2000 was largely influenced by the advent of big data. At that time, researchers developed a new collection of learning algorithms called Bandit algorithms [107–109] which made learning simpler and more adaptable to large-scale problems. By 2001, Random Forests (RF) [110] was created as an ensemble learning model that operated by creating a multitude of decision trees at the time of training and providing the class that is the most common among the tree classes. Meanwhile, the Logistic Regression has been rediscovered and redesigned for large scale ML problems [111]. With the introduction of the Semantic Web [112], Description Logics (DLs)/Web Ontology Language (OWL) became the standard representational language for knowledge bases. By around 2010, ILP broadened its scope by considering the concept learning descriptions presented in the DLs/OWL; constructing hypotheses in the form of class descriptions that could include conjunction, disjunction, and existential quantification. Several learning algorithms were developed at the time. Notably, implementations that were explored with the DL-Learner framework (e.g., OCEL and CELOE) [113, 114] attracted much interest in the 2010s. Section 2.3.3 presents a more detailed explanation on DL learning.

NN began to recover around 2005, in particular, during this period computer vision made efficient use of conventional NN. It has also been able to make their way into fields like natural language processing and voice recognition. In the 2010s, Deep Learning techniques [115] became popular in NN due to high learning capabilities and have been at the forefront of AI research, such as robotics, autonomous vehicles, and medical diagnostic systems. Deep learning tries to model high-level abstractions of training data using a deep graph with multiple processing layers that consist of several linear and non-linear transformations. The main limitation to deep-learning systems is that it is difficult to understand how the system came to a conclusion, particularly when there is a need to understand unexpected choices (i.e., particularly transparent learning is needed when it comes to critical applications such as self-driving cars). There is, therefore, a recent interest in combining deep learning with symbolic logics to provide explanation facilities. This also resulted in a new area of AI called Explainable-AI [22]. This will provide a way to understand the patterns in AI systems easily.

2.3.2 Attribute-Based Learning

Attribute-based learning is a supervised learning technique that uses the propositional attribute-value language to describe training sets and predictive models: objects in the training set must be described by their values for a fixed set of attributes (features), and the predictive model must be represented as a function of those same attributes. The key approaches to attribute-based learning include decision tree induction and rule induction [116].

The learning process is simple and yields efficient and comprehensible results. The main advantage comes from the expressive power of the propositional attribute-value language used to describe instances and concepts. Also, many learning algorithms which operate within this context do not use the background knowledge of the domain. Attribute-based learning techniques have proved useful in a wide range of applications. Also many of the practical data mining/ML tools that are used today often support attribute-based settings. WEKA[15] is a notable example of this.

Decision Tree Induction

Many attribute-based learning algorithms are used to express what is learned as a Decision Tree (DT). A DT is a flowchart like tree structure, where each node represents an input attribute, each branch represents a possible value that an input attribute can hold, and each leaf represents a value of the target attribute that is to be predicted.

These algorithms use divide-and-conquer strategy in top-down fashion to construct a DT structure from a training set of instances [117]. The algorithm first

15. `https://cs.waikato.ac.nz/ml/weka/` [accessed 02 Aug. 2020]

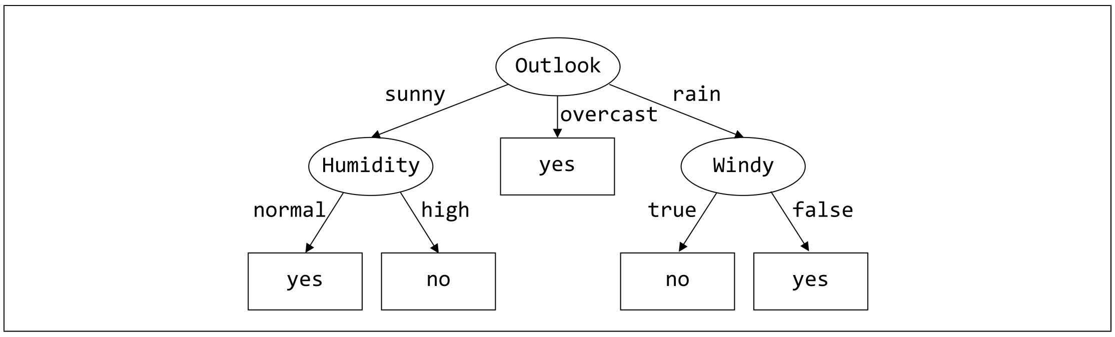


Figure 2.3: A Sample Decision Tree for GOLF Dataset

selects the best attribute for the root node based on a splitting criterion and generates a branch for each possible value of the attributes, then splits the instances into disjoint subsets, one for each branch that extends from the node. As the tree is being constructed, the procedure is applied recursively to each branch. If a set includes only examples from the same class, the corresponding node will be converted into a leaf node and labelled with the class. This process of forming a DT is straightforward and thus provides interpretable solutions. DTs can also be interpreted as a set of decision rules (each path from the root node to a leaf node represents a rule) [118]. Figure 2.3 shows a sample DT for a well-known GOLF dataset [119], where examples are explanations of weather conditions (Outlook, Humidity, Windy, and Temperature) and the target is whether or not these conditions are ideal for playing golf. Classification of a new instance starts at the root node and selects the branch that corresponds to the attribute value. This continues until a leaf node arrives and selects it as the target attribute value.

Over many years, the algorithms for constructing DTs were developed and improved, starting with ID3 (Iterative Dichotomizer 3). C4.5 is one of the best known and most widely used DT implementations.

- **C4.5 Decision Tree Algorithm**

  The C4.5 [26] algorithm has become the industry standard to produce DTs because it actually fits well for most types of problems. It is an improved version of ID3 [91]. In tree learning, C4.5 uses entropy (the measure of uncertainty in the dataset, the less entropy at the node, the more information is known about the classification of the data at this point of the tree) and the information gain (the estimate of the difference in entropy from before to after the set is split on the attribute) to select the best attribute to divide the dataset on each iteration; the attribute for which the resulting information gain is maximum (or smallest entropy) is selected to do the split. This makes C4.5 capable of achieving an optimal solution. The C4.5 algorithm enables the handling of training data with missing attribute values, given they are distributed statistically in relation to the known values of that attribute. C4.5 works with both discrete (has a finite number of possible values) and continuous (has an infinite set) attributes. In order to handle continuous attributes, C4.5 specifying a cut-off threshold and then splits the list into those whose attribute value is above the threshold and those that are less than or equal to it. C4.5 carries out pruning (removing nodes) of the constructed tree which results in smaller trees, simpler rules, and more

```
(Outlook=sunny)and(humidity=high)->PlayTennis=no
(Outlook=rainy)and(windy=true)->PlayTennis=no
(Outlook=overcast)->PlayTennis=yes
(Humidity=normal)->PlayTennis=yes
->PlayTennis=yes
```

Figure 2.4: A Sample Decision Ruleset for GOLF dataset

intuitive interpretations. This allows the model to well generalise any data from the problem domain (reduce overfitting).

Rule Induction

Other attribute-based learning algorithms represent induced knowledge as either a ruleset or a decision list (rules are ranked by their priority) for each class to be described. Any set of rules takes the form of a collection of *if-then* statements. Rule Induction algorithms use separate-and-conquer strategy (also referred to as covering approach) [120] to learn rules, whose basic idea is to look for a rule that explains part of the training data, delete the covered examples successively and repeat the process with the remaining examples by learning more rules until there are no examples left. This means each instance of the training set is covered by at least one rule. Rule learners make training models highly comprehensible and human-readable, just as DTs do [121]. Figure 2.4 presents a sample set of rules, a ML algorithm could be learned from the GOLF dataset [119]. The rules are tried in order to classify a new instance and chooses the class of the first rule that covers. If no rule is applicable, the default rule is chosen (which typically predicts the majority class)

There are different rule induction algorithms that share this simple separate-and-conquer technique (usually individual algorithms vary mainly in the way they learn single rules). RIPPER and PART are two dominant schemes.

- **RIPPER Rule Induction Algorithm**

  The RIPPER (Repeated Incremental Pruning to Produce Error Reduction) [27] is considered as the state-of-the-art in inductive rule learning. This is a refinement of Incremental Reduce Error Pruning (IREP) algorithm [122]. There are two main stages in the algorithm. First, the training data is randomly divided into a growing set and a pruning set (ratio 2:1). Using the growing set, a rule for each class is extracted from an empty rule, by greedily applying conditions to the rule until it is 100% accurate (does not cover negative instances). It tests every possible value of each attribute and selects the condition with the most information gain. Once the rule is grown, it is immediately pruned using the pruning dataset by considering removing the final condition sequence from the rule that maximises the worth of the rule (e.g., success rate). That is done until the deletion does not increase the worth. The process continues until rules are created for all classes. After constructing the initial set of rules, an optimisation step is performed. Here, two variations of each rule are produced in the initial ruleset. One of the variations is created by an empty rule by growing and pruning the rule where the pruning stage is guided to optimise the accuracy of the entire rule base. Another variant is generated by greedily adding conditions to the original rule. The rule with the minimum descriptive length will then be chosen for the final rule base. The algorithm continues to optimise the rules for the initial set of rules. As the rule is revised and constantly modified, this could lead to an increase in accuracy. By repeated incremental pruning, RIPPER effectively prevents overfitting. The algorithm also manages the missing values satisfactorily without the need for any preprocessing and is able to deal efficiently with continuous attributes.

- **PART Rule Induction Algorithm**

  PART [28] stands for Projective Adaptive Resonance Theory. It is a separate-and-conquer algorithm which learns a decision list of rules. For each iteration it produces a partial C4.5 DT, prunes the tree for the current set of instances (using the same C4.5 heuristics), makes the leaf with maximum coverage a rule, and discards the tree. The instances covered by the rule are then excluded, and the algorithm continues to create rules recursively for the remaining instances until none is left. The construction and discarding of partial DTs for the creation of rules avoids the tendency to over prune. Pruning is used to handle overfitting. Adapting C4.5, PART offers a simpler and more efficient way to extract optimised rules. It also has the ability to handle missing data as well as a number of data types, including discrete, binary, and continuous data, as does C4.5.

WEKA (Waikato Environment for Knowledge Analysis)

WEKA [123] is a popular machine learning and data mining workbench which contains numerous inbuilt algorithms for classification and prediction, accompanying with techniques for preprocessing and postprocessing of data. It supports a variety of file formats including most common CSV (Comma-Separated Values) and ARFF (Attribute-Relation File Format). WEKA also has a general API to embed other libraries.

In this study, experiments are conducted in the WEKA environment by utilising the J48 decision tree classification algorithm (Java implementation of the C4.5 in WEKA), JRip (WEKA's implementation of the RIPPER), and PART to construct classification models to infer response properties of HTTP services. This thesis examines the possibility of these algorithms producing accurate, human-readable logic for HTTP response features.

2.3.3 Description Logic Learning

Description Logic (DL) learning is a supervised ML technique that has its roots in Inductive Logic Programming [102, 124]. DL learning applies Description Logics (DLs) as the representation mechanism (DLs are a family of knowledge representation languages with well-known DLs like $\mathcal{ALC}$, $\mathcal{SHOIQ}$, $\mathcal{SHIQ}$, $\mathcal{SHOIN}$, and $\mathcal{SROIQ}$). The expressiveness of DLs gives DL learning the flexibility to specify more domain knowledge (capable of using the complex structures of available background knowledge in the learning process) and the understandability of learnt theories.

TBox

```
Class: Person         Class: Female       Class: Male
SubClassOf: Thing     SubClassOf: Person  SubClassOf: Person

ObjectProperty: hasChild
```

ABox

```
Individual: tom       Individual: eve     Individual: john
Types: Male           Types: Female       Types: Male
                                          Facts: hasChild tom

Individual: mary      Individual: peter
Types: Female         Types: Male
Facts: hasChild tom   Facts: hasChild mary
```

Figure 2.5: A Sample Ontology for FORTE Dataset

DLs describe domain knowledge in terms of concepts that model sets of objects, and roles that model binary relationships between objects. These atomic concepts and role are combined by using appropriate constructs such as negation ($\neg$), intersection ($\sqcap$), and existential restriction ($\exists$) to produce more complex terms. Connectives are also used to define relationships between terms, for example, inclusion ($\sqsubseteq$). Every such relationship is known as an axiom. In DL learning, data is stored in the Knowledge Base (KB) as a set of axioms. A KB is also referred to as a background knowledge base or ontology and consists of two components: TBox and ABox. The TBox encodes the terminology (domain vocabulary) that defines the general properties of concepts and roles. The ABox encodes assertions of individual objects in terms of concepts and roles. More details on DLs and their semantics can be found in [125]. For the purpose of learning a definition, certain instances within the KB are marked as positive examples and others as negative ones.

OWL [126] is considered to be one of the most common DL-based languages. OWL stands for Web Ontology Language and is the official W3C standard ontology language for the Semantic Web [112]. It inherits characteristics from Resource Description Framework (RDF) and RDF Schema (RDFS) resulting in enhanced expressive power. The KB at OWL is called *ontology.* In OWL, different naming conventions are usually used when compared to DLs. The concept in the DLs corresponds to the OWL *class*, the role corresponds to the *property* and the object represents the *individual. Thing* is the superclass of all classes in OWL ontology. Furthermore, OWL properties are differentiated between *object properties* and *data properties* where an object property describes the relationship between two instances and a data property describes the relationship between an instance and a literal. The ontologies can be processed by a reasoner to automatically infer new knowledge and also to check the logical consistency of the ontology. The first version of OWL included three major dialects: OWL Lite, OWL DL, and OWL Full. They were based on the $\mathcal{SHOIN}$ language. The most up-to-date version of OWL is OWL2 [127], which comes in two flavours: OWL2 DL and OWL2 Full. OWL 2 provides a number of extensions to OWL based on the $\mathcal{SROIQ}$ language.

All these various dialects of OWL and OWL2 have varying expressive capacities, and OWL2 DL is currently the most expressive with reasoning capabilities.

The most common setting of DL learning is to learn OWL class expressions that cover all given positive examples and none of the negative ones while taking into account the background knowledge. All positive examples are, therefore, representations of the induced expressions, and none of the negative examples are instances of it. Because of its logical representation, the learned class expressions are easy to understand. Figure 2.5 shows a sample ontology representing a part of the well-known FORTE family dataset [128]. This dataset contains observations of family relationships, such as Aunt, Brother, Daughter, Father, Uncle, etc. Given the set of positive examples = $\{john, peter\}$ (who are fathers) and negative examples = $\{mary, tom, eve\}$ (who are not fathers), an ML algorithm could, then, suggest that the `Father` relationship is equivalent to the following OWL class expression in Manchester OWL syntax[16]:
`Male and hasChild some Person`

DL-Learner[17] has become the central implementation of DL learning. The platform provides a wide range of ML algorithms for learning concepts in DLs/OWL. Among them, OCEL is the standard learning algorithm and is the commonly used and proved to be competitive.

- **OCEL Class Expression Learning Algorithm**

  OWL Class Expression Learner (OCEL) [29] is introduced and distributed with the DL-Learner as the standard learning algorithm for supervised learning in DL. It is designed specifically for learning class expressions that are concise and readable. The algorithm uses a top-down strategy to learn class expressions. It starts with the root of the class hierarchy (i.e., Thing) and applies the downward refinement operator and the criterion of horizontal expansion (sum of its length and number of times it has been refined) to generate descriptions in the search space (by means of specialisation) until an accurate description is found. The selection of descriptions in the search space for expansion is primarily based on the accuracy of the description with respect to the positive and negative examples. A description can be refined in the search space several times. However, it is only allowed for the refinement operator to produce descriptions of a length that is shorter or equal to the horizontal expansion of the refined description (in order to accommodate the infinity property of the refinement operator). These heuristics provide the best possible generalisation of examples, avoiding overfitting. The OCEL algorithm also supports nominal (categorical) and numerical data types.

DL-Learner

DL-Learner [130] is a prominent open-source software framework for DL concept learning. This offers a range of learning algorithms that support a variety of DLs, including $\mathcal{ALC}$, $\mathcal{ALCQ}$, and OWL/OWL2. It also contains reasoners, such as

16. See [129] for more details on Manchester OWL syntax.
17. `http://dl-learner.org` [accessed 02 Aug. 2020]

closed world reasoner, or can be connected to common OWL reasoners (e.g., Pellet[18], HermiT[19]). DL-Learner provides an API which allows the tool to be easily used and extended with new features.

In this study, the experiments are further extended by using the OCEL learning algorithm in DL-Learner tool to construct class descriptions for HTTP response features and to determine the suitability for generating reliable, comprehensible results.

2.4 HTTP Protocol

The Hypertext Transfer Protocol is the underlying protocol used by the Web and has become the popular application-layer protocol for distributed service-orientated applications. The following sections take a close look at the basics of the HTTP protocol, in particular, Section 2.4.1 briefly overviews the protocol, Section 2.4.2 details the structure of HTTP request and response messages as well as the encoding rules, and finally, Section 2.4.3 presents a sample representation of the HTTP message exchanges.

2.4.1 HTTP Overview

The HyperText Transfer Protocol (HTTP) is an application-layer protocol that was originally designed as the underlying protocol for the Web. It is an extensible protocol that has evolved over time. The initial version of HTTP (HTTP/0.9) was created in the early 1990s and as the Web grew in size and popularity, it eventually expanded to the HTTP/1.0 [132] and HTTP/1.1 [133], with improved performance and added features.

Recently, HTTP/2 [134] was launched as an enhancement of existing HTTP 1.x specifications for faster and safer data transfer. The demand for the HTTP protocol has been further enhanced due to the increased use of service-oriented computing in the development of distributed enterprise applications. HTTP is currently the most widely adopted protocol for building services across networks.

In general, HTTP is a client-server protocol. Like most network protocols, the basic HTTP communication process involves a relatively simple exchange of a request-response message pair between the client and the server. An HTTP client is usually a computer programme that initiates request messages to prompt the server to take actions. It establishes a Transmission Control Protocol (TCP) connection for sending HTTP request messages (HTTP allows multiple requests per connection) to a particular port on the host (default TCP/IP port is 80 but other ports can be used). An HTTP server is another computer programme that listens on that port and is waiting for a request message from the client. After receiving the request from the client, the server must take appropriate action specific to the request and send a response message indicating whether the request has been successful or not, and may also contain the requested content in its message body. Once the response has been delivered, both the server and the client

18. For info, see [131]
19. `http://hermit-reasoner.com` [accessed 02 Aug. 2020]

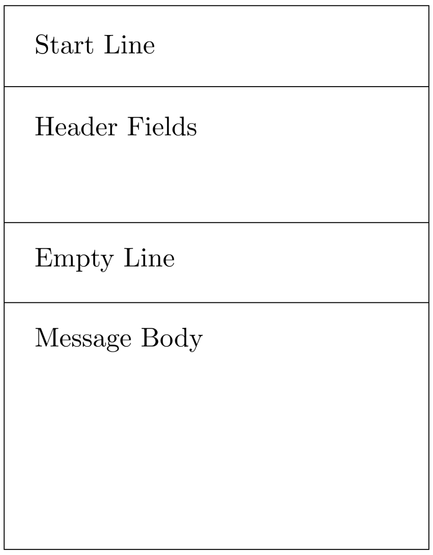


Figure 2.6: The HTTP Message Structure

forget each other and do not maintain state between the exchanges of messages. This feature makes the HTTP protocol *stateless*.

HTTP messages as described in HTTP/1.1 and older protocol versions are basically human-readable (text-based) and simple, making them easier to use. In HTTP/2, these simple messages are encapsulated into frames to improve performance, making it more difficult to read directly. But the basic message semantics has remained the same since HTTP/1.0. All HTTP messages are intended to follow the generic message format. A brief description of this standard message structure is presented in the following subsections.

2.4.2 HTTP Generic Message Format

The generic message format for HTTP is shown in Figure 2.6. It is based loosely on the RFC 822 message specifications for electronic mails [135] and the Multipurpose Internet Mail Extensions (MIME) specification [136]. Each HTTP message begins with a start line, then contains a number of message headers, followed by an empty line, and optionally, a message body.

Start Line

The start line is a special text line that reflects the nature of the HTTP message. The request line is the start line used for request messages. It states the action that the client wants to be performed, followed by a resource upon which the action should be taken, and the version of HTTP the client is using. Each element is separated by a space (SP) character. The formal syntax for the request line is:

```
<REQUEST-METHOD>SP<REQUEST-URI>SP<HTTP-VERSION>
```

The request method specifies the type of action to be performed on the resource defined by the request URI. Table 2.1 lists the most common HTTP methods. The request URI specifies the resource location and the resource name to identify

Table 2.1: Common HTTP Methods

Method	Description
GET	asks the server to send the data of a resource specified by the URI
HEAD	asks the server to send information about a resource but without its data
POST	asks the server to add a resource or take an action on an existing resource
PUT	asks the server to add or update (if the resource already exists) a resource in its entirety
PATCH	asks the server to update part of an existing resource
DELETE	asks the server to delete a resource

Table 2.2: Common HTTP Status Codes

Status Code	Reason Phrase	Description
200	OK	The request succeeded and the resulting resource is returned in the message body.
201	Created	The request succeeded and a new resource is created.
204	No Content	There is no content to send for this request
400	Bad Request	The server could not understand the request due to invalid syntax.
401	Unauthorized	The request needs user authentication
404	Not Found	The requested resource does not exist in the server
422	Unprocessable Entity	The request is well-formed but unable to be followed due to semantic errors.
500	Internal Server Error	An unexpected error occurred inside the server
503	Service Unavailable	The server is temporarily unavailable

the resource which the request should be applied. Every URI follows a particular form:

```
<SCHEME>://<AUTHORITY><PATH>?<QUERY>#<FRAGMENT>
```

where the scheme defines the mechanism to be used to reach the resource (for example, HTTP or HTTPS), the optional authority component consists of user identification details (host and optional port number), the path element identifies a specific resource, the query shows additional data that the resource can use, and the fragment contains an optional identifier to a specific part of the resource.

The status line is the start line used for response messages. It specifies the version of the protocol used by the server and provides a summary of the results of the request. The formal syntax for the status line is:

```
<HTTP-VERSION>SP<STATUS-CODE>SP<REASON-PHRASE>
```

The HTTP version indicates the version number the server uses for its response. The server must return a version number that is no greater than the one sent by the client in its request. The status code provides information about the outcome of the request. HTTP status codes are three-digit integer numbers where the first digit identifies the general response category: 1xx indicates the progress of the request prior to completion, 2xx indicates the success of the request, 3xx redirects the client to another URL, 4xx indicates an error on the client's side, 5xx indicates an error on the server's side. The reason phrase is a short descriptive text string of the status code. Table 2.2 lists the most common status codes, together with the standard reason phrases, and their meanings.

Header Fields

The HTTP header fields provide additional information about the HTTP message. The formal syntax of a header is as follows:

```
<HEADER-NAME>:SP<HEADER-VALUE>
```

Each HTTP header is defined as a key-value string where the key is the header name and it is followed by a colon and then the header value text. A number of headers are listed in HTTP and are grouped into four categories based on the function and type of message they serve.

- General headers: provide basic information about the message itself. General headers are used in both request and response messages.
- Request headers: provide additional information about the request or the client itself. Request headers are used only in HTTP request messages.
- Response headers: provide additional information about the response or the server itself. Response headers are used only in HTTP response messages.
- Entity headers: provide meta-data about the resource carried in the message body (when a resource is carried in the HTTP message body it is called an entity). Entity headers used in both request and response messages.

HTTP headers are optional for an HTTP message and it is possible to send headers in any order. However, the `Host` header must be present in each request according to HTTP/1.1. Besides the standard HTTP headers, custom application-specific header fields can be used. In particular, the use of `X-` prefix generally implies such custom headers.

Message Body

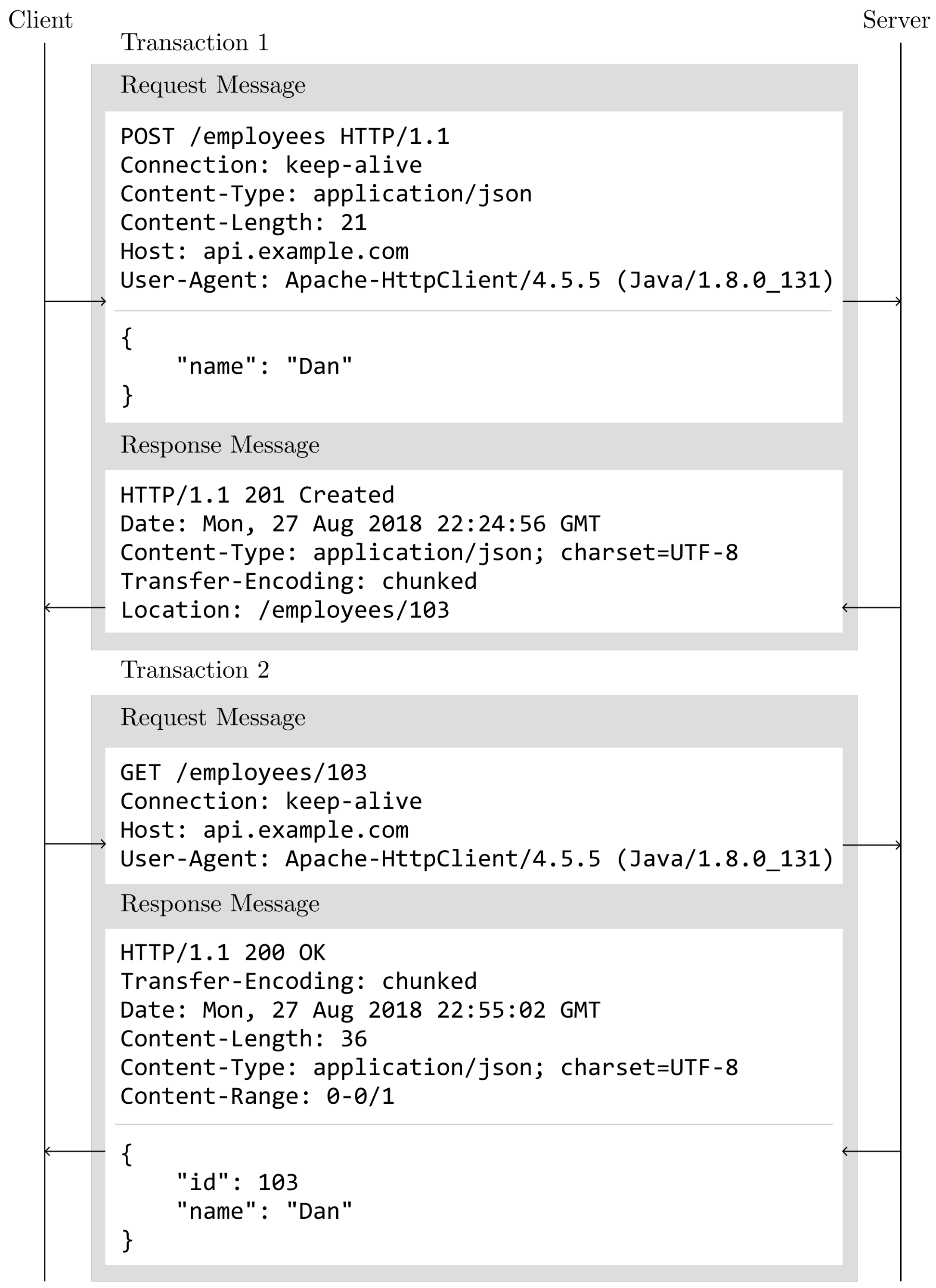
For HTTP messages, the message body is optional. It is where the data is sent from the client to the server (e.g., when using the methods `POST`, `PATCH`, and `PUT`), or where the resource requested by the client is returned, or where the descriptive

```
{
  "employees": [
    {
      "id": 101,
      "name": "John",
      "address": {
        "houseNumber": 21,
        "street": "2nd Street",
        "city": "Palmerston North",
        "state": "Manawatu",
        "zipCode": "4445"
      },
      "contacts": [
        "email1@employee1.com",
        "email2@employee1.com"
      ]
    },
    {
      "id": 102,
      "name": "Peter",
      "contacts": null
    }
  ]
}
```

Figure 2.7: A Sample JSON Representation of Employee Data

```
<employees>
  <employee>
    <id>101</id>
    <name>John</name>
    <address>
      <houseNumber>21</houseNumber>
      <street>2nd Street</street>
      <city>Palmerston North</city>
      <state>Manawatu</state>
      <zipCode>4445</zipCode>
    </address>
    <contacts>email1@employee1.com</contacts>
    <contacts>email2@employee1.com</contacts>
  </employee>
  <employee>
    <id>102</id>
    <name>Peter</name>
    <contacts/>
  </employee>
</employees>
```

Figure 2.8: A Sample XML Representation of Employee Data



```
POST /employees HTTP/1.1
Connection: keep-alive
Content-Type: application/json
Content-Length: 21
Host: api.example.com
User-Agent: Apache-HttpClient/4.5.5 (Java/1.8.0_131)

{
    "name": "Dan"
}
```

```
HTTP/1.1 201 Created
Date: Mon, 27 Aug 2018 22:24:56 GMT
Content-Type: application/json; charset=UTF-8
Transfer-Encoding: chunked
Location: /employees/103
```

```
GET /employees/103
Connection: keep-alive
Host: api.example.com
User-Agent: Apache-HttpClient/4.5.5 (Java/1.8.0_131)
```

```
HTTP/1.1 200 OK
Transfer-Encoding: chunked
Date: Mon, 27 Aug 2018 22:55:02 GMT
Content-Length: 36
Content-Type: application/json; charset=UTF-8
Content-Range: 0-0/1

{
    "id": 103
    "name": "Dan"
}
```

Figure 2.9: An Example Record of HTTP Interactions

text is passed when an error occurs. Some of the common data formats for services are as follows:

- Javascript Object Notation (JSON)

  JSON [56] is one of the most popular interchangeable data serialisation formats used in Web services. JSON syntax is derived from the JavaScript object literals, but its format is text only (does not do any processing or computation).

JSON is simpler and lightweight. It makes reading and writing much easier and simpler for humans and machines. The JSON syntax only defines two data structures: objects and arrays. An object is a set of key-value pairs, and an array is a list of values. JSON supports simple data type: strings, numbers, booleans, and null. JSON syntax also permits nested objects and arrays to represent complex data. As JSON has less extraneous information associated with it, it is easier and faster for machines to parse and generate. JSON can be simply parsed by a standard JavaScript function. JSON schemas describe the structure of JSON-based data. The JSON Schema specification[20] is a popular schema language of JSON.

An example JSON object is given in Figure 2.7. It defines an employees object (enclosed in { }) with an array of two employees (enclosed in [ ]). The object within the array contains another object named *address* and yet another array called *contacts*. This example also includes other data types such as string, number, null, etc.

- Extensible Markup Language (XML)

  XML [50] is another common format for data interchange between Web services. It is a text-based markup language derived from the Standard Generalized Markup Language (SGML)[21], i.e., it is not intended solely for data exchange purposes but developed to define unique markup languages where appropriate (to process and format documents and objects).

  XML is powerful, and can be extended. The XML syntax describes two basic data structures: elements and attributes. XML elements generally contain data text and markup by tags. Attributes are key-value pairs that describe the properties of the elements. Elements can be nested to any depth within other elements to represent complex data. XML does not specify a fixed set of tags or any predefined semantics. It provides facilities for the user to define custom tags. XML offers a wider selection of data types such as number, text, images, charts, graphs, etc. It also has built-in support for metadata, attributes, and namespaces. It powers query languages such as XPath and XQuery (to access data) as well as transformations with XSLT (to store data in one form and convert to any other format). In order to deal with XML, there are different types of parsers available: SAX and DOM. XML has the ability to define a schema that provides the constraints on valid data through standard W3C XML Schema Definitions[22] and then to validate it when producing and consuming the data.

  Figure 2.8 gives a sample XML representation of the same JSON example.

2.4.3 Sample HTTP Exchange

Consider a prototype HTTP-based employee record management service where the client can add, update or delete employees or access existing employee details. The sample interaction trace in Figure 2.9 shows the structural elements of the HTTP request and response messages, as well as an indication of the form of the

20. `http://json-schema.org/specification.html` [accessed 02 Aug. 2020]
21. `https://iso.org/standard/16387.html` [accessed 02 Aug. 2020]
22. `https://w3.org/TR/xmlschema11-1` [accessed 02 Aug. 2020]

headers that could be contained. `Transaction 1` uses the HTTP `POST` request to create a new employee record (resource) and `Transaction 2` uses an HTTP `GET` request to retrieve data from that particular employee.

The client sends the `POST` request to the URI `/employees` first. The HTTP body contains the `name` attribute value of the new resource, `Dan`. The HTTP request includes an example of both message headers and body. The HTTP server generates an `ID` for the new employee, creates an employee in its internal model, and sends a response with `201` status code to the client. Status code `201` means that this is a positive response to a request and that a new resource is being created. This response also includes a `Location` HTTP header that indicates the URI under which the created resource is available.

The client then submits a `GET` request to the URI `/employees/103` to access the details of the new employee. As shown in the figure, this HTTP request does not contain an entity, so there are no entity headers and the message body is empty. The request is handled by the HTTP server and a successful response message is sent with `200` status code. The response message contains the employee object.

2.5 Summary

In this chapter, we explored the basic concepts around different domains relevant to the study. First, we looked at the evolution of service-oriented computing and claimed the importance of lightweight HTTP-based services for the development of heterogeneous client-server applications. The fundamental principles of such services have also been explained. Next, the advances in application testing along with the approaches to test service-oriented applications were discussed and noted the benefits of using the service virtualisation technique to test applications that rely heavily on lightweight HTTP services. The fundamental principles of service virtualisation have also been presented. After that, we summed up the evolution of artificial intelligence techniques with an emphasis on machine learning and identified the significance of symbolic machine learning approaches in training models that are simple and easy to understand. We then outlined the typical symbolic techniques: attribute-based learning (i.e., decision tree and rule-induction) and description logic learning, and identified the most widely accepted implementations of each category. In particular, the C4.5 is the commonly used decision tree implementation, the RIPPER and PART are dominant schemes for rule learning, and the OCEL is the well-established algorithm for description logic learning. The selective algorithms have also been discussed in detail. Finally, we reviewed the HTTP protocol and explored its syntax and semantics.

Chapter 3

Systematic Literature Review

This chapter outlines the related work to the research discussed in this thesis. An overview of the review study is presented in Section 3.1. Section 3.2 briefly explains the adopted approach. Section 3.3 presents a summary of the findings.

3.1 Introduction

The development of modern enterprise applications enables applications to be composed using lightweight HTTP services (i.e., RESTful services). Testing such an application (from the service consumer perspective) requires the availability of services that the application makes requests to. However, continuous access to the real dependent services may be limited or expensive. Simulating the behaviour of such services is, therefore, useful in addressing their absence and in making progress on testing. This is basically what we base our research on. Accordingly, the main purpose of this review study is to comprehensively examine existing approaches to mimic the behaviour of constrained services to which the application-under-test is making requests. That would provide the state-of-the-art software testing practises in service simulation. The review further conducts relevant investigations into existing approaches to services testing (from the service provider perspective), particularly for RESTful services, to better understand the current state of the research.

There is also a growing demand for test automation in order to significantly improve the performance of software testing. Artificial Intelligence (AI) is an emerging approach for automating testing processes. Test practises could be optimised using AI technologies, i.e., AI is intended to deliver more accurate results in less time and offer promising means to make the entire testing process more efficient. Here, we are interested in getting insights into how AI has been used to automate software testing practises. The review, therefore, performs related investigations into AI applications in software testing.

To conduct our review study, we adopt the formal systematic literature review guidelines proposed by Kitchenham et al. [137]. It has attracted considerable interest from researchers in Software Engineering as it provides a more objective process for selecting, evaluating, and interpreting all relevant studies for a particular research compared to conventional review strategies.

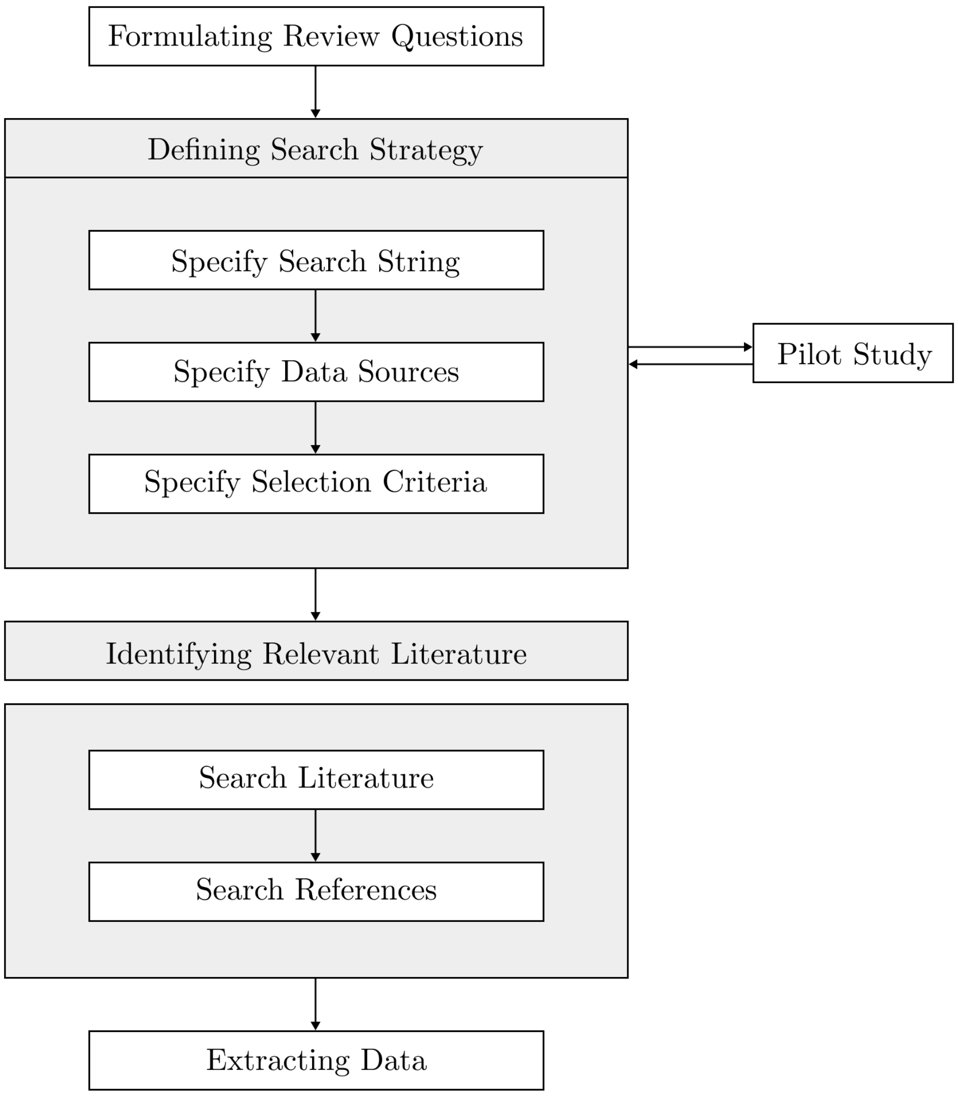


Figure 3.1: The Systematic Literature Review Process

3.2 Methodology

A general overview of the review approach is given in Figure 3.1. There are primarily five stages: formulating review questions, defining the search strategy, identifying applicable literature, extracting data, and finally, formalising a discussion to identify trends and research gaps. The rest of this section provides a detailed explanation of the process.

3.2.1 Review Questions

The scope of our literature review and its emphasis can be outlined using the following review questions (RQs):

RQ1 What software testing tools and frameworks are used to simulate the beha-

Table 3.1: Search Terms

Basic Term	**Alternative Terms**
Testing	Test, Application Testing, Software Testing, Unit Testing, Integration Testing, Quality Assurance, Test Automation, Test Oracle, Test Generation
Services	Web Services, External Services, Third-party Services, SOAP Services, RESTful Services
Simulation	Emulation, Modelling, Mocking, Mock Objects, Virtualisation
Artificial Intelligence	Machine Learning, Symbolic Reasoning, Knowledge Engineering, Neural Networks, Decision Trees, Rule Learning, Support Vector Machines, Deep Learning, Classification

viour of services that an application-under-test interacts with and what are the key benefits and limitations of each mechanism? This question would explicitly describe techniques that are used in service-oriented applications testing to mimic the behaviour of dependent services in order to address their absence in application tests.

RQ2 What is the current state of the research in services testing? This question would summarise what studies are being conducted in services testing. This question is specifically included in order to find research efforts which are used to test RESTful services.

RQ3 What is the current state of the research in which AI technologies are used for software testing? This question would give an overview of the application of AI techniques in software testing practises.

The review study would separately address the questions: the RQ1 will be examined in depth (as it deals with the main objective of this review), while the RQ2 and the RQ3 will be examined in less detail (only to identify the present state of the research).

3.2.2 Search Strategy

In order to retrieve the related literature, we perform the search process in two steps:

1. Primary search phase: search for relevant literature in digital databases, search engines, individual journals, conference proceedings, and grey literature sources
2. Secondary search phase: references are perused for each selected primary study to identify important studies which may be missed during the initial search process

A detailed summary of search strategies in the primary analysis phase is given below.

```
((Testing OR Test OR Application Testing OR Software Testing OR
   Unit Testing OR Integration Testing OR Quality Assurance OR
   Test Automation OR Test Oracle OR Test Generation)
                              AND
(((Services OR Web Services OR External Services OR Third-party
   Services OR SOAP Services OR RESTful Services) AND (Simulation
    OR Emulation OR Modelling OR Mocking OR Mock Objects OR
   Virtualisation))
                              OR
(Services OR Web Services OR External Services OR Third-party
   Services OR SOAP Services OR RESTful Services)
                              OR
(Machine Learning OR Symbolic Reasoning OR Knowledge Engineering
   OR Neural Networks OR Decision Trees OR Rule Learning OR
   Support Vector Machines OR Deep Learning OR Classification)))
```

Figure 3.2: The Search String

Search Strings

The following criteria are used to construct the search string that will be used in our search for relevant publications:

- Derive major search terms from RQs
- Identify synonyms for key terms
- Check keywords on any known publications
- Use boolean `OR` to incorporate synonyms
- Use boolean `AND` to restrict the search

The main terms derived from RQs and synonyms are outlined in Table 3.1. These words are then combined with boolean operators (i.e., `OR` and `AND`) to formulate the final search strings. Figure 3.2 lists the search string that is appropriate for the automated search.

Data Sources

Table 3.2 details the selected sources to be used in our search for related publications. The choice is based on our prior knowledge of the research domain.

Study Selection Criteria

Study selection refers to the assessment of retrieved articles. For this purpose, inclusion and exclusion criteria are defined to filter and eliminate any studies which are irrelevant, so that only relevant papers can be used for data extraction. Mainly, we choose to only consider primary studies published between January 2010 and May 2020.

Below is a description of other criteria for inclusion and exclusion of papers returned from the initial search phase.

Table 3.2: Data Sources

Databases	ACM Digital library IEEE Xplore Springer Link Google Scholar	
Journals	IEEE Transactions on Software Engineering (TSE) IEEE Transactions on Services Computing Journal of Systems and Software (JSS) Automated Software Engineering (ASE) Software Testing, Verification and Reliability Service Oriented Computing and Applications	
Proceedings	International Symposium on Software Testing and Analysis (ISSTA) International Conference on Software Testing Verification and Validation (ICST) International Conference on Service Oriented Computing (ICSOC) IEEE International Conference on Web Services (ICWS) IEEE International Conference on Services Computing (SCC) Enterprise Distributed Object Computing (EDOC) IEEE Congress on Services (SERVICES)	
Grey Literature	**Technical Reports**	IBM Oracle SAP HP CA
	Repositories	Maven GitHub
	Discussion Forums	Stack Overflow

Inclusion Criteria:

- Primary studies on approaches for emulating the interactive behaviour of services to test applications from the perspective of the service consumers
- Primary studies on approaches to services testing from the perspective of the service providers
- Primary studies on AI algorithms applied to software testing

Exclusion Criteria:

- Primary studies on approaches for emulating consumers who send requests to services
- Primary studies on simulation practises but not connected with software testing
- Primary studies on software testing approaches but not related to services testing
- Primary studies on AI-based approaches but not related to software testing
- Primary studies on techniques for testing AI systems

Before applying the search strategy to the actual large-scale search, we carry out a pilot study to validate the search proposal on the basis of the relevant papers that we already know and make a number of refinements.

3.3 Results and Discussion

The final list of 22, 25 and 46 papers linked to RQ1, RQ2 and RQ3 is obtained through the search process. The subsections below describe the extracted data addressing the RQs.

3.3.1 Service Simulation

A number of tools and approaches have been proposed to provide testing environments that are suitable for testing service-oriented applications independently of the services on which they depend. These solutions fall under two groups: mock objects and service virtualisation.

Mock Objects

A common practise for dealing with the absence of dependent services during test runs is to replace the behaviour of constrained services with mock objects. Marri et al. [138] describe the benefits of using mock objects for testing purposes. The creation of mock objects is often supported by generic mocking frameworks, e.g., Mockito [24], EasyMock [78], Jmock [139], and Moq [79], which allow engineers to manually create mock objects that serve as dummy implementations for actual dependencies. Mostafa et al. [140] report on the widespread use of mock frameworks in application development projects. In order to generate the environment required to adequately test an application using mock objects, engineers must explicitly define the interaction behaviour by specifying the expected return values for each method call. In particular, variations in these values require separate test cases, which take more time and effort to maintain. This typical form of mocking is also inexpressive and can not be reused [77].

Some authors aim to create more declarative and generalise mock objects. For example, Tillmann and Schulte [141, 142] introduce a generalised variant called *parameterised mock objects* (i.e., mock objects that represent symbolic variables) to automatically represent various possible return values. Thus, a single call to a mock object could return different values that the unit-under-test expects. In order to facilitate multiple executions without parameterisation, Achenbach and Ostermann [143] use non-deterministic choice technique (specify option sets of return values) to mock objects. These methods could, however, only be used to create mocks manually.

Various techniques are proposed to automatically construct generic mock objects as part of the automated test case generation. For example, Tillmann and Schulte [144] introduce a tool that automatically generates parameterised mock objects dependent on the symbolic execution of the .NET code. It uses symbolic execution to analyse how a .NET unit test and a programme-under-test use a mock object and applies a constraint solver to decide how different return values can affect different execution paths. The tool later produces code that builds a mock object for each observed execution path. Galler et al. [145] suggest an approach to creating mock objects based on formal contact specifications of the original classes (i.e., pre- and/or post-conditions). Islam and Csallner [146] propose a similar approach alongside a symbolic execution engine to Java with the exception that mock objects are based on interfaces (with no concrete implementations) rather than contracts. AutoMock [147] is another proposal that automatically synthesises mock objects based on post-conditions resulting from symbolic execution of path constraints. The work of Arcuri et al. [148] implements a technique in their automated test suite generation tool named EvoSuite to automatically create mock objects using the Mockito framework. However, there is no assurance in these proposals that the mocking behaviour is compatible with the behaviour expected from the components being mocked. Further, these mock objects are not reused across tests and other areas in the application development cycle.

Samimi et al. [149] provide an automated approach that defines and verifies the mock objects against a contract and is reused through tests and other codes where mock objects are required. It involves contract specifications written in a domain-specific language (PBnJ) which describes the intended responses to be mocked (requires engineers to learn the language). It uses a constraint solver to determine the return values of method calls on these mock objects. Solms and Marshall [150] offer a similar kind of solution where the component contract is specified in the standard Java language. It automatically constructs and maintains mock objects in accordance with the contract specification and allows for the reuse of mock objects. It also verifies the correctness of the mock objects upon the contract. However, all of the above techniques require source code, or explicit knowledge or documentation of the system protocols to successfully generate mock objects, most of which are not necessarily available to engineers who write mock tests.

Saff et al. [151] present a mocking technique using the recording and replay approach in their work on test factoring for Java. It automatically captures and records method calls and return values from a series of system tests. Afterwards, it replays the recorded results to the code-under-test by acting as a mock object. There are a few other approaches (e.g., [152, 153]) that suggest a similar mock generation technique to assist the testing process. With this strategy, it is possible to create mock objects without prior knowledge of the element being mocked. The behaviour of mock objects, however, depends on the richness of the recorded traces (availability of records for all possible method calls).

There are relatively few studies which explicitly apply the concept of mock objects to service-oriented applications testing. Mani et al. [154] present a framework that automatically generates *semantic service stubs* (mock objects like entities) from the behavioural contracts of the target services. The elements of a contract include semantic annotations such as pre-conditions that a consumer must meet in order to obtain the service, post-conditions that the service provides to the consumer, and exceptional conditions. These semantic annotations are added to the source code or WSDL specification of the service, and the annotated code is processed through an automated translator to construct the source code for the stub. A constraint solver is used to generate realistic test data to be returned when a service operation is invoked. A semantic service stub would verify request messages and return the appropriate response and exception messages along with test data, simulating some of the behaviour of the service. Test cases for a service-oriented application, therefore, can be carried out throughout the entire development cycle by invoking the service stubs. Ashikhmin et al. [155] use the RESTful API Modeling Language (RAML) specification to automatically build mocks of RESTful services. The RAML specification describes the service interface, endpoints, appropriate request format, and expected response to those requests (particularly a JSON scheme to describe the response body structure). The solution accepts the RAML specification as input and implements a *mock service* to be operated as a Docker [156] container. A mock service would handle incoming requests, starting with the validation of the request endpoints and parameters based on the RAML specification, and it would generate an appropriate response body based on the JSON schema specified in the RAML. Reza and Van Gilst [157] introduce RESTful service simulation framework that encompasses the need for engineers to input the API specification as XML scripts into the framework. The work of Soni et al. [158] proposes a similar framework called MockRest in Java. Overall, the existing solutions illustrate the possibility of simulating the behaviour of actual services by producing responses using mock objects. Then, there are Web service testing tools like SoapUI [83] that support the creation of mock services for both SOAP and RESTful services. Some of these tools allow mock services to be created automatically based on interface specifications while others are required to configure requests and associated responses to simulate actual service behaviour. However, the lack of detailed knowledge of the services or access to the source code and/or service documentations prohibits the use of either of the aforementioned techniques.

Service Virtualisation

Service Virtualisation (SV) is another practise used in the software development industry to address the dependency constraints in application testing. This corresponds to (automatically created) mock objects. It is an approach to replace the target services of an Application-Under-Test (AUT) by virtual service models, ensuring that engineers have continuous access to realistic testing environments. A virtual service model is often created by recording traffic between the AUT and the live service rather than creating the interaction pattern from scratch based on service documentations. SV can represent much realistic behaviour relative to simple mocks.

There are a number of white papers (e.g., [159–161]) that explain SV, its advantages and express the concept of using SV in testing service-oriented applications. Subbiah et al. [162] share the idea of constraint-free testing using SV. Similarly, Upadhyay et al. [163] discuss how SV could address the challenges in testing service-oriented applications.

Multiple vendors, including CA [8] and Parasoft [9], and some open-source projects (e.g., Wiremock [10] and Hoverfly [11]) offer SV tooling. These solutions simulate the behaviour of dependent services through synthesising responses mainly by recording and then replaying interaction messages. Most of these support multiple application-layer protocols. However, all rely on a priori knowledge of the service structure and message protocol. Possibly responses are manually modified after

Table 3.3: Service Virtualisation Tools Comparison

Tool	Protocol Supported	Reasons on State	Commercial	Uses AI
Parasoft	Most	Yes	Yes	No
CA	Most	Yes	Yes	Yes
Wiremock	HTTP	Yes	No	No
Hoverfly	HTTP	Yes	No	No

the recording (i.e., when responses are based on request attributes and data). The quality of synthesised responses depends on the availability of traffic recordings for every potential interaction scenario. In addition, the tools function as a black-box (engineers who use them can not understand how responses are produced by a specific service). Table 3.3 shows a general evaluation of these tools.

Opaque SV [12–15] is a proposal where dependent services are emulated by synthesising responses using semantic models inferred from recorded interactions. It allows responses to be created automatically, without requiring prior knowledge of the service protocols. The inference is done by means of clustering and bioinformatics-based learning techniques, i.e. the VAT cluster algorithm [164] is used to group recorded transactions by request type, the Multiple Sequence Alignment algorithm [165] is used to derive a request prototype for each cluster, and the Needleman-Wunsch algorithm [166] is used to locate the prototype closest to the incoming request (thereby to identify the matching cluster) and to identify common fields between both the request and the response messages of the centroid transaction of the selected cluster. The approach will later take the response from the centroid transaction of the cluster and dynamically modify the identified fields by copying the related information from the incoming request in order to generate a playback response. This approach has been incorporated successfully into the CA Service Virtualisation commercial product. FancyMock [18] is relatively similar to the Opaque SV that focuses on arbitrary message formats. It uses a different clustering algorithm (i.e., K-Nearest Neighbors classification algorithm [89]) to group requests and response messages, and sets a bound to each cluster size. It constructs a dictionary of payload fields for response clusters. The framework determines the closest matching request to an incoming request and selects a corresponding response to generate a response. Similar payload fields between selected messages will be modified from the incoming request values, and all other fields of the selected response will be randomly filled with dictionary values. However, all authors ignored the temporal properties of protocols when formulating responses, i.e., the response generation solely on the basis of the incoming request and the recorded interaction traces, but not the service state history. Therefore, it is recommended that these techniques are only appropriate if the target service is stateless (when all the information needed for a correct response is provided in the request) or if the test scenario does not require highly accurate responses.

The latest research by Hossain et al. [16, 17] extends previous SV studies and introduces a framework for simulating service behaviour, taking into account the state of the service when synthesising responses from recorded interactions. Clustering techniques are used to group requests as well as response messages into type-specific groups, and a single representation (prototype) is inferred for each batch. The kTail algorithm [167] is used to infer a message dependency model to learn the service state from the record-specific interaction history (use request/response types for each transaction). A data dependency model (i.e., a set of substitution rules) is derived by comparing the response prototypes with the corresponding request prototypes from each response cluster. The technique would then use the message dependency model to identify the correct response type for incoming requests (which then be used to select the response prototype). For message-specific fields of the selected response prototype, the data dependence model would be used to insert the required data values. The record-specific fields would be replaced by values from one of the randomly selected interactions of the respective response cluster. This SV solution, however, only takes into account the service's current status when choosing the response type. But, depending on the state of the service, the response body can also be changed, and such changes are not considered when formulating responses in this proposal. Further, it does not incorporate all service features (i.e., message headers) when constructing responses. The service models produced in all of the above-mentioned SV studies are also a black-box for the engineers who use them. It is not possible for engineers to comprehend the application-layer protocol and the underline service. But engineers often want to fine-tune and adjust the generated responses. This requires that virtual service models be produced in a format that is convenient for them to interpret and modify.

Another recent study by Enişer et al. [19, 20] proposes two separate SV methods which make use of past service history transactions when inferring service models from recorded interactions. One of the solutions proposed uses classification as an inference mechanism: [19] employs the rule-based RIPPER algorithm as the base classifier (which we use in our work as well) but [20] does not explicitly describe which classification algorithms are used. The other SV proposal makes use of deep neural networks. Information about the request type and content along with interaction history (use previous request/response types/contents up to 10 transactions) are the features that provide input to algorithms whose task is to induce models to predict the response type and content for a given request. One-hot encoding is used to provide numerical encoding for input features with categorical data. In terms of training time, the classification-based solution works better, but virtual service models trained by neural networks generate more accurate responses. It is, therefore, recommended that the methodology based on classification could be used when the test scenario needs a fast but not very accurate virtualisation. While this work is close to the study presented in this thesis, some significant differences do exist. Their approach lacks the potential for reliable prediction of HTTP-based services. The procedure is principally biased towards predicting the response state. When training models, it uses fixed history sizes and also does not include all service features. The datasets used are not satisfactory as experiments are small and may miss important aspects of real-world, state-of-the-art services. The interpretability of the resulting models is not considered to be a significant measure.

It is evident that there is a growing interest in SV related approaches in both industry and academia. SV is a promising method in service-oriented applications testing as it eliminates the constraints on accessing real dependent services and isolates the application under test. SV could be particularly beneficial in replacing lightweight HTTP services on which modern enterprise applications rely heavily. Existing SV studies propose solutions for the automated creation of virtual models of services based on recorded traffic and employ machine learning techniques as part of their implementations. However, the usability of these approaches remains limited. It is not possible to generate accurate approximations of real responses of HTTP/REST services with current practises (as valid responses should be generated depending on the internal service state). On top of that, their results lack provenance. There is, therefore, a need for a novel SV technique that can automatically generate behavioural responses, while achieving a high degree of accuracy with human comprehension. This is our major motivation in this thesis.

3.3.2 Services Testing

Services testing has been examined extensively in the literature. Some prior studies (e.g., [168–174]) cover broad surveys on Web service testing.

Most of the work focused primarily on black-box testing approaches (the testing process does not require the implementation of source code) for conventional SOAP-based services using their Web Services Description Language (WSDL) specification, like for example [175–182].

There are testing methods that use formal specifications or models to test RESTful services. For example, Chakrabarti and Kumar [183] present a functional testing framework called Test-the-REST to automatically create test cases from service specifications written in the Web Application Description Language (WADL) [184]. This approach is further extended in the work of Chakrabarti and Rodriquez [185] incorporating testing of service connectedness (possibility of accessing any other resource from a root resource) based on models representing the connections between resources. Ed-douibi et al. [186] also suggest a model-driven approach for testing RESTful services based on their OpenAPI specifications[1]. Fertig and Braun [187] present a domain-specific language to describe RESTful service models and further emphasise the automated generation of test cases from these models. UML protocol state machines are introduced by Pinheiro et al. [188] to construct behaviour models for test case development.

Property-Based Testing (PBT) has been used to test RESTful Web services. The key concept behind PBT is to test the validity of the properties by generating random input data and verifying the expected behaviour. Seijas et al. [189] propose a PBT methodology based on property-based models that depicts the idealised form of RESTful services. Similarly, Karlsson et al. [190] applies a PBT approach to automatically generate tests employing OpenAPI specifications. The method produces static test cases and arbitrary parameter values and sequences of operations that exploit previously returned results to perform stateful operations.

Segura et al. [191] propose a Metamorphic Testing (MT) approach for RESTful services to alleviate the *oracle problem* (difficulty in assessing whether the outcome of a method call is correct, within a reasonable amount of time). MT focuses on

1. `https://github.com/OAI/OpenAPI-Specification` [accessed 02 Aug. 2020]

the analysis of the relations among the inputs and outputs of multiple executions of the service-under-test.

Fuzz testing is an increasingly popular testing technique. While first-generation black-box fuzzers generated random input in order to expose defects, modern grey-box and white-box fuzzers use or even infer models of the AUT in order to increase the chances of discovering bugs. An example is fuzzers that can infer the grammar of the programming language or data format [192], or use dynamic feedback from test executions (such as coverage). Recently, some of those ideas have been applied to RESTful services testing [193]. Fuzzing uses AI to generate the actual tests, whereas our approach uses AI to generate services the (user-written) tests interact with, and we use fuzzing techniques to construct the datasets for evaluation (Section 4.3 details the data generation process).

There are other testing approaches that generate tests for RESTful Web services without any formal specification or model, but instead use the service's source code, as for example Arcuri [194] proposes a tool called EvoMaster[2] to automatically collect and exploit white-box information from code instrumentation to create test cases at system level using evolutionary algorithms. Zhang et al. [195] offer an extension to the EvoMaster tool that takes the resource relationships into account while creating test cases.

It is clear from these findings that a range of testing models, tools, and techniques are proposed as the popularity of RESTful services rises. Many of them address black-box testing (focusing on testing based on the API specifications). These studies propose specification-based test cases generation techniques for REST Web APIs by relying on their interface definitions, particularly the OpenAPI specifications. When the source code is fully available to developers, white-box testing would become a viable option. However, the related studies on white-box testing for REST services are rather limited. On the other hand, there is indeed a growing number of research studies suggesting fuzz test tools for REST APIs.

3.3.3 Artificial Intelligence in Software Testing

Artificial Intelligence (AI) techniques, in particular Machine Learning (ML) algorithms, have been adapted and used for the automation of software testing activities. Few review articles (i.e., [196–198]) contain an overview of existing literature on ML-based approaches to software testing. King et al. [196] discuss how AI is to be used to test applications from an industry perspective.

There are research studies which use ML to automate test case generation. For example, Bergadano and Gunetti [199] present an approach for generating test cases that distinguish a given programme from an alternative set of programmes. Inductive Logic Programming (ILP) [102, 103] induces alternative programmes that are equivalent to the original that users can interpret. Sant et al. [200] suggest a technique that automatically derives test cases for Web applications from statistical ML models (i.e., Markov models) based on user session data. Ahmed and Hermadi [201] and Wegener et al. [202] use evolutionary algorithms to produce

2. `https://github.com/EMResearch/EvoMaster` [accessed 02 Aug. 2020]

test cases. Rosenfeld et al. [203] also present a methodology that leverages ML algorithms to generate functional test cases for mobile applications. Chen et al. [204] apply a semi-supervised K-Means clustering algorithm for the selection of test cases during regression testing. The approach groups test cases into clusters (test cases in the same cluster are considered to have similar behaviours). Some studies detail the use of ML algorithms to support test case refinement and evaluation, e.g., Briand et al. [205,206] present a semi-automatic process for re-engineering test suites based on classification rules induced from the C4.5 tree learning algorithm that relate properties of test inputs to output equivalence classes. The engineers can easily interpret these rules to determine potential refinements in test suites. Mayrhauser et al. [207] use Artificial Neural Networks (ANNs) to build a model for estimating the effectiveness of the generated test cases. Gove and Faytong [208] also employ learners in Support Vector Machine (SVM) and grammar induction to identify infeasible test cases. Grammar induction yields results which make it clear for engineers to understand. In terms of efficiency, the SVM-trained classifiers perform well.

Wang et al. [209] explore how SVM could be used to automatically produce test oracles for reactive systems without relying on software specifications. The feature vectors created from the test traces are used as inputs for the proposed method. The generated model works as a test oracle. The work of Agarwal et al. [210] evaluates the effectiveness of Info Fuzzy Networks (IFN) and ANNs to implement test oracles. Test cases are used as inputs to the proposed approach. The model learned is able to predict the expected behaviour of the new test cases. Similarly, Singhal et al. [211] outline two ML solutions to generate test oracles. The first method is based on ANNs, and the second is based on decision trees. Vanmali et al. [212], Shahamiri et al. [213–215], and Gholami et al. [216] also recommend ANN algorithms to create automated test oracles. The work of Monsefi et al. [217] combines a deep learning and a fuzzy inference system to generate oracles. In addition, Hewson et al. [218] propose a ML-based approach to generate statistical test oracles for performance regression testing. Kanewala et al. [219], on the other hand, suggest a method based on the C4.5 tree algorithm to make metamorphic relationship predictions to support the test process without the need for test oracles.

Several research works use AI to assist with test planning, like for example, Cheatham et al. [220] use the COBWEB decision tree type learning algorithm to identify factors that influence the testing time. Software metrics (e.g., code complexity) are used as input for the learning process, and a tree model is built as output to predict the testing time for the new software. Some other research efforts are the work of Briand et al. [221] proposing a method based on the C4.5 decision tree algorithm to predict potential software bugs and locate the actual bugs to reduce debugging time, and the work of Wong and Qi [222] on suggesting an adaptive solution for locating errors based on a neural back-propagation (BP) network.

The application of AI to fuzz testing has received significant attention as well. The most widely adopted AI methodology for fuzz testing is deep learning. Samplefuzz [223], for example, uses Sequence-to-Sequence (Seq2Seq) Recursive Neural Networks to automatically learn a language model for PDF objects from the sample data. The generated model can produce a large number of new PDF objects that can be used to generate test cases. DeepFuzz [224] also uses Seq2Seq to learn the grammar of the correct C programme. It can generate grammatically correct C programmes on the basis of the learned model. NeuFuzz [225] learns about the known vulnerabilities programmes and unknown vulnerabilities in the sample through Long Short-Term Memory Networks (LSTMs) to discover the execution path that could contain the vulnerabilities. Fan and Chang [226] propose an approach that uses deep learning for black-box fuzzing of network protocols. Several other studies exist, such as Cheng et al [227], Hu et al. [228], Cummins et al. [229], Sablotny et al. [230], Rajpal et al. [231], and Gong et al. [232], which use deep learning in fuzzing.

In the context of Web services testing, Ioini et al. [182] propose a method based on the J48 decision tree algorithm (i.e., C4.5 [26]) to provide service providers with the means to expand the WSDL specifications that will allow service consumers to automatically validate their calls on the client-side. The learner receives interaction traces with request parameters and response messages as inputs. As output, it constructs a decision tree model for the input parameters of the service. The service provider could update existing WSDL specifications using the induced rules from the tree model. Such a new version of the WSDL file could be later used by service consumers to annotate their service calls. RESTler [193] offers an automatic black-box solution to AI-driven fuzz testing of REST services through its API. Existing service virtualisation research efforts [12–20] often use AI techniques to infer service semantic models based on recorded traffic that assist service consumers in testing applications that rely heavily on external black-box services in isolation (see Section 3.3.1 for an extensive survey).

All research works demonstrate that the use of AI/ML algorithms is a successful way of automating a wide range of software testing practises. Amongst them, studies based on ILP and decision tree/rules (i.e., symbolic AI) are capable of generating inference models that are easily interpretable. These techniques produce rules which can be customised to the requirements.

3.4 Summary

In this chapter, we reviewed literature from different domains that appear to be most relevant to our study in order to determine the scope of our research work. First, we examined existing approaches and tools to overcome the dependency issues in service-oriented application testing, and confirmed that service virtualisation is the most promising practise. We also analysed current approaches to service virtualisation and identified their major limitations which we aim to address in this thesis. We then surveyed different approaches to services testing and the latest work on AI-driven approaches to software testing which are important references to our research. To the best of our knowledge, this research is the first to study the virtualisation of HTTP-based services and to focus directly on producing HTTP responses with human-readable logic.

Chapter 4

Data Acquisition

The chapter outlines the network traffic datasets that will be used to perform the experiments in this study. These datasets are designed to be suitable for reproducible research in service-oriented computing in general and can be considered as one of the major outcomes of this thesis. Section 4.1 presents an overview of the intent of such datasets and their requirements. Details of the datasets can be found in Section 4.2 and Section 4.3.

4.1 Introduction

Service-Oriented Computing (SOC) is a popular computing paradigm that supports accelerated, low-cost development of distributed applications in heterogeneous environments. Over the past few years, RESTful HTTP-based services have been the dominant technology for realising SOC. This basically allows clients and servers in various languages and running in diverse platforms to work seamlessly together by sending HTTP requests and responding to them. This has, however, created new challenges both for the research and for the engineering community. Of particular interest are scalability, reliability, and security of (systems using and providing) services.

Like other fields of computing research, studies of SOC should aim for reproducibility [33,34]. There is a wider push for reproducibility in computing research, with some disciplines now including research artefact evaluation as part of the standard peer-review process [233]. One way to facilitate the reproducibility and also the dissemination of research is the provision of standardised datasets. By using carefully sourced and/or constructed datasets, research results become (1) easier to reproduce (2) comparable (i.e., results from different studies can be compared), and (3) generalisable (i.e., we can assume with a certain amount of confidence that results from a study can be applied to other data/systems that were not studied).

The provision of such datasets is a key contribution of this thesis. We present GHTraffic, a dataset comprising HTTP transactions extracted from a successful, large-scale service, GitHub, by reverse-engineering API interactions from existing repository snapshots, and augmented with synthetic API interactions that cannot be recovered from snapshots, namely (non-state-changing) queries. In addition, three other HTTP datasets are generated by creating random traffic targeting the services offered by Twitter, Google Tasks, and Slack. In order to form transactions, various operations to create, read, update, and delete (CRUD) service-specific resources are formed, and the respective responses are recorded, simulating service interactions by users through applications. The resources, the operations interacted with are tweets (Twitter), messages (Slack), and lists (Google Tasks). All these dataset creation processes result in large, rich, and diverse datasets. We argue that these datasets can be used for a wide range of studies, including performance benchmarking and service virtualisation (which is the basis of our research in this thesis).

The rest of the section is organised as follows: use cases and requirements are discussed in detail in Section 4.1.1 and an overview of related work is presented in Section 4.1.2.

4.1.1 Use Cases and Requirements

Three scenarios for the type of research that motivates the construction of the datasets are described below.

Performance Benchmarking

Modern enterprise applications usually cooperate with a variety of software services such as Web servers, application servers, databases, proxies, and Web service clients to perform their functionalities. These services need to be tested in order to ensure that they are able to deal with large data and transaction volumes. In particular, performance benchmarking can provide useful indications about how services behave under different load conditions. A typical benchmarking tool generates synthetic workloads or replays recorded real-world network traffic in order to simulate realistic workloads and measures performance-related metrics, such as latency and throughput.

A dataset that is large, complex, and extracted from actual network traffic facilitates the benchmarking of such systems with non-trivial, realistic workloads.

Functional Testing

A standard dataset can also be employed for functional testing. For instance, it can be used to test a generic REST framework with a CRUD back-end provided by a (non-SQL) database. This would take advantage of the fact that such a dataset encodes a certain semantics, usually a combination of the standard HTTP semantics (for instance, the idempotency of certain methods) plus additional, application-specific rules and constraints. In other words, a suitable dataset can provide an *oracle* of correct system and service behaviour. As an example, consider an HTTP `GET` request to a named resource. This request should result in a response with `200` status code if there was an earlier successful `POST` transaction for the resource and no successful `DELETE` transaction between the `POST` and the `GET`, and `404` otherwise. A suitable dataset should contain transaction sequences to reflect such behavioural patterns.

Service Virtualisation

Service Virtualisation (SV) [6,7] is an approach to build a semantic model of a service based on recorded traffic. For instance, SV will try to simulate the behaviour of an actual service by generating responses using models inferred from recorded transactions. This inference is usually done by means of supervised machine learning. The main application is to test systems that heavily rely on external *black-box* services in isolation. This corresponds to (automatically created) mock objects popular in application testing [77].

A suitable standardised dataset could be used to test SV. It would provide an oracle of actual service behaviour to be used in order to assess the quality of inferred behaviour.

Requirements

From the use cases above, we extract the following set of requirements to guide the construction of the datasets.

REQ1 **Large, yet manageable**: a good dataset should be of significant size to facilitate the use cases outlined and obtain results that are generalisable. However, this often conflicts with usability as experiments on large datasets are more difficult to setup and time-consuming. This can be addressed by providing several editions of different sizes.

REQ2 **Ease of use**: a good dataset should be presented in a format that is easy to process and preferably includes scripts to facilitate the processing and analysis of data, and a schema that (formally) describes the format used to represent data.

REQ3 **Reproducible, independent, and derived from principles**: a good dataset should not be produced ad-hoc, but extracted from real-world data or synthesised using a well-defined process unbiased by its use for one particular experiment.

REQ4 **Current**: a good dataset should reflect the state-of-the-art use of HTTP-based services. While this is difficult to assess in general, we argue that by extracting the dataset from the traffic of one of the most successful active Web services known for its excellent scalability and robustness, this can be achieved.

REQ5 **Precise and following standards**: a good dataset should contain transactions that comply with the syntax and semantics of HTTP, and the service(s) used.

REQ6 **Diverse**: a good dataset should support a wide set of HTTP features, such as various HTTP methods and status codes. In particular, it should go beyond the exclusive use of `POST` and `GET` requests which is a characteristic of older-generation Web applications designed for browser-based clients.

4.1.2 Related Work

This section provides an overview of the standard benchmarks and datasets containing HTTP message traces.

SPECweb2009 [234] is a standardised Web server benchmark produced by the Standard Performance Evaluation Corporation (SPEC). It is designed to evaluate a Web server ability to serve static and dynamic page requests. The benchmark comprises four distinct HTTP workloads to simulate three common types of consumer activities. The first workload is based on online banking, the second one is based on an e-commerce application, and the third one uses a scenario where support patches for computer applications are downloaded. All these workloads are developed by analysing log files of several popular Internet servers. The benchmark uses one or more client systems to generate HTTP workloads for the server according to the specified workload characteristics. Each client sends HTTP requests to the server and then validates the response received. However, the benchmark uses only HTTP 1.1 `GET` and `POST` requests and all of these requests are expected to result in responses with `200` status code. Server errors are especially communicated back to clients by generating error pages that return `200`.

TPC Benchmark W (TPC-W) [235] from the Transaction Processing Council is a notable open-source Web benchmark specifically targeted at measuring the performance of e-commerce systems. TPC-W simulates the principal transaction types of a retail store that sells products over the Internet. The workload of this benchmark specifies emulated browsers that generate Web interactions which represent typical browsing, searching, and ordering activities. It creates different `GET` and `POST` requests for specific documents and collects performance data. All these requests are expected to result in responses with `200` status code.

Rice University Bidding System (RUBiS) [236] is another open-source Web benchmark. It is based on an online auction site, modelled after eBay. This benchmark implements the core functionality of an auction site, in particular, selling, browsing, and bidding. The benchmark workload relies on a number of browser emulators that mimic the basic network interactions of real Web browsers. Read and write interactions are implemented using HTTP `GET` and `POST` requests.

DARPA dataset [237] by the MIT Lincoln Laboratory is a widely used evaluation dataset in intrusion detection research. There are three major releases (i.e., 1998, 1999, and 2000). Each release contains tcpdump files carrying a wide variety of simulated normal and malicious Web traffic in a military network setting. These network packet dumps can be used as a direct input to packet filtering engines such as Wireshark to extract sub-datasets that only contain HTTP request/response messages as relevant to our work. In particular, it is possible to use DARPA 2000 to obtain a dataset that holds 25,000 HTTP transactions (including HTTP 1.0 `GET` requests and responses with `200` status code).

CSIC 2010 [238] by the Information Security Institute of Spanish Research National Council is another publicly available dataset intended for the purpose of testing intrusion detection systems. It contains normal and anomalous HTTP 1.1 `POST` and `GET` requests targeting an e-commerce Web application. However, the dataset does not contain response data.

Table 4.1: Overview of HTTP Benchmarks and Datasets

Name	**HTTP Method**	**Response Code**	**Count**
TPC-W	GET, POST	200	13,500,000
RUBiS	GET, POST	200	4,030,000
DARPA 2000	GET	200	25,000
CSIC 2010	GET, POST	-	36,000
Opaque SV	GET, POST	200	1,825

There are few other HTTP datasets relating to SV studies. For example, the work of Versteeg et al. [12–15] describes a dataset with HTTP messages. The authors study **Opaque SV** using a relatively small HTTP dataset (consists of 1,825 request/response messages) collected through the Twitter REST API[1]. It includes both `POST` and `GET` requests returning `200`. However, none of the datasets employed in existing SV studies represent a wide range of HTTP features present in modern Web services or are publicly available for research purposes.

Table 4.1 summarises related benchmarks and datasets showing their request types, response codes, and transaction count. It is apparent that all these datasets only use a small fraction of the HTTP in terms of methods and status codes. They are somehow biased towards performance testing for older Web server where (static) pages are retrieved and in some cases created. They do not reflect the richness of modern Web APIs that take advantage of a much larger part of the HTTP.

Standard datasets have been widely used to support research in many other areas of Computer Science. For instance, the programming language and software engineering communities use datasets such as **DaCapo** [239] and **Qualitas Corpus/XCorpus** [240, 241] for benchmarking and empirical studies on source code. **Sourcerer** [242] is an infrastructure for large-scale collection and analysis of open-source code. The Sourcerer database is populated with more than 2,000 real-world open-source projects taken from Sourceforge, Apache, and Java.net.

The machine learning community uses several standardised datasets. This includes **UCI Machine Learning Repository** [243] by the Center for Machine Learning and Intelligent Systems at the University of California, Irvine. It provides a collection of benchmark datasets which can be used for the empirical analysis of learning algorithms. Another example is Google's **Kaggle**[2]. It offers ready-to-use public datasets for machine learning experiments that often include a description, usage examples and, in some cases, algorithms to solve the prediction problem associated with that particular dataset.

1. `https://developer.twitter.com` [accessed 02 Aug. 2020]
2. `https://kaggle.com/datasets/` [accessed 02 Aug. 2020]

4.2 GHTraffic Dataset

This section discusses the design of GHTraffic and the methods and tool used to construct it, then presents the results of some measurements on the dataset and provide basic instructions on how to obtain and use the GHTraffic. It also addresses versioning of the dataset and briefly reviews threats to validity.

4.2.1 Methodology

Input Data Selection

Over the past few years, GitHub[3] has emerged as the dominant platform for collaborative software engineering. It contains a rich set of features to manage code-related artefacts, including commits, pull requests, and issues.

There are several clients provided by GitHub that can be used to access its services, including the Web front-end and the desktop app. Many developers also use the standard git command line interface (CLI). In order to facilitate the development of a rich product ecosystem to access its services, GitHub also provides a REST API[4]. This allows third parties to integrate GitHub services into their products. Examples include mobile clients as well as IDE and build tool integrations (plugins).

The GitHub REST API provides a rich set of services to create, read, update, and delete resources related to the core GitHub functionality. It employs a large subset of HTTP features for this purpose and is, therefore, semantically richer than the datasets discussed on Section 4.1.2. Unfortunately, GitHub does not provide direct access to the recorded API interactions, so this information cannot be directly used for dataset construction.

An interesting use of the GitHub REST API for research purposes is GHTorrent [244]. This project uses the API to harvest information from repositories and stores that information by creating snapshots. These snapshots can then be downloaded and imported into a local MongoDB or MySQL database and queried. As of Apr. 2020, GHTorrent offers more than twenty terabytes of downloadable snapshots. These snapshots have already been used in empirical studies, examples include Gousios et al. [245] work on the pull-based software development model and Vasilescu et al. [246] work on the use of crowd-sourced knowledge in software development.

While GHTorrent provides a static view on the state of GitHub at certain points in time, we are interested in a more dynamic view of how interactions of clients with the repository have created this state. The basic idea is to reverse-engineer the respective API interactions (i.e., HTTP transactions) by cross referencing GHTorrent data with GitHub API functions. This has some obvious limitations. Firstly, we do not know whether all of these records were created via the REST API. They could have been created or altered using a different, or older version of the API, or via GitHub internal systems that bypass the API. We do not

3. `https://github.com/` [accessed 02 Aug. 2020]
4. `https://developer.github.com/v3/` [accessed 02 Aug. 2020]

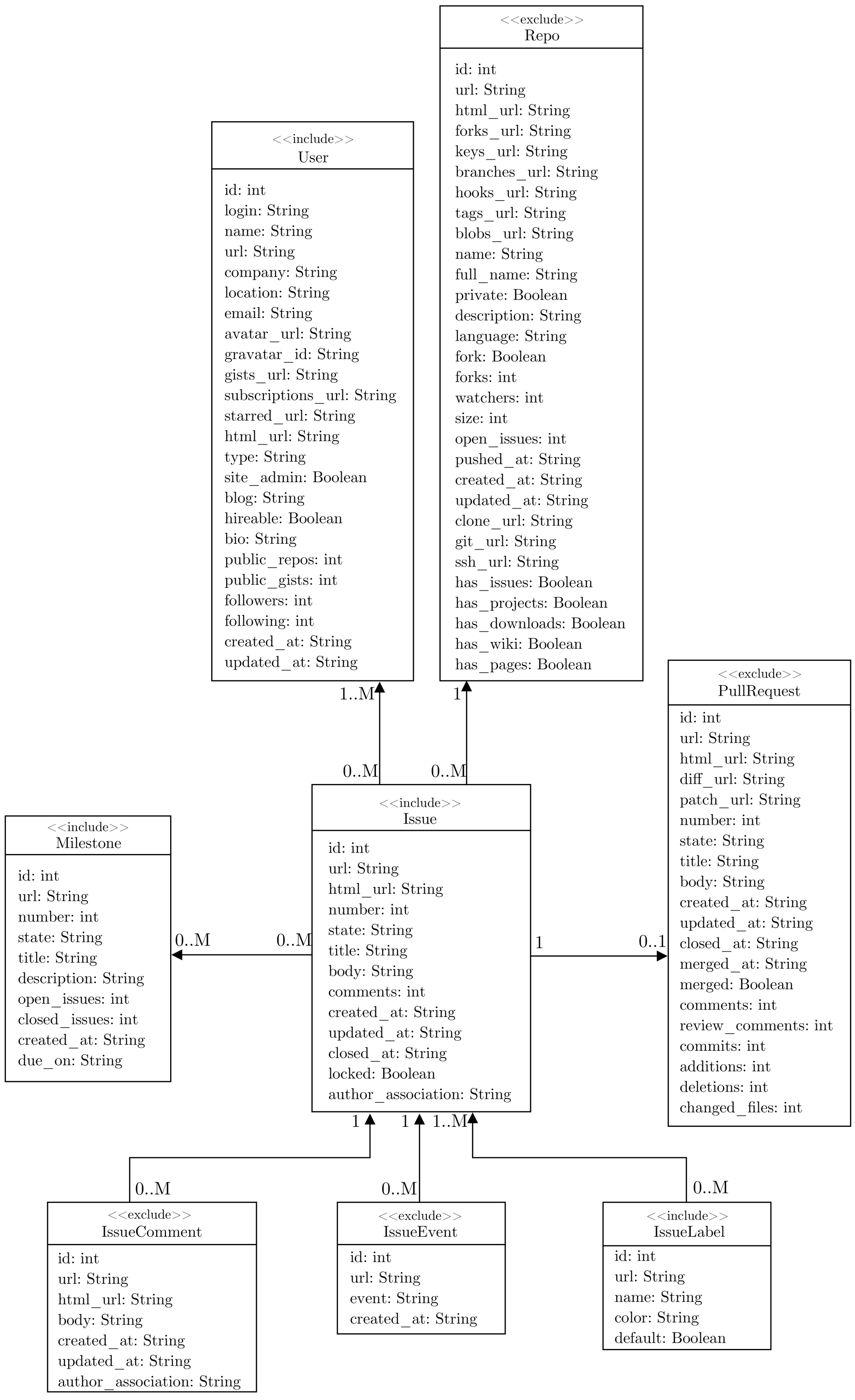


Figure 4.1: GitHub's Data Schema

consider this as a significant limitation. As far as the data inferred transactions are concerned, this will only have an impact on the `User-Agent` header. Secondly, the static data of the snapshots means that certain API interactions are not visible. This includes all read access (i.e., `GET` requests), requests that fail (e.g., a `DELETE` request resulting in a 404 response code will have no effect on the database), and shadowed requests (e.g., a successful PUT request followed by a successful DELETE request). To deal with those un-observable requests, we decided to augment the dataset with synthetic data.

Scope

GHTorrent collects a large amount of data on the terabyte scale. To make the data volume more manageable (REQ1), we decided to focus on a particular subset of GHTorrent, the Issue Tracking System. The issue tracking system itself references other entities[5] of the overall data model. The respective model (UML 2.0) is depicted in Figure 4.1. It is a refined version of the relational schema used in GHTorrent[6]. The stereotypes indicate which entities were included in the construction of the GHTraffic dataset.

Issues reference multiple other entities such as comments, milestones, labels, and users. While it is important to model some of them to facilitate our use cases, we decided to limit this to user, milestone, and label data. In particular, while issue comments look like integral parts of the issue tracking system, they are modelled in a relational style as one-to-many relationships via back-references. This means that comments reference the issue they are associated with, but issues do not directly reference comments.[7]

The design of the GHTraffic is driven by the use cases and the requirements derived from them. We wanted to construct a dataset that is large and diverse, and uses the features seen in modern Web services. This can be achieved by restricting the dataset to user, milestone, and label. Adding issue comments and other related data does increase the size further but does not add new features to the dataset. On the other hand, the increased size makes the dataset less manageable. As we will demonstrate in Section 4.2.2, the dataset is already sufficiently large.

Data represented in different entities is usually inlined in data returned by API calls. This means that if issue information is returned via the API, the JSON representation of the issue contains information about the issue and a summary of the users, labels, and milestones associated with it. Part of this information are URLs that can be used to query the full information for the respective entity. We treat these URLs as external, un-resolved references in the sense that our dataset does contain transactions to create, modify, delete or query these resources. Note that the GitHub API already uses references to external resources for which resolution cannot be guaranteed, an example for this is the `gravatar_id` attribute pointing to a picture of the user provided by the gravatar[8] service. The GHTraffic dataset is based on the 04 Aug. 2015 GHTorrent snapshot[9]. This is the largest

5. Entity is used in this paragraph in the context of entity-relationship data modelling [247], as opposed to the use of entity in the context of HTTP as defined by [133, Sect. 7]

6. `http://ghtorrent.org/relational.html` [accessed Apr. 03 2020]

7. The JSON representation of an issue contains a field `comments`, but this contains only the number of comments for the respective issue. This number can then be used to construct comments queries.

8. `https://pt.gravatar.com/` [accessed 02 Aug. 2020]

9. The respective dump is available from `http://ghtorrent-downloads.ewi.tudelft.nl/mongo-full/issues-dump.2015-08-04.tar.gz` [accessed 02 Aug. 2020]. The download size is

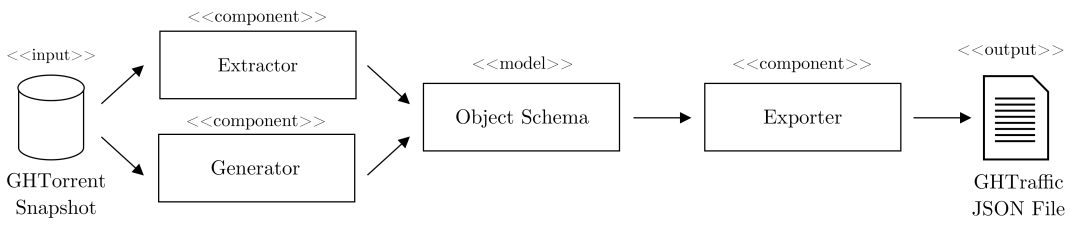


Figure 4.2: The Processing Pipeline

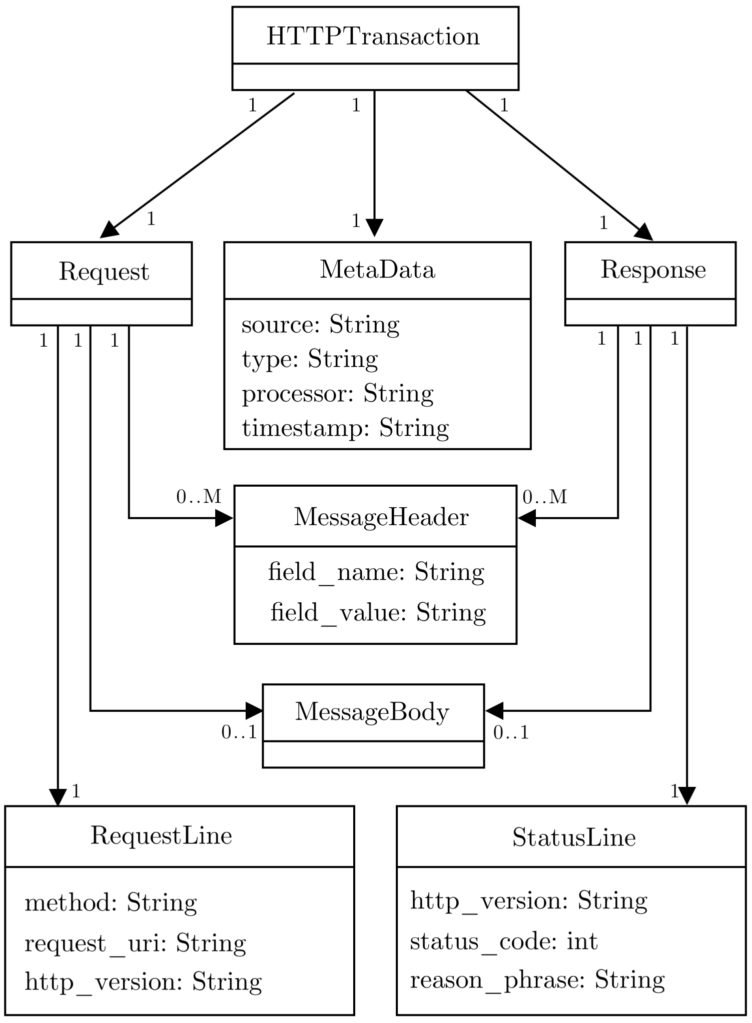


Figure 4.3: GHTraffic Schema

release of issues MongoDB database dumps as of Apr. 2020.

Processing Pipeline

An abstract overview of the infrastructure used to create the GHTraffic dataset is shown in Figure 4.2. GHTorrent snapshots are accessed by two core components, the **Extractor** and the **Generator**, the purpose of both is to create HTTP transactions. While the extractor builds transactions directly from snapshot data, the

6,128 MB which results in a 48.29 GB database with 21,077,018 records after restoring.

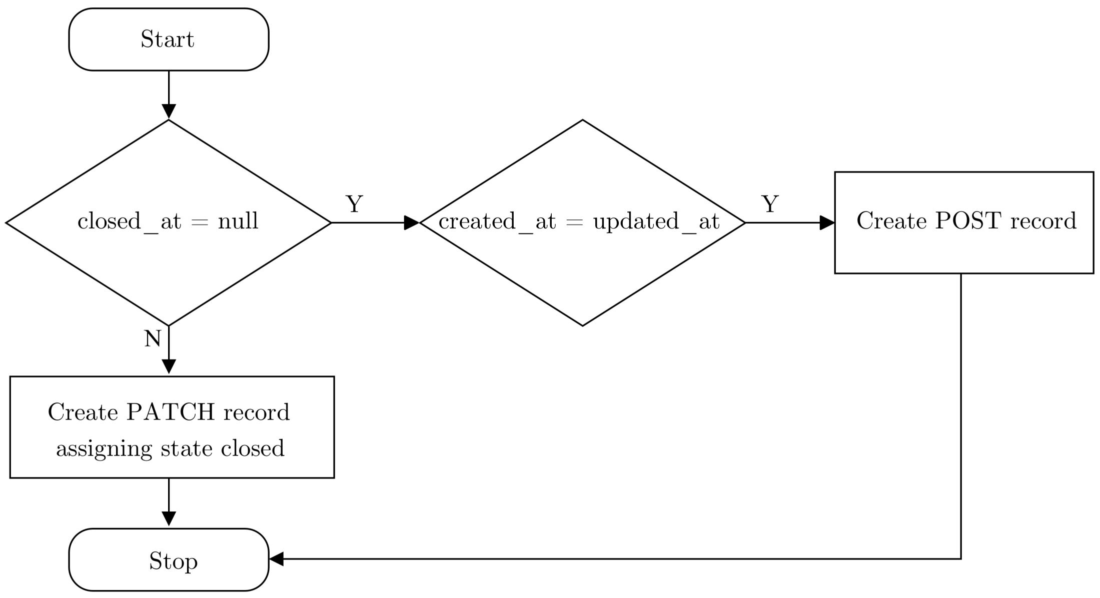


Figure 4.4: Extractor Algorithm to Process Records

generator infers synthetic transactions. In order to achieve this, it still needs access to the snapshot data. The reason for this is to get access to resource identifiers to be used in order to generate URLs. For instance, the generator creates queries, i.e., `GET` requests to query issues. If the respective resource names (i.e., issues ids) were generated randomly, almost all of those requests would fail with a `404`. This is not very realistic: in practise, most `GET` requests would try to access existing resources and succeed. In order to model this, the generator needs to access the GHTorrent snapshot.

The transactions generated by both the extractor and the generator instantiate a simple model depicted in Figure 4.3. This model is implemented in Java, i.e., each transaction has a transient in-memory representation as a Java object when the dataset is created. At the centre of this model are HTTP transactions, basically request/response pairs.

At the end of the pipeline is an **Exporter** component that processes the transactions represented as Java objects and persists them by encoding/serialising using JSON. The structure of the JSON files produced is defined by JSON schemas [248]. Note that there are separate schemas for each HTTP method.

The implementation of the components discussed have some abstractions to facilitate alternative extraction, inference, and data representations. The overall processing model is lazy and stream-like, i.e., only a small number of records remain in memory at any time in order to make processing scalable.

Processing can be customised by employing data filters (predicates). Only records matching certain criteria are processed. The main use case for this is filtering by URL and here in particular by the project. This allows us to build different editions of the dataset with certain target sizes. While there is a potentially easier way of doing this by just restricting the number of records being processed and included, using filters has an inherent advantage. GitHub data is fragmented by project and by filtering it accordingly, we are able to extract transactions that manipulate the same resources, reflecting the same issue being created and updated. This way, we can obtain *coherent* subsets of the overall dataset that still reflect the service semantics derived from issue tracking workflows.

Extraction

Each data record has `created_at`, `updated_at`, and `closed_at` timestamps which enable us to trace lifecycle events of the issue. Using this data, the GHTraffic scripts produce transaction records. An overview of the process is shown in Figure 4.4. For instance, a `POST` transaction record is created in order to represent the creation of an issue at the time stipulated in the `created_at` attribute. The value is converted to the standard date format used by the HTTP [133, Sect. 3.3] and set as the value of `Date` header.

Both the request and the response used headers as specified in the GitHub API documentation. This is a mix of standard HTTP headers and API-specific headers with names starting with "`x-`". In case the header values cannot be inferred from snapshot data, we use synthetic data. For example, we generate random token strings and use them as values for the `Authorization` headers. There is also a list of user agent strings to assign randomly as the value of the `User-Agent` headers. Further, for the request body, the script extracts the values of the `title, body, assignee, milestone, labels` parameters from the snapshot record and encodes it in JSON as stipulated in the API. The response creation process is analogous. Most of the values for the JSON-encoded response body are filled out with data directly taken from the snapshot. Besides, the GHTraffic script assigns the `created_at` value to the `updated_at` field. Further, it explicitly specifies `closed_at:null`, `closed_by:null`, `state:open`, and `locked:false`.

Every time a GitHub user updates an existing issue, its `updated_at` timestamp gets renewed with the date and time of the update. Marking an issue as closed is a special type of update, as an issue is not deleted, but its status is changed to `close`. In order to extract `PATCH` transactions used to close issues, the script queries issues whose `closed_at` value is not null and only those are processed by the extractor. The request and response messages are formed by following the GitHub Issues API documentation. Particularly, the `closed_at` value is converted to the standard HTTP date format and set as the value of `Date` response header and the `closed_at` value is also assigned to the `updated_at` field to set `closed_at` and `updated_at` columns' values same.

Besides, an update might be changing the title of an issue, changing its description, specifying users to assign the issue to, etc. However, we could not extract exactly what input data was used for editing an issue, therefore, we did not generate such transaction types.

Synthesising Queries

Only successful `POST` and `PATCH` transactions can be constructed by reverse-engineering the GHTorrent snapshot. In order to generate additional transaction records such as queries and delete requests, we had to resort to using synthetic data. The aim of generating synthetic data is to mimic transactions concerning several other

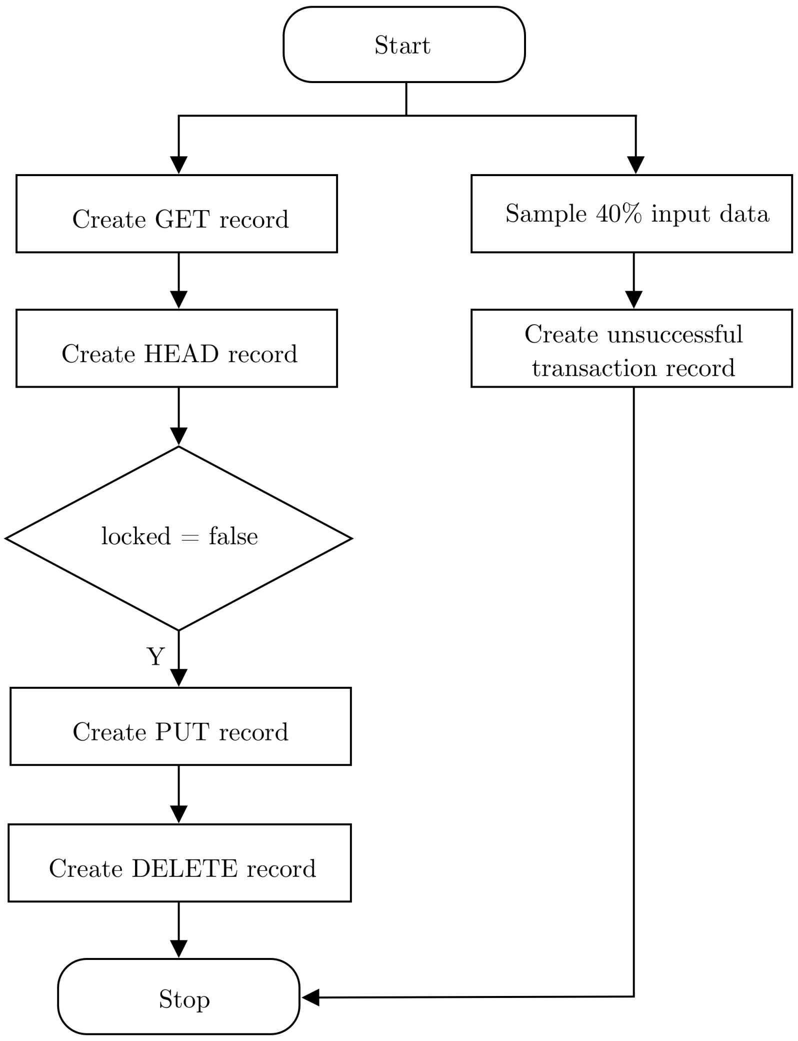


Figure 4.5: Algorithm to Generate Synthetic Data

HTTP request methods that are covered by the API and requests that fail, which is indicated by an error HTTP status code.

Figure 4.5 shows the process of synthetic data generation. The script generated `GET` and `HEAD` transactions for each record in the snapshot. The process is analogous to the process described in Section 4.2.1. However, the `Date` response header is set to the system date and time at which the request is formed. Similarly, `PUT` transactions for locking an issue are generated for records with `locked` value set to `false` and followed by `DELETE` transactions for unlocking the respective issues using the format described in the GitHub API.

Furthermore, the script produces unsuccessful transactions for all those HTTP methods by specifying requests:

- without authorisation token
- with badly formatted URL
- without request body
- with invalidly formatted JSON body

Table 4.2: GHTraffic Transactions Per HTTP Method

Method	**S**	**M**	**L**
POST	7,193	32,130	508,664
PATCH	4,286	30,807	468,080
GET	3,117	22,692	344,474
HEAD	1,796	15,130	245,127
PUT	3,662	15,945	238,115
DELETE	2,341	16,457	246,180

All the respective transactions have an error status code as defined in the API and are generated from a sample of 40% random records from the snapshot.[10] More specifically, a message explaining the error is added to the response body as specified in the GitHub API. For this purpose, we performed experiments on a *toy* project repository (i.e., a repository created to obtain real-time experience with the GitHub API) for creating synthetic data that closely resemble real-world representation as we found that certain aspects of the GitHub Issue API are undocumented. Additionally, we generated a small number of `GET` requests that returned `500` status code, in order to represent system failures.

Data Representation and Meta-Data

The target format of the GHTraffic dataset is described by the UML class diagram as shown in Figure 4.3. `HTTPTransaction` is the base element of the model. A transaction contains a single `Request` and `Response`. Each message could have any specific number of `MessageHeaders`. Additionally, a `MessageBody` is used to represent data associated with a request/response. `MetaData` is used to provide some additional information about the transaction record. The `source` attribute is set to `GHTorrent`, specifying the source of information. The `type` attribute is set to either `real-world` or `synthetic` depending on whether the data was directly derived from a GHTorrent record or synthesised as described above. The `processor` is the name of the script used to generate the record, i.e., this is the fully qualified name of a Java class. Finally, the `timestamp` field holds date and time when the record was created.

The actual JSON format of the dataset is defined by a set of JSON schemas for each transaction type (i.e., for each HTTP method). These schemes are presented in Appendix A.1, or can be found in the repository, in the *schemas* folder. The schemas comply with the JSON Schema Draft 4 specification [249]. Appendix A.2 presents sample GHTraffic records (over a single resource) for each request type.

4.2.2 Metrics

The GHTraffic dataset comprises three different editions: Small (S), Medium (M), and Large (L). The S dataset includes 22,395 HTTP transaction records cre-

10. The generator component needs to use at least 40% of GHTorrent snapshot records in order to extract an adequate amount of unsuccessful transactions on particular projects.

Table 4.3: GHTraffic Transactions Per HTTP Response Code

Response Code	**S**	**M**	**L**
200	4,649	22,163	391,903
201	1,796	8,808	150,662
204	3,588	5,756	82,554
400	2,717	13,807	196,474
401	547	19,302	291,831
404	5,909	43,658	646,346
422	1,868	12,626	196,678
500	1,321	7,041	94,192

Table 4.4: GHTraffic Transactions Per Record Type

Type	**S**	**M**	**L**
Real-world	2,853	13,355	241,241
Synthetic	19,542	119,806	1,809,399

ated from the **google/guava** [250] repository and takes up to 49.9 MB of disk space. Guava is a popular Java library containing utilities and data structures. It is a medium-sized large active project, and sourcing an edition from a single project has the advantage of creating a coherent dataset. The M dataset of size 345.2 MB includes 133,161 records from the **npm/npm** [251] repository. It is the popular de-facto standard package manager for JavaScript. The L dataset contains 3.73 GB of data with 2,050,640 records that were created by selecting eight repositories containing large and active projects on GitHub as of 2015, including **rails/rails** [252], **docker/docker** [253], **rust-lang/rust** [254], **angular/angular.js** [255], **twbs/bootstrap** [256], **kubernetes/kubernetes** [257], **Homebrew/homebrew** [258], and **symfony/symfony** [259].

Tables 4.2–4.4 present several selective metrics about the status of these three datasets.

4.2.3 Accessing and Using GHTraffic

The different editions of the GHTraffic dataset can be downloaded by using the following URLs[11]:

- `https://zenodo.org/record/1034573/files/ghtraffic-S-1.0.0.zip`
- `https://zenodo.org/record/1034573/files/ghtraffic-M-1.0.0.zip`
- `https://zenodo.org/record/1034573/files/ghtraffic-L-1.0.0.zip`

11. The dataset is published on Zenodo [260]. It is a data repository platform hosted at the European Organization for Nuclear Research Data Center, which was specifically designed to provide long-term preservation of all forms of research output.

Table 4.5: GHTraffic 2.0.0 Transactions Per HTTP Method

Method	**S**	**L**
POST	5,805	456,941
PATCH	7,577	454,475
GET	5,325	502,183
HEAD	3,491	279,395
PUT	5,285	369,834
DELETE	4,732	341,449

Table 4.6: GHTraffic 2.0.0 Transactions Per HTTP Response Code

Response Code	**S**	**L**
200	5,839	496,814
201	1,793	150,662
204	3,104	221,038
400	3,499	143,733
401	5,227	143,374
404	10,378	1,032,327
422	1,859	148,692
500	516	67,637

We also provide access to the scripts used to generate GHTraffic, including a VirtualBox image with a pre-configured setup. Note that due to the use of random data generation these scripts will produce slightly different datasets at each execution. Using the scripts, users can modify the configuration properties in *config.properties* file in order to create a customised version of the GHTraffic dataset for their own use. The *readme.md* file included in the distribution provides further information on how to build the code and run the scripts. GHTraffic scripts can be accessed by cloning the repository (`https://bitbucket.org/tbhagya/ghtraffic.git`) or by downloading the pre-configured VirtualBox image from `https://zenodo.org/record/1034573/files/ghtraffic-artifact-1.0.0.zip`.

4.2.4 GHTraffic Versioning

The original GHTraffic dataset is released as version 1.0.0. A new version of the dataset (version 2.0.0) is also being produced incorporating minor changes to the synthetic data generation process and adding another subset of unsuccessful transactions. The intention is to model enhanced transaction sequences that reflect more complex behavioural patterns.

The entire data generation process is quite similar to the original design as described in Section 4.2.1, except when transactions are synthesised, the original scripts fill in the `Date` response header with the current date and time and the

Table 4.7: GHTraffic 2.0.0 Transactions Per Record Type

Type	**S**	**L**
Real-world	2,845	241,241
Synthetic	29,370	2,163,036

transactions are composed accordingly, whereas the new scripts use a random date after a resource is successfully posted to make up the request and response. In particular, it uses `created_at` timestamps of resources to assign successive dates. The scripts also generate another form of failed transactions by stipulating requests prior to the successful creation of resources (emulating requests for unavailable resources). The `created_at` timestamps are again used here to generate preceding random dates to be set as the value of `Date` header. These result in a far more dynamic series of transactions to named resources.

This latest version consists of two different editions: Small (S) and Large (L) where the records were created by selecting the same repositories as the original Small and Large datasets (it does not, however, include a medium edition, since a variety of extensive datasets have already been achieved). The newest S dataset contains 32,215 records (64.5 MB) of data collected from **google/guava** repository. The L dataset of size 4.54 GB contains 2,404,277 records from eight repositories (i.e., **rails/rails**, **docker/docker**, **rust-lang/rust**, **angular/angular.js**, **twbs/bootstrap**, **kubernetes/kubernetes**, **Homebrew/homebrew**, and **symfony/symfony**). Tables 4.5–4.7 display some metrics that characterise the dataset. The datasets are available for download using the following URLs:

- `https://zenodo.org/record/4007589/files/ghtraffic-S-2.0.0.zip`
- `https://zenodo.org/record/4007589/files/ghtraffic-L-2.0.0.zip`

The project scripts are also available from the repository (`https://bitbucket.org/tbhagya/ghtraffic-version-2.0.0`) and can easily be customised and extended to generate dataset variants. The small edition of GHTraffic 2.0.0 is selected for use in this study.

4.2.5 Threats to Validity

As seen in Tables 4.4 and 4.7, the size of synthetic data drastically exceeds the size of data extracted from the snapshot. This leaves the possibility that the GHTraffic does not reflect realistic workloads. To mitigate this threat, we ensured that the request/response formats for the transaction types that were synthesised had been sampled and validated using the toy GitHub repository. That is, we used a few samples of each transaction type, sent similar requests to the toy repository, and checked whether the sample response formats were equivalent to what the actual GitHub API returned. This practise guaranteed that the transactions were syntactically and semantically correct, even though they did not actually occur. In addition to this, the representation of the transactions has information about whether they are synthetic or not, and users of the GHTraffic can use this to completely remove or reduce the ratio of synthetic data by applying filters.

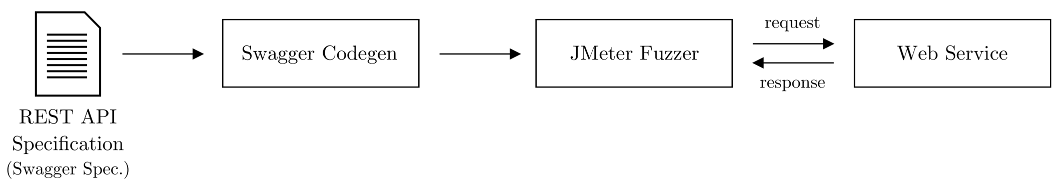


Figure 4.6: The Process of Generating Network Traffic

We acknowledge that the GHTraffic was generated from a 2-year-old snapshot of GHTorrent. As noted earlier, this design decision was made to produce a dataset large enough to facilitate the use cases described, but still manageable with typical resources available to researchers and practitioners. We also provide access to the scripts used to generate the GHTraffic, and users can utilise these scripts in order to generate customised versions from newer instances of GHTorrent if needed.

4.3 Twitter, Google Tasks, and Slack Datasets

This section discusses the method used to create Twitter, Google Tasks, and Slack datasets, followed by some selective metrics that characterise the datasets, and includes basic instructions on obtaining and using the data.

4.3.1 Methodology

Subject Services

In addition to GitHub, three of the popular and active Web services currently in use are Twitter[12], Google Tasks[13], and Slack[14]. Twitter is a social media platform that allows users to send and receive short messages called tweets. Google Tasks is a service offered by Google Inc. to manage tasks and task lists. Slack is a channel-based messaging platform. Each of these platforms has a wide variety of different features and also offers a REST API which enables application developers to flexibly access the available services through simple HTTP requests. In addition, each REST API provides a rich set of services to create, read, update, and delete resources related to the core functionalities using a set of HTTP features, including various HTTP methods and status codes. Thereby, datasets constructed directly from capturing API interactions represent the richness of modern Web APIs and are semantically richer than the ones mentioned in Section 4.1.2.

We are interested in generating three separate HTTP datasets by creating random traffic for the services offered by Twitter, Google Tasks, and Slack. The basic idea is to simulate a variety of service interactions by users via applications. The creation of the datasets is driven by the use cases and the requirements set out in Section 4.1.1. In order to form transactions, different CRUD operations are created to service-specific resources and corresponding responses are recorded. The

12. `https://twitter.com` [accessed 02 Aug. 2020]
13. `https://play.google.com/store/apps/details?id=com.google.android.apps.tasks` [accessed 02 Aug. 2020]
14. `https://slack.com` [accessed 02 Aug. 2020]

resources most intuitively used by the users are chosen to produce random API calls, namely tweets[15] (Twitter), lists[16] (Google Tasks), and messages[17] (Slack). The following is a list of operations chosen from each category (which are frequently used to interact with specified resources).

- Tweets
  - POST /statuses/update.json

    Updates the authenticated user's current status
  - GET /statuses/show/`:id`

    Returns the status specified by the `id` parameter
  - POST /statuses/destroy/`id`.json

    Deletes the status specified by the `id`
- Lists
  - POST /lists

    Creates a new task list and adds it to the authenticated user's task lists
  - GET /lists/`id`

    Returns the authenticated user's task list specified by the `id`
  - PATCH /lists/`id`

    Updates the authenticated user's task list specified by the `id`
  - DELETE /lists/`id`

    Deletes the authenticated user's task list specified by the `id`
- Messages
  - POST /chat.postMessage

    Posts a message to a channel
  - POST /chat.update/`:id`

    Updates a message specified by the `id` parameter in a channel
  - POST /chat.delete/`:id`

    Deletes a message specified by the `id` parameter from a channel

15. `https://developer.twitter.com/docs/tweets/post-and-engage/api-reference/post-statuses-update` [accessed 02 Aug. 2020]
16. `https://developers.google.com/tasks` [accessed 02 Aug. 2020]
17. `https://api.slack.com/methods` [accessed 02 Aug. 2020]

Table 4.8: Transactions Per HTTP Method on Twitter, Google Tasks, and Slack Datasets

Method	Twitter	Google Tasks	Slack
POST	20,445	1,124	17,422
PATCH	-	1,082	-
GET	5,608	1,367	-
DELETE	-	1,129	-

Table 4.9: Transactions Per HTTP Response Code on Twitter, Google Tasks, and Slack Datasets

Response Code	Twitter	Google Tasks	Slack
200	1,515	2,607	17,422
204	-	606	-
404	24,538	1,478	-
503	-	11	-

Network Traffic Generation

An overview of the approach used to generate traffic is shown in Figure 4.6. The actual input generation uses fuzzing techniques. In particular, Apache JMeter [82] is used as it has the functionality to fuzz RESTful services (randomly generate various types of API calls by providing different inputs) and recording interactions in a suitable textual format for further processing. The fuzzing is guided by a light-weight semantic service model provided as Swagger[18] spec. Swagger (recently renamed as OpenAPI) has emerged as the standard approach for specifying and documenting HTTP APIs in a way that is both human- and machine-readable. Swagger defined APIs can be directly used with the Swagger Codegen tool[19] to create API clients and server stubs (which are configured to get input data that the API expects to pass in from CSV files). Since the services used to construct datasets all possess Swagger APIs[20], we are able to use Swagger Codegen to auto-generate JMeter scripts. This approach allows us to automate much of the data generation process.

The JMeter scripts produced require input data (parameter values) from CSV files in order to render API calls. As we intended to construct large and diverse datasets (to facilitate our use cases), these files need to be filled out with a variety of values to support different types of calls. However, if resource identifiers are provided arbitrarily almost all requests would fail (with HTTP `404`). This is not realistic, since most requests would attempt to access existing resources. Therefore, first, we specify parameter values in order to randomly create resources, then use those resource identifiers to randomly read, update, and delete resources. We are

18. `https://github.com/OAI/OpenAPI-Specification` [accessed 02 Aug. 2020]
19. `https://github.com/swagger-api/swagger-codegen` [accessed 02 Aug. 2020]
20. The swagger spec for Twitter, Google Tasks, and Slack can be accessed from `https://api.apis.guru/v2/specs/twitter.com/current/2.1/openapi.json`, `https://api.apis.guru/v2/specs/googleapis.com/tasks/v1/openapi.json`, and `https://api.apis.guru/v2/specs/slack.com/1.5.0/openapi.json`, respectively [accessed 02 Aug. 2020]

able to achieve more realistic behavioural patterns with this strategy. As seen in Section 4.3.2 below, the resultant datasets explicitly support a wide range of HTTP features and are sufficiently large in size.

4.3.2 Metrics

The Twitter dataset contains 26,053 HTTP transaction records and the total size of the dataset is 58 MB. The Slack dataset in size 33.9 MB has 17,422 records and the Google Tasks dataset contains 4,702 transactions in 8.8 MB. Further details of these three datasets are summarised in Tables 4.8 and 4.9 showing the transaction counts based on the request type as well as the response code. The datasets clearly demonstrate the standard datasets criteria set out in Section 4.1.1 for facilitating studies related to the motivating use cases.

4.3.3 Accessing and Using Datasets

The following URLs can be used to download the datasets:

- `https://zenodo.org/record/4007570/files/twitter-1.0.0.zip`
- `https://zenodo.org/record/4007570/files/googletasks-1.0.0.zip`
- `https://zenodo.org/record/4007570/files/slack-1.0.0.zip`

Each dataset is available in XML format. Appendix A.3, A.4, and A.5 present sample transaction records over a resource in each dataset for each operation type. The JMeter scripts used for datasets construction are accessible from the repository (`https://bitbucket.org/tbhagya/http-traffic-datasets`), giving the flexibility to extend the datasets.

4.3.4 Threats to Validity

As seen in Tables 4.8 and 4.9, the Twitter, Google Tasks, and Slack datasets do not have a broad variety of HTTP features (such as different HTTP methods and/or status codes). This leaves the possibility that the datasets do not represent the features seen in modern Web APIs. The subject HTTP-based services, however, are being extensively used at present and these datasets were constructed directly from capturing API interactions (different CRUD operations were created to service-specific resources and corresponding responses were recorded). On the one hand, it could be observed that these APIs do not allow certain actions on resources or do not use HTTP features to provide semantic meaning for the intention of the action being taken and the response being received. For example, the Twitter API uses the `POST` HTTP method for both create and delete operations and is not allowed to modify a single Tweet (no explicit usage of the `DELETE` and `PUT/PATCH` request types defined) and that the Slack API does post, update, and delete operations through the `POST` HTTP method and is not allowed to retrieve a particular message from a channel (no explicit use of any other request type except for HTTP `POST`) as well as it always returns the HTTP `200` code even when it has unexpected behaviour. This means that these APIs literally only use a small portion of the HTTP in terms of methods and status codes. However, in the data generation process, we were able to issue a number of common CRUD operations, simulating service interactions that exist in realistic service scenarios as much as possible (it was, therefore, not necessary to create additional syntactic transactions). Thereby, each constructed dataset provides an oracle of actual service behaviour and of substantial size to facilitate experiments with realistic network traffic.

4.4 Summary

In this chapter, we outlined the network traffic datasets to be used in the experiments. First, we discussed use cases, and requirements of such datasets, followed by an overview of related work. Then, we presented the GHTraffic, a dataset comprising HTTP transactions extracted from GitHub data and augmented with synthetic transaction data. After that, we introduced three other HTTP traffic datasets that were obtained from fuzzing the services offered by Twitter, Google Tasks, and Slack. All four datasets are carefully sourced and/or built from successful active Web services reflecting realistic HTTP workloads. The datasets are also of significant size and relatively easy to use and customise. These datasets could be considered as the standard datasets for reproducible research on many aspects of service-oriented computing. We hope that the datasets will find uses in many areas of research. These are chosen for use in this study in order to facilitate the reproducibility of research results.

Chapter 5

Experimental Methodology

This chapter presents the experimental setup and describes the methodology used to achieve the research aims of this thesis. Section 5.1 provides a summary of the study. Sections 5.2–5.5 describe the key phases of the experiment. Section 5.6 sets out instructions to reproduce the experiment results.

5.1 Introduction

Modern application development makes extensive use of lightweight HTTP services to implement certain functionalities that applications require (more on this in Section 2.1). One of the significant engineering challenges is how to adequately test such applications with the services on which they depend. The concept of Service Virtualisation (SV) is gaining popularity in addressing this problem. It is a practise to mimic the behaviours of real dependent services by synthesising the anticipated responses using semantic models inferred from recorded traffic (details on SV are in Section 2.2.2).

SV often applies Artificial Intelligence (AI) techniques when it comes to inferring service semantics. Among the different categories of AI, there is growing demand for Symbolic Machine Learning (SML) algorithms due to the provenance of their results (see Section 2.3.1). SML techniques represent learned knowledge in simple logical rules that are generally easier for humans to analyse and understand. Typical symbolic learning approaches include Attribute-Based Learning and Inductive Logic Programming/Description Logic (DL) Learning.

In reality, however, fully automated SV techniques are unlikely to provide a sufficient level of accuracy to mock complex services, i.e., response properties could exist that are obviously much more complex to model using automated methods (described in Section 1.1). Therefore, we recommend using SV to infer some attributes of the HTTP service responses. Besides, we understand that engineers often want to inspect the SV results and make changes. This requires the presentation of results to engineers in an appropriate format that is easier to understand and customise. We assume that if SV is based on inference rules, engineers will have the potential to fine-tune and adjust the rules to create mocks suitable for testing.

Accordingly, the primary aim of this research is to understand the potential of SML techniques for generating HTTP services' mock skeletons directly from traffic records. We examine three attribute-based learning algorithms, namely the

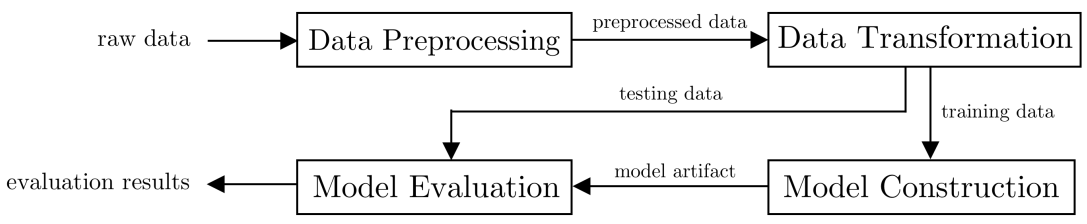


Figure 5.1: The Machine Learning Framework

C4.5 decision tree algorithm and the RIPPER and PART rule learners. For these algorithms, we use the WEKA implementations (as WEKA is commonly used in the community). The algorithm that we consider in DL learning is the OCEL class expression learning algorithm implemented in DL-Learner. These algorithms have been used extensively and have proven their ability to handle both numerical and nominal data properly. See Section 2.3.2 and 2.3.3 for details on the algorithms. All the experiments will be performed employing network traffic datasets extracted from a few different successful, large-scale HTTP services (i.e., GitHub, Twitter, Google, and Slack) described in Chapter 4. The experimental design focuses on the production of reliable results, fair comparisons, and accurate evaluations of the achievements of the algorithms chosen. This also focuses on the production of reproducible results (addressing the secondary research aim). Section 1.3 provides a comprehensive explanation of these two research aims. For the experiments, we use 2.2 GHz Quad-Core Intel Core i7 processor, 16 GB memory and macOS (Catalina) operating system with a Java Runtime Environment (JRE) 8 (64-bit) Java Virtual Machine (JVM). The default heap size of the JVM is 8 GB.

The general overview of the experimental methodology (i.e., machine learning workflow) can be found in the Figure 5.1. There are four stages: first the **Data Preprocessing** step corresponding to the cleaning of the raw datasets, then the **Data Transformation** step converts preprocessed data into data formats appropriate for learning algorithms, the **Model Construction** step builds model artefacts from training data, and finally the **Model Evaluation** step assesses the predictive ability of the models generated. The following sections will discuss the construction of these phases in more detail.

5.2 Data Preprocessing

The experimental datasets do not contain any error interaction records nor duplicates. So all data records can be taken into account in this study.

A basic data cleaning process is performed to remove the contents of data records that impede raw datasets from being processed and/or parsed correctly. In particular, it identifies and deletes all data properties with long text. Examples of this will be `title` and `body` in the GHTraffic dataset. Texts found in `title` and `body` often contain extremely long strings with whitespace, line breaks, and blank lines, that need to be excluded from the original before any processing takes place. These modifications have little or no effect on learning (i.e., datasets are still meaningful without them). The cleansed data will be stored separately from the raw datasets for the next step to load up and process.

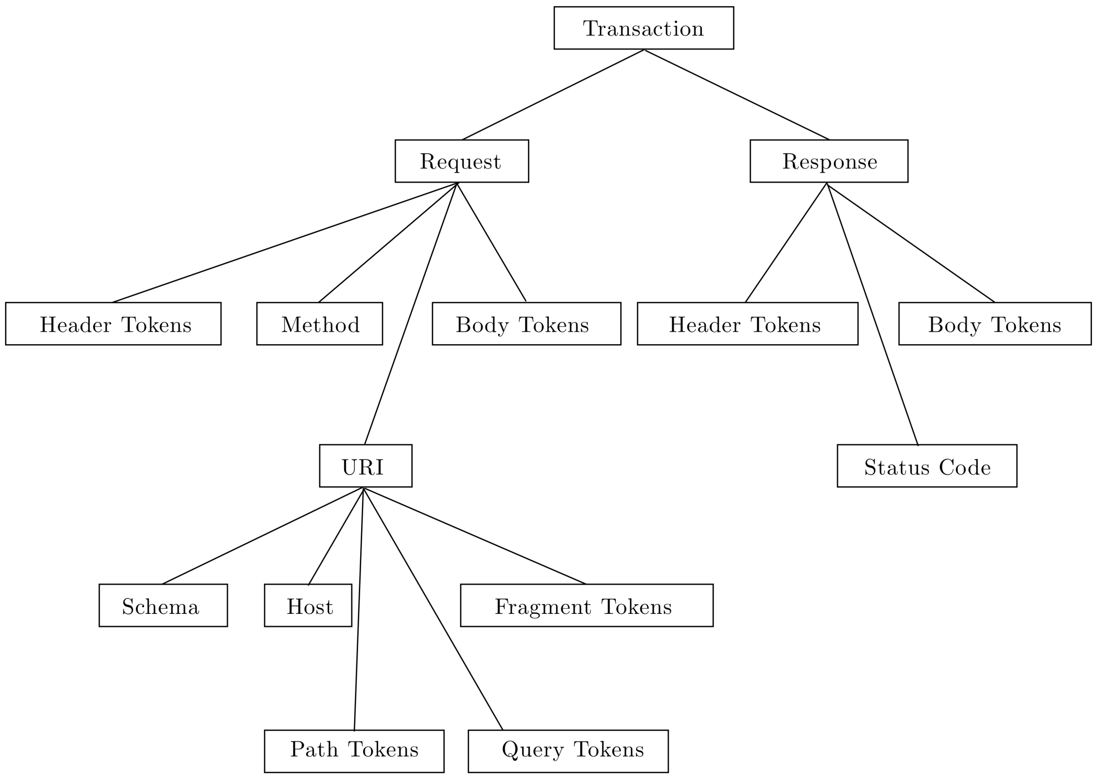


Figure 5.2: The Feature Tree

5.3 Data Transformation

The attribute-based learning algorithms expect input data described in the propositional attribute-value format, whereas algorithms based on DL learning require data with rich domain knowledge expressed in description logics. We, therefore, pursue two different directions for the transformation of preprocessed data. Each of the approaches focus on capturing as many important information as possible contained in HTTP datasets (in particular data about request/response messages and service state history) with sound generalisation.

5.3.1 Attribute-Based Learning Approach

As described in Section 2.3.2, attribute-based learning takes as input a set of labelled instances (i.e., examples) where each instance is described in terms of values for a fixed set of attributes (an attribute vector). The goal of learning is to induce the mapping from attribute vectors to target class labels.

Data transformation in attribute-based learning is a two-step process: **Feature Extraction** and **Data Preparation**. The feature extraction step enables the collection of attributes from HTTP datasets. Once the features are extracted, the attributes are filtered and processed in the data preparation step (using the WEKA API[1] written in Java) to be properly accessible by learning algorithms.

1. `https://weka.sourceforge.io/doc.stable` [accessed 02 Aug. 2020]

https://www.slack.com/api/chat.postmessage?q=foo#param=bar

$\downarrow$

RequestUriSchema - https
RequestUriHost - www.slack.com
RequestUriPathToken1 - api
RequestUriPathToken2 - chat.postmessage
RequestUriPathToken3 - not-exist
RequestUriPathToken4 - not-exist
RequestUriPathToken5 - not-exist
RequestUriPathToken6 - not-exist
RequestUriQueryToken1 - q=foo
RequestUriQueryToken2 - not-exist
RequestUriQueryToken3 - not-exist
RequestUriQueryToken4 - not-exist
RequestUriFragmentToken1 - param=bar
RequestUriFragmentToken2 - not-exist

Figure 5.3: Attribute and Value Derivation From a Sample URI

Feature Extraction

Through interpreting the structural properties of HTTP messages, features are extracted to be used as attributes by the learning algorithms. Figure 5.2 presents a high-level overview of the features used to capture the structure of HTTP messages (features are organised in a simple hierarchy, the *feature tree*). The following is a description of the process for extracting the features.

The first group of attributes is the general characteristics that each HTTP transaction has, in particular, the `RequestMethod` and the `ResponseStatusCode`. Several other attributes are derived from the request URIs. The URI has a canonical structure which consists of the schema, host, path, query, and fragment. Elements such as the schema and the host can be explicitly used as attributes (i.e., the `RequestUriSchema` and the `RequestUriHost`). The path segment can be tokenised using standard delimiters (i.e., `/`), and each token can be used as an attribute named `RequestUriPathToken` suffixed by the position index of the token in the path (use up to six indexes by studying general API implementations). If no value is presented for the respective token, the value of the attribute is set to `not-exist`, otherwise, the value of the token will be used. The query string parameters (key-value pairs) can be decoded and separated by the `&` delimiter to be used as attributes named `RequestUriQueryToken` suffixed by the position index (use up to four indexes). We can also perform fragment tokenisation to obtain a separate set of attributes (i.e., the `RequestUriFragmentToken1` and the `RequestUriFragmentToken2`). Figure 5.3 illustrates this process using an example.

It is also possible to derive attributes based on the characteristics of request and response bodies. The `HasRequestPayload` and the `HasValidRequestPayload` are boolean attributes that are introduced to indicate whether a request has a message body and whether the message body is encoded correctly according to its content type. Although HTTP services can use arbitrary content types, we only support JSON as it is by far the most widely used format for data exchange via

HTTP services. Basic tokenisation is also done by separately parsing the request and the response bodies extracting two different sets of all possible service-specific keys based the set of HTTP transactions (usually the body is a JSON object). These keys can lead to a wide collection of attributes prefixed by `RequestBody` or `ResponseBody` with the name of the key. We also perform deep tokenisation if JSON objects contain object arrays. If the respective key is presented in the message body, the attribute value is set to value of the key, otherwise, the value will be set to `not-exist`.

Furthermore, we extract attributes from the request and the response headers that include both standard HTTP headers as well as API-specific headers (i.e., usually identifiable by header names starting with x-). The process is analogous to the approach previously described and applies the same logic with the stipulation to name attributes as `RequestHeader` or `ResponseHeader` along with the name of the header (key) and return `not-exist` if there is no header in the transaction. Moreover, an inferred `HasAuthorisationToken` attribute will be extracted when encountering information about the request authentication. The inference is based on the presence of an authentication token with either the HTTP `Authorization` request header or the URI query parameter (e.g., Slack sends an authentication token through a query string parameter).

Finally, we generate attributes to represent the features (i.e., state) of the predecessors on each transaction (a transaction that is just before and a set of transactions that precedes a named resource).[2] The `HasImmediatePreviousTransaction` boolean attribute is added to indicate whether or not another transaction happened immediately prior to a particular transaction interacting with the same resource. The attribute `ImmediatelyPreviousRequestMethod` is used to specify the CRUD operation that transaction performed. Additional boolean attributes are also introduced in relation to this, based on the presence of certain naming patterns in the URI tokens (e.g., Slack uses postMessage, update, delete, Twitter uses update, destroy, show). Both the `ImmediatelyPreviousResponseStatusCode` and `HasImmediatePreviousTransactionSucceeded` are introduced to depict the immediate predecessor status. The `HasImmediatePreviousTransactionSucceeded`, in particular, is an inferred attribute dependent on the use of certain status codes, and/or the availability of certain properties in the response body (e.g., Slack includes a top-level boolean property `ok` indicating success or failure). Concerning all predecessors, a list of boolean attributes can also be inferred indicating whether or not successful CRUD operations have occurred on the specific resource in the past.

As mentioned above, the names are given to attributes in such a way that humans can clearly recognise the intent of attributes. That could also potentially lead to the development of more human-readable models.

2. In order to easily identify the order of transactions, the value of the `Date` header in GMT (`day-name, day month year hour:minute:second GMT`) is converted into the Unix Epoch format (an integer representation of time in the form of seconds from January 1, 1970, UTC). This would provide a simple numerical value for the transaction date/time.

Table 5.1: Overview of Input Data for Attribute-Based Learning

Dataset	Input Attributes	Targets	Examples
GHTraffic	34	50	32,215
Twitter	32	63	26,053
Google Tasks	33	16	4,702
Slack	32	8	17,422

Data Preparation

The data extracted from recorded HTTP transactions is converted into Attribute-Relation File Format (ARFF) to be used in WEKA. Appendix B.1 includes a complete list of attributes obtained from each dataset. The ARFF files created can be found in the project repository, in the *training-data* folder. Appendix B.2 presents a summary of the contents of each ARFF file.

Data type conversion is done as the classification algorithms require attribute values to be nominal or binary (numeric or string attributes can have a de facto infinite domain). All numeric values are converted into a set of nominal attributes by using the `NumericToNominal` filter of WEKA. The filter basically takes all the integer values and adds them to the list of nominal values of the attribute (e.g., `ResponseStatusCode` has a predefined finite set of all possible values after the filter is applied). A similar approach is applied to string attributes using the `StringToNominal` filter.

The aim of the study is to generate HTTP response skeletons with multiple attributes (e.g. status code, response headers, response body). From the machine-learning point of view, this is, therefore, a multi-class or multi-target learning problem. Unfortunately, none of the machine learning toolboxes currently allow multivariate predictions. The only option is to train separate models for each target feature. When training a model for a target, all other target attributes are excluded, so that they do not influence the output (as only the request and the service history will determine the response feature).

The classification algorithms have confined the use of classifiers to non-unary targets (the target attribute must have at least two values). Accordingly, all target attributes which only have one distinct value are also ignored, as they have no discriminative value. On the other hand, target attributes holding a fairly large set of distinct values can also be excluded from learning as they are non-optimal for predictions. An example for a response feature with only one value is the *server* (assuming that service runs always on the same server), an example for a response feature with too many values is a high-precision *timestamp* (assuming that each transaction has a unique value). Similarly, it is possible to exclude high-variety input attributes as they are unreliable as inputs to a model. Using such excessive features decreases training speed, takes up a larger amount of memory, lowers model interpretability, and, most importantly, can result in overfitting and poor generalisation. The default threshold for exclusion is set to 10 distinct values. In all of these cases, the WEKA's `Remove` filter is used to eliminate attributes before the data is passed to the algorithms.

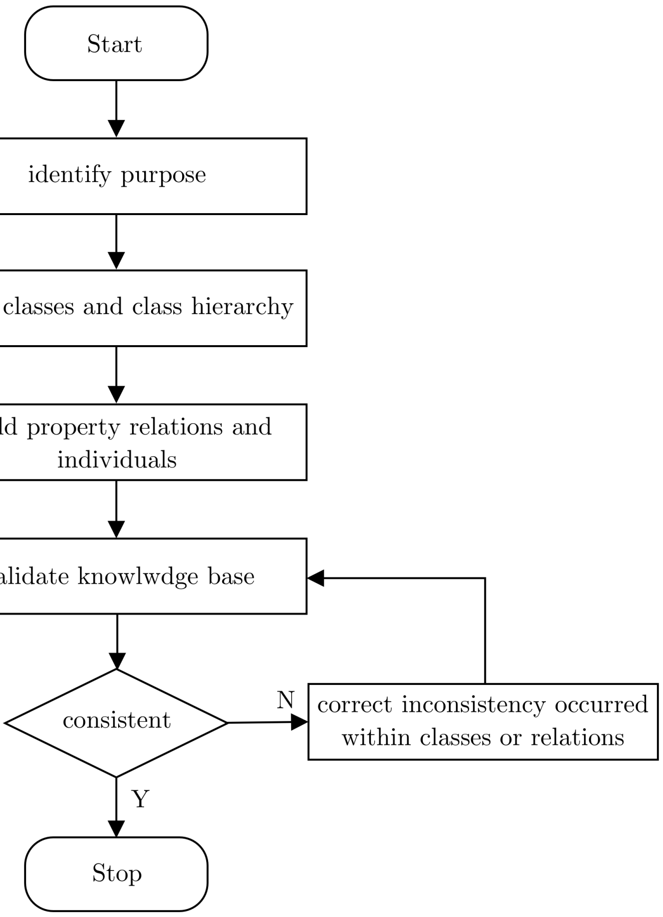

Start
identify purpose
build classes and class hierarchy
build property relations and individuals
validate knowlwdge base
consistent
N
correct inconsistency occurred within classes or relations
Y
Stop

Figure 5.4: The Knowledge Base Generation Process

A summary of the number of features associated with each input dataset after attribute removal is listed in Table 5.1. As can be seen, the datasets contain a variable number of features and targets. This depends on the amount of data each subject service maintains on its resources and how complex the resources are. The GHTraffic dataset, for instance, is based on the GitHub issue tracking system, where each issue has data on itself and other associated entities (like milestones, labels, and users), such that a considerable amount of data was received from the GitHub API leading to a range of features and targets. Appendix A shows examples of transaction records over service-specific resources.

5.3.2 Description Logic Learning Approach

As described in Section 2.3.3, DL learning requires two inputs. The first is a background knowledge base (or ontology) expressed in DL, which includes a set of concepts (i.e., classes) and relations (i.e., properties) encoded in a TBox, and sets of individual objects and their relationships to one another captured by an ABox. The second is a list of positive and negative examples based on the target class to be learnt (serve as a subset of individuals in the knowledge base). The goal of DL learning is to find a definition (logical formula) for the target class covering all/many positive examples and none/few negative examples.

The key steps in the data transformation task in DL learning involve building knowledge bases for experimental datasets using the OWL2 DL knowledge representation language (the most expressive OWL dialect) utilising the OWL API[3] written in Java, and identifying certain individuals within knowledge bases as examples and further processing to be adequately parsed to learning algorithms.

Background Knowledge Base Construction

An outline of the background knowledge base development process is shown in Figure 5.4. Each step is explained in detail in the section below.

We plan to use background knowledge bases to assist in learning definitions for the values of the different response features from the values associated with the incoming request features and the service state history. It is, therefore, necessary to include different classes in the knowledge base referring to all those attributes-values. `Transaction` is the root class in the knowledge base that represents all transaction records. It is also possible to add subclasses of `Transaction` (i.e., leaf classes) which defines the basic features that can be found in HTTP request/response messages. We use the same set of attributes obtained on the basis of the structural properties of HTTP messages in attribute-value learning (as described in Section 5.3.1) and extract the distinct values retained by each attribute. These attribute values result in an extensive set of subclasses (the key idea is to generate one class for each distinct value of an attribute). These distinctions are vital in this domain as objects with different values for an attribute are different kinds of objects. However, in this approach, attributes that hold a fairly large set of distinct values (e.g., a common case would be a *date* that belongs to exactly one transaction) are ignored as having low predictive power. We filter out these irrelevant features by applying the threshold limit of 10 (as same as in attribute-value learning). If a given property holds distinct values above the specified threshold limit then it will be ignored without creating new classes.

We follow the naming convention that class names start with the attribute name and use underscore to join the value name (e.g., `RequestMethod_POST`). In many attributes, however, the value names hold special characters and are lengthy, therefore, we use short human-readable strings (derived from the real values). For example, the `ResponseHeader_Content-Type` attribute contains a unique occurrence named `application/json;charset=UTF-8` where `Json` is inferred as the name of the value to create the `ResponseHeader_Content-Type_Json` subclass. The full name can be stored using the annotation property `rdfs:label`. This convention helps to make the intent of the property clearer to humans, which in turn allows learning algorithms to generate more human-readable expressions for class descriptions.

We also define some more details about these classes. The classes related to the same attribute have been made disjoint from one another as it is not possible for an individual to be an instance of more than one of these classes (e.g., the `ResponseStatusCode_200` and the `ResponseStatusCode_204` classes are disjoint: an instance of `200` can not be an instance of `204` at the same time). This is expressed in OWL using `owl:disjointWith` construct.

3. `http://owlcs.github.io/owlapi` [accessed 02 Aug. 2020]

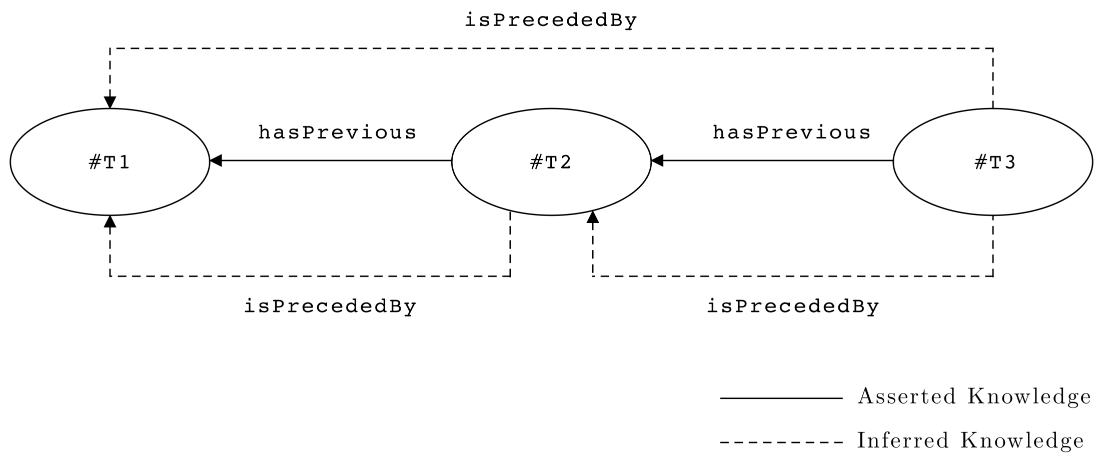


Figure 5.5: A Sample Inference on Object Properties

A set of object properties are added to the ontology to depict semantics on service state history. These properties relate `Transaction` class to itself, setting the order of transactions that happened to a named resource. A property called `hasPrevious` is defined. It is used to relate an instance of `Transaction` that occurred just prior to another instance of `Transaction` for a given resource. It is defined to be a *functional property* indicating that for any instance there can be at most one instance to be directly preceded. For example, given two instances of `Transaction` class, say, `T1` and `T2`, if `T2 hasPrevious T1`, then `T2` has no other instances immediately prior to that. The `hasPrevious` is also defined as an *asymmetric property.* This means that if `T2 hasPrevious T1`, then `T1` does not relate to `T2` along the same `hasPrevious` property (as `T1` did not occur before `T2`). It is also defined as *irreflexive property* because an individual cannot be related to itself through the property. A property called `isPrecededBy` is defined as a super property of the `hasPrevious`. It is used to relate a set of instances of `Transaction` that have taken place prior to another instance of `Transaction` for a given resource. It is defined to be a *transitive property*, therefore, for three given instances of `Transaction` class, say `T1`, `T2`, and `T3`, if `T2 isPrecededBy T1`, and `T3 isPrecededBy T2`, then `T3 isPrecededBy T1`.

The next step is to create individuals, then add them to classes, and add relationships between individuals through object properties. Every transaction in the dataset is presented as an individual in the knowledge base (named `T` suffixed with a unique integer) and is allocated as a member of the class `Transaction`. These individuals can be further assigned to the subclasses by means of the values that each transaction carries for the request/response attributes. The order of transactions for a given resource is asserted by relating two consecutive individuals using the object property `hasPrevious`. However, it is not necessary to make the `isPrecededBy` relation between individuals explicit. This can be automatically computed using a DL reasoner[4]. Such a reasoner can use the sub properties and simply infer the super properties. For example, in three instances of `Transaction` (`T1`, `T2`, and `T3`), if `T2 hasPrevious T1`, the reasoner may infer that

4. DL learning algorithms use reasoners to infer new knowledge automatically that is not explicitly contained within the background knowledge bases and to confirm the logical validity of knowledge bases. There are also multiple reasoner components introduced in the DL-Learner.

`T2 isPrecededBy T1`, since `hasPrevious` is a subset of `isPrecededBy`. In addition, if `T3 hasPrevious T2`, the reasoner may easily infer that `T3 isPrecededBy T1`, since the property `isPrecededBy` has been declared as transitive. Figure 5.5 illustrates this graphically.

Ensuring that the knowledge base is logically consistent is an important part of this development. Inconsistent knowledge bases may have a significant impact on the quality of results. Therefore, the resulting knowledge bases are checked for consistency by the reasoner before being used in model generation.

Identification of Examples and Data Preparation

DL learning is intended to be used to obtain descriptions for classes related to the response properties (known as target classes) given the background knowledge about the incoming request and the preceding request/response messages. It is, therefore, a multi-class learning problem that DL learning algorithms do not currently support. The only option is to perform separate learning tasks per target class.

Not all individuals on the background knowledge base serve as examples. We select the individuals that have happened most recently to each resource as the example set, and remove each of them from the target classes. Thus, the examples are the individuals presented in the form of the incoming request and the service history on the knowledge base. Some instances in the example set can be then listed as positive examples and others as negative ones to learn class definitions. Assume that the class description to be learned is the `ResponseStatusCode_200`, where instances in the example set with the status code value of `200` are grouped into the positive set, while the remaining instances in the example set are into the negative set that is not applicable to the learning.

DL learning algorithms have confined their application to learning problems whose background knowledge bases ranging from several to hundreds of thousands of axioms (where datasets range from small to large). This allows the entire knowledge bases to be loaded into OWL reasoners used in the algorithms for processing and reasoning. The Google Tasks dataset is a mid-sized and mid-complex dataset (the complexity of a dataset depends particularly on the number of resources in the dataset and transactions occurred per resource) that meets the above-mentioned criteria. However, the other three datasets, i.e., GHTraffic, Twitter, and Slack, are very large and complex, containing millions of axioms in their knowledge bases. Therefore, the DL Learning algorithms cannot be applied directly to learning problems with such knowledge bases (as reasoners cannot load the entirely of the large ontologies into memory). The potential solution for this problem is to use sub-datasets of the originals (making them manageable) that potentially cover the behaviour that occurs in the original datasets as much as possible, enabling the inferencing to be performed with medium-size knowledge bases. As a consequence, sub-datasets from GHTraffic, Twitter, and Slack are extracted for the development of background knowledge bases and examples (for conducting DL learning experiments).[5] Appendix B.3 provides details of all three sub-datasets and the URLs

5. Up to 20% of resources from the GHTraffic dataset, 40% from the Twitter, and 70% from the Slack need to be considered in order to extract transactions to form manageable datasets

Table 5.2: Overview of Input Data for Description Logic Learning

Dataset	Targets/Leaf Classes	Classes Assertions	Object Properties	Object Property Assertions	Examples/Individuals
GHTraffic	81/196	177,696	2	4,106	377/2,430
Twitter	126/162	201,209	2	4,042	504/2,525
Google Tasks	39/68	163,146	2	7,156	1,124/4,702
Slack	14/58	205,995	2	7,490	1,499/5,244

for accessing datasets.

DL learning algorithms have also limited their use to learning problems where both positive and negative examples exist. Accordingly, all target classes with no negative examples (the response properties with a single distinct value) are ignored from learning definitions. The target classes which do not include any positive examples would also be exempt from learning.

Appendix B.4 shows a complete list of classes and properties of the resulting background knowledge bases associated with each HTTP dataset visualised in Protege[6]. These knowledge bases can be found in the repository, in *training-data* folder. A summary of each background knowledge base after data preparation is given in Table 5.2.

5.4 Model Construction

Model construction in attribute-based learning is carried out using the WEKA 3.5 workbench. The WEKA implementation of C4.5 (implemented as J48), RIPPER (implemented as JRip), and PART algorithms are applied to train multiple models to predict different attributes associated with the response properties on each dataset. The generated ARFF files (as discussed in Section 5.3.1) are given as input to the algorithms. All algorithms are executed with the default configuration settings[7] since they have been empirically proven to perform well in general. In addition, there are no major hyperparameters available that could optimise the performance, so that a positive bias could be introduced if the parameters are adjusted (to maximise performance).

Model construction in DL learning is performed using the DL-Learner 1.4.0. The OCEL algorithm is utilised to learn class expressions for classes which relate to different values associated with the response properties on each dataset (each response property value is treated as a separate class learning problem). The background ontology built (in Section 5.3.2) is used as the input to OCEL along with a list of positive and negative examples related to the target class to be learnt. We use the `ClosedWorldReasoner` available in DL-Learner, so that those learning problems assume that all related information is known about the individuals in the domain and included in the knowledge base to be inferred. Moreover, there

while presumably retaining interaction patterns in the initial datasets. These percentages are determined depending on complexity of each dataset.

6. Protege is a free and open-source ontology editor and knowledge base framework available from `https://protege.stanford.edu` [accessed 02 Aug. 2020]. It has an attractive user interface that allows users to create, edit, and manage background knowledge bases.

7. The default WEKA parameters are explained in `https://weka.sourceforge.io/doc.dev/weka/classifiers/AbstractClassifier.html` [accessed 02 Aug. 2020]

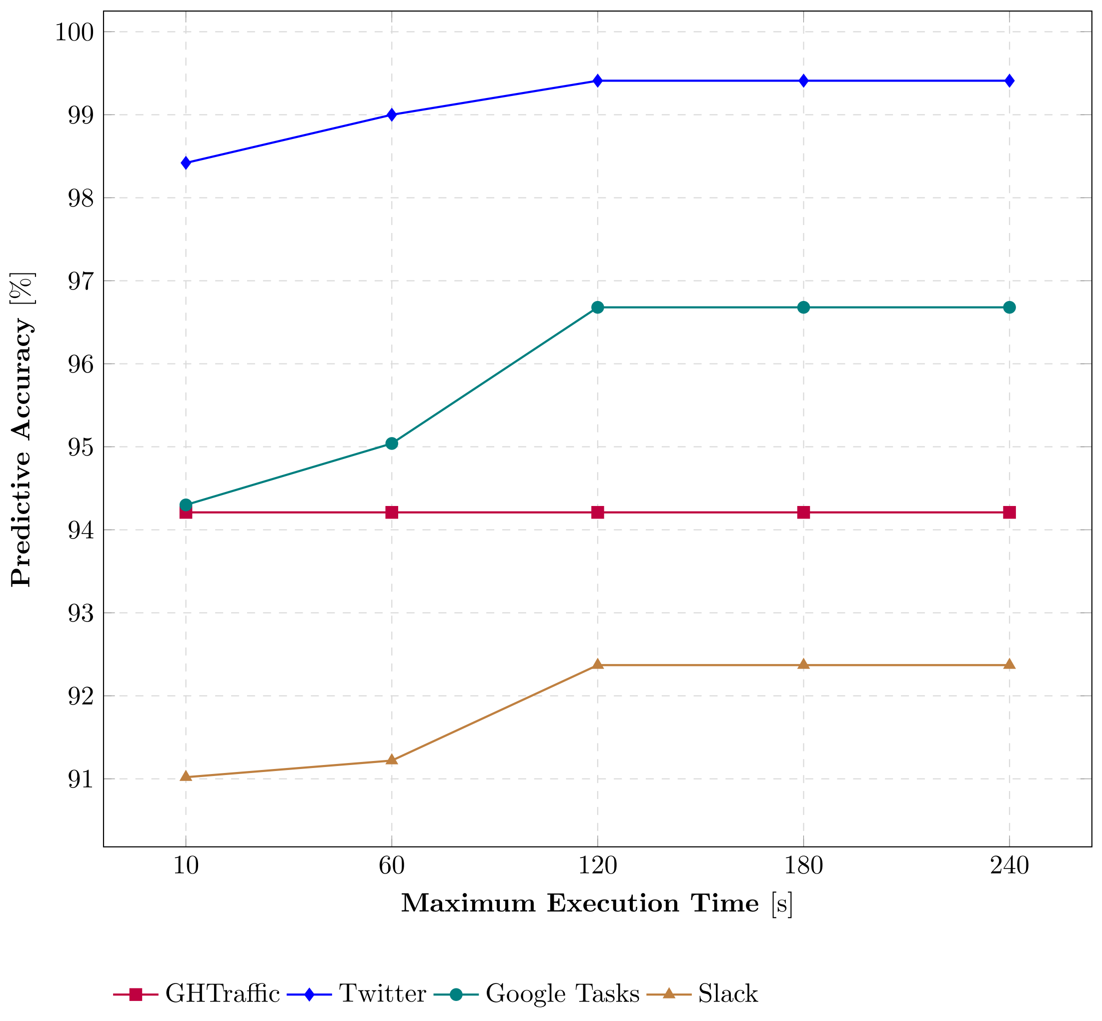


Figure 5.6: Effect of Maximum Execution Time on Predictive Accuracy

are hyperparameters in OCEL that influence the training process.[8] The most important hyperparameter in this case is the `MaxExecutionTimeInSeconds` (which stops the algorithm when the pre-set time is reached). The current default is 10 seconds, which could limit the ability of the algorithm to find an optimal solution. We perform a pilot study to determine the optimum value of this parameter that would yield the best results. This is achieved by setting five commonly used values for the maximum execution time (10, 60, 120, 180, and 240 seconds), constructing models targeting classes that only correspond to the response state of each dataset, and measuring the average predictive accuracy of the 2-fold cross validation (see Section 5.5 for more details on the evaluation methodology). The experimental results are plotted on the graph in Figure 5.6. As is evident from the results, the increase of the maximum execution time in the GHTraffic does not improve the predictive accuracy (almost similar results can be achieved with any time). In all other datasets, the predictive accuracy increases slightly from 10 seconds, resulting in the highest value at 120 seconds and almost constant from then on. This implies that it is possible to use any time from 120 seconds to achieve nearly

8. The configuration options used by DL-Learner is available in `https://cdn.rawgit.com/AKSW/DL-Learner/master/interfaces/doc/configOptions.html` [accessed 02 Aug. 2020]

similar results. Thus, we would choose the maximum execution time to be 120 seconds as the setting for OCEL on all datasets. This value stays constant during the training process.

5.5 Model Evaluation

Based on the aims of this thesis, it is important to evaluate the classification models generated by algorithms in terms of their performance and customisability. In the following sections, we discuss the selection of evaluation metrics and methods to calculating the metrics.

5.5.1 Evaluation Metrics

The performance evaluation metrics are selected with a particular focus on measuring the predictive capability of models (how well classes are identified). Predictive accuracy is the most commonly reported metric in the literature [261–264]. It simply stands for the capability of a model to identify instances correctly. To provide a more thorough evaluation of the models, we are also interested at two other widely used metrics: precision and recall. Precision is the ability of a model to not classify an instance as positive when it is not, whereas recall can be seen as the ability of a model to identify all positive instances. Below are the formulas for calculating the metrics:

- $\texttt{Predictive Accuracy} = \frac{\texttt{TruePositive + TrueNegative}}{\texttt{TruePositive + TrueNegative + FalsePositive + FalseNegative}}$

where: `TruePositive` is a number of positive instances correctly predicted to be positive, `TrueNegative` is a number of negative instances correctly predicted to be negative, `FalsePositive` is a number of negative instances erroneously predicted to be positive, and `FalseNegative` is a number of positive instances erroneously predicted to be negative.

- $\texttt{Precision} = \frac{\texttt{TruePositive}}{\texttt{TruePositive + FalsePositive}}$

- $\texttt{Recall} = \frac{\texttt{TruePositive}}{\texttt{TruePositive + FalseNegative}}$

The customisability of the models is evaluated mainly by emphasising the comprehensibility (how easy the models are to understand). The model size is often used to measure the comprehensibility with the implicit assumption that small/short models are easier to interpret (less complex models are more easily readable) [265–267]. The number of leaves and size of the tree (total number of nodes) or the number of rules produced by attribute-based learning algorithms is measured to assess the comprehensibility (as the number of nodes in a tree is roughly equivalent to the size of the corresponding ruleset, this can lead to a reasonable comparison). The definition length in the OWL class expression (total number of concepts, role, qualifiers, and connective symbols occurring that appear in the expression) is measured as a proxy for model size for DL learning.

The measures set out above will be calculated automatically by WEKA. DL-Learner, however, does not calculate precision or recall, so that the original scripts

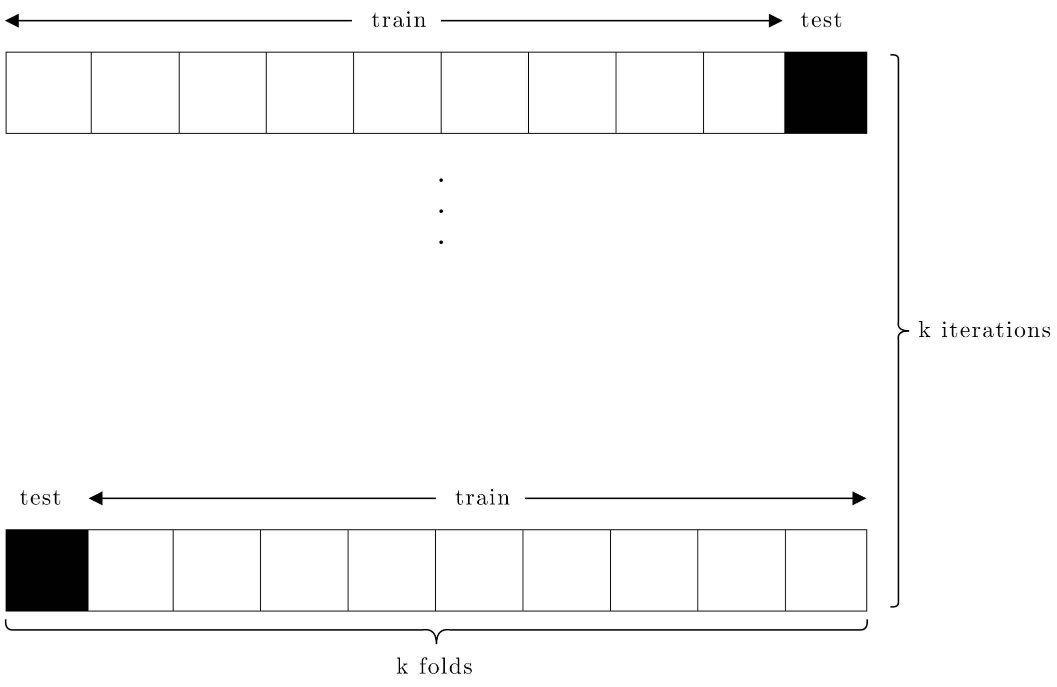


Figure 5.7: K-Fold Cross Validation

have to be modified to suit the needs. The corresponding implementation code can be found in the repository[9]. In order to get an idea of how well the learning algorithms perform on average on each dataset, the mean and the standard deviation of all these measures are later computed from all observed results of the collection of single-targeted models.

The experimental results of the selected algorithms will be presented and compared with each other in Chapter 6.

5.5.2 Cross Validation Approach

Cross Validation (CV) is a popular method used to evaluate ML models. The basic idea is to train a set of models on sub-datasets of the input data and to test them on the complementary sub-dataset. Typically, it is the preferred technique for assessing the generalisation capability of models (how well models perform on unseen data).

In k-fold CV, the examples in the input dataset are randomly divided into k equal sub-datasets (i.e., folds). Each of the k sub-dataset is then iteratively used as a test set, with the remaining k-1 used to train the model in each iteration. A graphical explanation can be found in Figure 5.7. This process is repeated k times, so that each example is used at least once in both the training and the test set. The chosen evaluation metrics are simply calculated as the average of all of the k folds. Repeating the experiment helps to avoid the issue of overfitting the data to the models and to have a clear understanding of the generalisation of the models.

9. `https://bitbucket.org/tbhagya/http-mock-skeletons` [accessed 02 Aug. 2020]

We apply the commonly used 10-fold CV in our evaluation of models generated from attribute-based learning algorithms. However, evaluation in OCEL with 10-fold CV turns out to be too computational intensive for the given datasets and ends up in `out of memory` exception. The CV may require more memory depending on how the OCEL algorithm is implemented. Further, the size of the knowledge bases is much larger (as expressed in OWL) for the algorithm to operate under the evaluation condition. Therefore, we use basic 2-fold CV for OCEL on each dataset. This approach is still superior as models have the potential to be trained and tested from all data and usually results in robust estimates. The evaluations in attribute-based learning algorithms are also repeated with 2-fold CV to allow the implicit comparison. This is reported in the evaluation results in Chapter 6.

5.5.3 Statistical Significance Test

The measures obtained from experiments (i.e., mean and standard deviation computed for each algorithm in each dataset) can be used to determine the suitability of symbolic learning techniques for predicting HTTP service responses (such as more or less accurate). However, it is important to decide whether the differences between the experimental outcomes reflect a trend or occur only by chance. In this respect, a statistical significance test can be carried out to draw relevant conclusions.

Every significance test usually starts with a null hypothesis and an alternative hypothesis. A null hypothesis states that there is no significant difference in the experimental results (assumes that the algorithms perform the same) and an alternative hypothesis states that there are significant differences in the results (assumes that algorithms perform differently). An appropriate test statistic is then utilised to measure the sampling distribution under the null hypothesis. The result of the test statistic is used to determine (usually by means of a p-value [268]) whether the null hypothesis can be accepted or denied with respect to a significant level. In the case where there is no evidence to reject the null hypothesis, there is no significant difference between the measures in relation to the given significant level. The significant level is the probability of rejecting the null hypothesis when it is valid. The most commonly used significance level is 0.05 which corresponds to a 5% chance of the results occurring randomly.

There are several statistically significant testing methodologies available (e.g., z-test [269], t-test [270], f-test [271]). Amongst, we are interested in using the t-test as it is the standard to be followed where the population variance is unknown and is measured using the sample variance (through the standard deviation). It compares the mean and the standard deviation (for a particular metric) of the two groups to see whether there is a significant difference between them. We thus perform pairwise tests between algorithms over all datasets to conclude the results of each metric at the statistical level.

Given the experimental results of two learning algorithms on a dataset, the t-value of the t-test for a single metric is calculated using the following formula:

- $\texttt{t-value} = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{(s^2(\frac{1}{n_1} + \frac{1}{n_2}))}}$

where: `x1` and `x2` are the means of the metric for the two groups being compared, `s2` is the pooled standard error of the two groups of the metric, and `n1` and `n2` are the numbers of observations in each of the groups. The p-value can then be estimated in relation to the t-value by using the t-distribution table [272]. If the p-value is higher than the statistical significance level, the difference between the evaluation results is not statistically significant (i.e., the null hypothesis is not rejected). It means that the algorithms being studied performed the same. The statistical results are used in Chapter 6 for the formulation of the discussion on the overall experimental results.

5.6 Reproducing Experimental Results

The experimental methodology described above is implemented in the Java language and is developed using the Maven software project management architecture. We provide access to these scripts, including a VirtualBox image with a pre-configured setup, to run the experiments and to reproduce the results.

Obtaining Scripts

The project scripts can be accessed either by cloning the repository from `https://bitbucket.org/tbhagya/http-mock-skeletons` or by downloading the pre-configured VirtualBox image from `https://zenodo.org/record/4025955/files/http-mock-skeletons-artifact-1.0.0.zip`

Environment Setup

The following settings are required to use the experimental scripts by cloning the repository:

1. Install Java Runtime Environment (JRE) 8[10] or above
2. Install Apache Maven 3.5.0[11] or above
3. Create *scripts/src/resources* folder where all raw and training data is stored
4. Download the appropriate dataset[12], based on the learning type you intend to experiment with (i.e., attribute-based learning or description logic learning)
5. Extract the zipped file and move the raw dataset file to the *scripts/src/resources*

10. `https://oracle.com/java/technologies/javase-jre8-downloads.html` [accessed 16 Aug. 2020]

11. `https://maven.apache.org/download.cgi` [accessed 16 Aug. 2020]

12. The GHTraffic dataset (small edition of 2.0.0) to conduct experiments with attribute-based learning can be downloaded from `https://zenodo.org/record/4007589/files/ghtraffic-S-2.0.0.zip`, and Twitter, Google Tasks and Slack can be downloaded using the links provided in Section 4.3.3. Use the links provided in Appendix B.3 to download the GHTraffic, Twitter, and Slack datasets (sub-datasets of the originals) to conduct experiments with description logic learning. The original Google Tasks dataset from `https://zenodo.org/record/4007570/files/googletasks-1.0.0.zip` is used to perform experiments on description logic learning.

Running Experiments

The three shell scripts resided in the *scripts* folder can be used to perform different experimental phases.

1. The script `datapreparator.sh` is configured to execute the **Data Preprocessing** and **Data Transformation** phases in which the downloaded raw data is cleaned and converted into data formats suitable for different learning types.

   It requires two arguments:

   - `-l` with either `abl` (for attribute-based learning) or `dll` (for description logic learning) specifying the learning type intended for experimentation
   - `-d` with either `GHTraffic`, `Twitter`, `GoogleTasks` or `Slack` specifying the type of dataset downloaded

   Depending on the learning type and dataset specified, the training data (e.g., ARFF or OWL) will be generated and stored in the *resources* folder.

2. The script `modellearner.sh` is configured to execute the **Model Construction** step in which the model artefacts are generated from training data created by `datapreparator.sh`.

   It requires three arguments:

   - `-a` with either `C4.5`, `RIPPER`, `PART` or `OCEL` specifying the learning algorithm intended for experimentation
   - `-d` with either `GHTraffic`, `Twitter`, `GoogleTasks` or `Slack` specifying the type of dataset
   - `-i` with the `target index` to train in attribute-based learning or `-c` with the `name of target class` to train in description logic learning

   Based on the specified algorithm and the target, the model will be generated and output to the terminal.

   The script also contains options `-ilist` and `-clist` with either `GHTraffic`, `Twitter`, `GoogleTasks` or `Slack` for retrieving the full list of target attribute indexes or target class names that are optimal for predictions in the specified dataset. By referring to these lists, users can identify the target attribute index or the target class name that they want to learn.

3. The script `modelevaluator.sh` is configured to execute the **Model Evaluation** step in which the generated model artefacts are tested using the cross validation technique.

   It requires the same input arguments as `modellearner.sh`.

   Depending on the specified algorithm and the target, the model will be evaluated and the results will be output to the terminal.

The `readme.md` file included in the repository provides further information on how to build and run the scripts. Note that the evaluation results can slightly differ in each execution as cross validation procedure randomly generates folds.

5.7 Summary

In this chapter, we presented a comprehensive overview of our experimental setup. The process was designed and implemented to preserve the reproducibility, generalisation, and comparability of results. It started with the cleaning of raw datasets. The preprocessed datasets were then converted into data formats compatible with selected symbolic learning algorithms and further filtered for best exposure to algorithms. This was followed by the development of multiple models to predict different response properties from the training data. Finally, the models generated were evaluated for their correctness and comprehensibility by cross validation and the t-test was used to assess the statistical significant differences between the results. The experimental results will be presented in the subsequent chapter of this thesis. We also provided instructions on how to reproduce the results.

Chapter 6

Experimental Results

This chapter presents and discusses the results of our experiments. A brief introduction to the experimental setup is provided in Section 6.1. The experimental evaluation results will be outlined and discussed in detail in Section 6.2 and potential threats to the validity of the results will be highlighted in Section 6.3.

6.1 Introduction

As was described in the introductory chapter, the main purpose of this study is to understand the appropriateness of Symbolic Machine Learning (SML) techniques for constructing mock skeletons of HTTP services directly from network traffic recordings. And as such, the study examined four promising SML algorithms, including three of the attribute-based learning algorithms, i.e., the C4.5 decision tree algorithm, the RIPPER and PART rule learners (utilising their WEKA implementations), and the OCEL Description Logic (DL) learning algorithm (from the DL-Learner). All the experiments were carried out on the interaction traces obtained from four different real HTTP services (i.e., GitHub, Twitter, Google, and Slack). Chapter 5 outlines the experimental approach adopted: the raw datasets were first cleaned up, followed by the transformation of the data into formats suitable for the selected learning algorithms. The algorithms and training datasets were then used to train multiple models to predict different features associated with the response properties, and finally, the resultant models were evaluated. There has also been a significant emphasis on the reproducibility of the experimental results.

We, particularly, employed the 10-fold Cross Validation (CV) technique to evaluate the models built by attribute-based learning algorithms. However, due to memory constraints, we could not perform 10-fold CV on OCEL and therefore opted to perform tests with 2-fold. A 2-fold CV was carried out subsequently on attribute-based learning algorithms to see whether any significant differences exist. Section 5.5.2 provides a complete overview of the method. In CV, the inferred models were assessed in two different aspects: predictive ability and comprehensibility. The predictive capability of the models was evaluated in terms of predictive accuracy, precision, and recall. To evaluate the comprehensibility of the models, the model size was used (i.e., tree size or number of rules was measured in attribute-based learning and class expression length was measured in DL learning).

Table 6.1: Average Predictive Accuracy Achieved by Algorithms on Datasets

	Technique			
Dataset	**C4.5**	**RIPPER**	**PART**	**OCEL**
GHTraffic	0.9835±0.0165	0.9835±0.0165	0.9835±0.0165	0.9533±0.0972
Twitter	1.0000±0.0000	1.0000±0.0000	1.0000±0.0000	0.9941±0.0040
Google Tasks	0.9988±0.0009	0.9986±0.0009	0.9988±0.0009	0.9733±0.0452
Slack	0.9544±0.1290	0.9544±0.1290	0.9544±0.1290	0.9216±0.1855

Table 6.2: Average Precision Achieved by Algorithms on Datasets

	Technique			
Dataset	**C4.5**	**RIPPER**	**PART**	**OCEL**
GHTraffic	0.4481±0.2789	0.4480±0.2789	0.4481±0.2789	0.9341±0.1506
Twitter	1.0000±0.0000	1.0000±0.0000	1.0000±0.0000	0.9852±0.0151
Google Tasks	0.9000±0.1532	0.8998±0.1531	0.9000±0.1532	0.8263±0.3310
Slack	0.9147±0.2413	0.9147±0.2413	0.9147±0.2413	0.9188±0.1848

Table 6.3: Average Recall Achieved by Algorithms on Datasets

	Technique			
Dataset	**C4.5**	**RIPPER**	**PART**	**OCEL**
GHTraffic	0.4509±0.2733	0.4510±0.2735	0.4509±0.2733	0.9851±0.0637
Twitter	1.0000±0.0000	1.0000±0.0000	1.0000±0.0000	0.9986±0.0015
Google Tasks	0.9005±0.1524	0.9003±0.1523	0.9005±0.1524	0.8690±0.3376
Slack	0.9375±0.1768	0.9375±0.1768	0.9375±0.1768	1.0000±0.0000

Section 5.5.1 outlines the details of each metric. In order to gain a better understanding of the overall (average) performance of algorithms on each dataset, the mean and the standard deviation of all these measurements were calculated from the observed results of the single-targeted model sets (plus, as in Section 5.5.3, the t-tests were performed to determine whether there were statistically significant differences in the averages obtained). It is also important to note that although the techniques proposed in [16,17] and [19,20] are aimed for stateful service virtualisation, we did not compare our results against them, due to their major limitations as listed in Chapter 3, especially because of their inability to generate accurate approximations of the actual behavioural responses of the HTTP/REST services depending on the internal service state, and the lack of provenance of their results. The research reported in this thesis is the very first to study the virtualisation of HTTP-based services and to focus directly on the generation of accurate HTTP responses with human-readable logics.

In the following sections, a detailed discussion of the experimental evaluation results will be presented.

Table 6.4: Average Model Size Achieved by Algorithms on Datasets

	Technique			
Dataset	**C4.5**	**RIPPER**	**PART**	**OCEL**
GHTraffic	6.4000±11.6986	2.2000±2.9416	3.2400 ±5.2238	7.9753±4.1261
Twitter	9.8889±0.8819	3.9365±0.3044	2.9841±0.1260	9.3651±1.1771
Google Tasks	5.3750±1.8212	2.5000±0.6325	3.3125±1.0782	6.8974±4.2165
Slack	9.3750±4.5020	3.3750±1.1877	4.0000±1.3093	3.5714±2.5933

6.2 Results and Discussion

Tables 6.1–6.4 present overall performance measurements (mean and standard deviation of predictive accuracy, precision, recall, and model size) based on different techniques applied (i.e., C4.5, RIPPER, PART, and OCEL), over all four datasets (i.e., GHTraffic, Twitter, Google Tasks, and Slack). For detailed quantitative results of attribute-based learning algorithms at target level in each dataset, see Tables 6.5–6.8. Appendix C.5 presents the 2-fold CV results of attribute-based learning algorithms. Evidently, there is no substantial difference in model performance when adjusting the number of folds in CV (between 2 and 10) and evaluating models generated from attribute-based learning algorithms. Therefore, an implicit comparison can be made between the obtained results from attribute-based learning and DL learning strategies. Tables 6.9–6.12 provide detailed quantitative OCEL results at the target level of each dataset. Appendix C.6 presents the results from t-tests comparing performance of algorithms on each dataset.

In general, the algorithms perform with an average accuracy of around 0.9216-1.0000 on all datasets, which means that the error rate is low and most of the inferred classifiers are reliable. The observed precision and recall averages of each algorithm are often quite close to the predictive accuracy on each dataset with the exception of the attribute-based learning algorithms (i.e., C4.5, RIPPER, and PART) on the GHTraffic[1], so we can reasonably be confident that most classifiers return accurate results (high precision corresponds to low false positive rate) and that the majority of the results are positive (high recall corresponds to low false negative rate). Moreover, the standard deviations of all of those measures are quite low (range from 0.0000-0.3376), which confirms that there are low variations in the measurements for different training and testing sets in the CV. The results of the t-tests further indicate that there are no statistically significant differences in

1. The GHTraffic dataset especially contains a set of response properties that are not optimal for predictions (which do not have associations with the request features and the service state history) and also have extremely unbalanced value distributions (a large proportion of instances from one value and relatively few instances spread across the rest). In attribute-based learning, the input attributes are irrelevant to the learning problem, and the algorithms use the central tendency of each target for classification, which results in high accuracy but low precision and recall. This will be addressed in more detail in Section 2.1. In DL learning, the inferred models for these targets always reach high measures. However, the algorithm can only learn the mapping for targets from the incoming request features and non-state-related response features of previous transactions (as the language used by DL learning account for much more expressive input than others).

the average measures between the C4.5, RIPPER, and PART across all datasets (estimated p-values between the attribute-based learning algorithms are larger than the considered significance level of 0.5, showing that we cannot reject the null hypothesis that there is no significant difference in the experimental results), whereas the averages of OCEL are generally slightly lower than those of the C4.5, RIPPER, and PART (estimated p-values between the attribute-based learning algorithms and the DL learning algorithm are less than 0.05, showing that we can reject the null hypothesis and conclude that there are differences). These results statistically provide convincing evidence that attribute-based learning and DL learning techniques perform slightly differently, despite both present higher predictive capabilities.

On the other hand, the average size of the classifiers produced by algorithms on all datasets is around 2.2000-9.8889, which means that the models are most compact and in a format that can be more easily interpreted. The t-test statistics from the experimental results show that that there is a slight statistical significance difference between the model size of algorithms across datasets. This is apparent because there are differences in the way the models are expressed by the algorithms.

While the classification algorithms are capable of achieving highly accurate models with human-readable logics for most of the targets associated with the response properties in each dataset, it is worth to note that there are certain targets where algorithms obtain comparatively low results. This can be attributed to factors such as the complexity of the learning problem (it is much more complex to model certain response features depending on the characteristics of the request and the status inferred from the service history), as well as insufficient training data (a limited number of transaction sequences representing various behavioural patterns). The following is a comprehensive review of the experimental results, along with a discussion of a few other circumstances which significantly impact the quality of the predictions.

6.2.1 Attribute-Based Learning Approach

This section discusses experimental findings in attribute-based learning based on different datasets. For each dataset, C4.5, RIPPER, and PART results are analysed with respect to the target attributes associated with the key HTTP response features, including the status, response headers, and response body.

Results on GHTraffic Dataset

Table 6.5 outlines the experimental results produced by C4.5, RIPPER, and PART for each target attribute in the GHTraffic dataset. See Appendix C.1.1, C.2.1, and C.3.1 for the sample classification models. A summary of all attributes found in the dataset is given in Appendix B.1.1.

As presented in Section 4.2, the GHTraffic dataset consists of HTTP transactions extracted from the GitHub's Issue Tracking system. It contains a set of `POST` requests to create issues returning `201`, `400`, `404` and `422`, `GET` and `HEAD` requests to query issues returning `200` and `404`, `PATCH` requests to update issues returning

Table 6.5: Results of Attribute-Based Learning Algorithms per Response Feature (Target) in GHTraffic. The calculated mean and standard deviation indicate the overall performance of each algorithm in predicting response properties of the dataset (as seen in Tables 6.1, 6.2, 6.3, and 6.4).

Target	**C4.5**					**RIPPER**				**PART**			
	Accuracy	**Precision**	**Recall**	**Leaves**	**Size**	**Accuracy**	**Precision**	**Recall**	**Rules**	**Accuracy**	**Precision**	**Recall**	**Rules**
ResponseStatusCode	0.9858	0.9948	0.8896	38	51	0.9858	0.9948	0.8896	9	0.9858	0.9948	0.8896	23
ResponseHeader_Cache-Control	1.0000	1.0000	1.0000	16	23	1.0000	1.0000	1.0000	5	1.0000	1.0000	1.0000	8
ResponseHeader_Vary	1.0000	1.0000	1.0000	16	23	1.0000	1.0000	1.0000	5	1.0000	1.0000	1.0000	8
ResponseHeader_X-Accepted-OAuth-Scopes	1.0000	1.0000	1.0000	15	21	1.0000	1.0000	1.0000	7	1.0000	1.0000	1.0000	7
ResponseHeader_X-OAuth-Scopes	1.0000	1.0000	1.0000	2	3	1.0000	1.0000	1.0000	2	1.0000	1.0000	1.0000	2
ResponseBody_assignee.site_admin	0.9634	0.4815	0.5000	1	1	0.9634	0.4815	0.5000	1	0.9634	0.4815	0.5000	1
ResponseBody_assignee.type	0.9634	0.4815	0.5000	1	1	0.9634	0.4815	0.5000	1	0.9634	0.4815	0.5000	1
ResponseBody_assignees.site_admin	0.9825	0.4915	0.5000	1	1	0.9825	0.4915	0.5000	1	0.9825	0.4915	0.5000	1
ResponseBody_assignees.type	0.9825	0.4915	0.5000	1	1	0.9825	0.4915	0.5000	1	0.9825	0.4915	0.5000	1
ResponseBody_closed_by.avatar_url	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.events_url	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.followers_url	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.following_url	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.gists_url	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.html_url	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.id	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.login	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.organizations_url	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.received_events_url	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.repos_url	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.site_admin	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.starred_url	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.subscriptions_url	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.type	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.url	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_documentation_url	0.9856	0.9895	0.9887	34	43	0.9842	0.9885	0.9935	17	0.9856	0.9895	0.9887	23
ResponseBody_locked	0.9999	0.6663	0.6667	16	23	0.9999	0.6663	0.6667	4	0.9999	0.6663	0.6667	8
ResponseBody_message	0.9959	0.8308	0.8333	18	27	0.9959	0.8308	0.8333	9	0.9959	0.8308	0.8333	14
ResponseBody_milestone.creator.avatar_url	0.9654	0.3217	0.3333	1	1	0.9654	0.3217	0.3333	1	0.9654	0.3217	0.3333	1
ResponseBody_milestone.creator.events_url	0.9654	0.3217	0.3333	1	1	0.9654	0.3217	0.3333	1	0.9654	0.3217	0.3333	1
ResponseBody_milestone.creator.followers_url	0.9654	0.3217	0.3333	1	1	0.9654	0.3217	0.3333	1	0.9654	0.3217	0.3333	1
ResponseBody_milestone.creator.following_url	0.9654	0.3217	0.3333	1	1	0.9654	0.3217	0.3333	1	0.9654	0.3217	0.3333	1
ResponseBody_milestone.creator.gists_url	0.9654	0.3217	0.3333	1	1	0.9654	0.3217	0.3333	1	0.9654	0.3217	0.3333	1
ResponseBody_milestone.creator.html_url	0.9654	0.3217	0.3333	1	1	0.9654	0.3217	0.3333	1	0.9654	0.3217	0.3333	1
ResponseBody_milestone.creator.id	0.9654	0.3217	0.3333	1	1	0.9654	0.3217	0.3333	1	0.9654	0.3217	0.3333	1
ResponseBody_milestone.creator.login	0.9654	0.3217	0.3333	1	1	0.9654	0.3217	0.3333	1	0.9654	0.3217	0.3333	1
ResponseBody_milestone.creator.organizations_url	0.9654	0.3217	0.3333	1	1	0.9654	0.3217	0.3333	1	0.9654	0.3217	0.3333	1
ResponseBody_milestone.creator.received_events_url	0.9654	0.3217	0.3333	1	1	0.9654	0.3217	0.3333	1	0.9654	0.3217	0.3333	1

Continued on next page

Table 6.5 – continued from previous page

Target	C4.5					RIPPER				PART			
	Accuracy	**Precision**	**Recall**	**Leaves**	**Size**	**Accuracy**	**Precision**	**Recall**	**Rules**	**Accuracy**	**Precision**	**Recall**	**Rules**
ResponseBody_milestone.creator.repos_url	0.9654	0.3217	0.3333	1	1	0.9654	0.3217	0.3333	1	0.9654	0.3217	0.3333	1
ResponseBody_milestone.creator.site_admin	0.9654	0.3217	0.3333	1	1	0.9654	0.3217	0.3333	1	0.9654	0.3217	0.3333	1
ResponseBody_milestone.creator.starred_url	0.9654	0.3217	0.3333	1	1	0.9654	0.3217	0.3333	1	0.9654	0.3217	0.3333	1
ResponseBody_milestone.creator.subscriptions_url	0.9654	0.3217	0.3333	1	1	0.9654	0.3217	0.3333	1	0.9654	0.3217	0.3333	1
ResponseBody_milestone.creator.type	0.9654	0.3217	0.3333	1	1	0.9654	0.3217	0.3333	1	0.9654	0.3217	0.3333	1
ResponseBody_milestone.creator.url	0.9654	0.3217	0.3333	1	1	0.9654	0.3217	0.3333	1	0.9654	0.3217	0.3333	1
ResponseBody_milestone.due_on	0.9654	0.3217	0.3333	1	1	0.9654	0.3217	0.3333	1	0.9654	0.3217	0.3333	1
ResponseBody_milestone.open_issues	0.9654	0.2500	0.2412	1	1	0.9654	0.2500	0.2412	1	0.9654	0.2500	0.2412	1
ResponseBody_milestone.state	0.9654	0.3217	0.3333	1	1	0.9654	0.3217	0.3333	1	0.9654	0.3217	0.3333	1
ResponseBody_state	0.9790	0.9350	0.9277	13	21	0.9790	0.9350	0.9277	5	0.9790	0.9350	0.9277	14
ResponseBody_user.site_admin	1.0000	1.0000	1.0000	16	23	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	8
ResponseBody_user.type	1.0000	1.0000	1.0000	16	23	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	8
Mean	0.9835	0.4481	0.4509	4.7800	6.4000	0.9835	0.4480	0.4510	2.2000	0.9835	0.4481	0.4509	3.2400
Standard Deviation	0.0165	0.2789	0.2733	8.4498	11.6986	0.0165	0.2789	0.2735	2.9416	0.0165	0.2789	0.2733	5.2238

200, 400, 401, 404 and 422, and PUT and DELETE requests to lock and unlock issues returning 204, 401 and 404. There are also a few GET requests which return 500 status code. Both in terms of size and complexity, this dataset is the largest and complex experimental dataset used in this study.

Basically, the GitHub API uses the response status code to indicate whether or not a request is successful. In case of a failure, the documentation_url and message fields in the response body are used to describe the error in depth. Accordingly, the ResponseStatusCode, ResponseBody_documentation_url, and ResponseBody_message are the target attributes associated with the response state in the GHTraffic dataset. As can be seen in Table 6.5, the inferred semantics models by C4.5, RIPPER, and PART have high predictive performance for each of these state-related targets and are small in size and less complex. For example, consider the ResponseStatusCode target attribute which has 8 distinct values: 200, 201, 204, 400, 401, 404, 422, and 500. It can be observed that the C4.5, RIPPER, and PART models for ResponseStatusCode score 0.9858 predictive accuracy by correctly classifying 31,758 instances out of 32,215, and the precision and recall rates are 0.9948 and 0.8896, respectively. Note that all incorrectly classified instances are from HTTP status 500 (which indicates that the server has encountered an unexpected condition). In particular, the GHTraffic dataset contains 516 records with HTTP 500 (see Table 4.6) of which 59 are correctly classified, and the remaining 457 instances are misclassified as HTTP 404 (i.e., 457 false positives for class 404 and 457 false negatives for class 500). This is mainly due to the fact that it is much more complex to infer rules that precisely model such unexpected server behaviour. However, it can be confirmed that all generic response codes in the GHTraffic dataset could be 100% accurately identified by the models. In reality, such a server error occurs at a very low frequency. Figure 6.1 shows the textual representation of the sample C4.5 decision tree. The first number in brackets at the end of each leaf is the total instances which reach the particular leaf. The second number represents the instances misclassified. The tree is 51 (nodes) in size and includes 38 leaves (due to the complexity of the dataset, the model size is fairly large compared to other datasets, but is readable and describes the scenario well). The resultant tree can also be represented in the form of unordered rules (a total of 38 rules could be obtained by extracting one rule per leaf). Some of the rules drawn from the tree are as follows:

```
1 IF (HasSuccessfulCreateOperationOccurredBefore = false) AND
  (HasRequestPayload = true) AND (HasValidRequestPayload = true) AND
  (RequestMethod = POST) AND
  (HasAuthorisationToken = true) THEN
  ResponseStatusCode = 201

2 IF (HasSuccessfulCreateOperationOccurredBefore = false) AND
  (HasRequestPayload = false) AND
  (RequestMethod = PATCH) THEN
  ResponseStatusCode = 404
```

Rule 1 encodes the semantics of HTTP 201 (Created) status code (which indicates the successful creation of an issue in the GitHub's Issue Tracking system). It stipulates that the response status code for a request will be HTTP 201 if the particular resource has not been successfully created previously and the request made using the method POST, valid authentication tokens, and a payload with correctly encoded values. Conversely, Rule 2 encodes the semantics of HTTP 404 (Not Found) status code to an edit (PATCH) request. It states that the response

```
HasSuccessfulCreateOperationOccurredBefore = false
|   HasRequestPayload = false
|   |   RequestMethod = HEAD: 404 (1769.0)
|   |   RequestMethod = DELETE: 404 (1698.0)
|   |   RequestMethod = POST: 422 (542.0)
|   |   RequestMethod = GET
|   |   |   ImmediatelyPreviousStatusCode = not-exist: 404 (295.0/67.0)
|   |   |   ImmediatelyPreviousStatusCode = 404
|   |   |   |   ImmediatelyPreviousMethod = not-exist: 404 (0.0)
|   |   |   |   ImmediatelyPreviousMethod = HEAD: 404 (303.0/56.0)
|   |   |   |   ImmediatelyPreviousMethod = DELETE: 404 (332.0/71.0)
|   |   |   |   ImmediatelyPreviousMethod = POST: 404 (254.0/43.0)
|   |   |   |   ImmediatelyPreviousMethod = GET: 500 (59.0)
|   |   |   |   ImmediatelyPreviousMethod = PUT: 404 (275.0/52.0)
|   |   |   |   ImmediatelyPreviousMethod = PATCH: 404 (274.0/61.0)
|   |   |   ImmediatelyPreviousStatusCode = 400: 404 (291.0/52.0)
|   |   |   ImmediatelyPreviousStatusCode = 201: 404 (0.0)
|   |   |   ImmediatelyPreviousStatusCode = 200: 404 (0.0)
|   |   |   ImmediatelyPreviousStatusCode = 422: 404 (118.0/55.0)
|   |   |   ImmediatelyPreviousStatusCode = 401: 404 (0.0)
|   |   |   ImmediatelyPreviousStatusCode = 204: 404 (0.0)
|   |   |   ImmediatelyPreviousStatusCode = 500: 404 (59.0)
|   |   RequestMethod = PUT: 404 (1720.0)
|   |   RequestMethod = PATCH: 404 (0.0)
|   HasRequestPayload = true
|   |   HasValidRequestPayload = false: 400 (1752.0)
|   |   HasValidRequestPayload = true
|   |   |   RequestMethod = HEAD: 404 (0.0)
|   |   |   RequestMethod = DELETE: 404 (0.0)
|   |   |   RequestMethod = POST
|   |   |   |   HasAuthorisationToken = true: 201 (1793.0)
|   |   |   |   HasAuthorisationToken = false: 404 (1718.0)
|   |   |   RequestMethod = GET: 404 (0.0)
|   |   |   RequestMethod = PUT: 404 (0.0)
|   |   |   RequestMethod = PATCH: 404 (1729.0)
HasSuccessfulCreateOperationOccurredBefore = true
|   HasAuthorisationToken = true
|   |   RequestUriPathToken6 = not-exist
|   |   |   HasRequestPayload = false
|   |   |   |   RequestMethod = HEAD: 200 (1722.0)
|   |   |   |   RequestMethod = DELETE: 200 (0.0)
|   |   |   |   RequestMethod = POST: 200 (0.0)
|   |   |   |   RequestMethod = GET: 200 (3065.0)
|   |   |   |   RequestMethod = PUT: 200 (0.0)
|   |   |   |   RequestMethod = PATCH: 422 (1317.0)
|   |   |   HasRequestPayload = true
|   |   |   |   HasValidRequestPayload = false: 400 (1747.0)
|   |   |   |   HasValidRequestPayload = true: 200 (1052.0)
|   |   RequestUriPathToken6 = lock: 204 (3104.0)
|   HasAuthorisationToken = false: 401 (5227.0)
```

Figure 6.1: The Sample C4.5 Tree for `ResponseStatusCode` in GHTraffic

status code for a request will be HTTP `404` if the particular resource has not been successfully created before and the request made using the `PATCH` method, and no payload has been transferred along with the request. Likewise, the inferred decision tree model is very simple to comprehend. Figures 6.2 and 6.3 present the sample RIPPER and PART decision lists (ordered rulesets) for `ResponseStatusCode` with 9 and 23 rules. Numbers in brackets at the end of each rule indicate the total number of instances that classified into the particular rule and the number of misclassified instances. We can observe that both RIPPER and PART models are simple to interpret: the rules are analysed in order, and the last rule is the default rule that applies if none of the other rules are true. In terms of induced knowledge, the models share similarities with C4.5, but do have some variations

```
(RequestMethod = GET) and
(HasSuccessfulCreateOperationOccurredBefore = false) and
(ImmediatelyPreviousMethod = GET) and
(ImmediatelyPreviousStatusCode = 404) => ResponseStatusCode=500 (59.0/0.0)

(RequestMethod = POST) and
(HasValidRequestPayload = true) and
(HasAuthorisationToken = true) => ResponseStatusCode=201 (1793.0/0.0)

(RequestMethod = PATCH) and
(HasRequestPayload = false) => ResponseStatusCode=422 (1317.0/0.0)

(RequestMethod = POST) and
(HasRequestPayload = false) => ResponseStatusCode=422 (542.0/0.0)

(RequestUriPathToken6 = lock) and
(HasAuthorisationToken = true) and
(HasSuccessfulCreateOperationOccurredBefore = true) => ResponseStatusCode=204
    (3104.0/0.0)

(HasRequestPayload = true) and
(HasValidRequestPayload = false) and
(HasAuthorisationToken = true) => ResponseStatusCode=400 (3499.0/0.0)

(HasAuthorisationToken = false) and
(HasValidRequestPayload = false) => ResponseStatusCode=401 (5227.0/0.0)

(HasSuccessfulCreateOperationOccurredBefore = true) => ResponseStatusCode=200
    (5839.0/0.0)

=> ResponseStatusCode=404 (10835.0/457.0)
```

Figure 6.2: The Sample RIPPER Ruleset for `ResponseStatusCode` in GHTraffic

due to the way algorithms work. It is clear from all of these examples that the models produced can be used to easily understand the possible causes of particular status codes. Having such information at hand makes it easy for the engineers to inspect and understand the logic behind status codes of service responses.

On the other hand, there are few target attributes associated with the response headers in the GHTraffic dataset that hold a single distinct value and are therefore excluded from the learning process, as the use of attribute-based learning algorithms is limited to non-unary targets. The `ResponseHeader_Server`, `ResponseHeader_Content-Type`, and `ResponseHeader_X-GitHub-Media-Type` are examples for targets with only one value (each record has the same value). In addition, targets with a fairly large set of distinct values are also exempt from learning because they are not appropriate for predictions. The `ResponseHeader_Date`, `ResponseHeader_ETag`, and `ResponseHeader_X-GitHub-Request-Id` are examples for targets with too many values (each record has a unique value). However, we can see that all targets with a limited number of distinct values relating to the response headers in the GHTraffic dataset (i.e., `ResponseHeader_Vary`, `ResponseHeader_Cache-Control`, `ResponseHeader_X-Accepted-OAuth-Scopes`, and `ResponseHeader_X-OAuth-Scopes`) achieve the maximum predictive performance across all algorithms (means 1.0 predictive accuracy, precision, and recall). The classification models are also small in size, which in turn, can be readily understood. As an instance, for the target `ResponseHeader_X-OAuth-Scopes` (related to the `X-OAuth-Scopes` API-specific header) containing 2 distinct values: `public_repo` and `not-exist`, C4.5 produces a tree with size 3 (including 2 leaves), while RIPPER and PART generate rulesets of 5 and 8 rules. Figures 6.4–6.6 depict

```
HasSuccessfulCreateOperationOccurredBefore = false AND
HasRequestPayload = false AND
RequestMethod = HEAD: 404 (1769.0)

HasSuccessfulCreateOperationOccurredBefore = false AND
HasRequestPayload = false AND
RequestMethod = PUT: 404 (1720.0)

HasSuccessfulCreateOperationOccurredBefore = false AND
HasValidRequestPayload = true AND
RequestMethod = POST AND
HasAuthorisationToken = true: 201 (1793.0)

HasSuccessfulCreateOperationOccurredBefore = false AND
HasRequestPayload = true AND
HasValidRequestPayload = true: 404 (3447.0)

HasAuthorisationToken = false: 401 (5227.0)

RequestUriPathToken6 = lock AND
HasSuccessfulCreateOperationOccurredBefore = true: 204 (3104.0)

HasRequestPayload = true AND
HasValidRequestPayload = false: 400 (3499.0)

HasSuccessfulCreateOperationOccurredBefore = true AND
RequestMethod = GET: 200 (3065.0)

HasSuccessfulCreateOperationOccurredBefore = true AND
RequestMethod = HEAD: 200 (1722.0)

HasRequestPayload = false AND
HasSuccessfulCreateOperationOccurredBefore = false AND
RequestMethod = DELETE: 404 (1698.0)

HasRequestPayload = false AND
RequestMethod = PATCH: 422 (1317.0)

RequestMethod = PATCH: 200 (1052.0)

RequestMethod = POST: 422 (542.0)

ImmediatelyPreviousStatusCode = 404 AND
ImmediatelyPreviousMethod = DELETE: 404 (332.0/71.0)

ImmediatelyPreviousStatusCode = 404 AND
ImmediatelyPreviousMethod = HEAD: 404 (303.0/56.0)

ImmediatelyPreviousStatusCode = not-exist: 404 (295.0/67.0)

ImmediatelyPreviousStatusCode = 400: 404 (291.0/52.0)

ImmediatelyPreviousStatusCode = 404 AND
ImmediatelyPreviousMethod = PUT: 404 (275.0/52.0)

ImmediatelyPreviousStatusCode = 404 AND
ImmediatelyPreviousMethod = PATCH: 404 (274.0/61.0)

ImmediatelyPreviousStatusCode = 404 AND
ImmediatelyPreviousMethod = POST: 404 (254.0/43.0)

ImmediatelyPreviousStatusCode = 422: 404 (118.0/55.0)

ImmediatelyPreviousStatusCode = 404: 500 (59.0)

: 404 (59.0)
```

Figure 6.3: The Sample PART Ruleset for `ResponseStatusCode` in GHTraffic

```
HasAuthorisationToken = true: public_repo (25270.0)
HasAuthorisationToken = false: not-exist (6945.0)
```

Figure 6.4: The Sample C4.5 Tree for ResponseHeader_X-OAuth-Scopes in GHTraffic

```
(HasAuthorisationToken = false) => ResponseHeader_X-OAuth-Scopes=not-exist
    (6945.0/0.0)

=> ResponseHeader_X-OAuth-Scopes=public_repo (25270.0/0.0)
```

Figure 6.5: The Sample RIPPER Ruleset for ResponseHeader_X-OAuth-Scopes in GHTraffic

```
HasAuthorisationToken = true: public_repo (25270.0)

: not-exist (6945.0)
```

Figure 6.6: The Sample PART Ruleset for ResponseHeader_X-OAuth-Scopes in GHTraffic

the sample models by C4.5, RIPPER, and PART. The first number in parentheses at the end of each leaf/rule is the total instances reaching the particular leaf/rule and the second number represents misclassified instances. These models can be used to easily interpret the semantics of `X-OAuth-Scopes` header. They stipulate that all responses for requests made with valid authorisation tokens will include the `X-OAuth-Scopes` header with the value `public_repo`, otherwise, there will be no `X-OAuth-Scopes` header in the responses.

In addition, the GHTraffic dataset includes a wide variety of target attributes associated with the content of the response body. Some have a fairly large set of distinct values (e.g., `ResponseBody_user.id`, `ResponseBody_created_at`, and `ResponseBody_comments`) and thus are ignored when generating models. It is noted that there are several other targets relevant to the response body (e.g., `ResponseBody_assignee.type` and `ResponseBody_milestone.open_issues`) that attain high predictive accuracy but fail to perform equally on other metrics. The induced models are either a single node tree or a ruleset with only the default class, resulting in high accuracy but low precision and recall (leading to a relatively high variance in the average measurements). It is very likely that the standard pruning options are preventing the models from growing. This is probably because the input attributes are irrelevant to the learning problem and the best an algorithm can do is to use the central tendency of that target for classification. We observe that these targets relate to the data returned by the server with respect to multiple entities to which the Issue Tracking system refers itself (such as *users*, *labels*, and *milestones*) and that correlation between/across targets and input attributes (features of the respective request message and service state history) cannot be guaranteed. It is also particularly notable that these targets have extremely unbalanced value distributions (a large proportion of instances from one value and relatively few instances spread across the rest of the values) contributing to high predictive accuracy. For example, the `ResponseBody_assignee.type` has two values (`not-exist` and `User`) and comprises 31,035 instances from the total dataset of 32,215 records related to `not-exist` (96.34% of the overall dataset), and even if the output is a single node (class `not-exist`), it could simply predict each instance as `no-exist`, thus achieving 0.9634 predictive accuracy. However, this will lead to low precision and recall values, as not even a single instance can be accurately identified in `User` class. Such target features are, therefore, non-optimal for predictions. Yet, there are few targets from the response body (such as `ResponseBody_locked` and `ResponseBody_state`) for which the algorithms produce very accurate, simple classifiers. For example, the target `ResponseBody_state` represents the `state` parameter that specifies whether the particular issue is `open` or `closed`. Accordingly, the target has 3 distinct values: `open`, `closed`, and `not-exist`. As can be seen in Table 6.5, the inferred models by C4.5, RIPPER, and PART for the `ResponseBody_state` have high predictive performance (all the instances misclassified belong to the HTTP status code `500`) and lower complexity. See Appendix C.1.1, C.2.1, and C.3.1 for sample classification models. It is obvious that these models can be used to easily understand the semantics of the `state` field in the response body.

Overall, C4.5, RIPPER, and PART are able to achieve highly accurate models with human-readable logics for the targets associated with the main response features of the GHTraffic dataset, with the exceptions for certain targets linked to the content of the response body. These targets represent data returned by the server, where no correlation can be guaranteed between/across targets and input attributes (features of the respective request message and service state history). In addition, the dataset includes records of unexpected service behaviour that often hinder algorithms from learning perfectly accurate models.

Results on Twitter Dataset

The results obtained from C4.5, RIPPER, and PART for each target in the Twitter dataset are shown in Table 6.6. The sample classification models are provided in Appendix C.1.2, C.2.2, and C.3.2. See Appendix B.1.2 for a complete list of attributes in the dataset.

The Twitter dataset is made up of HTTP messages collected from Twitter's Tweets feature. This includes `POST` requests to create and delete tweets that return `200` and `404`, and `GET` requests to retrieve tweets that returns `200` and `404`. Note that all of the requests have been directed to a single user account. See Section 4.3 for further details on the dataset.

Typically, the Twitter API uses the request method along with the naming patterns in the URI to define which Create, Read, Update, and Delete (CRUD) operation to execute on a resource. The API also incorporates both the `status code` and the `status` response header to express the state information for a request. When an error happens, the `errors` object is added to the response body with a concise error code and error text (`code` and `message` properties). As such, the target attributes corresponding to the response state in the Twitter dataset include the `ResponseStatusCode`, `ResponseHeader_status`, `ResponseBody_errors.code`, and `ResponseBody_errors.message`. As can be seen from Table 6.6, the semantic classifiers inferred by C4.5, RIPPER, and PART achieve the highest predictive performance scores for all of the targets associated with the response status. In particular, the models result in 1.0 predictive accuracy by correctly classifying all 26,053 instances of records. The precision and recall measures are also 1.0. It can

Table 6.6: Results of Attribute-Based Learning Algorithms per Response Feature (Target) in Twitter. The calculated mean and standard deviation indicate the overall performance of each algorithm in predicting response properties of the dataset (as seen in Tables 6.1, 6.2, 6.3, and 6.4).

Target	**C4.5**					**RIPPER**				**PART**			
	Accuracy	**Precision**	**Recall**	**Leaves**	**Size**	**Accuracy**	**Precision**	**Recall**	**Rules**	**Accuracy**	**Precision**	**Recall**	**Rules**
ResponseStatusCode	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	3	1.0000	1.0000	1.0000	3
ResponseHeader_status	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	3	1.0000	1.0000	1.0000	3
ResponseHeader_x-rate-limit-limit	1.0000	1.0000	1.0000	2	3	1.0000	1.0000	1.0000	2	1.0000	1.0000	1.0000	2
ResponseBody_contributors	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_coordinates	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_entities.hashtags	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_entities.symbols	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_entities.urls	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_entities.user_mentions	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_errors.code	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_errors.message	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_favorite_count	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_favorited	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_geo	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_in_reply_to_screen_name	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_in_reply_to_status_id	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_in_reply_to_status_id_str	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_in_reply_to_user_id	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_in_reply_to_user_id_str	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_is_quote_status	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_place	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_retweet_count	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_retweeted	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_truncated	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.contributors_enabled	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.created_at	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.default_profile	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.default_profile_image	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.favourites_count	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.follow_request_sent	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.followers_count	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.following	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.friends_count	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.geo_enabled	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.has_extended_profile	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.id	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.id_str	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.is_translation_enabled	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3

Continued on next page

Table 6.6 – continued from previous page

Target	C4.5					RIPPER				PART			
	Accuracy	Precision	Recall	Leaves	Size	Accuracy	Precision	Recall	Rules	Accuracy	Precision	Recall	Rules
ResponseBody_user.is_translator	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.lang	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.listed_count	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.location	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.name	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.notifications	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.profile_background_color	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.profile_background_image_url	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.profile_background_image_url_https	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.profile_background_tile	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.profile_banner_url	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.profile_image_url	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.profile_image_url_https	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.profile_link_color	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.profile_sidebar_border_color	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.profile_sidebar_fill_color	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.profile_text_color	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.profile_use_background_image	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.protected	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.screen_name	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.time_zone	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.translator_type	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.url	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.utc_offset	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.verified	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
Mean	1.0000	1.0000	1.0000	5.9365	9.8889	1.0000	1.0000	1.0000	3.9365	1.0000	1.0000	1.0000	2.9841
Standard Deviation	0.0000	0.0000	0.0000	0.5040	0.8819	0.0000	0.0000	0.0000	0.3044	0.0000	0.0000	0.0000	0.1260

be further observed that the resultant models are reasonably small in size and less complex. Consider, for example, the target `ResponseStatusCode` with 2 distinct values: `200` and `404`, where C4.5 builds a tree with 10 nodes and 6 leaves, and each RIPPER and PART rule learner builds an ordered ruleset with 3 rules. The textual representation of the sample C4.5 classification model is depicted in Figure 6.7. The first number in brackets the end of each leaf is the total instances which reach that particular leaf and the second number represents the instances misclassified. This tree can also be represented as a set of unordered decision rules by extracting paths from root to leaf. The derived classification rules are as follows:

```
1 IF (RequestHeader_Content-Type = application/x-www-form-urlencoded) THEN
  ResponseStatusCode = 200

2 IF (RequestHeader_Content-Type = not-exist) AND
  (HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false) AND
  (HasImmediatePreviousTransactionSucceeded = false) THEN
  ResponseStatusCode = 404

3 IF (RequestHeader_Content-Type = not-exist) AND
  (HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false) AND
  (HasImmediatePreviousTransactionSucceeded = true) AND
  (ImmediatelyPreviousMethod = not-exist) THEN
  ResponseStatusCode = 404

4 IF (RequestHeader_Content-Type = not-exist) AND
  (HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false) AND
  (HasImmediatePreviousTransactionSucceeded = true) AND
  (ImmediatelyPreviousMethod = POST) THEN
  ResponseStatusCode = 404

5 IF (RequestHeader_Content-Type = not-exist) AND
  (HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false) AND
  (HasImmediatePreviousTransactionSucceeded = true) AND
  (ImmediatelyPreviousMethod = GET) THEN
  ResponseStatusCode = 200

6 IF (RequestHeader_Content-Type = not-exist) AND
  (HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) THEN
  ResponseStatusCode = 200
```

The semantics of `ResponseStatusCode` are very clear. Take Rule 1, for example, which encodes one of the possibilities of getting HTTP `200` (`OK`) status code in the dataset (indicates that the request has been successful). It simply states that the status code for a request will be HTTP `200` if the request contains the `Content-Type` header with the `application/x-www-form-urlencoded` value. As we take a closer look at the dataset, it can be observed that each resource (tweet) has a `POST` request (with the `update.json` token held by the URI) to create the corresponding tweet (to update the user's current status with a text) as the very first transaction that returns `200` and these are the only records containing the `Content-Type` request header (as requests for retrieval and deletion of tweets do not have payloads). Thus, any request with the `Content-Type` header with the value `application/x-www-form-urlencoded` in this dataset may apparently represent the successful creation of a tweet. Evidently, the model can accurately capture the presented scenario. Figures 6.8 and 6.9 display the resultant RIPPER and PART models. Numbers in brackets at the end of each rule represent the total number of instances that classified under the given rule and the number of misclassified instances. We can note that they are very simple classification rules and even share parallels with C4.5 in terms of induced knowledge. It is obvious that the models can be easily interpreted by engineers to identify which properties

```
  RequestHeader_Content-Type = application/x-www-form-urlencoded: 200 (867.0)
  RequestHeader_Content-Type = not-exist
  |   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
  |   |   HasImmediatePreviousTransactionSucceeded = false: 404 (24025.0)
  |   |   HasImmediatePreviousTransactionSucceeded = true
  |   |   |   ImmediatelyPreviousMethod = not-exist: 404 (0.0)
  |   |   |   ImmediatelyPreviousMethod = POST: 404 (513.0)
  |   |   |   ImmediatelyPreviousMethod = GET: 200 (101.0)
  |   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: 200
      (547.0)
```

Figure 6.7: The Sample C4.5 Tree for `ResponseStatusCode` in Twitter

```
(RequestUriPathToken3 = update.json) => ResponseStatusCode=200 (867.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false) =>
    ResponseStatusCode=200 (648.0/0.0)

=> ResponseStatusCode=404 (24538.0/0.0)
```

Figure 6.8: The Sample RIPPER Ruleset for `ResponseStatusCode` in Twitter

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: 404 (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: 200
    (1515.0)

: 404 (513.0)
```

Figure 6.9: The Sample PART Ruleset for `ResponseStatusCode` in Twitter

are likely to lead to specific status codes. The differences between how the three techniques work also become very apparent from these samples.

Besides, most of the target attributes related to the response headers in the Twitter dataset have either a single value or a significant number of distinct values, such that they are excluded from learning. Examples for targets with a one value are the `ResponseHeader_content-type`, `ResponseHeader_server`, and `ResponseHeader_x-xss-protection`. The `ResponseHeader_last-modified`, `ResponseHeader_date`, and `ResponseHeader_x-response-time` are examples for targets with number of multiple values. Further, the `ResponseHeader_status` and `ResponseHeader_x-rate-limit-limit` are the targets with a few distinct values. As discussed above, the `ResponseHeader_status` is linked to the response status and algorithms produce 100% accurate classification models that can be easily comprehended. The remaining `ResponseHeader_x-rate-limit-limit` represents the response header `x-rate-limit-limit`, indicating the current limit in the number of API requests, which usually appears to be `900` for GET requests in the dataset. The target has thus 2 separate values, `900` and `not-exist`. As shown in the experimental results, this target also reaches the maximum degree of predictive performance (1.0 predictive accuracy, precision, and recall) and the classifiers are smaller, meaning the classifiers produced are reliable and interpretable. Figure 6.10 displays the sample classification model by C4.5. The number of leaves is 2 and the size of the tree is 3. Figures 6.11 and 6.12 show the RIPPER and PART

```
RequestMethod = POST: not-exist (20445.0)
RequestMethod = GET: 900 (5608.0)
```

Figure 6.10: The Sample C4.5 Tree for `ResponseHeader_x-rate-limit-limit` in Twitter

```
(RequestMethod = GET) => ResponseHeader_x-rate-limit-limit=900 (5608.0/0.0)

=> ResponseHeader_x-rate-limit-limit=not-exist (20445.0/0.0)
```

Figure 6.11: The Sample RIPPER Ruleset for `ResponseHeader_x-rate-limit-limit` in Twitter

```
RequestMethod = POST: not-exist (20445.0)

: 900 (5608.0)
```

Figure 6.12: The Sample PART Ruleset for `ResponseHeader_x-rate-limit-limit` in Twitter

models where each ruleset includes 2 rules. The first number in parentheses at the end of each leaf/rule is the total instances reaching the particular leaf/rule and the second number represents misclassified instances. It is noticeable that the models lead to a simple understanding of the semantics of the `x-rate-limit-limit` header and are capable of inferring the values precisely.

Moreover, there is a relatively larger number of target attributes related to the response body. Among them, some of the targets represent the status information and, as mentioned earlier in this section, it is apparent that the derived models are accurate and have a structure which can be readily comprehended. Table 6.6 shows that for all remaining targets associated with the response body, the classification models generated by C4.5, RIPPER, and PART have the highest predictive performance values and are as smaller as the others. However, it can be observed that all of these targets are correlated with the data values returned from the server referring to multiple other entities, such as *tweets*, *user*, and *timelines* objects, which could not normally be inferred from the features of the respective request and state history (the same issue has been detected with the response payload targets of the GHTraffic). But, as we look at the dataset more closely, we discover that each target has only two distinct values depending upon the success or failure of the request. Specifically, successful transactions have the same payload values (as directed to a single user) and there are no such data values for unsuccessful transactions. This means that the target values are centred on the status of the response. For the sample classification models, see Appendix C.1.2, C.2.2, and C.3.2. It can be clearly seen in all cases that the generated models are equivalent to the results of `ResponseStatusCode`. It is obvious that the algorithms generate precise and simple models that correctly capture this particular scenario in the dataset.

In all, the results demonstrate that the C4.5, RIPPER, and PART algorithms are significantly outperformed in training models that are both accurate and human-readable for all targets related to the core features of HTTP responses in the Twitter dataset.

Results on Google Tasks Dataset

Table 6.7 reports the experimental results for each target attribute in the Google Tasks dataset on C4.5, RIPPER, and PART. All of the sample classification models generated can be found in Appendix C.1.3, C.2.3, and C.3.3. Appendix B.1.3 is a complete list of attributes in the dataset.

As presented in Section 4.3, the Google Tasks dataset comprises HTTP transactions extracted from the Task Lists system in Google Tasks. It contains a collection of `POST` requests to create task lists that return `200`, `GET` requests to view task lists that return `200` and `404`, `PATCH` requests to update task lists that return `200` and `404`, and `DELETE` requests to remove task lists that return `204` and `404`. There are also a few `GET` and `PATCH` requests that return `503`.

Usually, the Google Tasks API embeds the success or failure of a request into the `status code` of the response. If an error occurs with a request, the API additionally returns a detailed description of the error in the response body (an `error` object with the properties `code` and `message` and `errors` array). Accordingly, the `ResponseStatusCode`, `ResponseBody_error.code`, `ResponseBody_error.message`, `ResponseBody_error.errors.domain`, `ResponseBody_error.errors.reason`, and `ResponseBody_error.errors.message` are the target attributes related to the response state in the Google Tasks dataset. The results presented in Table 6.7 demonstrate that the classifiers induced by C4.5, RIPPER, and PART have good predictive performance for all of these state-related targets and are quite small and less complex. As an example, for the `ResponseStatusCode` (with 4 distinct values: `200`, `204`, `404`, and `503`), the inferred models achieve 0.9977 predictive accuracy by correctly classifying 4,691 instances from the total dataset of 4,702 records and the precision and recall are 0.7487 and 0.75, respectively. It is noted that the models misclassify all 11 instances that belong to HTTP status code `503` (indicates that the server is temporarily unable to process the request) in the Google Tasks dataset (see Table 4.9 for details on the dataset), resulting in 7 false positives for class `200`, 4 false positives for class `404`, and 11 false negatives for class `503`. This is primarily because it is certainly not feasible to infer rules modelling the response codes of such unpredictable server behaviour based on the features of the request message and service state history. Nor is there a lack of training data for class `503`. In reality, the frequency at which such an error occurs is also very low. However, all generic response codes in the Google Tasks dataset can be 100% accurately defined by the models. The textual representation of the sample model for `ResponseStatusCode` is shown in Figure 6.13. The first number in parentheses at the end of each leaf is the total number of instances that reach the leaf, and the second number is the number of instances that are misclassified. The tree has 7 nodes and 5 leaves. The following set of unordered rules can also be retrieved by traversing the tree from root to leaf:

```
1 IF (HasSuccessfulDeleteOperationOccurredBefore = false) AND
(RequestMethod = POST) THEN
ResponseStatusCode = 200

2 IF (HasSuccessfulDeleteOperationOccurredBefore = false) AND
(RequestMethod = GET) THEN
ResponseStatusCode = 200

3 IF (HasSuccessfulDeleteOperationOccurredBefore = false) AND
(RequestMethod = PATCH) THEN
```

```
ResponseStatusCode = 200

4 IF (HasSuccessfulDeleteOperationOccurredBefore = false) AND
(RequestMethod = DELETE) THEN
ResponseStatusCode = 204

5 IF (HasSuccessfulDeleteOperationOccurredBefore = true) THEN
ResponseStatusCode = 404
```

Rules 1 to 3 encode the semantics of HTTP `200` (`OK`) status code (which indicates the successful creation/access/update of a task list). These rules stipulate that if the particular resource has not been successfully deleted before and the request made using either `POST`, `GET`, or `PATCH`, the response status code to the request will be HTTP `200`. Rule 4 encodes the semantics of HTTP `204` (`No Content`) status code (which indicates the successful deletion of a task list). It states that the response status code will be `204` if the particular resource has not been successfully deleted before and the request made using the `DELETE` method. On the contrary, Rule 5 encodes HTTP `404` (`Not Found`) semantics (which shows that the server can not find the requested resource). It specifies that if the particular resource has been successfully deleted before, then the response status code for the request will be HTTP `404`. In fact, it is evident that all these classification rules are very easy to comprehend. Figures 6.14 and 6.15 show the sample RIPPER and PART decision lists for `ResponseStatusCode` where each ruleset has 4 and 5 rules, respectively. The number in parentheses at the end of each rule is the instances reaching that particular rule and the instances that are misclassified. It can be observed that RIPPER and PART models also encode the same knowledge as C4.5, even though there are structural differences depending on how algorithms operate. It is very apparent that all models can be used to easily figure out the cause of a particular status code.

Besides, the Google Tasks dataset contains few target attributes related to the response headers that are either single-valued (i.e., `ResponseHeader_Server` and `ResponseHeader_Alt-Svc`) or have a large number of distinct values (i.e., `ResponseHeader_Date`, `ResponseHeader_ETag`, and `ResponseHeader_Expires`), thus exempt when constructing models. However, some of the targets with a limited number of distinct values linked to the headers, such as `ResponseHeader_Vary` and `ResponseHeader_Cache-Control` have near-perfect predictive performance measurements on C4.5, RIPPER, and PART (misclassifying few of `503`). The classification models produced are also smaller. The `ResponseHeader_Content-Type`, `ResponseHeader_X-Content-Type-Options`, `ResponseHeader_X-Frame-Options`, and `ResponseHeader_X-XSS-Protection` are the targets associated with the response headers in the dataset for which the inferred classifiers are 100% correct. Figure 6.16 displays the C4.5 model for `ResponseHeader_Content-Type`. The constructed tree is size 7 (nodes) and contains 5 leaves. Figures 6.17 and 6.18 show the sample RIPPER and PART ordered rulesets where each includes 2 and 4 rules. The first number in parentheses at the end of each leaf/rule is the total instances reaching the particular leaf/rule. The second number represents misclassified instances.

All algorithms perform predictive accuracy of 1.0 including precision and recall rates of 1.0. This target represents the `Content-Type` response header which specifies the type of content returned in the responses. As can be seen, the samples models explicitly state that all responses to requests will contain the `Content-Type`

```
HasSuccessfulDeleteOperationOccurredBefore = false
|   RequestMethod = POST: 200 (1124.0)
|   RequestMethod = GET: 200 (1005.0/3.0)
|   RequestMethod = PATCH: 200 (485.0/4.0)
|   RequestMethod = DELETE: 204 (606.0)
HasSuccessfulDeleteOperationOccurredBefore = true: 404 (1482.0/4.0)
```

Figure 6.13: The Sample C4.5 Tree for `ResponseStatusCode` in Google Tasks

```
(RequestMethod = DELETE) and
(HasSuccessfulDeleteOperationOccurredBefore = false) => ResponseStatusCode=204
    (606.0/0.0)

(HasSuccessfulDeleteOperationOccurredBefore = true) => ResponseStatusCode=404
    (1482.0/4.0)

=> ResponseStatusCode=200 (2614.0/7.0)
```

Figure 6.14: The Sample RIPPER Ruleset for `ResponseStatusCode` in Google Tasks

```
HasSuccessfulDeleteOperationOccurredBefore = true: 404 (1482.0/4.0)

RequestMethod = POST: 200 (1124.0)

RequestMethod = GET: 200 (1005.0/3.0)

RequestMethod = DELETE: 204 (606.0)

: 200 (485.0/4.0)
```

Figure 6.15: The Sample PART Ruleset for `ResponseStatusCode` in Google Tasks

```
RequestMethod = POST: application/json; charset=UTF-8 (1124.0)
RequestMethod = GET: application/json; charset=UTF-8 (1367.0)
RequestMethod = PATCH: application/json; charset=UTF-8 (1082.0)
RequestMethod = DELETE
|   HasSuccessfulDeleteOperationOccurredBefore = false: not-exist (606.0)
|   HasSuccessfulDeleteOperationOccurredBefore = true: application/json; charset=
    UTF-8 (523.0)
```

Figure 6.16: The Sample C4.5 Tree for `ResponseHeader_Content-Type` in Google Tasks

```
(RequestMethod = DELETE) and
(HasSuccessfulDeleteOperationOccurredBefore = false) => ResponseHeader_Content-
    Type=not-exist (606.0/0.0)

=> ResponseHeader_Content-Type=application/json; charset=UTF-8 (4096.0/0.0)
```

Figure 6.17: The Sample RIPPER Ruleset for `ResponseHeader_Content-Type` in Google Tasks

```
RequestMethod = GET: application/json; charset=UTF-8 (1367.0)

HasRequestPayload = true: application/json; charset=UTF-8 (2206.0)

HasSuccessfulDeleteOperationOccurredBefore = false: not-exist (606.0)

: application/json; charset=UTF-8 (523.0)
```

Figure 6.18: The Sample PART Ruleset for `ResponseHeader_Content-Type` in Google Tasks

Table 6.7: Results of Attribute-Based Learning Algorithms per Response Feature (Target) in Google Tasks. The calculated mean and standard deviation indicate the overall performance of each algorithm in predicting response properties of the dataset (as seen in Tables 6.1, 6.2, 6.3, and 6.4).

Target	**C4.5**					**RIPPER**				**PART**			
	Accuracy	**Precision**	**Recall**	**Leaves**	**Size**	**Accuracy**	**Precision**	**Recall**	**Rules**	**Accuracy**	**Precision**	**Recall**	**Rules**
ResponseStatusCode	0.9977	0.7485	0.7500	5	7	0.9977	0.7485	0.7500	3	0.9977	0.7485	0.7500	5
ResponseHeader_Accept-Ranges	0.9994	0.9990	0.9995	3	5	0.9985	0.9980	0.9985	3	0.9994	0.9990	0.9995	3
ResponseHeader_Cache-Control	0.9985	0.9983	0.9983	5	7	0.9983	0.9983	0.9980	3	0.9985	0.9983	0.9983	4
ResponseHeader_Content-Type	1.0000	1.0000	1.0000	5	7	1.0000	1.0000	1.0000	2	1.0000	1.0000	1.0000	4
ResponseHeader_Pragma	0.9991	0.9990	0.9990	5	7	0.9990	0.9990	0.9990	4	0.9991	0.9990	0.9990	4
ResponseHeader_Transfer-Encoding	0.9994	0.9990	0.9995	3	5	0.9981	0.9975	0.9980	3	0.9994	0.9990	0.9995	3
ResponseHeader_Vary	0.9994	0.9990	0.9995	3	5	0.9983	0.9975	0.9985	3	0.9994	0.9990	0.9995	3
ResponseHeader_X-Content-Type-Options	1.0000	1.0000	1.0000	5	7	1.0000	1.0000	1.0000	2	1.0000	1.0000	1.0000	4
ResponseHeader_X-Frame-Options	1.0000	1.0000	1.0000	5	7	1.0000	1.0000	1.0000	2	1.0000	1.0000	1.0000	4
ResponseHeader_X-XSS-Protection	1.0000	1.0000	1.0000	5	7	1.0000	1.0000	1.0000	2	1.0000	1.0000	1.0000	4
ResponseBody_error.code	0.9977	0.6650	0.6667	2	3	0.9977	0.6650	0.6667	2	0.9977	0.6650	0.6667	2
ResponseBody_error.errors.domain	0.9985	0.9990	0.9975	2	3	0.9985	0.9990	0.9975	2	0.9985	0.9990	0.9975	2
ResponseBody_error.errors.message	0.9977	0.6650	0.6667	2	3	0.9977	0.6650	0.6667	2	0.9977	0.6650	0.6667	2
ResponseBody_error.errors.reason	0.9977	0.6650	0.6667	2	3	0.9977	0.6650	0.6667	2	0.9977	0.6650	0.6667	2
ResponseBody_error.message	0.9977	0.6650	0.6667	2	3	0.9977	0.6650	0.6667	2	0.9977	0.6650	0.6667	2
ResponseBody_kind	0.9985	0.9985	0.9985	5	7	0.9985	0.9985	0.9985	3	0.9985	0.9985	0.9985	5
Mean	0.9988	0.9000	0.9005	3.6875	5.3750	0.9986	0.8998	0.9003	2.5000	0.9988	0.9000	0.9005	3.3125
Standard Deviation	0.0009	0.1532	0.1524	1.4009	1.8212	0.0009	0.1531	0.1523	0.6325	0.0009	0.1532	0.1524	1.0782

```
HasSuccessfulDeleteOperationOccurredBefore = false
|   RequestMethod = POST: tasks#taskList (1124.0)
|   RequestMethod = GET: tasks#taskList (1005.0/3.0)
|   RequestMethod = PATCH: tasks#taskList (485.0/4.0)
|   RequestMethod = DELETE: not-exist (606.0)
HasSuccessfulDeleteOperationOccurredBefore = true: not-exist (1482.0)
```

Figure 6.19: The Sample C4.5 Tree for `ResponseBody_kind` in Google Tasks

```
(HasSuccessfulDeleteOperationOccurredBefore = true) => ResponseBody_kind=not-
    exist (1482.0/0.0)

(RequestMethod = DELETE) => ResponseBody_kind=not-exist (606.0/0.0)

=> ResponseBody_kind=tasks#taskList (2614.0/7.0)
```

Figure 6.20: The Sample RIPPER Ruleset for `ResponseBody_kind` in Google Tasks

header with the value '`application/json; charset=UTF-8`' except for successful delete requests (which usually do not contain any content in the response body). It is noticeable that these models lead to a simple interpretation of the semantics of the `Content-Type` header and are capable of inferring the exact values.

The Google Tasks API typically includes relatively small payloads for responses, such that there are only a few associated targets in the Google Tasks dataset. Among them are targets representing the status information and, as discussed above, we found that the resultant models are accurate and have a basic structure that is easy to interpret. Additionally, there are few targets associated with the response body in the Google Tasks dataset that hold a fairly large set of distinct values and therefore ignored without learning. Apart from all that, the `ResponseBody_kind` is the only other target with respect to the response body in the Google Tasks dataset. The `ResponseBody_kind` represents a property called `kind` in the response body that specifies the type of resource, that appears to always be `tasks#tasks` for successful `POST`, `GET`, and `PATCH` requests. Accordingly, the target has 2 distinct values: `tasks#tasks` and `not-exist`. As can be seen from Table 6.7, the inferred models by C4.5, RIPPER, and PART for the `ResponseBody_kind` obtain 0.9985 predictive accuracy by correctly classifying 4,695 instances out of 4,702 and also have 0.9985 precision and recall. In all of these contexts, 7 instances belonging to HTTP `503` are classified incorrectly. The corresponding C4.5, RIPPER, and PART models can be seen respectively in Figures 6.19–6.21, where the tree is size 7 (contains 5 leaves), and each set of rules contains 4 and 5 rules. The first number in brackets at the end of each leaf/rule is the total instances which reach the particular leaf/rule, whereas the second number represents the instances misclassified. From the examples, it is apparent that the semantics of the `kind` property can be readily understood. It is noticed that the target values are centred on the response status and hence the generated models are equivalent to the `ResponseStatusCode` models.

In view of the results, it can be concluded that the C4.5, RIPPER, and PART algorithms achieve higher predictive performance results when predicting all the core features of the HTTP responses in the Google Tasks dataset and the models are readable. However, because the dataset includes records of unexpected service

```
HasSuccessfulDeleteOperationOccurredBefore = true: not-exist (1482.0)

RequestMethod = POST: tasks#taskList (1124.0)

RequestMethod = GET: tasks#taskList (1005.0/3.0)

RequestMethod = DELETE: not-exist (606.0)

: tasks#taskList (485.0/4.0)
```

Figure 6.21: The Sample PART Ruleset for `ResponseBody_kind` in Google Tasks

behaviour, it hinders algorithms from learning perfectly accurate models.

Results on Slack Dataset

The experimental results obtained for the target attributes in the Slack dataset on C4.5, RIPPER, and PART are shown in Table 6.8. The sample classification models can be found in Appendix C.1.4, C.2.4, and C.3.4. A list of all attributes in the dataset is provided in Appendix B.1.4.

The Slack dataset contains a collection of `POST` requests for sending, updating, and deleting chat messages to/from a channel that returns `200`. Note that all of the requests have been directed from a single user account to a single channel. See Section 4.3 for further details on the dataset.

In general, the Slack API uses HTTP `POST` for each request and presents certain naming patterns in URI tokens to signify which CRUD operation to perform. For example, to operate on a chat message, the API adds either a `chat.postMessage`, `chat.update`, or `chat.delete` to the URI. The Slack API also always passes HTTP `200` status code (even if a request failed) and adds more substantive information about the state to the response body. In particular, the response body contains the boolean property `ok` indicating the success or failure of a request. In the event of a failure, the response body further holds the `error` property with a short error message. Accordingly, the `ResponseStatusCode` target attribute in the Slack dataset has only one distinct value (i.e., `200`) and therefore is ignored when generating models. In accordance with Table 6.8, the inferred models by C4.5, RIPPER, and PART for all other targets associated with the response status in the Slack dataset, i.e., `ResponseBody_ok` and `ResponseBody_error`, score the highest degree of predictive performance. They reach 1.0 predictive accuracy by correctly classifying all 17,422 instances of records, and the precision and recall rates are 1.0. The models are also smaller and less complex, which in effect, can be interpreted more easily. As an example, Figure 6.22 presents the textual representation of the sample model for `ResponseBody_ok` by C4.5. The constructed tree is 8 nodes in size and has 14 leaves. Figures 6.23 and 6.24 show the RIPPER and PART models where each ruleset includes 4 rules, respectively. The number in parentheses at the end of each leaf/rule is the total instances reaching that particular leaf/rule and the misclassified instances. These models demonstrate that the semantics of `ok` can be well understood. One of the logics identified is that the `ok` will be `true` if the particular resource has not been successfully deleted before and if the URI path of the request carries the `chat.postMessage` token at its second position (means if the request is to create the resource) and if the

Table 6.8: Results of Attribute-Based Learning Algorithms per Response Feature (Target) in Slack. The calculated mean and standard deviation indicate the overall performance of each algorithm in predicting response properties of the dataset (as seen in Tables 6.1, 6.2, 6.3, and 6.4).

Target	**C4.5**					**RIPPER**				**PART**			
	Accuracy	**Precision**	**Recall**	**Leaves**	**Size**	**Accuracy**	**Precision**	**Recall**	**Rules**	**Accuracy**	**Precision**	**Recall**	**Rules**
ResponseHeader_x-slack-router	0.6350	0.3175	0.5000	1	1	0.6350	0.3175	0.5000	1	0.6350	0.3175	0.5000	1
ResponseBody_channel	1.0000	1.0000	1.0000	8	14	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	4
ResponseBody_error	1.0000	1.0000	1.0000	8	14	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	4
ResponseBody_message.bot_id	1.0000	1.0000	1.0000	5	8	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	5
ResponseBody_message.edited.user	1.0000	1.0000	1.0000	5	8	1.0000	1.0000	1.0000	2	1.0000	1.0000	1.0000	4
ResponseBody_message.type	1.0000	1.0000	1.0000	5	8	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	5
ResponseBody_message.user	1.0000	1.0000	1.0000	5	8	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	5
ResponseBody_ok	1.0000	1.0000	1.0000	8	14	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	4
Mean	0.9544	0.9147	0.9375	5.6250	9.3750	0.9544	0.9147	0.9375	3.3750	0.9544	0.9147	0.9375	4.0000
Standard Deviation	0.1290	0.2413	0.1768	2.3867	4.502	0.1290	0.2413	0.1768	1.1877	0.1290	0.2413	0.1768	1.3093

```
HasSuccessfulDeleteOperationOccurredBefore = false
|   HasURLInImmediatelyPreviousTransactionContainsATokenToUpdate = false
|   |   RequestUriPathToken2 = chat.update
|   |   |   HasImmediatePreviousTransaction = false: false (309.0)
|   |   |   HasImmediatePreviousTransaction = true: true (1292.0)
|   |   RequestUriPathToken2 = chat.delete
|   |   |   HasImmediatePreviousTransaction = false: false (206.0)
|   |   |   HasImmediatePreviousTransaction = true: true (1826.0)
|   |   RequestUriPathToken2 = chat.postMessage: true (3985.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToUpdate = true
|   |   HasImmediatePreviousTransactionSucceeded = false: false (335.0)
|   |   HasImmediatePreviousTransactionSucceeded = true: true (540.0)
HasSuccessfulDeleteOperationOccurredBefore = true: false (8929.0)
```

Figure 6.22: The Sample C4.5 Tree for `ResponseBody_ok` in Slack

immediately preceding transaction does not have an update token in the URI of the request (means that no transaction exists to update the resource immediately preceding). This particularly encodes the semantics of the successful creation of a message in Slack's Messages system. Clearly, the samples show that the generated classifiers are very easy to understand and explain well the scenario presented in the dataset. The semantic knowledge inferred by C4.5, RIPPER, and PART can also be observed to be close.

Further, the Slack dataset contains a relatively high number of targets related to the response headers with either a single value or a large set of different values that are not appropriate for predictions. The `ResponseHeader_Cache-Control`, `ResponseHeader_Content-Type`, and `ResponseHeader_Server` are examples for targets with a single value. The `ResponseHeader_Date` and `ResponseHeader_Via` are examples for targets with many different values. Besides, the only target trained in relation to the response headers is `ResponseHeader_x-slack-router`. The target has two values: `p` and `not-exist` and comprises 11,063 instances from the value `p` (63.5002% of the overall dataset). It can be observed that the generated C4.5 model is a single node tree (`p`) and that the RIPPER and PART models have only the default class `p`. The models reach 0.635 predictive accuracy by simply predicting each instance as `p` and the precision is 0.3175 whereas the recall is 0.5. This is probably due to the fact there that is no correlation between the target and input attributes, so that algorithms can not learn anything useful from the data (algorithms do not have adequate information to grow the models). It is noted that the `x-slack-router` response header contains data returned by the server and there is no relation between its values and the respective request features and the service state history. The relatively low scores of `ResponseHeader_x-slack-router` also lead to comparatively low average scores in the Slack dataset with a notable value distribution. Although a target of this sort has a few distinct values, it is not optimal for predictions.

On the other hand, there are few targets associated with the response payload. Among, the `ResponseBody_ts`, `ResponseBody_text`, `ResponseBody_message.ts`, `ResponseBody_message.text`, and `ResponseBody_message.edited.ts` are ignored from the learning process as they contain many unique values (as such targets are not predictable). However, we can note that all other targets with few distinct values (e.g., `ResponseBody_channel` and `ResponseBody_message.user`) have the highest predictive performance statistics (1.0 predictive accuracy, and 1.0 preci-

```
(HasSuccessfulDeleteOperationOccurredBefore = false) and
(RequestUriPathToken2 = chat.postMessage) => ResponseBody_ok=true (3985.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_ok=true (3118.0/0.0)

(HasSuccessfulDeleteOperationOccurredBefore = false) and
(HasImmediatePreviousTransactionSucceeded = true) => ResponseBody_ok=true
    (540.0/0.0)

=> ResponseBody_ok=false (9779.0/0.0)
```

Figure 6.23: The Sample RIPPER Ruleset for ResponseBody_ok in Slack

```
HasSuccessfulDeleteOperationOccurredBefore = true: false (8929.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToUpdate = false AND
RequestUriPathToken2 = chat.postMessage: true (3985.0)

HasImmediatePreviousTransactionSucceeded = true: true (3658.0)

: false (850.0)
```

Figure 6.24: The Sample PART Ruleset for ResponseBody_ok in Slack

```
HasSuccessfulDeleteOperationOccurredBefore = false
|   HasURLInImmediatelyPreviousTransactionContainsATokenToUpdate = false
|   |   RequestUriPathToken2 = chat.update
|   |   |   HasImmediatePreviousTransaction = false: not-exist (309.0)
|   |   |   HasImmediatePreviousTransaction = true: CCGRWTRKQ (1292.0)
|   |   RequestUriPathToken2 = chat.delete
|   |   |   HasImmediatePreviousTransaction = false: not-exist (206.0)
|   |   |   HasImmediatePreviousTransaction = true: CCGRWTRKQ (1826.0)
|   |   RequestUriPathToken2 = chat.postMessage: CCGRWTRKQ (3985.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToUpdate = true
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (335.0)
|   |   HasImmediatePreviousTransactionSucceeded = true: CCGRWTRKQ (540.0)
HasSuccessfulDeleteOperationOccurredBefore = true: not-exist (8929.0)
```

Figure 6.25: The Sample C4.5 Tree for ResponseBody_channel in Slack

sion and recall) and the inferred semantic models are smaller and simple (more convenient to interpret). As an instance, for the target ResponseBody_channel containing 2 distinct values: CCGRWTRKQ and not-exist, C4.5 produces a tree with size 8 (including 5 leaves), while RIPPER and PART generate rulesets of 4 rules. Figures 6.25–6.27 depict the sample models induced by C4.5, RIPPER, and PART. The numbers in brackets at the end of each leaf/rule denote the total number of instances classified into the particular leaf/rule and the misclassified instances. This target reflects the ResponseBody_channel property in the response body that lists the channel ID where requests have been directed to, which always happens to be UC8J6APLN for successful requests in the Slack dataset. We observe that the sample models express exactly the same semantics as the ResponseBody_ok because the ResponseBody_channel values are based on the response state. Once again, the models illustrate good relevance and interpretability.

Therefore, it can be concluded that C4.5, RIPPER, and PART are capable of obtaining promising predictive performance results predicting all of the main response features in the Slack dataset and clearly outperforming model size fa-

```
(HasSuccessfulDeleteOperationOccurredBefore = false) and
(RequestUriPathToken2 = chat.postMessage) => ResponseBody_channel=CCGRWTRKQ
    (3985.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_channel=CCGRWTRKQ (3118.0/0.0)

(HasSuccessfulDeleteOperationOccurredBefore = false) and (
    HasImmediatePreviousTransactionSucceeded = true) => ResponseBody_channel=
    CCGRWTRKQ (540.0/0.0)

=> ResponseBody_channel=not-exist (9779.0/0.0)
```

Figure 6.26: The Sample RIPPER Ruleset for `ResponseBody_channel` in Slack

```
HasSuccessfulDeleteOperationOccurredBefore = true: not-exist (8929.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToUpdate = false AND
RequestUriPathToken2 = chat.postMessage: CCGRWTRKQ (3985.0)

HasImmediatePreviousTransactionSucceeded = true: CCGRWTRKQ (3658.0)

: not-exist (850.0)
```

Figure 6.27: The Sample PART Ruleset for `ResponseBody_channel` in Slack

vouring comprehensibility, with the exception of one header field specific to the data values returned from the server.

6.2.2 Description Logic Learning Approach

In this section, OCEL findings are discussed based upon different datasets. With each dataset, the results are analysed with respect to the target classes corresponding to the values of the different attributes associated with the main HTTP response features (i.e., status, headers, and body).

Results on GHTraffic Dataset

Table 6.9 summarises the number of positive and negative examples for each of the target class in the GHTraffic knowledge base and shows the experimental results for each target on OCEL. The generated sample class definitions can be found in Appendix C.4.1. Appendix B.4.1 provides a complete list of classes in the Protege environment.

As stated in Section 6.2.1, the target attributes associated with the response status in the GHTraffic dataset are `ResponseStatusCode`, `ResponseBody_message`, and `ResponseBody_documentation_url`. Thereby, a number of target classes referring to the distinct values of these attributes are available in the background knowledge base of the GHTraffic dataset. As shown in the experimental results, for the majority of status-related target classes, the inferred OCEL descriptions attain high to excellent predictive performance and are shorter. Consider, for example, the target classes derived from the attribute `ResponseStatusCode` (`ResponseStatusCode_200`, `ResponseStatusCode_201`, `ResponseStatusCode_204`, `ResponseStatusCode_400`, `ResponseStatusCode_401`, `ResponseStatusCode_404`, `ResponseStatusCode_422`, and `ResponseStatusCode_500`). Among these targets,

```
ResponseStatusCode_204:
RequestHeader_HasAuthorisationToken_true and
RequestUriPathToken6_lock and
(isPrecededBy some ResponseStatusCode_201)

ResponseStatusCode_400:
RequestHeader_HasAuthorisationToken_true and
RequestHeader_HasRequestPayload_true
and RequestHeader_HasValidRequestPayload_false

ResponseStatusCode_401:
RequestHeader_HasAuthorisationToken_false

ResponseStatusCode_422:
RequestHeader_HasRequestPayload_false and
RequestMethod_PATCH

ResponseStatusCode_200:
(RequestHeader_HasValidRequestPayload_true or
(RequestUriPathToken6_not-exist and
(not (RequestMethod_PATCH)))) and
(isPrecededBy some ResponseStatusCode_201)
```

Figure 6.28: The Best OCEL Descriptions for Some `ResponseStatusCode` Values in GHTraffic

the `ResponseStatusCode_201` and `ResponseStatusCode_500` have no positive examples and are ignored when generating models, as the use of OCEL is confined to learning problems where there are both positive and negative examples. Out of the others, `ResponseStatusCode_204` (with 56 positive and 321 negative examples), `ResponseStatusCode_400` (with 38 positive and 339 negative examples), `ResponseStatusCode_401` (with 98 positive and 279 negative examples), and `ResponseStatusCode_422` (with 23 positive and 354 negative examples) are targets where OCEL produces 100% accurate classification models by covering all positive examples and zero negative cases. Each of these classifiers is 7, 5, 1, and 3 in length, respectively. And for the `ResponseStatusCode 200` with 139 positives and 238 negatives, OCEL almost reaches maximum accuracy. The definition learned by OCEL fully defines positive examples with 6 more negative (false positives) cases. This leads to a predictive accuracy of 0.9841 as shown in Table 6.9, together with a precision of 0.9605 and a recall of 1.0. The length of the definition produced is 10. In this context, the resultant classifier appears to be rather general, i.e., since OCEL is a top-down algorithm, it could not generate more specialised descriptions before the timeout (120 seconds) and thus accepts some false positives. This is because some examples include past transactions with HTTP `500` (Internal Server Error) indicating unexpected server behaviour (and as stated in Section 6.2.1, it is more challenging to model the exact relations from such a knowledge base), while some others include a small number of instances describing different behavioural patterns (no sufficient instances to learn about certain transaction sequences). However, it is obvious that all these learning problems are simple queries that require less reasoning, leading to top scores. The remaining class `ResponseStatusCode_404` with 23 positive and 354 negative examples is one of the most difficult learning problems in the GHTraffic dataset. The inferred OCEL model is more general and describes 19 positive examples which also include 121 negative ones (4 false negatives and 121 false positives). It hardly reaches 0.6687 predictive accuracy, 0.1456 precision, and 0.8258 recall. Although there are few positive examples, it can be observed that they reflect a variety

Table 6.9: Results of Description Logic Learning Algorithm per Response Feature Value (Target) in GHTraffic. The number of positive and negative examples is also shown for each target concept.

Class	Positives	Negatives	Accuracy	Precision	Recall	Length
ResponseStatusCode_200	139	238	0.9841	0.9605	1.0000	10
ResponseStatusCode_204	56	321	1.0000	1.0000	1.0000	7
ResponseStatusCode_400	38	339	1.0000	1.0000	1.0000	5
ResponseStatusCode_401	98	279	1.0000	1.0000	1.0000	1
ResponseStatusCode_404	23	354	0.6687	0.1456	0.8258	12
ResponseStatusCode_422	23	354	1.0000	1.0000	1.0000	3
ResponseHeader_Cache-Control_max-age	139	238	0.9841	0.9605	1.0000	10
ResponseHeader_Cache-Control_not-exist	238	139	0.7005	0.6821	1.0000	10
ResponseHeader_Vary_accept	139	238	0.9841	0.9605	1.0000	10
ResponseHeader_Vary_not-exist	238	139	0.7005	0.6821	1.0000	10
ResponseHeader_X-Accepted-OAuth-Scopes_not-exist	98	279	1.0000	1.0000	1.0000	1
ResponseHeader_X-Accepted-OAuth-Scopes_accepted-public-repo	195	182	1.0000	1.0000	1.0000	10
ResponseHeader_X-Accepted-OAuth-Scopes_repo	84	293	0.5251	0.3156	0.9643	1
ResponseHeader_X-OAuth-Scopes_not-exist	98	279	1.0000	1.0000	1.0000	1
ResponseHeader_X-OAuth-Scopes_public-repo	279	98	1.0000	1.0000	1.0000	1
ResponseBody_assignee.site_admin_false	17	360	1.0000	1.0000	1.0000	13
ResponseBody_assignee.site_admin_not-exist	360	17	0.9656	0.9728	0.9917	5
ResponseBody_assignee.type_not-exist	360	17	0.9656	0.9728	0.9917	5
ResponseBody_assignee.type_User	17	360	1.0000	1.0000	1.0000	13
ResponseBody_assignees.site_admin_false	11	366	1.0000	1.0000	1.0000	6
ResponseBody_assignees.site_admin_not-exist	366	11	0.9788	0.9917	0.9863	5
ResponseBody_assignees.type_not-exist	366	11	0.9788	0.9917	0.9863	5
ResponseBody_assignees.type_User	11	366	1.0000	1.0000	1.0000	6
ResponseBody_documentation_url_edit-an-issue	92	285	1.0000	1.0000	1.0000	3
ResponseBody_documentation_url_lock-an-issue	46	331	1.0000	1.0000	1.0000	3
ResponseBody_documentation_url_not-exist	199	178	0.9655	0.9389	1.0000	12
ResponseBody_documentation_url_unlock-an-issue	21	356	1.0000	1.0000	1.0000	3
ResponseBody_documentation_url_v3	178	199	0.6151	0.5586	0.9663	10
ResponseBody_locked_false	109	268	1.0000	1.0000	1.0000	9
ResponseBody_locked_not-exist	268	109	0.7721	0.7715	0.9776	7
ResponseBody_message_invalid-request	23	354	1.0000	1.0000	1.0000	3
ResponseBody_message_not-exist	199	178	0.9655	0.9389	1.0000	12
ResponseBody_message_not-found	19	358	0.8219	0.5328	0.4722	12

Continued on next page

Table 6.9 – continued from previous page

Class	Positives	Negatives	Accuracy	Precision	Recall	Length
ResponseBody_message_problems-passing-json	38	339	1.0000	1.0000	1.0000	5
ResponseBody_message_requires-authentication	98	279	1.0000	1.0000	1.0000	1
ResponseBody_milestone.creator.avatar_url_avatar-v3	16	361	1.0000	1.0000	1.0000	13
ResponseBody_milestone.creator.avatar_url_not-exist	361	16	0.9709	0.9706	1.0000	5
ResponseBody_milestone.creator.events_url_events-privacy	16	361	1.0000	1.0000	1.0000	13
ResponseBody_milestone.creator.events_url_not-exist	361	16	0.9709	0.9706	1.0000	5
ResponseBody_milestone.creator.followers_url_followers-cgdecker	16	361	1.0000	1.0000	1.0000	13
ResponseBody_milestone.creator.followers_url_not-exist	361	16	0.9709	0.9706	1.0000	5
ResponseBody_milestone.creator.following_url_following-cgdecker	16	361	1.0000	1.0000	1.0000	13
ResponseBody_milestone.creator.following_url_not-exist	361	16	0.9709	0.9706	1.0000	5
ResponseBody_milestone.creator.gists_url_gists-cgdecker	16	361	1.0000	1.0000	1.0000	13
ResponseBody_milestone.creator.gists_url_not-exist	361	16	0.9709	0.9706	1.0000	5
ResponseBody_milestone.creator.html_url_html-cgdecker	16	361	1.0000	1.0000	1.0000	13
ResponseBody_milestone.creator.html_url_not-exist	361	16	0.9709	0.9706	1.0000	5
ResponseBody_milestone.creator.id_101568	16	361	1.0000	1.0000	1.0000	13
ResponseBody_milestone.creator.id_not-exist	361	16	0.9709	0.9706	1.0000	5
ResponseBody_milestone.creator.login_cgdecker	16	361	1.0000	1.0000	1.0000	13
ResponseBody_milestone.creator.login_not-exist	361	16	0.9709	0.9706	1.0000	5
ResponseBody_milestone.creator.organizations_url_not-exist	361	16	0.9709	0.9706	1.0000	5
ResponseBody_milestone.creator.organizations_url_organizations-cgdecker	16	361	1.0000	1.0000	1.0000	13
ResponseBody_milestone.creator.received_events_url_events-cgdecker	16	361	1.0000	1.0000	1.0000	13
ResponseBody_milestone.creator.received_events_url_not-exist	361	16	0.9709	0.9706	1.0000	5
ResponseBody_milestone.creator.repos_url_not-exist	361	16	0.9709	0.9706	1.0000	5
ResponseBody_milestone.creator.repos_url_repos-cgdecker	16	361	1.0000	1.0000	1.0000	13
ResponseBody_milestone.creator.site_admin_false	16	361	1.0000	1.0000	1.0000	13
ResponseBody_milestone.creator.site_admin_not-exist	361	16	0.9709	0.9706	1.0000	5
ResponseBody_milestone.creator.starred_url_not-exist	361	16	0.9709	0.9706	1.0000	5
ResponseBody_milestone.creator.starred_url_starred-cgdecker	16	361	1.0000	1.0000	1.0000	13
ResponseBody_milestone.creator.subscriptions_url_not-exist	361	16	0.9709	0.9706	1.0000	5
ResponseBody_milestone.creator.subscriptions_url_subscriptions-cgdecker	16	361	1.0000	1.0000	1.0000	13
ResponseBody_milestone.creator.type_not-exist	361	16	0.9709	0.9706	1.0000	5
ResponseBody_milestone.creator.type_User	16	361	1.0000	1.0000	1.0000	13
ResponseBody_milestone.creator.url_creator-cgdecker	16	361	1.0000	1.0000	1.0000	13
ResponseBody_milestone.creator.url_not-exist	361	16	0.9709	0.9706	1.0000	5
ResponseBody_milestone.due_on_not-exist	361	16	0.9709	0.9706	1.0000	5

Continued on next page

Table 6.9 – continued from previous page

Class	**Positives**	**Negatives**	**Accuracy**	**Precision**	**Recall**	**Length**
ResponseBody_milestone.due_on_null	16	361	1.0000	1.0000	1.0000	13
ResponseBody_milestone.open_issues_0	9	368	0.9974	1.0000	0.9974	12
ResponseBody_milestone.open_issues_1	7	370	0.9974	1.0000	0.8750	5
ResponseBody_milestone.open_issues_not-exist	361	16	0.9709	0.9706	1.0000	5
ResponseBody_milestone.state_closed	16	361	1.0000	1.0000	1.0000	13
ResponseBody_milestone.state_not-exist	361	16	0.9709	0.9706	1.0000	5
ResponseBody_state_closed	65	312	0.9256	0.7759	0.9697	13
ResponseBody_state_not-exist	268	109	0.7721	0.7715	0.9776	7
ResponseBody_state_open	44	333	0.9576	0.7497	0.9546	15
ResponseBody_user.site_admin_false	109	268	1.0000	1.0000	1.0000	10
ResponseBody_user.site_admin_not-exist	268	109	0.7721	0.7715	0.9776	7
ResponseBody_user.type_not-exist	268	109	0.7721	0.7715	0.9776	7
ResponseBody_user.type_User	109	268	1.0000	1.0000	1.0000	11
Mean			0.9533	0.9341	0.9851	7.9753
Standard Deviation			0.0972	0.1506	0.0637	4.1261

of behavioural patterns that trigger HTTP 404 (meaning that there are not adequate instances to represent each specific transaction sequence). Some examples include past transaction sequences containing HTTP 500 (which make it more complex to trace the correlations). Such that the algorithm is unable to produce more specialised class descriptions that capture the respective behavioural patterns prior to the timeout. It is clear that the algorithm performs poorly when the data is varied and when contains unexpected service behaviour. Figure 6.28 shows the best descriptions produced for the target classes `ResponseStatusCode_204`, `ResponseStatusCode_400`, `ResponseStatusCode_401`, `ResponseStatusCode_422`, `ResponseStatusCode_200` in the Manchester OWL Syntax. Take into account, for example, the sample class description for `ResponseStatusCode_204` which encodes the semantics of HTTP 204 (`No Content`) status code (indicates the successful locking or unlocking of an issue in the GitHub's Issue Tracking system). The rule stipulates that the response status code for a request will be HTTP 204 if the request is made using a valid authentication token and the request URI path carries the `lock` token at its sixth position, and the request has a preceding HTTP 201 transaction for that particular resource (means that issue has already been successfully created). Another example is that the class description for `ResponseStatusCode_400` which encodes the semantics of HTTP 400 (`Bad Request`) status code (indicates that the server is unable to handle the request due to malformed message syntax) as if the request is made using a valid authentication token and has a request payload which does not have correctly encoded values. The samples demonstrate that the generated relational concept descriptions are very simple to understand and describe well each scenario. It can also be observed that the induced knowledge exchanges similarities with knowledge derived from attribute-based learning algorithms as described in Section 6.2.1. Clearly, the class expressions produced can easily be used to understand the exact causes of particular status codes.

On the other hand, the background knowledge base of the GHTraffic dataset contains target classes for attribute values associated with the response headers. Among these are target classes with only positive examples (the target attribute has just one distinct value) that are ignored when constructing models. The `ResponseHeader_Server_GitHub.com`, `ResponseHeader_Content-Type_json`, and `ResponseHeader_X-GitHub-Media-Type_v3-json` are examples of this. However, we can see that most of the other target classes related to the response headers in the GHTraffic knowledge base achieve high to maximum predictive performance statistics on OCEL. The generated classifiers are also shorter. As example, for the target classes `ResponseHeader_X-OAuth-Scopes_public_repo` (including 279 positive and 98 negative examples) and `ResponseHeader_X-OAuth-Scopes_not-exist` (including 98 positive and 279 negative examples) obtained from the attribute `ResponseHeader_X-OAuth-Scopes`, OCEL produces class descriptions of length 1 which are 100% accurate (result in 1.0 predictive accuracy, precision, and recall). Figure 6.29 depicts the best-learned OCEL class expressions. As can be seen, the `ResponseHeader_X-OAuth-Scopes_public_repo` description specifies that if the request is made with a valid authorisation token, the response will contain the `ResponseHeader_X-OAuth-Scopes` header with the value `public_repo`. In contrast, the description for `ResponseHeader_X-OAuth-Scopes_not-exist` specifies that if the request is made without a valid authorisation token, there will be no

```
ResponseHeader_X-OAuth-Scopes_public_repo:
RequestHeader_HasAuthorisationToken_true

ResponseHeader_X-OAuth-Scopes_not-exist:
RequestHeader_HasAuthorisationToken_false
```

Figure 6.29: The Best OCEL Descriptions for `ResponseHeader_X-OAuth-Scopes` Values in GHTraffic

`ResponseHeader_X-OAuth-Scopes` header. It is evident that the semantics of the header values of `X-OAuth-Scopes` can be correctly and easily interpreted by these definitions. We also note that the models express exactly the same knowledge that is learned from attribute-based learning algorithms. Besides, for some targets relevant to the response headers values, such as the `ResponseHeader_Vary_not-exist` related to the `ResponseHeader_Vary`, the generated concept definitions are not sufficiently specialised to fully describe all the positive examples. Many of the negatives are also covered. These learning problems are difficult to learn as the background knowledge base and certain examples represent HTTP `500` (unpredictable server behaviour). The lack of examples describing some of the transaction sequences also makes it impossible to generate perfectly accurate class expressions.

Moreover, there is a wide range of target classes associated with the attribute values of the response body. Many have only positive or negative examples and are thus ignored from learning definitions. Table 6.9 reveals that there are many other target classes for which the inferred OCEL models are highly accurate and less sophisticated. Even so, it can be observed that, in all these cases, OCEL is able to learn the mapping for attribute values of the response body from just the values associated with the incoming request features and the non-state-related response body features of the previous transactions (however, as discussed in Section 2.1.3, the HTTP response message should depend on the values of the request and the state inferred from transaction history). This is probably because the values related to the features of interaction status history are not relevant to the learning problem. These target classes can be observed to be correlated to the data values returned by the server. There is, therefore, no potential relation between the target classes and the internal service status, for which the same happens with attribute-based learning. Consequently, such target classes are not optimal for predictions. Despite all these, there are few target classes, such as `ResponseBody_state_open`, `ResponseBody_state_closed`, and `ResponseBody_state_not-exist`, for which the algorithm produces precise and simple descriptions. See Appendix C.4.1 for the best-discovered class descriptions.

In view of the results, it can be concluded that OCEL is capable of learning highly accurate and human-readable concepts for targets associated with the key response feature values in the GHTraffic dataset, apart from targets with complex learning problems (when examples are diverse and/or are linked to past transactions with unexpected server behaviour) and for targets linked to the content of the response body that are correlated with the data values returned by the server where there is no relation between the targets and the features of the request and the service state history.

Table 6.10: Results of Description Logic Learning Algorithm per Response Feature Value (Target) in Twitter. The number of positive and negative examples is also shown for each target concept.

Class	**Positives**	**Negatives**	**Accuracy**	**Precision**	**Recall**	**Length**
ResponseStatusCode_200	156	348	0.9901	0.9699	1.0000	10
ResponseStatusCode_404	348	156	0.9980	1.0000	0.9971	9
ResponseHeader_status_200OK	156	348	0.9901	0.9699	1.0000	10
ResponseHeader_status_404NotFound	348	156	0.9980	1.0000	0.9971	9
ResponseHeader_x-rate-limit-limit_900	93	411	1.0000	1.0000	1.0000	1
ResponseHeader_x-rate-limit-limit_not-exist	411	93	1.0000	1.0000	1.0000	1
ResponseBody_contributors_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_contributors_null	156	348	0.9901	0.9699	1.0000	10
ResponseBody_coordinates_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_coordinates_null	156	348	0.9901	0.9699	1.0000	10
ResponseBody_entities.hashtags_empty-list	156	348	0.9901	0.9699	1.0000	10
ResponseBody_entities.hashtags_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_entities.symbols_empty-list	156	348	0.9901	0.9699	1.0000	10
ResponseBody_entities.symbols_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_entities.urls_empty-list	156	348	0.9901	0.9699	1.0000	10
ResponseBody_entities.urls_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_entities.user_mentions_empty-list	156	348	0.9901	0.9699	1.0000	10
ResponseBody_entities.user_mentions_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_errors.code_144	348	156	0.9980	1.0000	0.9971	9
ResponseBody_errors.code_not-exist	156	348	0.9901	0.9699	1.0000	10
ResponseBody_errors.message_no-status-found	348	156	0.9980	1.0000	0.9971	9
ResponseBody_errors.message_not-exist	156	348	0.9901	0.9699	1.0000	10
ResponseBody_favorite_count_0	156	348	0.9901	0.9699	1.0000	10
ResponseBody_favorite_count_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_favorited_false	156	348	0.9901	0.9699	1.0000	10
ResponseBody_favorited_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_geo_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_geo_null	156	348	0.9901	0.9699	1.0000	10
ResponseBody_in_reply_to_screen_name_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_in_reply_to_screen_name_null	156	348	0.9901	0.9699	1.0000	10
ResponseBody_in_reply_to_status_id_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_in_reply_to_status_id_null	156	348	0.9901	0.9699	1.0000	10
ResponseBody_in_reply_to_status_id_str_not-exist	348	156	0.9980	1.0000	0.9971	9

Continued on next page

Table 6.10 – continued from previous page

Class	Positives	Negatives	Accuracy	Precision	Recall	Length
ResponseBody_in_reply_to_status_id_str_null	156	348	0.9901	0.9699	1.0000	10
ResponseBody_in_reply_to_user_id_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_in_reply_to_user_id_null	156	348	0.9901	0.9699	1.0000	10
ResponseBody_in_reply_to_user_id_str_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_in_reply_to_user_id_str_null	156	348	0.9901	0.9699	1.0000	10
ResponseBody_is_quote_status_false	156	348	0.9901	0.9699	1.0000	10
ResponseBody_is_quote_status_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_place_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_place_null	156	348	0.9901	0.9699	1.0000	10
ResponseBody_retweet_count_0	156	348	0.9901	0.9699	1.0000	10
ResponseBody_retweet_count_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_retweeted_false	156	348	0.9901	0.9699	1.0000	10
ResponseBody_retweeted_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_truncated_false	156	348	0.9901	0.9699	1.0000	10
ResponseBody_truncated_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.contributors_enabled_false	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.contributors_enabled_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.created_at_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.created_at_WedMar0709	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.default_profile_false	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.default_profile_image_false	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.default_profile_image_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.default_profile_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.favourites_count_64	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.favourites_count_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.follow_request_sent_false	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.follow_request_sent_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.followers_count_185	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.followers_count_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.following_false	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.following_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.friends_count_249	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.friends_count_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.geo_enabled_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.geo_enabled_true	156	348	0.9901	0.9699	1.0000	10

Continued on next page

Table 6.10 – continued from previous page

Class	Positives	Negatives	Accuracy	Precision	Recall	Length
ResponseBody_user.has_extended_profile_false	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.has_extended_profile_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.id_517417816	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.id_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.id_str_517417816	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.id_str_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.is_translation_enabled_false	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.is_translation_enabled_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.is_translator_false	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.is_translator_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.lang_en	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.lang_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.listed_count_3	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.listed_count_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.location_Kurunegala	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.location_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.name_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.name_ThiliniBhagya	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.notifications_false	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.notifications_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.profile_background_color_1A1B1F	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.profile_background_color_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.profile_background_image_url_http-abc.twimg.com	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.profile_background_image_url_https_https-abc.twimg.com	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.profile_background_image_url_https_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.profile_background_image_url_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.profile_background_tile_false	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.profile_background_tile_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.profile_banner_url_https-pbs.twimg.com-profile-banners	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.profile_banner_url_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.profile_image_url_http-pbs.twimg.com-profile-images	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.profile_image_url_https_https-pbs.twimg.com-profile-images	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.profile_image_url_https_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.profile_image_url_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.profile_link_color_3E4547	156	348	0.9901	0.9699	1.0000	10

Continued on next page

Table 6.10 – continued from previous page

Class	**Positives**	**Negatives**	**Accuracy**	**Precision**	**Recall**	**Length**
ResponseBody_user.profile_link_color_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.profile_sidebar_border_color_FFFFFF	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.profile_sidebar_border_color_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.profile_sidebar_fill_color_252429	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.profile_sidebar_fill_color_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.profile_text_color_666666	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.profile_text_color_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.profile_use_background_image_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.profile_use_background_image_true	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.protected_false	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.protected_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.screen_name_bhagyasl	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.screen_name_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.time_zone_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.time_zone_null	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.translator_type_none	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.translator_type_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.url_http-t.co	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.url_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.utc_offset_not-exist	348	156	0.9980	1.0000	0.9971	9
ResponseBody_user.utc_offset_null	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.verified_false	156	348	0.9901	0.9699	1.0000	10
ResponseBody_user.verified_not-exist	348	156	0.9980	1.0000	0.9971	9
Mean			0.9941	0.9852	0.9986	9.3651
Standard Deviation			0.0040	0.0151	0.0015	1.1771

Results on Twitter Dataset

The number of positive and negative examples together with experimental results from OCEL for each target class in the Twitter knowledge base are shown in Table 6.10. The sample classifiers can be found in Appendix C.4.2. Appendix B.4.2 provides the full list of classes in the Protege environment.

As defined in Section 6.2.1, the following are the target attributes related to the response status in the dataset: `ResponseStatusCode`, `ResponseHeader_status`, `ResponseBody_errors.message`, and `ResponseBody_errors.code`. And as such, the background knowledge base of the Twitter dataset contains separate target classes for each distinct value of those attributes. The findings in Table 6.10 demonstrate that the trained class descriptions by OCEL for the target classes associated with the response status obtain near-perfect predictive performance scores and are shorter and less complex. Take into account, for example, the `ResponseStatusCode_200` and `ResponseStatusCode_404` targets derived from the `ResponseStatusCode` attribute. It can be observed that the OCEL classifier for the `ResponseStatusCode_200` (with 156 positive examples and 348 negative examples) achieves 0.9901 predictive accuracy by covering all positive ones including 5 of the negatives (false positives). This also has 0.9699 precision and 1.0 recall. The length of the definition produced is 10. In a similar way, for the target `ResponseStatusCode_404` with 348 positive and 156 negative examples, the OCEL classifier reaches 0.998 predictive accuracy by retrieving 347 positives (1 false negative) and not capturing any negative, and the precision and recall rates are 1.0 and 0.9971, respectively. It has a length of 9. Both are simple learning problems. However, we can see that the algorithm can not completely locate a class description that defines all the positive and none of the negative examples before the timeout. This means that the generated concepts are not successively specialised. That is because some of the transaction sequences are not represented by sufficient examples in the training dataset. As a result, the algorithm is unable to specialise the solutions until it includes all positive examples but no negative ones. Figure 6.30 depicts the best OCEL class definitions induced for `ResponseStatusCode_200` and `ResponseStatusCode_404` (in Manchester OWL syntax). The class description for `ResponseStatusCode_200` describes the semantics of HTTP 200 (`OK`) status code (indicating the successful creation/retrieval/deletion of a tweet) as if the request contains the `Content-Type` header with the value `form-urlencoded` (which suggests the semantics of successful tweet creation) or if the request does not contain the `Content-Type` header and the request has an immediately preceding HTTP `200` transaction for that particular resource where the request URI path does not carry the `destroy` token at its third position (means that a retrieval or deletion of a tweet will be successful if the request has a successful transaction immediately preceding it without a successful deletion). It is evident from the samples that the resulting relational concepts are very easy to comprehend and explain well each scenario. It can also be found that the inferred semantic knowledge has parallels with the knowledge extracted from the attribute-based learning algorithms (described in Section 6.2.1). Apparently, the class expressions generated can easily be used to obtain accurate insights into the exact causes that lead to different status codes.

Furthermore, several of the target classes related to the distinct values of the

```
ResponseStatusCode_200:
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))

ResponseStatusCode_404:
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

Figure 6.30: The Best OCEL Descriptions for `ResponseStatusCode` Values in Twitter

```
ResponseHeader_x-rate-limit-limit_900:
RequestMethod_GET

ResponseHeader_x-rate-limit-limit_not-exist:
RequestMethod_POST
```

Figure 6.31: The Best OCEL Descriptions for `ResponseHeader_x-rate-limit-limit` Values in Twitter

response headers, such as the `ResponseHeader_content-type_json`, are limited to positive examples, so that they are excluded from the learning process. There are also target classes for the response header values that represent the status information and, as discussed above, the OCEL algorithm is able to produce highly accurate and readily understandable classification models. Other than that, the only remaining targets are the `ResponseHeader_x-rate-limit-limit_900` and `ResponseHeader_x-rate-limit-limit_not-exist`, which are aligned with the values of the `x-rate-limit-limit` response header. The results indicate that for all these targets, the class definitions induced by OCEL have the maximum degree of predictive performance (1.0 predictive accuracy, precision, and recall) and are shorter in length, thus, potentially perfectly reliable and comprehensible. The best class expressions obtained are shown in Figure 6.31. As can be seen, the `ResponseHeader_x-rate-limit-limit_900` description states that if the request is made using the `GET` method, the response will include the header `x-rate-limit-limit` with the value `900`. Conversely, the other expression states that if the request is made using the `POST`, there will be no `x-rate-limit-limit` header in the response. It is evident that the inferred definitions explicitly lead to an exact and simpler interpretation of the semantics of the header values. It is also noticeable that the knowledge learned has similarities with the knowledge obtained from attribute-based learning algorithms as defined in Section 6.2.1.

The Twitter knowledge base also contains a fairly large number of target classes associated with the attribute values of the response body. Some target classes correspond to the state of response as previously mentioned, and OCEL is clearly capable of providing highly accurate classification models that can be easily comprehended. According to Table 6.10, even for all the remaining targets related to the response body values, the generated OCEL expressions attain higher predictive performance and are smaller as well. These target classes, however, are found to be correlated with the data values retrieved by the server (similar to those identified in the attribute-based learning). This means that there is also no possible correlation between the targets and the values of the request and the internal service status.

```
ResponseStatusCode_200:
(not (RequestMethod_DELETE)) and
(hasPrevious some ResponseStatusCode_200)

ResponseStatusCode_204:
RequestMethod_DELETE and
(hasPrevious some ResponseStatusCode_200)

ResponseStatusCode_404:
isPrecededBy some ResponseStatusCode_204
```

Figure 6.32: The Best OCEL Descriptions for Some `ResponseStatusCode` Values in Google Tasks

At the same time, as we discussed in Section 6.2.1, the targets are observed to be dependent on the success or failure of the request. The findings demonstrate the point very clearly. In all these cases, the inferred OCEL models are equivalent to those inferred for the `ResponseStatusCode_200` and `ResponseStatusCode_404`. Obviously, the generated models are accurate and readable and correctly represent this particular scenario in the dataset. See Appendix C.4.2 for the best class expressions learned for each target in Manchester OWL syntax.

In all, the results demonstrate that the OCEL algorithm is significantly outperformed in training class expressions that are both accurate and human-readable for all targets related to the values of the core features of HTTP responses in the Twitter dataset. It is clear that the algorithm is much better at answering simple queries that require less reasoning. However, having a limited number of examples representing certain behavioural patterns hinders the algorithm from learning perfectly correct definitions.

Results on Google Tasks Dataset

Table 6.11 outlines the number of positive and negative examples and the results produced by OCEL for each target classes in the Google Tasks knowledge base. See Appendix C.4.3 for the sample class definitions. A complete classes list found in the knowledge base (in the Protege environment) is given in Appendix B.4.3.

As specified in Section 6.2.1, the Google Tasks dataset contains multiple target attributes related to the response state. Like, for example, `ResponseStatusCode`, `ResponseBody_error.code`, and `ResponseBody_error.message`. As a result, the background knowledge base of the Google Tasks dataset includes multiple target classes relating to the distinct values of those attributes. As can be seen in Table 6.11, the class descriptions induced by OCEL for most of the target classes associated with the response status have higher predictive performance and are shorter, and thus more reliable and readable. For example, consider the targets `ResponseStatusCode_200`, `ResponseStatusCode_204`, `ResponseStatusCode_404`, and `ResponseStatusCode_503` related to the `ResponseStatusCode`. For the target `ResponseStatusCode_200` with 517 positive and 607 negative examples, the generated classifier scores 0.9884 predictive accuracy by covering 514 positive examples and 10 negatives (false positives), and the precision and recall rates are 0.9813 and 0.9942, respectively. The definition length is 6. The OCEL classifier for `ResponseStatusCode_204` (with 100 positive and 1124 negative examples) obtains 0.9947 predictive accuracy, 0.9546 precision, and 0.99 recall by covering 99

Table 6.11: Results of Description Logic Learning Algorithm per Response Feature Value (Target) in Google Tasks. The number of positive and negative examples is also shown for each target concept.

Class	**Positives**	**Negatives**	**Accuracy**	**Precision**	**Recall**	**Length**
ResponseStatusCode_200	517	607	0.9884	0.9813	0.9942	6
ResponseStatusCode_204	100	1024	0.9947	0.9545	0.9900	8
ResponseStatusCode_404	505	619	0.9991	0.9980	1.0000	3
ResponseStatusCode_503	2	1122	0.9893	0.0000	0.0000	17
ResponseHeader_Accept-Ranges_none	570	554	0.9973	1.0000	0.9947	5
ResponseHeader_Accept-Ranges_not-exist	554	570	0.9991	1.0000	0.9982	5
ResponseHeader_Cache-Control_must-revalidate	454	670	1.0000	1.0000	1.0000	5
ResponseHeader_Cache-Control_no-cache-must-revalidate	163	961	0.9056	0.6674	0.9938	7
ResponseHeader_Cache-Control_private	507	617	0.9227	0.8735	0.9980	3
ResponseHeader_Content-Type_json	1024	100	0.9555	0.9555	1.0000	6
ResponseHeader_Content-Type_not-exist	100	1024	0.9947	0.9546	0.9900	5
ResponseHeader_Pragma_no-cache	163	961	0.9056	0.6674	0.9938	7
ResponseHeader_Pragma_not-exist	961	163	0.8862	0.8833	1.0000	5
ResponseHeader_Transfer-Encoding_chunked	570	554	0.9973	1.0000	0.9947	5
ResponseHeader_Transfer-Encoding_not-exist	554	570	0.9991	1.0000	0.9982	5
ResponseHeader_Vary_origin	570	554	0.9973	1.0000	0.9947	5
ResponseHeader_Vary_X-Origin	554	570	0.9991	1.0000	0.9982	5
ResponseHeader_X-Content-Type-Options_nosniff	1024	100	0.9555	0.9555	1.0000	6
ResponseHeader_X-Content-Type-Options_not-exist	100	1024	0.9947	0.9546	0.9900	5
ResponseHeader_X-Frame-Options_not-exist	100	1024	0.9947	0.9546	0.9900	5
ResponseHeader_X-Frame-Options_SAMEORIGIN	1024	100	0.9555	0.9555	1.0000	6
ResponseHeader_X-XSS-Protection_block	1024	100	0.9555	0.9555	1.0000	6
ResponseHeader_X-XSS-Protection_not-exist	100	1024	0.9991	1.0000	0.9900	5
ResponseBody_error.code_404	505	619	0.9991	0.9980	1.0000	3
ResponseBody_error.code_503	2	1122	0.9893	0.0000	0.0000	17
ResponseBody_error.code_not-exist	617	507	0.9876	0.9797	0.9984	8
ResponseBody_error.errors.domain_global	507	617	0.9227	0.8735	0.9980	3
ResponseBody_error.errors.domain_not-exist	617	507	0.9876	0.9797	0.9984	8
ResponseBody_error.errors.message_BackendError	2	1122	0.9893	0.0000	0.0000	17
ResponseBody_error.errors.message_not-exist	617	507	0.9876	0.9797	0.9984	8
ResponseBody_error.errors.message_NotFound	505	619	0.9991	0.9980	1.0000	3
ResponseBody_error.errors.reason_backendError	2	1122	0.9893	0.0000	0.0000	17
ResponseBody_error.errors.reason_not-exist	617	507	0.9876	0.9797	0.9984	8

Continued on next page

Table 6.11 – continued from previous page

Class	**Positives**	**Negatives**	**Accuracy**	**Precision**	**Recall**	**Length**
ResponseBody_error.errors.reason_notFound	505	619	0.9991	0.9980	1.0000	3
ResponseBody_error.message_BackendError	2	1122	0.9893	0.0000	0.0000	17
ResponseBody_error.message_not-exist	617	507	0.9876	0.9797	0.9984	8
ResponseBody_error.message_NotFound	505	619	0.9991	0.9980	1.0000	3
ResponseBody_kind_not-exist	607	517	0.7700	0.7705	0.9984	5
ResponseBody_kind_taskList	517	607	0.9884	0.9813	0.9942	6
Mean			0.9733	0.8263	0.8690	6.8974
Standard Deviation			0.0452	0.3310	0.3376	4.2165

positive examples and 5 negative ones, whereas for the ResponseStatusCode_404 (with 505 positive and 619 negative examples), the inferred OCEL classifier has 0.9991 predictive accuracy, 0.998 precision, and 1.0 recall covering all positive examples and 1 of the negative examples. The length of the constructed class descriptions is 8 and 3, respectively. It can be noted that, in all targets, the incorrectly identified positive and/or negative cases are either HTTP status 503 (Service Unavailable) or those with past transaction sequences that contain HTTP 503. As explained in the attribute-based learning, this is due to the fact that it is much more difficult to learn class descriptions that precisely represent all positive and none of the negative examples from the background knowledge base and examples comprising such unexpected server behaviour. In the same way, for the class ResponseStatusCode_503 with 2 positive examples and 1122 negative examples, OCEL reaches 0.9893 predictive accuracy, 0.0 precision and 0.0 recall. It is because the inferred class description does not describe any positive examples correctly. This can be also attributed to difficulties in discovering the right concept for this kind of unforeseeable behaviour. There is also an imbalance between positive and negative examples that contributes further to this outcome. In reality, however, such server failures occur very rarely, such that these variations can be disregarded. The highest predictive performance can be mostly achieved by excluding the instances. Figure 6.32 presents the sample descriptions obtained for the ResponseStatusCode_200, ResponseStatusCode_204, and ResponseStatusCode_404 in the Manchester OWL syntax. The class description for ResponseStatusCode_200 encodes the semantics of HTTP 200 (OK) status code (indicating the successful creation/access/modification of a task list). It states that if the request is not made using the DELETE method (means if the request method is either POST, GET, or PATCH) and if the request has an immediately preceding transaction with HTTP 200 for that particular resource (that implies the request has a successful immediately preceding transaction), the response status code to the request will be HTTP 200. The class description for ResponseStatusCode_204 encodes the semantics of HTTP 204 (No Content) status code (indicating the successful deletion of a task list). This states that if the request is made using the DELETE method and had an immediately preceding transaction with HTTP 200 for that specific resource, the response status code will be 204. Conversely, the class description for ResponseStatusCode_404 encodes HTTP 404 (Not Found) semantics (indicating that the server can not locate a task list). It specifies that if the request has a prior transaction with HTTP 204 for that particular resource (meaning the resource has been successfully deleted before), the response status code for the request will be HTTP 404. It can be observed that the inferred semantic knowledge is close to the knowledge derived from attribute-based learning algorithms (described in Section 6.2.1). In general, it is very evident that all of these classification rules are easier to read and provide useful insights as to what properties are likely to lead to a certain status code.

Besides, the background knowledge base of the Google Tasks dataset contains the ResponseHeader_Alt-Svc_quic and ResponseHeader_Server_GSE targets related to the response header values which have no negative examples and are therefore excluded from the learning process. Table 6.11 shows that for all other target classes relevant to the response headers in the knowledge base, the definitions produced by OCEL are accurate and shorter. One example is the target classes associ-

```
ResponseHeader_Content-Type_json:
(not (RequestMethod_DELETE)) or
(isPrecededBy some ResponseStatusCode_204)

ResponseHeader_Content-Type_not-exist:
RequestMethod_DELETE and
(hasPrevious some ResponseStatusCode_200)
```

Figure 6.33: The Best OCEL Descriptions for `ResponseHeader_Content-Type` Values in Google Tasks

ated with the distinct values of the attribute `ResponseHeader_Content-Type`. The OCEL classifier for `ResponseHeader_Content-Type_json` (with 1024 positive and 100 negative examples) obtains 0.9555 predictive accuracy (by covering all positives and 50 negatives) and has 0.9555 precision and 1.0 recall as well. The definition is 6 in length. The classifier for `ResponseHeader_Content-Type_not-exist` (with 100 positive and 1024 negative examples) reaches 0.9947 predictive accuracy. It covers 99 positive and 5 negative examples and archives 0.9546 precision and 0.99 recall rates. The length of the definition is 5. In these contexts, the generated concepts are not sufficiently specialised to only represent all the positive examples and not the negative ones. There are two reasons for this: firstly, the background knowledge base and examples include unpredictable server behaviour, and secondly, inadequate examples of some of the transaction sequences in the knowledge base. The best OCEL descriptions learned are presented (in the Manchester OWL syntax) in Figure 6.33. As can be seen, the `ResponseHeader_Content-Type_json` description clearly specified that if the request is not made using the `DELETE` method (means if the request method is either `POST`, `GET`, or `PATCH`) or if the request has a preceding transaction with HTTP `204` for that particular resource (meaning the resource has been successfully deleted before), the response will contain the `Content-Type` header with the value `json` (means '`application/json; charset=UTF-8`'). The `ResponseHeader_Content-Type_json` description shows that if the request is made using the `DELETE` method and has an immediately preceding transaction with HTTP `200` for that specific resource (means preceded by a succesful requests), the response will not contain the `Content-Type` header. These definitions follow the exact same knowledge generated by the attribute-based learning algorithms for the semantics of the header values in `Content-Type` (suggesting that all responses to requests contain the `Content-Type` header with the value '`application/json; charset=UTF-8`' except for successful delete requests).

Additionally, there are different target classes in the knowledge base for distinct values of the response's payload attributes. Amongst them are the targets representing the response status and, as discussed above, the resultant OCEL classifiers are correct and have an easy-to-understand structure. Apart from these, the remaining target classes are the `ResponseBody_kind_taskList` (with 517 positive examples and 607 negative examples) and `ResponseBody_kind_not-exist` (with 607 positive examples and 517 negative examples). Both classes are associated with the `ResponseBody_kind` attribute in the Google Tasks dataset, which tends to have the value `tasks#tasks` for successful `POST`, `GET`, and `PATCH` requests (means for requests resulting in status code `200`). The experiment result reveals

```
ResponseBody_kind_taskList:
(not (RequestMethod_DELETE)) and
(hasPrevious some ResponseStatusCode_200)

ResponseBody_kind_not-exist:
RequestMethod_DELETE or
(isPrecededBy some ResponseStatusCode_204)
```

Figure 6.34: The Best OCEL Descriptions for `ResponseBody_kind` Values in Google Tasks

that the inferred model by OCEL for `ResponseBody_kind_taskList` has higher predictive performance since the learning problem is relatively simple. It reaches 0.9884 predictive accuracy by retrieving 514 positive examples and 10 negatives, as well as 0.9813 precision and 0.9942 recall. The length of the definition produced is 6. The model can be observed as being similar to `ResponseStatusCode 200`. The target `ResponseBody_kind_not-exist` is quite complex, as is the case with examples from HTTP `204`, `404`, and even `503`. As such, the model achieves slightly low predictive performance scores (0.77 predictive accuracy, 0.9984 recall, and precision 0.7705). It has a length of 5. In this case, OCEL is unable to compute a more precise class description from the training set before 120 seconds. Figure 6.34 displays the best OCEL description learned. However, it is obvious from the samples that the concept descriptions can be easily readable.

Overall, the experimental results confirm that OCEL is able to achieve highly reliable models with human-readable logics for targets associated with the main HTTP response feature values of the Google Tasks dataset, except for targets with complex learning problems (when predicting unforeseeable service behaviour). However, in general, having unpredictable server behaviour and insufficient examples of some transaction sequences hinder the algorithm from learning definitions that are 100% accurate.

Results on Slack Dataset

Table 6.12 summarises the number of positive and negative examples for each target class along with the OCEL results obtained for targets in the Slack knowledge base. The sample class definitions can be found in Appendix C.4.4. A full list of classes in the knowledge base (in the Protege environment) is provided in Appendix B.4.4.

As mentioned in Section 6.2.1, the `ResponseStatusCode` target attribute in the Slack dataset has only one distinct value (i.e., `200`). For this reason, the corresponding target class `ResponseStatusCode_200` in the background knowledge base of the Slack dataset has no negative examples and is ignored in model creation. The `ResponseBody_ok` and `ResponseBody_error` are the other two state-related target attributes in the dataset and the background knowledge base includes target classes associated with the distinct values of those attributes. The experimental results presented in Table 6.12 indicate that the class definitions induced by OCEL for all these targets have high to excellent predictive performance and are shorter in length and less complex, which in turn, are more precise and can be interpreted easier. Take into account, for example, the `ResponseBody_ok_true` (with 477 positive and 1,022 negative examples) and the `ResponseBody_ok_false` (with

Table 6.12: Results of Description Logic Learning Algorithm per Response Feature Value (Target) in Slack. The number of positive and negative examples is also shown for each target concept.

Class	**Positives**	**Negatives**	**Accuracy**	**Precision**	**Recall**	**Length**
ResponseHeader_x-slack-router_not-exist	565	934	0.3769	0.3769	1.0000	1
ResponseHeader_x-slack-router_p	934	565	0.6231	0.6231	1.0000	1
ResponseBody_channel_CCGRWTRKQ	477	1022	1.0000	1.0000	1.0000	8
ResponseBody_channel_not-exist	1022	477	0.9673	0.9543	1.0000	2
ResponseBody_error_messagenotfound	1022	477	0.9673	0.9543	1.0000	2
ResponseBody_error_not-exist	477	1022	1.0000	1.0000	1.0000	8
ResponseBody_message.bot_id_BCEPNCQDN	428	1071	1.0000	1.0000	1.0000	4
ResponseBody_message.bot_id_not-exist	1071	428	1.0000	1.0000	1.0000	2
ResponseBody_message.type_message	428	1071	1.0000	1.0000	1.0000	4
ResponseBody_message.type_not-exist	1071	428	1.0000	1.0000	1.0000	2
ResponseBody_message.user_not-exist	1071	428	1.0000	1.0000	1.0000	2
ResponseBody_message.user_UC8J6APLN	428	1071	1.0000	1.0000	1.0000	4
ResponseBody_ok_false	1022	477	0.9673	0.9543	1.0000	2
ResponseBody_ok_true	477	1022	1.0000	1.0000	1.0000	8
Mean			0.9216	0.9188	1.0000	3.5714
Standard Deviation			0.1855	0.1849	0.0000	2.5933

1,022 positive and 477 negative examples). As can be seen, the OCEL model for `ResponseBody_ok_true` reaches 1.0 predictive accuracy by completely covering every positive example and none of the negatives, and precision and recall rates are both 1.0. The class description built is 8 in length. The OCEL model for `ResponseBody_ok_false` achieves 0.9673 predictive accuracy along with 0.9534 precision and 1.0 recall rates by retrieving all positives but also 49 additional negatives (49 false positives), and the length of the definition is 2. Figure 6.35 displays the best definitions learned in the Manchester OWL syntax. The class definition for `ResponseBody_ok_true` encodes the semantics of the `true` value of the `ok` as if the request URI path carries the `chat.postMessage` token in its second location (means if the request is to create a message), or if the request URI path does not carry the `chat.update` and has an immediately preceding transaction for that particular resource with the `ok` value `true` (means if the request is to delete a message and it directly precedes by a successful transaction). Looking at the knowledge base closely, we can see that all the positive examples of `ResponseBody_ok_true` are related to the successful creation and deletion of messages (there are no examples with message modification in the sub-dataset that we used to build the knowledge base) and that all the instances for the creation of messages are successful. It is therefore clear that the inferred description describes the situation well. On the contrary, the class description for `ResponseBody_ok_false` encodes the semantics of the `false` value of the `ok` field (which implies the failure of a request). This specifies that the `ok` field will be `false` if the request URI path does not have the `chat.postMessage` token in its second location (means if the request is not to create a message). The resultant definition is general in this case, which retrieves some false positives. This is probably that the train set is not adequate to learn the mapping (insufficient examples to represent certain sequences of transactions in the knowledge base).

Further, the background knowledge base of the Slack dataset includes certain target classes relevant to the response header values with no negative examples, that are not suitable for learning class expressions. Examples include the `ResponseHeader_Content-Type_json`, `ResponseHeader_Server_Apache`, and `ResponseHeader_Cache-Control_private`. The only target classes trained with respect to the response headers are the `ResponseHeader_x-slack-router_p` and `ResponseHeader_x-slack-router_not-exist` (built from the attribute values of `ResponseHeader_x-slack-router`). In the `ResponseHeader_x-slack-router_p`, there are 934 positive examples (62.31% of total examples) and 565 negative examples. The target `ResponseHeader_x-slack-router_not-exist` has 565 positive examples (37.69% of total examples) and 934 negative cases. It can be observed that the inferred OCEL classification models for these two classes consist only of the root node `owl:Thing` (all instances in the knowledge base are individuals of `owl:Thing`). By simply covering all positive and negative examples with `owl:Thing`, the model for `ResponseHeader_x-slack-router_p` achieves 0.6231 predictive accuracy, 0.6231 precision, and 1.0 recall, while the classification model for `ResponseHeader_x-slack-router_not-exist` achieves 0.3769 predictive accuracy, 0.3769 precision, and 1.0 recall. This can be attributed to the fact that the algorithm is unable to derive more specific class definitions starting from the most general concept `owl:Thing` prior to timeout. This is probably because there is no mapping between the given examples and background knowledge. As already

```
ResponseBody_ok_true:
RequestUriPathToken2_chat.postMessage or
((not (RequestUriPathToken2_chat.update)) and
(hasPrevious some ResponseBody_ok_true))

ResponseBody_ok_false:
not (RequestUriPathToken2_chat.postMessage)
```

Figure 6.35: The Best OCEL Descriptions for `ResponseBody_ok` Values in Slack

```
ResponseBody_message.user_UC8J6APLN:
RequestHeader_Content-Type_x-www and
(not (RequestUriPathToken2_chat.update))

ResponseBody_message.user_not-exist:
not (RequestUriPathToken2_chat.postMessage)
```

Figure 6.36: The Best OCEL Descriptions for `ResponseBody_ok` Values in Slack

stated in Section 6.2.1, it is clear that the `ResponseHeader_x-slack-router` values and the respective request features and the service status history do not correlate. Therefore, the targets are not ideal for learning. These fairly low scores also lead to comparatively low average scores in the Slack dataset with a significant value distribution.

There are also several target classes associated with the property values of the response body. Specifically, the `ResponseBody_message.edited.user_UC8J6APLN` and `ResponseBody_message.edited.user_not-exist` targets represent the user who edits a message. However, as noted above, there are no examples in the background knowledge base of the Slack dataset that involve modifications of messages. Therefore, the `ResponseBody_message.edited.user_UC8J6APLN` is ignored when generating models because no positive examples can be found in the knowledge base, and the `ResponseBody_message.edited.user_not-exist` is also ignored since it has only positive examples. Apart from this, as we discussed above, there are some of the target classes correspond to state information. In addition, the `ResponseBody_channel_CCGRWTRKQ` and `ResponseBody_channel_not-exist` targets represent the `ResponseBody_channel` property values in the response body, and we note that the models express exactly the same performance measures and semantics as `ResponseBody_ok` because the values of the `ResponseBody_channel` are based on the response state. All the remaining target classes associated with the response body in the knowledge base of the Slack dataset are found to be dependent on whether or not the transaction is successful in creating a message. The concept descriptions generated very precisely demonstrate this particular scenario. It can be observed that the inferred OCEL descriptions reach the highest rate of predictive performance (1.0 predictive accuracy, precision, and recall) and that they are shorter, illustrating excellent relevance and interpretability. Figure 6.36 depicts the best class expressions learned in the Manchester OWL syntax for target classes derived from the attribute `ResponseBody_message.user`.

It can, therefore, be concluded that OCEL is capable of providing promising predictive performance results predicting the key HTTP response feature values in the Slack dataset and is able to produce comprehensible models, with the exception of certain targets linked to the response header values relevant to the data returned by the server. In addition, the lack of sufficient examples of certain transaction sequences prevents the algorithm from learning fully accurate concepts at times.

6.3 Threats to Validity

Three threats that could potentially influence the results of the experiments and the generalisability of the findings should be considered:

1. The study examined four SML algorithms, including three of the attribute-based learning algorithms, i.e., the C4.5 decision tree algorithm, the RIPPER and PART rule learners, and the OCEL description logic learning algorithm, to assess the potential of SML algorithms in producing mock response skeletons of HTTP services. This leaves the possibility that the chosen algorithms do not necessarily cover a broader context of SML techniques, thereby limiting the ability to generalise the results of the study. However, we ensured that the selection of algorithms was based on how well-established and efficient the algorithms are in each category of symbolic learning techniques: attribute-based learning (i.e., decision tree learning and rule learning) and description logic learning, and their ability to handle both continuous and discrete data types (as required when considering the typical nature of HTTP services). All of these chosen algorithms have been established and effective in their respective category of symbolic technique and have been well-explored by several researchers and, in particular, have demonstrated their ability to work directly with nominal and numerical attributes in model learning. Hence, it is very likely that the study has drawn reliable conclusions which can be extended to SML techniques in general.

2. The HTTP datasets used in the study were collected by reverse-engineering REST API interactions from repository snapshots or by recording traffic through fuzzing REST APIs. This leaves the possibility that such datasets do not represent realistic workloads, thereby, research results may not be realistic and generalisable. However, we extracted datasets from the most successful active Web services and synthesised them using well-defined processes. All API interactions were derived on the basis of the syntax and semantics of HTTP and the underline services, and a wide set of HTTP features was also implemented. Each dataset was large enough to facilitate the research described. We managed to preserve the behaviour that exists in real HTTP traffic as much as possible and believe that all datasets reflect the state-of-the-art use of HTTP-based services in general. We are, therefore, confident that the results of the research were accurate and can be applied to other HTTP-based services that were not studied.

3. The classification models generated by algorithms were mainly evaluated by emphasising their predictive capability and comprehensibility. Metrics such as predictive accuracy, precision, and recall have been used to assess predictive validity and model size was used to measure comprehensibility. Such metrics may not be sufficient to determine how suitable the symbolic learning algorithms are for producing accurate and customisable mock response skelet-

ons. Further analysis is expected in the future, in particular, to evaluate models by a targeted group of end-users in order to verify the extent of their usability.

6.4 Summary

In this chapter, we presented and discussed the experimental results in order to evaluate the appropriateness of Symbolic Machine Learning techniques in producing mock response skeletons for HTTP-based services. Specifically, we explored results based on GHTraffic, Twitter, Google Tasks, and Slack datasets. With each dataset, the results of C4.5, RIPPER and PART (from attribute-based learning) and OCEL (from description logic learning) were analysed with respect to the target attributes associated with the key HTTP response features, including status, response headers, and response body. At the end of the chapter, we discussed possible threats to the validity of the results. Overall, the results obtained reveal that the significance of the Symbolic Machine Learning algorithms in the training of accurate, human-readable models for predicting the core features of HTTP service responses. It can be also observed that all algorithms have similarities in terms of induced semantic knowledge. Moreover, it confers the usefulness of the proposed attributes in building classification models.

Chapter 7

Conclusions and Future Work

This chapter summarises the results and contribution of this thesis and outlines some future works. Section 7.1 presents the key contributions of our research. Section 7.2 discusses directions for future work.

7.1 Summary of Contributions

The research in this thesis is primarily aimed at identifying the suitability of Symbolic Machine Learning (SML) techniques in automatically producing mock response skeletons of HTTP-based services that are both accurate and customisable. This goal has been achieved by examining four promising symbolic learning algorithms: the C4.5 decision tree algorithm, the RIPPER and PART rule learners (from attribute-based learning) and the OCEL class expression learning algorithm (from description logic learning), utilising network traffic datasets derived from the services offered by GitHub, Twitter, Google Tasks, and Slack. Several experiments were conducted on the chosen algorithms targeting at training models to predict some of the properties of HTTP service responses by directly inferring knowledge of the protocol structure and service status from recorded interactions, and testing the predictive ability and comprehensibility of generated models through cross validation. The predictive performance was measured by using predictive accuracy, precision, and recall, while the model size was used to calculate the comprehensibility.

The experimental results discussed in Chapter 6 demonstrate that the selected SML algorithms are suitable for making highly accurate and human-readable predictions for the key aspects of HTTP service responses, including the status, response headers, and response body. Such a set of output predictions is known as a mock skeleton. The generated models for most of the response properties have higher predictive performance and are smaller in size and less complex, which in turn, are more precise and can be easily interpreted by domain specialists. All the output predictions enable a better comprehension of the service semantics through proper logic. In terms of induced knowledge, the algorithms (both attribute-based learning and description logic learning) share similarities, even though there are structural differences depending on how the algorithms operate. Having such knowledge at hand would make it easier for the engineers used to write mock tests to inspect and understand the causes behind the different response property values, enabling them to easily edit or refine the predictions to create mocks that can generate service responses suitable for application testing. The engineers could add missing response property values that could not be predicted in practise (filling the gaps in the skeletons). They can identify which response properties are not available in skeletons by referring to a few sample responses in the network traffic traces collected. The fact that key characteristics of the HTTP service responses can be precisely predicted in human-readable format help engineers to easily make decisions about whether the mock service is a suitable representation of the actual service. It promotes the trust of the engineers in the generated mock service. Even if the service evolves from time to time (i.e., if there are certain changes in the new version of the service than the one created mock skeletons with), consider for example, that if the service now changes its usage from `PUT` to `PATCH`, engineers could simply modify the generated mock service accordingly as they understand the semantics rather than going through the much more expensive process of capturing the service interactions again and re-learning the service.

It is observed that the algorithms could not be used with some response properties to learn perfectly accurate models based on factors such as the lack of training data (a limited number of transaction sequences representing various behavioural patterns) and where the training data contains unexpected service behaviour (such as HTTP `500 Internal Server Error`). Of the two causes listed above, the first is more common and the accuracy can be improved by increasing the number of individuals in the training set. Unfortunately, this would not be a possibility for OCEL at present, because it could cause the reasoner to run out of memory. The second one is a rarely occurring issue in reality, so that such variations can be disregarded. There are also some response properties which have either a single value (e.g., `Host`, assuming that the service may always use the same host) or a fairly large number of distinct values (e.g., `Date`, assuming that each transaction has a unique value) that are not suited for predictions. Yet, these types of characteristics of the responses can be handled in practice with very little manual effort. Some other response properties, such as the content of the data returned by the server (such as data on external entities to which the service itself refers) and the data on unexpected service behaviour, also remain unpredictable, as no correlation can be guaranteed with the service structure and status. Such features are often not incorporated into response mocks in practice as well.

In general, this is the very first research that studies the virtualisation of HTTP-based services and directly emphasises the generation of HTTP responses with human-readable explanations. Considering that the SML techniques produce inherently interpretable results (contrary to the sub-symbolic techniques), our study is primarily intended to explore the suitability of SML techniques to accurately predict HTTP response properties with human-readable constructs. It is important to note that the existing SV solutions [8–20] are not able to produce accurate approximations of the actual responses of HTTP-based services. And the techniques largely lack transparency. This research, therefore, advances the state-of-the-art SV techniques. By concluding the research, SML algorithms are recommended for generating mock skeletons for HTTP-based services testing.

In addition to the findings mentioned above, this thesis has also made several other considerable contributions. All the experimental datasets comprising HTTP transactions (which were introduced in Chapter 4) are publicly available to enable users to track the provenance of our findings (we also provide access to the scripts used to generate the datasets). These datasets are ideal for reproducible research on many aspects of service-oriented computing. The scripts used to automate all the experiments (in Chapter 5) are also publicly accessible and explicit guidelines (on how to re-run experiments) are given, such that users could run experiments themselves using the same datasets and scripts to obtain the same results from this study (presented in Chapter 6). In addition, a pre-configured VirtualBox image (i.e., artefact) is provided that replicates the experimental environment ensuring that all experiments can be reproduced with little manual effort on any computer that has VirtualBox installed and satisfies the minimum system requirements. All of these practices particularly covered the secondary aim of the research, i.e., to produce research results that are easier to reproduce. It is notable that there is no open access to datasets and/or scripts used in existing SV studies [12–20] to reproduce the published results, so our study has taken a lead in this regard.

The experimental methodology presented in Chapter 5 also introduced a well-designed, robust machine learning framework for reliable predictions of HTTP response properties. The framework orchestrates and automates sequences of machine learning tasks by allowing data preprocessing, data transformation, and training and evaluating models to achieve outcomes. These tasks were specially designed to comply with any HTTP-based service in general. The framework offers a reproducible machine learning workflow that could be used for future research towards HTTP service simulation. Moreover, Chapter 3 contributed a systematic overview of the literature on existing approaches and tools in service-oriented application testing that address dependency issues. It also presented literature surveys in the fields of services testing and AI-driven software testing.

7.2 Future Work

There are a few other potential research directions that can be followed to further improve the quality of the findings or to advance the scope of the research.

As stated in the discussion of potential threats to the validity of the research (in Chapter 6), it would be useful to assess the usability of the generated models with real end-users (i.e., quality assurance engineers). This would provide a more thorough empirical investigation into the ability of algorithms to produce customisable skeletons for mocked services. It would be an indispensable direction to help the results to be more useful for real-world applications.

It would also be interesting to extend the experiments by adding similar datasets from other service providers using similar processes and algorithms and/or by adding additional SML algorithms using similar datasets and processes.

Another potential direction for future research work is to implement a service virtualisation approach based on the symbolic learning techniques studied to replace the real target services for testing purposes. Such a technique could make it possible to construct accurate mock response skeletons in simple logics that can facilitate easy comprehension of service semantics and can easily be tuned and adjusted to create mocks that can generate service responses. This would significantly increase the flexibility and usability of mocks in service-oriented applications testing. The generated mocks would be transparent to the engineers using them. The machine learning framework presented in this thesis (in Chapter 5) could be used as the basis for such research.

Appendix A

HTTP Datasets

A.1 JSON Schemas for GHTraffic Dataset

A.1.1 POST HTTP Transaction

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "description": "A representation of an HTTP POST transaction",
  "type": "object",
  "properties": {
    "MetaData": {
      "type": "object",
      "properties": {
        "processor": {
          "type": "string",
          "description": "asserted"
        },
        "source": {
          "type": "string",
          "description": "asserted"
        },
        "type": {
          "type": "string",
          "description": "asserted"
        },
        "timestamp": {
          "type": "string",
          "description": "asserted"
        }
      }
    },
    "Request": {
      "type": "object",
      "properties": {
        "Message-Body": {
          "type": "string"
        },
        "Message-Header": {
          "type": "object",
          "properties": {
            "Authorization": {
              "type": "string",
              "description": "asserted"
            },
            "Accept": {
              "type": "string",
              "description": "asserted"
            },
            "Content-Length": {
              "type": "integer",
              "description": "asserted"
            },
            "Content-Type": {
              "type": "string",
              "description": "asserted"
            },
            "Host": {
              "type": "string",
              "description": "asserted"
            },
            "User-Agent": {
              "type": "string",
              "description": "asserted"
            }
          },
          "required": [
            "Authorization",
            "Accept",
            "Content-Length",
            "Content-Type",
            "Host",
            "User-Agent"
            ]
        },
        "Method": {
          "type": "string",
          "description": "asserted",
          "enum": [
            "POST"
          ]
        },
        "Request-URI": {
          "type": "string",
          "description": "inferred"
        },
        "HTTP-Version": {
          "type": "string",
          "description": "asserted"
        }
      },
      "required": [
      "Message-Body",
      "Message-Header","Method","Request-URI","HTTP-Version"],
      "additionalProperties": false
    },
    "Response": {
      "type": "object",
      "properties": {
        "Message-Body": {
          "type": "string"
        },
        "Message-Header": {
          "type": "object",
          "properties": {
            "Date": {
              "type": "string",
              "description": "inferred"
            },
            "Server": {
              "type": "string",
              "description": "asserted"
            },
            "Content-Length": {
              "type": "integer",
              "description": "inferred"
            },
            "Content-Type": {
              "type": "string",
              "description": "asserted"
            },
            "Location": {
              "type": "string",
              "description": "inferred"
            },
            "X-GitHub-Request-Id": {
              "type": "string",
              "description": "asserted"
            },
            "Vary": {
              "type": "string",
              "description": "asserted"
            },
            "ETag": {
              "type": "string",
              "description": "asserted"
            },
            "Access-Control-Allow-Origin": {
              "type": "string",
              "description": "asserted"
            },
            "X-GitHub-Media-Type": {
              "type": "string",
              "description": "asserted"
            },
            "Cache-Control": {
              "type": "string",
              "description": "asserted"
            },
            "X-OAuth-Scopes": {
              "type": "string",
              "description": "asserted"
            },
            "X-Accepted-OAuth-Scopes": {
              "type": "string",
              "description": "asserted"
            },
            "Access-Control-Expose-Headers": {
              "type": "string",
              "description": "asserted"
            }
          },
          "required": [
            "Date",
            "Server",
            "Content-Length",
            "Content-Type",
            "Location",
            "X-GitHub-Request-Id",
            "Vary",
            "ETag",
            "X-Accepted-OAuth-Scopes",
            "X-OAuth-Scopes",
            "Access-Control-Allow-Origin",
            "X-GitHub-Media-Type",
            "Cache-Control",
            "Access-Control-Expose-Headers"
          ]
        },
        "Reason-Phrase": {
          "type": "string",
          "description": "asserted"
        },
        "Status-Code": {
          "type": "integer",
          "description": "asserted"
        },
        "HTTP-Version": {
          "type": "string",
          "description": "asserted"
        }
      },
      "required": [
        "Message-Body",
        "Message-Header",
        "Reason-Phrase",
        "Status-Code",
        "HTTP-Version"
      ],
          "additionalProperties": false
        }
      },
      "required": [
        "Meta-Data",
        "Request",
        "Response"
      ]
    }
```

A.1.2 PATCH HTTP Transaction

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "description": "A representation of an HTTP PATCH transaction",
  "type": "object",
  "properties": {
    "MetaData": {
      "type": "object",
      "properties": {
        "processor": {
          "type": "string",
          "description": "asserted"
        },
        "source": {
          "type": "string",
          "description": "asserted"
        },
        "type": {
          "type": "string",
          "description": "asserted"
        },
        "timestamp": {
          "type": "string",
          "description": "asserted"
        }
      }
    },
  "Request": {
      "type": "object",
      "properties": {
        "Message-Body": {
          "type": "string"
        },
        "Message-Header": {
          "type": "object",
          "properties": {
            "Authorization": {
              "type": "string",
              "description": "asserted"
            },
            "Accept": {
              "type": "string",
              "description": "asserted"
            },
            "Content-Length": {
              "type": "integer",
              "description": "asserted"
            },
            "Content-Type": {
              "type": "string",
              "description": "asserted"
            },
            "Host": {
              "type": "string",
              "description": "asserted"
            },
            "User-Agent": {
              "type": "string",
              "description": "asserted"
            }
          },
          "required": [
            "Authorization",
            "Accept",
            "Content-Length",
            "Content-Type",
            "Host",
            "User-Agent"
          ]
        },
        "Method": {
          "type": "string",
          "description": "asserted",
          "enum": [
            "PATCH"
          ]
        },
        "Request-URI": {
          "type": "string",
          "description": "inferred"
        },
        "HTTP-Version": {
          "type": "string",
          "description": "asserted"
        }
      },
      "required": [
        "Message-Body",
        "Message-Header",
        "Method",
        "Request-URI",
        "HTTP-Version"
      ],
      "additionalProperties": false
    },
    "Response": {
      "type": "object",
      "properties": {
        "Message-Body": {
          "type": "string"
        },
        "Message-Header": {
          "type": "object",
          "properties": {
            "Date": {
              "type": "string",
              "description": "inferred"
            },
            "Server": {
              "type": "string",
              "description": "asserted"
            },
            "Content-Length": {
              "type": "integer",
              "description": "inferred"
            },
            "Content-Type": {
              "type": "string",
              "description": "asserted"
            },
            "X-GitHub-Request-Id": {
              "type": "string",
              "description": "asserted"
            },
            "Vary": {
              "type": "string",
              "description": "asserted"
            },
            "ETag": {
              "type": "string",
              "description": "asserted"
            },
            "Access-Control-Allow-Origin": {
              "type": "string",
              "description": "asserted"
            },
            "X-GitHub-Media-Type": {
              "type": "string",
              "description": "asserted"
            },
            "Cache-Control": {
              "type": "string",
              "description": "asserted"
            },
            "X-OAuth-Scopes": {
              "type": "string",
              "description": "asserted"
            },
            "X-Accepted-OAuth-Scopes": {
              "type": "string",
              "description": "asserted"
            },
            "Access-Control-Expose-Headers": {
              "type": "string",
              "description": "asserted"
            }
          },
          "required": [
            "Date",
            "Server",
            "Content-Length",
            "Content-Type",
            "X-GitHub-Request-Id",
            "Vary",
            "ETag",
            "X-Accepted-OAuth-Scopes",
            "X-OAuth-Scopes",
            "Access-Control-Allow-Origin",
            "X-GitHub-Media-Type",
            "Cache-Control",
            "Access-Control-Expose-Headers"
          ]
        },
        "Reason-Phrase": {
          "type": "string",
          "description": "asserted"
        },
        "Status-Code": {
          "type": "integer",
          "description": "asserted"
        },
        "HTTP-Version": {
          "type": "string",
          "description": "asserted"
        }
      },
      "required": [
        "Message-Body",
        "Message-Header",
        "Reason-Phrase",
        "Status-Code",
        "HTTP-Version"
      ],
      "additionalProperties": false
    }
  },
  "required": [
    "Meta-Data",
    "Request",
    "Response"
  ]
}
```

A.1.3 GET HTTP Transaction

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "description": "A representation of an HTTP GET transaction",
  "type": "object",
  "properties": {
    "MetaData": {
      "type": "object",
      "properties": {
        "processor": {
          "type": "string",
          "description": "asserted"
        },
        "source": {
          "type": "string",
          "description": "asserted"
        },
        "type": {
          "type": "string",
          "description": "asserted"
        },
        "timestamp": {
          "type": "string",
          "description": "asserted"
        }
      }
    },
    "Request": {
      "type": "object",
      "properties": {
        "Message-Header": {
          "type": "object",
          "properties": {
            "Authorization": {
              "type": "string",
              "description": "asserted"
            },
            "Accept": {
              "type": "string",
              "description": "asserted"
            },
            "Host": {
              "type": "string",
              "description": "asserted"
            },
            "User-Agent": {
              "type": "string",
              "description": "asserted"
            }
          },
          "required": [
            "Authorization",
            "Accept",
            "Host",
            "User-Agent"
          ]
        },
        "Method": {
          "type": "string",
          "description": "asserted",
          "enum": [
            "GET"
          ]
        },
        "Request-URI": {
          "type": "string",
          "description": "inferred"
        },
        "HTTP-Version": {
          "type": "string",
          "description": "asserted"
      }
    },
    "required": [
      "Message-Header",
      "Method",
      "Request-URI",
      "HTTP-Version"
    ],
    "additionalProperties": false
  },
  "Response": {
    "type": "object",
    "properties": {
      "Message-Body": {
        "type": "string"
      },
      "Message-Header": {
        "type": "object",
        "properties": {
          "Date": {
            "type": "string",
            "description": "asserted"
          },
          "Server": {
            "type": "string",
            "description": "asserted"
          },
          "Content-Length": {
            "type": "integer",
            "description": "inferred"
          },
          "Content-Type": {
            "type": "string",
            "description": "asserted"
          },
          "X-GitHub-Request-Id": {
            "type": "string",
            "description": "asserted"
          },
          "Vary": {
            "type": "string",
            "description": "asserted"
          },
          "ETag": {
            "type": "string",
            "description": "asserted"
          },
          "Last-Modified": {
            "type": "string",
            "description": "inferred"
          },
          "Access-Control-Allow-Origin": {
            "type": "string",
            "description": "asserted"
          },
          "X-GitHub-Media-Type": {
            "type": "string",
            "description": "asserted"
          },
          "Cache-Control": {
            "type": "string",
            "description": "asserted"
          },
          "X-OAuth-Scopes": {
            "type": "string",
            "description": "asserted"
          },
          "X-Accepted-OAuth-Scopes": {
            "type": "string",
            "description": "asserted"
          },
          "Access-Control-Expose-Headers": {
            "type": "string",
              "description": "asserted"
            }
          },
          "required": [
            "Date",
            "Server",
            "Content-Length",
            "Content-Type",
            "X-GitHub-Request-Id",
            "Vary",
            "ETag",
            "Last-Modified",
            "X-Accepted-OAuth-Scopes",
            "X-OAuth-Scopes",
            "Access-Control-Allow-Origin",
            "X-GitHub-Media-Type",
            "Cache-Control",
            "Access-Control-Expose-Headers"
          ]
        },
        "Reason-Phrase": {
          "type": "string",
          "description": "asserted"
        },
        "Status-Code": {
          "type": "integer",
          "description": "asserted"
        },
        "HTTP-Version": {
          "type": "string",
          "description": "asserted"
        }
      },
      "required": [
        "Message-Body",
        "Message-Header",
        "Reason-Phrase",
        "Status-Code",
        "HTTP-Version"
      ],
      "additionalProperties": false
    }
  },
  "required": [
    "Meta-Data",
    "Request",
    "Response"
  ]
}
```

A.1.4 HEAD HTTP Transaction

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "description": "A representation of an HTTP HEAD transaction",
  "type": "object",
  "properties": {
    "MetaData": {
      "type": "object",
      "properties": {
        "processor": {
          "type": "string",
          "description": "asserted"
        },
        "source": {
          "type": "string",
          "description": "asserted"
        },
        "type": {
          "type": "string",
          "description": "asserted"
        },
        "timestamp": {
          "type": "string",
          "description": "asserted"
        }
      }
    },
    "Request": {
      "type": "object",
      "properties": {
        "Message-Header": {
          "type": "object",
          "properties": {
            "Authorization": {
              "type": "string",
              "description": "asserted"
            },
            "Accept": {
              "type": "string",
              "description": "asserted"
            },
            "Host": {
              "type": "string",
              "description": "asserted"
            },
            "User-Agent": {
              "type": "string",
              "description": "asserted"
            }
          },
          "required": [
            "Authorization",
            "Accept",
            "Host",
            "User-Agent"
          ]
        },
        "Method": {
          "type": "string",
          "description": "asserted",
          "enum": [
            "HEAD"
          ]
        },
        "Request-URI": {
          "type": "string",
          "description": "inferred"
        },
        "HTTP-Version": {
          "type": "string",
          "description": "asserted"
        }
      },
      "required": [
      "Message-Header",
      "Method",
      "Request-URI",
      "HTTP-Version"
      ],
      "additionalProperties": false
    },
    "Response": {
      "type": "object",
      "properties": {
        "Message-Header": {
          "type": "object",
          "properties": {
            "Date": {
              "type": "string",
              "description": "asserted"
            },
            "Server": {
              "type": "string",
              "description": "asserted"
        },
        "Content-Length": {
          "type": "integer",
          "description": "inferred"
        },
        "Content-Type": {
          "type": "string",
          "description": "asserted"
        },
        "X-GitHub-Request-Id": {
          "type": "string",
          "description": "asserted"
        },
        "Vary": {
          "type": "string",
          "description": "asserted"
        },
        "ETag": {
          "type": "string",
          "description": "asserted"
        },
        "Last-Modified": {
          "type": "string",
          "description": "inferred"
        },
        "Access-Control-Allow-Origin": {
          "type": "string",
          "description": "asserted"
        },
        "X-GitHub-Media-Type": {
          "type": "string",
          "description": "asserted"
        },
        "Cache-Control": {
          "type": "string",
          "description": "asserted"
        },
        "X-OAuth-Scopes": {
          "type": "string",
          "description": "asserted"
        },
        "X-Accepted-OAuth-Scopes": {
          "type": "string",
          "description": "asserted"
        },
        "Access-Control-Expose-Headers": {
          "type": "string",
          "description": "asserted"
        }
      },
      "required": [
        "Date",
        "Server",
        "Content-Length",
        "Content-Type",
        "X-GitHub-Request-Id",
        "Vary",
        "ETag",
        "Last-Modified",
        "X-Accepted-OAuth-Scopes",
        "X-OAuth-Scopes",
        "Access-Control-Allow-Origin",
        "X-GitHub-Media-Type",
        "Cache-Control",
        "Access-Control-Expose-Headers"
      ]
    },
    "Reason-Phrase": {
      "type": "string",
      "description": "asserted"
    },
    "Status-Code": {
      "type": "integer",
              "description": "asserted"
            },
            "HTTP-Version": {
              "type": "string",
              "description": "asserted"
            }
          },
          "required": [
            "Message-Header",
            "Reason-Phrase",
            "Status-Code",
            "HTTP-Version"
          ],
          "additionalProperties": false
        }
      },
      "required": [
        "Meta-Data",
        "Request",
        "Response"
      ]
    }
```

A.1.5 PUT HTTP Transaction

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "description": "A representation of an HTTP PUT transaction",
  "type": "object",
  "properties": {
    "MetaData": {
      "type": "object",
      "properties": {
        "processor": {
          "type": "string",
          "description": "asserted"
        },
        "source": {
          "type": "string",
          "description": "asserted"
        },
        "type": {
          "type": "string",
          "description": "asserted"
        },
        "timestamp": {
          "type": "string",
          "description": "asserted"
        }
      }
    },
    "Request": {
      "type": "object",
      "properties": {
        "Message-Header": {
          "type": "object",
          "properties": {
            "Authorization": {
              "type": "string",
              "description": "asserted"
            },
            "Accept": {
              "type": "string",
              "description": "asserted"
            },
            "Host": {
              "type": "string",
              "description": "asserted"
            },
            "User-Agent": {
              "type": "string",
              "description": "asserted"
          }
        },
        "required": [
        "Authorization",
        "Accept",
        "Host",
        "User-Agent"
        ]
      },
      "Method": {
        "type": "string",
        "description": "asserted",
        "enum": [
          "PUT"
        ]
      },
      "Request-URI": {
        "type": "string",
        "description": "inferred"
      },
      "HTTP-Version": {
        "type": "string",
        "description": "asserted"
      }
    },
    "required": [
      "Message-Header",
      "Method",
      "Request-URI",
      "HTTP-Version"
    ],
    "additionalProperties": false
  },
  "Response": {
    "type": "object",
    "properties": {
      "Message-Header": {
        "type": "object",
        "properties": {
          "Date": {
            "type": "string",
            "description": "asserted"
          },
          "Server": {
            "type": "string",
            "description": "asserted"
          },
          "Content-Type": {
            "type": "string",
            "description": "asserted"
          },
          "X-GitHub-Request-Id": {
            "type": "string",
            "description": "asserted"
          },
          "Access-Control-Allow-Origin": {
            "type": "string",
            "description": "asserted"
          },
          "X-GitHub-Media-Type": {
            "type": "string",
            "description": "asserted"
          },
          "X-OAuth-Scopes": {
            "type": "string",
            "description": "asserted"
          },
          "X-Accepted-OAuth-Scopes": {
            "type": "string",
            "description": "asserted"
          },
          "Access-Control-Expose-Headers": {
            "type": "string",
              "description": "asserted"
            }
          },
          "required": [
            "Date",
            "Server",
            "Content-Type",
            "X-GitHub-Request-Id",
            "X-Accepted-OAuth-Scopes",
            "X-OAuth-Scopes",
            "Access-Control-Allow-Origin",
            "X-GitHub-Media-Type",
            "Access-Control-Expose-Headers"
          ]
        },
        "Reason-Phrase": {
          "type": "string",
          "description": "asserted"
        },
        "Status-Code": {
          "type": "integer",
          "description": "asserted"
        },
        "HTTP-Version": {
          "type": "string",
          "description": "asserted"
        }
      },
      "required": [
        "Message-Header",
        "Reason-Phrase",
        "Status-Code",
        "HTTP-Version"
      ],
      "additionalProperties": false
    }
  },
  "required": [
    "Request",
    "Response"
  ]
}
```

A.1.6 DELETE HTTP Transaction

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "description": "A representation of an HTTP DELETE transaction",
  "type": "object",
  "properties": {
    "MetaData": {
      "type": "object",
      "properties": {
        "processor": {
          "type": "string",
          "description": "asserted"
        },
        "source": {
          "type": "string",
          "description": "asserted"
        },
        "type": {
          "type": "string",
          "description": "asserted"
        },
        "timestamp": {
          "type": "string",
          "description": "asserted"
        }
      }
    },
    "Request": {
```

```
    "type": "object",
    "properties": {
      "Message-Header": {
        "type": "object",
        "properties": {
          "Authorization": {
            "type": "string",
            "description": "asserted"
          },
          "Accept": {
            "type": "string",
            "description": "asserted"
          },
          "Host": {
            "type": "string",
            "description": "asserted"
          },
          "User-Agent": {
            "type": "string",
            "description": "asserted"
          }
        },
        "required": [
          "Authorization",
          "Accept",
          "Host",
          "User-Agent"
        ]
      },
      "Method": {
        "type": "string",
        "description": "asserted",
        "enum": [
          "DELETE"
        ]
      },
      "Request-URI": {
        "type": "string",
        "description": "inferred"
      },
      "HTTP-Version": {
        "type": "string",
        "description": "asserted"
      }
    },
    "required": [
      "Message-Header",
      "Method",
      "Request-URI",
      "HTTP-Version"
    ],
    "additionalProperties": false
  },
  "Response": {
    "type": "object",
    "properties": {
      "Message-Header": {
        "type": "object",
        "properties": {
          "Date": {
            "type": "string",
            "description": "asserted"
          },
          "Server": {
            "type": "string",
            "description": "asserted"
          },
          "Content-Type": {
            "type": "string",
            "description": "asserted"
          },
          "X-GitHub-Request-Id": {
            "type": "string",
              "description": "asserted"
            },
            "Access-Control-Allow-Origin": {
              "type": "string",
              "description": "asserted"
            },
            "X-GitHub-Media-Type": {
              "type": "string",
              "description": "asserted"
            },
            "X-OAuth-Scopes": {
              "type": "string",
              "description": "asserted"
            },
            "X-Accepted-OAuth-Scopes": {
              "type": "string",
              "description": "asserted"
            },
            "Access-Control-Expose-Headers": {
              "type": "string",
              "description": "asserted"
            }
          },
          "required": [
            "Date",
            "Server",
            "Content-Type",
            "X-GitHub-Request-Id",
            "X-Accepted-OAuth-Scopes",
            "X-OAuth-Scopes",
            "Access-Control-Allow-Origin",
            "X-GitHub-Media-Type",
            "Access-Control-Expose-Headers"
          ]
        },
        "Reason-Phrase": {
        "type": "string",
        "description": "asserted"
        },
        "Status-Code": {
        "type": "integer",
        "description": "asserted"
        },
        "HTTP-Version": {
        "type": "string",
        "description": "asserted"
        }
      },
      "required": [
        "Message-Header",
        "Reason-Phrase",
        "Status-Code",
        "HTTP-Version"
      ],
      "additionalProperties": false
    }
  },
  "required": [
    "Request",
    "Response"
  ]
}
```

A.2 Sample Records on GHTraffic Dataset

A.2.1 POST HTTP Transaction (Create Issue)

```
{
  "Request": {
    "Message-Body": "{\"milestone\":6,\"title\":\"Multimaps.newListMultimap
        should document that map must be empty\",\"body\":\"..If the map passed
        to Multimaps.newListMultimap is not empty an IllegalArgumentException is
        thrown..\",\"labels\":[\"status: fixed\",\"type: defect\"]}",
    "Message-Header": {
      "Authorization": "token 9f8ea2dd9a582c92488de26f7725bd6eaa561df3",
      "Accept": "*/*",
      "User-Agent": "Mozilla/5.0 (Macintosh; U; Intel Mac OS X; en) AppleWebKit
          /418.9 (KHTML, like Gecko) Safari/419.3",
      "Host": "api.github.com",
      "Content-Length": 223,
      "Content-Type": "application/json; charset=utf-8"
    },
    "Request-URI": "/repos/google/guava/issues",
    "Method": "POST",
    "HTTP-Version": "HTTP/1.1"
  },
  "Response": {
    "Message-Body": "{\"comments\":0,\"closed_at\":null,\"user\":{\"login\":\"
        gissuebot\",\"id\":8091570,\"avatar_url\":\"https:\\/\\/avatars.
        githubusercontent.com\\/u\\/8091570?v=3\",\"gravatar_id\":\"\",\"url\":\"
        https:\\/\\/api.github.com\\/users\\/gissuebot\",\"html_url\":\"https
        :\\/\\/github.com\\/gissuebot\",\"followers_url\":\"https:\\/\\/api.
        github.com\\/users\\/gissuebot\\/followers\",\"following_url\":\"https
        :\\/\\/api.github.com\\/users\\/gissuebot\\/following{\\/other_user}\",\"
        gists_url\":\"https:\\/\\/api.github.com\\/users\\/gissuebot\\/gists{\\/
        gist_id}\",\"starred_url\":\"https:\\/\\/api.github.com\\/users\\/
        gissuebot\\/starred{\\/owner}{\\/repo}\",\"subscriptions_url\":\"https
        :\\/\\/api.github.com\\/users\\/gissuebot\\/subscriptions\",\"
        organizations_url\":\"https:\\/\\/api.github.com\\/users\\/gissuebot\\/
        orgs\",\"repos_url\":\"https:\\/\\/api.github.com\\/users\\/gissuebot\\/
        repos\",\"events_url\":\"https:\\/\\/api.github.com\\/users\\/gissuebot
        \\/events{\\/privacy}\",\"received_events_url\":\"https:\\/\\/api.github.
        com\\/users\\/gissuebot\\/received_events\",\"type\":\"User\",\"
        site_admin\":false},\"assignees\":[],\"created_at\":\"2014-10-31T17:22:22
        Z\",\"title\":\"Multimaps.newListMultimap should document that map must
        be empty\",\"body\":\"..If the map passed to Multimaps.newListMultimap is
         not empty an IllegalArgumentException is thrown..\",\"url\":\"https
        :\\/\\/api.github.com\\/repos\\/google\\/guava\\/issues\\/736\",\"labels
        \":[{\"url\":\"https:\\/\\/api.github.com\\/repos\\/google\\/guava\\/
        labels\\/status%3A+fixed\",\"name\":\"status: fixed\",\"color
        \":\"009800\"}],\"labels_url\":\"https:\\/\\/api.github.com\\/repos\\/
        google\\/guava\\/issues\\/736\\/labels{\\/name}\",\"number\":736,\"
        milestone\":{\"url\":\"https:\\/\\/api.github.com\\/repos\\/google\\/
        guava\\/milestones\\/6\",\"labels_url\":\"https:\\/\\/api.github.com\\/
        repos\\/google\\/guava\\/milestones\\/6\\/labels\",\"id\":849118,\"number
        \":6,\"title\":\"11.0\",\"description\":\"\",\"creator\":{..another user
        },\"open_issues\":0,\"closed_issues\":25,\"state\":\"closed\",\"
        created_at\":\"2014-11-01T03:46:46Z\",\"updated_at\":\"2014-11-06T23
        :12:24Z\",\"due_on\":null,\"closed_at\":\"2014-11-06T23:12:24Z\"},\"
        updated_at\":\"2014-10-31T17:22:22Z\",\"html_url\":\"https:\\/\\/github.
        com\\/google\\/guava\\/issues\\/736\",\"id\":47420734,\"state\":\"open
        \",\"assignee\":null,\"locked\":false}",
    "Message-Header": {
      "X-Accepted-OAuth-Scopes": "public_repo, repo",
      "Server": "GitHub.com",
      "Access-Control-Allow-Origin": "*",
      "Date": "Fri, 31 Oct 2014 17:22:22 GMT",
      "Access-Control-Expose-Headers": "ETag, X-OAuth-Scopes, X-Accepted-OAuth-
          Scopes",
      "Cache-Control": "private, max-age=60",
      "ETag": "44210f0e808dc4eaf31dd49631a71979",
      "X-GitHub-Media-Type": "github.v3; format=json",
      "Vary": "Accept, Authorization, Cookie",
      "Content-Length": 1004,
      "X-OAuth-Scopes": "public_repo",
      "Content-Type": "application/json; charset=utf-8",
      "X-GitHub-Request-Id": "7B12:CABC5:5FCE305:F7EDEF9:BDA312D8",
      "Location": "https://api.github.com/repos/google/guava/issues/736"
    },
    "Status-Code": 201,
    "HTTP-Version": "HTTP/1.1",
    "Reason-Phrase": "Created"
  },
  "Meta-Data": {
    "source": "GHTorrent",
    "type": "real-world",
    "processor": "nz.ac.massey.ghtraffic.scripts.extractor.
      GHTorrentTransactionFactoryForIssueCreation",
    "timestamp": "Mon, 04 Dec 2017 00:08:38 GMT"
  }
}
```

A.2.2 PATCH HTTP Transaction (Update Issue 736)

```
{
  "Request": {
    "Message-Body": "{\"state\":\"closed\"}",
    "Message-Header": {
      "Authorization": "token 1d67ee5c6a67127672b42bdfa1528111e03f2d74",
      "Accept": "*/*",
      "User-Agent": "Opera/9.27 (Windows NT 5.1; U; en)",
      "Host": "api.github.com",
      "Content-Length": 55,
      "Content-Type": "application/json; charset=utf-8"
    },
    "Request-URI": "/repos/google/guava/issues/736",
    "Method": "PATCH",
    "HTTP-Version": "HTTP/1.1"
  },
  "Response": {
    "Message-Body": "{\"comments\":5,\"closed_at\":\"2014-10-31T20:08:23Z\",\"
        user\":{\"login\":\"gissuebot\",\"id\":8091570,\"avatar_url\":\"https
        :\\/\\/avatars.githubusercontent.com\\/u\\/8091570?v=3\",\"gravatar_id
        \":\"\",\"url\":\"https:\\/\\/api.github.com\\/users\\/gissuebot\",\"
        html_url\":\"https:\\/\\/github.com\\/gissuebot\",\"followers_url\":\"
        https:\\/\\/api.github.com\\/users\\/gissuebot\\/followers\",\"
        following_url\":\"https:\\/\\/api.github.com\\/users\\/gissuebot\\/
        following{\\/other_user}\",\"gists_url\":\"https:\\/\\/api.github.com\\/
        users\\/gissuebot\\/gists{\\/gist_id}\",\"starred_url\":\"https:\\/\\/api
        .github.com\\/users\\/gissuebot\\/starred{\\/owner}{\\/repo}\",\"
        subscriptions_url\":\"https:\\/\\/api.github.com\\/users\\/gissuebot\\/
        subscriptions\",\"organizations_url\":\"https:\\/\\/api.github.com\\/
        users\\/gissuebot\\/orgs\",\"repos_url\":\"https:\\/\\/api.github.com\\/
        users\\/gissuebot\\/repos\",\"events_url\":\"https:\\/\\/api.github.com
        \\/users\\/gissuebot\\/events{\\/privacy}\",\"received_events_url\":\"
        https:\\/\\/api.github.com\\/users\\/gissuebot\\/received_events\",\"type
        \":\"User\",\"site_admin\":false},\"assignees\":[],\"created_at
        \":\"2014-10-31T17:22:22Z\",\"title\":\"Multimaps.newListMultimap should
        document that map must be empty\",\"body\":\"..If the map passed to
        Multimaps.newListMultimap is not empty an IllegalArgumentException is
        thrown..\",\"url\":\"https:\\/\\/api.github.com\\/repos\\/google\\/guava
        \\/issues\\/736\",\"labels\":[{\"url\":\"https:\\/\\/api.github.com\\/
        repos\\/google\\/guava\\/labels\\/status%3A+fixed\",\"name\":\"status:
        fixed\",\"color\":\"009800\"}],\"labels_url\":\"https:\\/\\/api.github.
        com\\/repos\\/google\\/guava\\/issues\\/736\\/labels{\\/name}\",\"number
        \":736,\"milestone\":{\"url\":\"https:\\/\\/api.github.com\\/repos\\/
        google\\/guava\\/milestones\\/6\",\"labels_url\":\"https:\\/\\/api.github
        .com\\/repos\\/google\\/guava\\/milestones\\/6\\/labels\",\"id
        \":849118,\"number\":6,\"title\":\"11.0\",\"description\":\"\",\"creator
        \":{..another user},\"open_issues\":0,\"closed_issues\":25,\"state\":\"
        closed\",\"created_at\":\"2014-11-01T03:46:46Z\",\"updated_at
        \":\"2014-11-06T23:12:24Z\",\"due_on\":null,\"closed_at\":\"2014-11-06T23
        :12:24Z\"},\"updated_at\":\"2014-10-31T20:08:23Z\",\"html_url\":\"https
        :\\/\\/github.com\\/google\\/guava\\/issues\\/736\",\"id\":47420734,\"
        state\":\"closed\",\"assignee\":null,\"locked\":false}",
    "Message-Header": {
      "X-Accepted-OAuth-Scopes": "public_repo, repo",
      "Server": "GitHub.com",
      "Access-Control-Allow-Origin": "*",
      "Date": "Fri, 31 Oct 2014 20:08:23 GMT",
      "Access-Control-Expose-Headers": "ETag, X-OAuth-Scopes, X-Accepted-OAuth-
          Scopes",
      "Cache-Control": "private, max-age=60",
      "ETag": "e190e34f817af76bee13b2c45eae724c",
      "X-GitHub-Media-Type": "github.v3; format=json",
      "Vary": "Accept, Authorization, Cookie",
      "Content-Length": 997,
      "X-OAuth-Scopes": "public_repo",
      "Content-Type": "application/json; charset=utf-8",
      "X-GitHub-Request-Id": "86A0:A40B3:2BC9CE2:1899064:33066C43"
    },
    "Status-Code": 200,
    "HTTP-Version": "HTTP/1.1",
    "Reason-Phrase": "OK"
  },
  "Meta-Data": {
    "source": "GHTorrent",
    "type": "real-world",
    "processor": "nz.ac.massey.ghtraffic.scripts.extractor.
        GHTorrentTransactionFactoryForIssueClosing",
    "timestamp": "Mon, 04 Dec 2017 20:17:54 GMT"
  }
}
```

A.2.3 GET HTTP Transaction (Issue 736)

```
{
  "Request": {
    "Message-Header": {
      "Authorization": "token 90b10e0c3cce4aca8215d75d11261af364b0d968",
      "Accept": "*/*",
      "User-Agent": "Opera/9.27 (Windows NT 5.1; U; en)",
      "Host": "api.github.com"
    },
    "Request-URI": "/repos/google/guava/issues/736",
    "Method": "GET",
    "HTTP-Version": "HTTP/1.1"
  },
  "Response": {
    "Message-Body": "{\"comments\":5,\"closed_at\":\"2014-10-31T20:08:23Z\",\"
        created_at\":\"2014-10-31T17:22:22Z\",\"user\":{\"login\":\"gissuebot
        \",\"id\":8091570,\"avatar_url\":\"https:\\/\\/avatars.githubusercontent.
        com\\/u\\/8091570?v=3\",\"gravatar_id\":\"\",\"url\":\"https:\\/\\/api.
        github.com\\/users\\/gissuebot\",\"html_url\":\"https:\\/\\/github.com\\/
        gissuebot\",\"followers_url\":\"https:\\/\\/api.github.com\\/users\\/
        gissuebot\\/followers\",\"following_url\":\"https:\\/\\/api.github.com\\/
        users\\/gissuebot\\/following{\\/other_user}\",\"gists_url\":\"https
        :\\/\\/api.github.com\\/users\\/gissuebot\\/gists{\\/gist_id}\",\"
        starred_url\":\"https:\\/\\/api.github.com\\/users\\/gissuebot\\/starred
        {\\/owner}{\\/repo}\",\"subscriptions_url\":\"https:\\/\\/api.github.com
        \\/users\\/gissuebot\\/subscriptions\",\"organizations_url\":\"https
        :\\/\\/api.github.com\\/users\\/gissuebot\\/orgs\",\"repos_url\":\"https
        :\\/\\/api.github.com\\/users\\/gissuebot\\/repos\",\"events_url\":\"
        https:\\/\\/api.github.com\\/users\\/gissuebot\\/events{\\/privacy}\",\"
        received_events_url\":\"https:\\/\\/api.github.com\\/users\\/gissuebot\\/
        received_events\",\"type\":\"User\",\"site_admin\":false},\"title\":\"
        Multimaps.newListMultimap should document that map must be empty\",\"body
        \":\"..If the map passed to Multimaps.newListMultimap is not empty an
        IllegalArgumentException is thrown..\",\"url\":\"https:\\/\\/api.github.
        com\\/repos\\/google\\/guava\\/issues\\/736\",\"labels\":[{\"url\":\"
        https:\\/\\/api.github.com\\/repos\\/google\\/guava\\/labels\\/status%3A+
        fixed\",\"name\":\"status: fixed\",\"color\":\"009800\"}],\"labels_url
        \":\"https:\\/\\/api.github.com\\/repos\\/google\\/guava\\/issues
        \\/736\\/labels{\\/name}\",\"number\":736,\"milestone\":{\"url\":\"https
        :\\/\\/api.github.com\\/repos\\/google\\/guava\\/milestones\\/6\",\"
        labels_url\":\"https:\\/\\/api.github.com\\/repos\\/google\\/guava\\/
        milestones\\/6\\/labels\",\"id\":849118,\"number\":6,\"title
        \":\"11.0\",\"description\":\"\",\"creator\":{..another user},\"
        open_issues\":0,\"closed_issues\":25,\"state\":\"closed\",\"created_at
        \":\"2014-11-01T03:46:46Z\",\"updated_at\":\"2014-11-06T23:12:24Z\",\"
        due_on\":null,\"closed_at\":\"2014-11-06T23:12:24Z\"},\"updated_at
        \":\"2014-11-01T03:49:40Z\",\"html_url\":\"https:\\/\\/github.com\\/
        google\\/guava\\/issues\\/736\",\"id\":47420734,\"state\":\"closed\",\"
        assignee\":null,\"locked\":false}",
    "Message-Header": {
      "X-Accepted-OAuth-Scopes": "public_repo, repo",
      "Server": "GitHub.com",
      "Access-Control-Allow-Origin": "*",
      "Last-Modified": "Sat, 01 Nov 2014 03:49:40 GMT",
      "Date": "Wed, 25 Nov 2015 13:42:36 GMT",
      "Access-Control-Expose-Headers": "ETag, X-OAuth-Scopes, X-Accepted-OAuth-
          Scopes",
      "Cache-Control": "private, max-age=60",
      "ETag": "5d71e477bfd135cca5f6ea0b795de667",
      "X-GitHub-Media-Type": "github.v3; format=json",
      "Vary": "Accept, Authorization, Cookie",
      "Content-Length": 951,
      "X-OAuth-Scopes": "public_repo",
      "Content-Type": "application/json; charset=utf-8",
      "X-GitHub-Request-Id": "5CB3:3F9CC:9AD4FE3:9B17BC8:5B3E2854"
    },
    "Status-Code": 200,
    "HTTP-Version": "HTTP/1.1",
    "Reason-Phrase": "OK"
  },
  "Meta-Data": {
    "source": "GHTorrent",
    "type": "synthetic",
    "processor": "nz.ac.massey.ghtraffic.scripts.generator.
        CreateGETTransactionsForIssueListing",
    "timestamp": "Sun, 03 Dec 2017 23:15:56 GMT"
  }
}
```

A.2.4 HEAD HTTP Transaction (Issue 736)

```
{
  "Request":{
    "Message-Header":{
      "Authorization":"token af8bd688c0d7855fd822fde931f2682e46d167e2",
      "Accept":"*/*",
      "User-Agent":"Mozilla/5.0 (Macintosh; U; Intel Mac OS X; en) AppleWebKit
          /418.9 (KHTML, like Gecko) Safari/419.3",
      "Host":"api.github.com"
    },
    "Request-URI":"/repos/google/guava/issues/736",
    "Method":"HEAD",
    "HTTP-Version":"HTTP/1.1"
  },
  "Response":{
    "Message-Header":{
      "X-Accepted-OAuth-Scopes":"public_repo, repo",
      "Server":"GitHub.com",
      "Access-Control-Allow-Origin":"*",
      "Last-Modified":"Sat, 01 Nov 2014 03:49:40 GMT",
      "Date":"Mon, 09 Mar 2015 23:28:26 GMT",
      "Access-Control-Expose-Headers":"ETag, X-OAuth-Scopes, X-Accepted-OAuth-
          Scopes",
      "Cache-Control":"private, max-age=60",
      "ETag":"2e98b04cdd292be895f5827952476987",
      "X-GitHub-Media-Type":"github.v3; format=json",
      "Vary":"Accept, Authorization, Cookie",
      "Content-Length":945,
      "X-OAuth-Scopes":"public_repo",
      "Content-Type":"application/json; charset=utf-8",
      "X-GitHub-Request-Id":"982C:D7AEC:96E9240:6D2608F:7EC9768F"
    },
    "Status-Code":200,
    "HTTP-Version":"HTTP/1.1",
    "Reason-Phrase":"OK"
  },
  "Meta-Data":{
    "source":"GHTorrent",
    "type":"synthetic",
    "processor":"nz.ac.massey.ghtraffic.scripts.generator.
        CreateHEADTransactionsForGettingHeaderInfo",
    "timestamp":"Tue, 05 Dec 2017 01:07:24 GMT"
  }
}
```

A.2.5 PUT HTTP Transaction (Lock Issue 736)

```
{
  "Request":{
    "Message-Header":{
      "Authorization":"token cdb7938ef6d53a8098e86d7acaf68f2452cbbb3f",
      "Accept":"*/*",
      "User-Agent":"Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9) Gecko
          /2008052906 Firefox/3.0",
      "Host":"api.github.com"
    },
    "Request-URI":"/repos/google/guava/issues/736/lock",
    "Method":"PUT",
    "HTTP-Version":"HTTP/1.1"
  },
  "Response":{
    "Message-Header":{
      "X-Accepted-OAuth-Scopes":"public_repo, repo",
      "Access-Control-Expose-Headers":"ETag, X-OAuth-Scopes, X-Accepted-OAuth-
          Scopes",
      "Server":"GitHub.com",
      "X-GitHub-Media-Type":"github.v3; format=json",
      "Access-Control-Allow-Origin":"*",
      "X-OAuth-Scopes":"public_repo",
      "Date":"Fri, 17 Apr 2015 14:06:03 GMT",
      "Content-Type":"application/json; charset=utf-8",
      "X-GitHub-Request-Id":"F004:E8FD6:D25CA40:565CCA2:D7A986D6"
    },
    "Status-Code":204,
    "HTTP-Version":"HTTP/1.1",
    "Reason-Phrase":"No Content"
  },
  "Meta-Data":{
    "source":"GHTorrent",
    "type":"synthetic",
    "processor":"nz.ac.massey.ghtraffic.scripts.generator.
        CreatePUTTransactionsForIssueLocking",
    "timestamp":"Tue, 05 Dec 2017 02:56:24 GMT"
  }
}
```

A.2.6 DELETE HTTP Transaction (Unlock Issue 736)

```
{
  "Request":{
    "Message-Header":{
      "Authorization":"token 90e94922c2533e6fc0a44dcb9674d38095e37a83",
      "Accept":"*/*",
      "User-Agent":"Mozilla/5.0 (Macintosh; U; Intel Mac OS X; en) AppleWebKit
          /418.9 (KHTML, like Gecko) Safari/419.3",
      "Host":"api.github.com"
    },
    "Request-URI":"/repos/google/guava/issues/736/lock",
    "Method":"DELETE",
    "HTTP-Version":"HTTP/1.1"
  },
  "Response":{
    "Message-Header":{
      "X-Accepted-OAuth-Scopes":"public_repo, repo",
      "Access-Control-Expose-Headers":"ETag, X-OAuth-Scopes, X-Accepted-OAuth-
          Scopes",
      "Server":"GitHub.com",
      "X-GitHub-Media-Type":"github.v3; format=json",
      "Access-Control-Allow-Origin":"*",
      "X-OAuth-Scopes":"public_repo",
      "Date":"Wed, 26 Aug 2015 22:14:09 GMT",
      "Content-Type":"application/json; charset=utf-8",
      "X-GitHub-Request-Id":"E2EE:1C0CC:C9CB561:E1B633E:508E8422"
    },
    "Status-Code":204,
    "HTTP-Version":"HTTP/1.1",
    "Reason-Phrase":"No Content"
  },
  "Meta-Data":{
    "source":"GHTorrent",
    "type":"synthetic",
    "processor":"nz.ac.massey.ghtraffic.scripts.generator.
        CreateDELETETransactionsForIssueUnlocking",
    "timestamp":"Mon, 04 Dec 2017 21:40:48 GMT"
  }
}
```

A.3 Sample Records on Twitter Dataset

A.3.1 Create Tweet

```
<httpSample>
   <responseHeader class="java.lang.String">HTTP/1.1 200 OK
   cache-control: no-cache, no-store, must-revalidate, pre-check=0, post-check=0
   content-disposition: attachment; filename=json.json
   content-length: 2240
   content-type: application/json;charset=utf-8
   date: Tue, 12 Jun 2018 01:28:57 GMT
   expires: Tue, 31 Mar 1981 05:00:00 GMT
   last-modified: Tue, 12 Jun 2018 01:28:56 GMT
   pragma: no-cache
   server: tsa_l
   set-cookie: personalization_id="v1_+lt2VvjC7bCOuKVAD3PJlQ=="; Expires=Thu, 11
       Jun 2020 01:28:56 GMT; Path=/; Domain=.twitter.com
   set-cookie: lang=en; Path=/
   set-cookie: guest_id=v1%3A152876693686675953; Expires=Thu, 11 Jun 2020 01
       :28:56 GMT; Path=/; Domain=.twitter.com
   status: 200 OK
   strict-transport-security: max-age=631138519
   x-access-level: read-write-directmessages
   x-connection-hash: 1f061e276683143e95b32011c18a0f45
   x-content-type-options: nosniff
   x-frame-options: SAMEORIGIN
   x-response-time: 263
   x-transaction: 0059ae40006deb8e
   x-tsa-request-body-time: 0
   x-twitter-response-tags: BouncerCompliant
   x-xss-protection: 1; mode=block; report=https://twitter.com/i/xss_report
   </responseHeader>
   <requestHeader class="java.lang.String">Connection: keep-alive
   Authorization: OAuth oauth_consumer_key="bicHvLbtKnuU9tFomq3grx7GI",
       oauth_signature_method="HMAC-SHA1",oauth_timestamp="1528766855",
       oauth_nonce="37a0188427b640b7a5ff432fd26819e8",oauth_version="1.0",
       oauth_signature="I90GLGyUGX5RuJk4oqusO3Fci2c%3D",oauth_token="517417816-
       bQOa8fStCdHsHB4pt8uYsELEdyUO0E6NN3qOhY8z"
   Content-Type: application/x-www-form-urlencoded
   Content-Length: 17
   Host: api.twitter.com
   User-Agent: Apache-HttpClient/4.5.5 (Java/1.8.0_131)
   </requestHeader>
   <responseData class="java.lang.String">{
      "created_at":"Tue Jun 12 01:28:56 +0000 2018",
      "id":1006347588856045569,
      "id_str":"1006347588856045569",
      "text":"yywSNRtUzC",
      "truncated":false,
"entities":{
   "hashtags": [],
   "symbols":[],
   "user_mentions":[],
   "urls":[]
},
"source":"<a href="https://sudam.co.nz" rel="nofollow">MockTweetsApp</a>",
"in_reply_to_status_id":null,
"in_reply_to_status_id_str":null,
"in_reply_to_user_id":null,
"in_reply_to_user_id_str":null,
"in_reply_to_screen_name":null,
"user":{
   "id":517417816,
   "id_str":"517417816",
   "name":"Thilini Bhagya",
   "screen_name":"bhagyasl",
   "location":"Kurunegala",
   "description":"",
   "url":"http://t.co/sQduiwqJiy",
   "entities":{
      "url":{
         "urls":[
         {
            "url":"http://t.co/sQduiwqJiy",
            "expanded_url":"http://inkedin.com/in/bhagyasl/",
            "display_url":"inkedin.com/in/bhagyasl/",
            "indices":[
               0,22
            ]
         }
      ]
   },
   "description":{
      "urls":[]
      }
   },
   "protected":false,
   "followers_count":185,
   "friends_count":249,
   "listed_count":3,
   "created_at":"Wed Mar 07 09:41:33 +0000 2012",
   "favourites_count":64,
   "utc_offset":null,
   "time_zone":null,
   "geo_enabled":true,
   "verified":false,
   "statuses_count":308,
   "lang":"en",
   "contributors_enabled":false,
   "is_translator":false,
   "is_translation_enabled":false,
   "profile_background_color":"1A1B1F",
   "profile_background_image_url":"http://abs.twimg.com/images/themes/
       theme9/bg.gif",
   "profile_background_image_url_https":"https://abs.twimg.com/images/
       themes/theme9/bg.gif",
   "profile_background_tile":false,
   "profile_image_url":"http://pbs.twimg.com/profile_images
       /950100192048562176/LKr7Ay21_normal.jpg",
   "profile_image_url_https":"https://pbs.twimg.com/profile_images
       /950100192048562176/LKr7Ay21_normal.jpg",
   "profile_banner_url":"https://pbs.twimg.com/profile_banners
       /517417816/1399047954",
   "profile_link_color":"3E4547",
   "profile_sidebar_border_color":"FFFFFF",
   "profile_sidebar_fill_color":"252429",
   "profile_text_color":"666666",
   "profile_use_background_image":true,
   "has_extended_profile":false,
   "default_profile":false,
   "default_profile_image":false,
   "following":false,
        "follow_request_sent":false,
        "notifications":false,
        "translator_type":"none"
      },
      "geo":null,
      "coordinates":null,
      "place":null,
      "contributors":null,
      "is_quote_status":false,
      "retweet_count":0,
      "favorite_count":0,
      "favorited":false,
      "retweeted":false,
      "lang":"eu"
    }
    </responseData>
    <cookies class="java.lang.String"></cookies>
    <method class="java.lang.String">POST</method>
    <queryString class="java.lang.String">status=yywSNRtUzC</queryString>
    <java.net.URL>https://api.twitter.com/1.1/statuses/update.json</java.net.URL>
</httpSample>
```

A.3.2 Retrieve Tweet 1006347588856045569

```
<httpSample>
  <responseHeader class="java.lang.String">HTTP/1.1 200 OK
  cache-control: no-cache, no-store, must-revalidate, pre-check=0, post-check=0
  content-disposition: attachment; filename=json.json
  content-length: 2240
  content-type: application/json;charset=utf-8
  date: Tue, 12 Jun 2018 02:43:07 GMT
  expires: Tue, 31 Mar 1981 05:00:00 GMT
  last-modified: Tue, 12 Jun 2018 02:43:07 GMT
  pragma: no-cache
  server: tsa_l
  set-cookie: personalization_id="v1_Va+j+c1nPX4SOFuU51pnXg=="; Expires=Thu, 11
      Jun 2020 02:43:07 GMT; Path=/; Domain=.twitter.com
  set-cookie: lang=en; Path=/
  set-cookie: guest_id=v1%3A152877138757955854; Expires=Thu, 11 Jun 2020 02:43:07
       GMT; Path=/; Domain=.twitter.com
  status: 200 OK
  strict-transport-security: max-age=631138519
  x-access-level: read-write-directmessages
  x-connection-hash: d50dff659e3101d499b3443da33db09e
  x-content-type-options: nosniff
  x-frame-options: SAMEORIGIN
  x-rate-limit-limit: 900
  x-rate-limit-remaining: 898
  x-rate-limit-reset: 1528772287
  x-response-time: 189
  x-transaction: 002789b80029c96c
  x-twitter-response-tags: BouncerCompliant
  x-xss-protection: 1; mode=block; report=https://twitter.com/i/xss_report
  </responseHeader>
  <requestHeader class="java.lang.String">Connection: keep-alive
  Authorization: OAuth oauth_consumer_key="bicHvLbtKnuU9tFomq3grx7GI",
      oauth_signature_method="HMAC-SHA1",oauth_timestamp="1528771283",oauth_nonce
      ="5dc53671321d4f2f8cff0267df1953c9",oauth_version="1.0",oauth_signature="
      Vn9ZyiFUDG0hmzAn2edxCqFNoBU%3D",oauth_token="517417816-
      bQ0a8fStCdHsHB4pt8uYsELEdyUO0E6NN3qOhY8z"
  Host: api.twitter.com
  User-Agent: Apache-HttpClient/4.5.5 (Java/1.8.0_131)
  </requestHeader>
  <responseData class="java.lang.String">{
    "created_at":"Tue Jun 12 01:28:56 +0000 2018",
    "id":1006347588856045569,
    "id_str":"1006347588856045569",
    "text":"yywSNRtUzC",
    "truncated":false,
    "entities":{
      "hashtags":[],
      "symbols":[],
  "user_mentions":[],
  "urls":[]
},
"source":"<a href="https://sudam.co.nz" rel="nofollow">MockTweetsApp</a>",
"in_reply_to_status_id":null,
"in_reply_to_status_id_str":null,
"in_reply_to_user_id":null,
"in_reply_to_user_id_str":null,
"in_reply_to_screen_name":null,
"user":{
  "id":517417816,
  "id_str":"517417816",
  "name":"Thilini Bhagya",
  "screen_name":"bhagyasl",
  "location":"Kurunegala",
  "description":"",
  "url":"http://t.co/sQduiwqJiy",
  "entities":{
    "url":{
      "urls":[
      {
        "url":"http://t.co/sQduiwqJiy",
        "expanded_url":"http://inkedin.com/in/bhagyasl/",
        "display_url":"inkedin.com/in/bhagyasl/",
        "indices":[
          0,22
        ]
      }
    ]
  },
  "description":{
    "urls":[]
    }
  },
  "protected":false,
  "followers_count":185,
  "friends_count":249,
  "listed_count":3,
  "created_at":"Wed Mar 07 09:41:33 +0000 2012",
  "favourites_count":64,
  "utc_offset":null,
  "time_zone":null,
  "geo_enabled":true,
  "verified":false,
  "statuses_count":369,
  "lang":"en",
  "contributors_enabled":false,
  "is_translator":false,
  "is_translation_enabled":false,
  "profile_background_color":"1A1B1F",
  "profile_background_image_url":"http://abs.twimg.com/images/themes/theme9/
     bg.gif",
  "profile_background_image_url_https":"https://abs.twimg.com/images/themes/
     theme9/bg.gif",
  "profile_background_tile":false,
  "profile_image_url":"http://pbs.twimg.com/profile_images
     /950100192048562176/LKr7Ay21_normal.jpg",
  "profile_image_url_https":"https://pbs.twimg.com/profile_images
     /950100192048562176/LKr7Ay21_normal.jpg",
  "profile_banner_url":"https://pbs.twimg.com/profile_banners
     /517417816/1399047954",
  "profile_link_color":"3E4547",
  "profile_sidebar_border_color":"FFFFFF",
  "profile_sidebar_fill_color":"252429",
  "profile_text_color":"666666",
  "profile_use_background_image":true,
  "has_extended_profile":false,
  "default_profile":false,
  "default_profile_image":false,
  "following":false,
  "follow_request_sent":false,
  "notifications":false,
  "translator_type":"none"
```

```
    },
    "geo":null,
    "coordinates":null,
    "place":null,
    "contributors":null,
    "is_quote_status":false,
    "retweet_count":0,
    "favorite_count":0,
    "favorited":false,
    "retweeted":false,
    "lang":"eu"
  }
  </responseData>
  <cookies class="java.lang.String">
  </cookies>
  <method class="java.lang.String">GET</method>
  <queryString class="java.lang.String">
  </queryString>
  <java.net.URL>https://api.twitter.com/1.1/statuses/show.json?id
      =1006347588856045569</java.net.URL>
</httpSample>
```

A.3.3 Delete Tweet 1006347588856045569

```
<httpSample>
  <responseHeader class="java.lang.String">HTTP/1.1 200 OK
  cache-control: no-cache, no-store, must-revalidate, pre-check=0, post-check=0
  content-disposition: attachment; filename=json.json
  content-length: 2240
  content-type: application/json;charset=utf-8
  date: Tue, 12 Jun 2018 02:43:07 GMT
  expires: Tue, 31 Mar 1981 05:00:00 GMT
  last-modified: Tue, 12 Jun 2018 02:43:07 GMT
  pragma: no-cache
  server: tsa_l
  set-cookie: personalization_id="v1_ZDYAbtrmlpVn4HwZ/36aEQ=="; Expires=Thu, 11
      Jun 2020 02:43:07 GMT; Path=/; Domain=.twitter.com
  set-cookie: lang=en; Path=/
  set-cookie: guest_id=v1%3A152877138756118875; Expires=Thu, 11 Jun 2020 02:43:07
       GMT; Path=/; Domain=.twitter.com
  status: 200 OK
  strict-transport-security: max-age=631138519
  x-access-level: read-write-directmessages
  x-connection-hash: 49f40fcd3a367c548d46f9d700c2f394
  x-content-type-options: nosniff
  x-frame-options: SAMEORIGIN
  x-response-time: 210
  x-transaction: 00c122f500a0512a
  x-twitter-response-tags: BouncerCompliant
  x-xss-protection: 1; mode=block; report=https://twitter.com/i/xss_report
  </responseHeader>
  <requestHeader class="java.lang.String">Connection: keep-alive
  Authorization: OAuth oauth_consumer_key="bicHvLbtKnuU9tFomq3grx7GI",
       oauth_signature_method="HMAC-SHA1",oauth_timestamp="1528771162",oauth_nonce
       ="077befedc9bf4d5aa51796db6c0a4a12",oauth_version="1.0",oauth_signature="
       X4tPxkrSV7kAgKy17hWi77ZnekY%3D",oauth_token="517417816-
       bQ0a8fStCdHsHB4pt8uYsELEdyUO0E6NN3qOhY8z"
  Content-Type: application/x-www-form-urlencoded
  Content-Length: 0
  Host: api.twitter.com
  User-Agent: Apache-HttpClient/4.5.5 (Java/1.8.0_131)
  </requestHeader>
  <responseData class="java.lang.String">{
    .. same response data as previous record
  }
  </responseData>
  <cookies class="java.lang.String">
  </cookies>
  <method class="java.lang.String">POST</method>
  <queryString class="java.lang.String">
  </queryString>
  <java.net.URL>https://api.twitter.com/1.1/statuses/destroy/1006347588856045569.
```

```
    json</java.net.URL>
</httpSample>
```

A.4 Sample Records on Google Tasks Dataset

A.4.1 Create List

```
<httpSample>
  <responseHeader class="java.lang.String">HTTP/1.1 200 OK
  Cache-Control: no-cache, no-store, max-age=0, must-revalidate
  Pragma: no-cache
  Expires: Mon, 01 Jan 1990 00:00:00 GMT
  Date: Mon, 27 Aug 2018 22:26:26 GMT
  Vary: X-Origin
  Content-Type: application/json; charset=UTF-8
  X-Content-Type-Options: nosniff
  X-Frame-Options: SAMEORIGIN
  X-XSS-Protection: 1; mode=block
  Server: GSE
  Alt-Svc: quic=":443"; ma=2592000; v="44,43,39,35"
  Accept-Ranges: none
  Vary: Origin,Accept-Encoding
  Transfer-Encoding: chunked
  </responseHeader>
  <requestHeader class="java.lang.String">Connection: keep-alive
  Authorization: Bearer ya29.GlsGBl1zrjbnkq3md8zOcY5ckJFt42xg2YfRFbL8gHLVwLiHSb-
      bIAdlzp0d9VLxkqSLDXASS5MxM3RB2Qf2iudWSQLBxOyyZ_W7DuYgqsA78BnojJyS_RnveXXL
  Content-Type: application/json
  Content-Length: 24
  Host: www.googleapis.com
  User-Agent: Apache-HttpClient/4.5.5 (Java/1.8.0_131)
  </requestHeader>
  <responseData class="java.lang.String">{
    "kind": "tasks#taskList",
    "id": "MTcyMDU1OTI5NDQ0MDQ0ODg3NjI6NDkwODMyNjA4MDE0NTUyMTow",
    "etag": "\"FhCqMAsBrrKDkDLKevwtJykQ9I8/jrhhSpFMBCqafUWVdyMd0X4tB2U\"",
    "title": "VSUmpg",
    "updated": "2018-08-27T22:26:26.000Z",
    "selfLink": "https://www.googleapis.com/tasks/v1/users/@me/lists/
        MTcyMDU1OTI5NDQ0MDQ0ODg3NjI6NDkwODMyNjA4MDE0NTUyMTow"
  }
  </responseData>
  <responseFile class="java.lang.String"></responseFile>
  <cookies class="java.lang.String"></cookies>
  <method class="java.lang.String">POST</method>
  <queryString class="java.lang.String">{
    "title": "VSUmpg"
  }
  </queryString>
  <java.net.URL>https://www.googleapis.com/tasks/v1/users/@me/lists</java.net.URL
      >
</httpSample>
```

A.4.2 Return List MTcyMDU1OTI5NDQ0MDQ0ODg3NjI6NDkwODMyNjA4MDE0NTUyMTow

```
<httpSample>
  <responseHeader class="java.lang.String">HTTP/1.1 200 OK
  Expires: Mon, 27 Aug 2018 23:56:57 GMT
  Date: Mon, 27 Aug 2018 23:56:57 GMT
  Cache-Control: private, max-age=0, must-revalidate, no-transform
  ETag: &quot;FhCqMAsBrrKDkDLKevwtJykQ9I8/jrhhSpFMBCqafUWVdyMd0X4tB2U&quot;
  Vary: Origin
  Vary: X-Origin
  Content-Type: application/json; charset=UTF-8
  X-Content-Type-Options: nosniff
  X-Frame-Options: SAMEORIGIN
  X-XSS-Protection: 1; mode=block
  Content-Length: 346
  Server: GSE
  Alt-Svc: quic=":443"; ma=2592000; v="44,43,39,35"
  </responseHeader>
  <requestHeader class="java.lang.String">Connection: keep-alive
  Host: www.googleapis.com
  User-Agent: Apache-HttpClient/4.5.5 (Java/1.8.0_131)
  Authorization: Bearer ya29.GlsGBl1zrjbnkq3md8zOcY5ckJFt42xg2YfRFbL8gHLVwLiHSb-
      bIAdlzp0d9VLxkqSLDXASS5MxM3RB2Qf2iudWSQLBxOyyZ_W7DuYgqsA78BnojJyS_RnveXXL
  </requestHeader>
  <responseData class="java.lang.String">{
    "kind": "tasks#taskList",
    "id": "MTcyMDU1OTI5NDQ0MDQ0ODg3NjI6NDkwODMyNjA4MDE0NTUyMTow",
    "etag": "\"FhCqMAsBrrKDkDLKevwtJykQ9I8/jrhhSpFMBCqafUWVdyMd0X4tB2U\"",
    "title": "VSUmpg",
    "updated": "2018-08-27T22:26:26.000Z",
    "selfLink": "https://www.googleapis.com/tasks/v1/users/@me/lists/
        MTcyMDU1OTI5NDQ0MDQ0ODg3NjI6NDkwODMyNjA4MDE0NTUyMTow"
  }
  </responseData>
  <responseFile class="java.lang.String"></responseFile>
  <cookies class="java.lang.String"></cookies>
  <method class="java.lang.String">GET</method>
  <queryString class="java.lang.String"></queryString>
  <java.net.URL>https://www.googleapis.com/tasks/v1/users/@me/lists/
      MTcyMDU1OTI5NDQ0MDQ0ODg3NjI6NDkwODMyNjA4MDE0NTUyMTow</java.net.URL>
</httpSample>
```

A.4.3 Update List MTcyMDU1OTI5NDQ0MDQ0ODg3NjI6NDkwODMyNjA4MDE0NTUyMTow

```
<httpSample>
  <responseHeader class="java.lang.String">HTTP/1.1 200 OK
  Cache-Control: no-cache, no-store, max-age=0, must-revalidate
  Pragma: no-cache
  Expires: Mon, 01 Jan 1990 00:00:00 GMT
  Date: Tue, 28 Aug 2018 04:20:24 GMT
  Vary: X-Origin
  Content-Type: application/json; charset=UTF-8
  X-Content-Type-Options: nosniff
  X-Frame-Options: SAMEORIGIN
  X-XSS-Protection: 1; mode=block
  Server: GSE
  Alt-Svc: quic=":443"; ma=2592000; v="44,43,39,35"
  Accept-Ranges: none
  Vary: Origin,Accept-Encoding
  Transfer-Encoding: chunked
  </responseHeader>
  <requestHeader class="java.lang.String">Connection: keep-alive
  Authorization: Bearer ya29.GlsHBm6kl2pDgXcSc8dSUBfRI0oDWSTR3tMw_wrzzSsRnm-
      HY1bN6T33vOmx025iVBd18Ayfg_pW8MMTb4xsdaAnapfc9TB5ab6ODvVVTFU789V_SA-RXp4f-
      qnK
  Content-Type: application/json
  Content-Length: 24
  Host: www.googleapis.com
  User-Agent: Apache-HttpClient/4.5.5 (Java/1.8.0_131)
  </requestHeader>
  <responseData class="java.lang.String">{
    "kind": "tasks#taskList",
    "id": "MTcyMDU1OTI5NDQ0MDQ0ODg3NjI6NDkwODMyNjA4MDE0NTUyMTow",
    "etag": "\"FhCqMAsBrrKDkDLKevwtJykQ9I8/Gv7_Jj_E1F9qe_U0rP0pyxSA4aE\"",
    "title": "PYUbjq",
    "updated": "2018-08-28T04:20:24.000Z",
    "selfLink": "https://www.googleapis.com/tasks/v1/users/@me/lists/
        MTcyMDU1OTI5NDQ0MDQ0ODg3NjI6NDkwODMyNjA4MDE0NTUyMTow"
  }
  </responseData>
  <responseFile class="java.lang.String"></responseFile>
  <cookies class="java.lang.String"></cookies>
  <method class="java.lang.String">PATCH</method>
  <queryString class="java.lang.String">{
    "title": "PYUbjq"
  }
  </queryString>
  <java.net.URL>https://www.googleapis.com/tasks/v1/users/@me/lists/
      MTcyMDU1OTI5NDQ0MDQ0ODg3NjI6NDkwODMyNjA4MDE0NTUyMTow</java.net.URL>
</httpSample>
```

A.4.4 Delete List MTcyMDU1OTI5NDQ0MDQ0ODg3NjI6NDkwODMyNjA4MDE0NTUyMTow

```
<httpSample>
  <responseHeader class="java.lang.String">HTTP/1.1 204 No Content
  Cache-Control: no-cache, no-store, max-age=0, must-revalidate
  Pragma: no-cache
  Expires: Mon, 01 Jan 1990 00:00:00 GMT
  Date: Tue, 28 Aug 2018 04:21:32 GMT
  ETag: "FhCqMAsBrrKDkDLKevwtJykQ9I8/vyGp6PvFo4RvsFtPoIWeCReyIC8"
  Vary: Origin
  Vary: X-Origin
  Server: GSE
  Alt-Svc: quic=":443"; ma=2592000; v="44,43,39,35"
  </responseHeader>
  <requestHeader class="java.lang.String">Connection: keep-alive
  Authorization: Bearer ya29.GlsHBm6kl2pDgXcSc8dSUBfRIOoDWSTR3tMw_wrzzSsRnm-
      HY1bN6T33vOmx025iVBd18Ayfg_pW8MMTb4xsdaAnapfc9TB5ab6ODvVVTFU789V_SA-RXp4f-
      qnK
  Content-Length: 0
  Host: www.googleapis.com
  User-Agent: Apache-HttpClient/4.5.5 (Java/1.8.0_131)
  </requestHeader>
  <responseData class="java.lang.String"></responseData>
  <responseFile class="java.lang.String"></responseFile>
  <cookies class="java.lang.String"></cookies>
  <method class="java.lang.String">DELETE</method>
  <queryString class="java.lang.String"></queryString>
  <java.net.URL>https://www.googleapis.com/tasks/v1/users/@me/lists/
      MTcyMDU1OTI5NDQ0MDQ0ODg3NjI6NDkwODMyNjA4MDE0NTUyMTow</java.net.URL>
</httpSample>
```

A.5 Sample Records on Slack Dataset

A.5.1 Send Message

```
<httpSample>
  <responseHeader class="java.lang.String">HTTP/1.1 200 OK
  Content-Type: application/json; charset=utf-8
  Transfer-Encoding: chunked
  Connection: keep-alive
  Date: Mon, 27 Aug 2018 06:49:30 GMT
  Server: Apache
  x-slack-router: p
  X-Slack-Req-Id: a27dd9ef-de0a-464b-aeda-264135557811
  X-OAuth-Scopes: identify,channels:write,chat:write:user
  X-Accepted-OAuth-Scopes: chat:write:user
  Expires: Mon, 26 Jul 1997 05:00:00 GMT
  Cache-Control: private, no-cache, no-store, must-revalidate
  Vary: Accept-Encoding
  Pragma: no-cache
  X-XSS-Protection: 0
  X-Content-Type-Options: nosniff
  X-Slack-Exp: 1
  X-Slack-Backend: h
  Referrer-Policy: no-referrer
  Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
  Access-Control-Allow-Origin: *
  X-Via: haproxy-www-d7fx
  X-Cache: Miss from cloudfront
  Via: 1.1 d070a57995f9fd009c96043a24002ef2.cloudfront.net (CloudFront)
  X-Amz-Cf-Id: DnFnpcnDyTAvRDl4JyvJUdesXs1REP4_J2syd5DpaJhQJN2I5L1DWg==
  </responseHeader>
  <requestHeader class="java.lang.String">Connection: keep-alive
  Content-Type: application/x-www-form-urlencoded
  Content-Length: 111
  Host: slack.com
  User-Agent: Apache-HttpClient/4.5.5 (Java/1.8.0_131)
  </requestHeader>
  <responseData class="java.lang.String">{
    "ok":true,
    "channel":"CCGRWTRKQ",
    "ts":"1535352570.000200",
    "message":{
      "type":"message",
      "user":"UC8J6APLN",
      "text":"PmWoE",
      "bot_id":"BCEPNCQDN",
      "ts":"1535352570.000200"
      }
    }</responseData>
  <cookies class="java.lang.String"></cookies>
  <method class="java.lang.String">POST</method>
  <queryString class="java.lang.String">token=xoxp
      -415149719555-416618363702-422804431936-b30a935a430030bec72e399b896b0bcc&
      channel=CCGRWTRKQ&text=PmWoE</queryString>
  <java.net.URL>https://slack.com/api/chat.postMessage</java.net.URL>
</httpSample>
```

A.5.2 Update Message 1535352570.000200

```
<httpSample>
  <responseHeader class="java.lang.String">HTTP/1.1 200 OK
  Content-Type: application/json; charset=utf-8
  Transfer-Encoding: chunked
  Connection: keep-alive
  Date: Mon, 27 Aug 2018 08:18:00 GMT
  Server: Apache
  x-slack-router: p
  X-Slack-Req-Id: 71cef103-2415-4afe-8083-94de9d8d13da
  X-OAuth-Scopes: identify,channels:write,chat:write:user
  X-Accepted-OAuth-Scopes: chat:write:user
  Expires: Mon, 26 Jul 1997 05:00:00 GMT
  Cache-Control: private, no-cache, no-store, must-revalidate
  Vary: Accept-Encoding
  Pragma: no-cache
  X-XSS-Protection: 0
  X-Content-Type-Options: nosniff
  X-Slack-Exp: 1
  X-Slack-Backend: h
  Referrer-Policy: no-referrer
  Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
  Access-Control-Allow-Origin: *
  X-Via: haproxy-www-elvs
  X-Cache: Miss from cloudfront
  Via: 1.1 62b57aad96f95c086d713d3c9a7e766a.cloudfront.net (CloudFront)
  X-Amz-Cf-Id: 4R6I1YPQCOdbn4PaenngVV0rGquNuUnQHM6eGdu9lwrRRVr4FFDkaA==
  </responseHeader>
  <requestHeader class="java.lang.String">Connection: keep-alive
  Content-Type: application/x-www-form-urlencoded
  Content-Length: 133
  Host: slack.com
  User-Agent: Apache-HttpClient/4.5.5 (Java/1.8.0_131)
  </requestHeader>
  <responseData class="java.lang.String">{
    "ok":true,
    "channel":"CCGRWTRKQ",
    "ts":"1535352570.000200",
    "text":"nKMOgn",
    "message":{
      "type":"message",
      "user":"UC8J6APLN",
      "text":"nKMOgn",
      "bot_id":"BCEPNCQDN"
    }
  }
  </responseData>
  <cookies class="java.lang.String"></cookies>
  <method class="java.lang.String">POST</method>
  <queryString class="java.lang.String">token=xoxp
      -415149719555-416618363702-422804431936-b30a935a430030bec72e399b896b0bcc&
      channel=CCGRWTRKQ&ts=1535352570.000200&text=nKMOgn</queryString>
  <java.net.URL>https://slack.com/api/chat.update</java.net.URL>
</httpSample>
```

A.5.3 Delete Message 1535352570.000200

```
<httpSample>
  <responseHeader class="java.lang.String">HTTP/1.1 200 OK
  Content-Type: application/json; charset=utf-8
  Transfer-Encoding: chunked
  Connection: keep-alive
  Date: Mon, 27 Aug 2018 10:52:23 GMT
  Server: Apache
  X-Content-Type-Options: nosniff
  X-Slack-Req-Id: fa5431af-453c-457f-bda0-d01247f2760c
  Expires: Mon, 26 Jul 1997 05:00:00 GMT
  Cache-Control: private, no-cache, no-store, must-revalidate
  X-OAuth-Scopes: identify,channels:write,chat:write:user
  Vary: Accept-Encoding
  Pragma: no-cache
  X-Accepted-OAuth-Scopes: chat:write:user
  X-XSS-Protection: 0
  Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
  Referrer-Policy: no-referrer
  X-Slack-Exp: 1
  X-Slack-Backend: h
  Access-Control-Allow-Origin: *
  X-Via: haproxy-www-npmn
  X-Cache: Miss from cloudfront
  Via: 1.1 d070a57995f9fd009c96043a24002ef2.cloudfront.net (CloudFront)
  X-Amz-Cf-Id: UpadK_fa_fcHooLbmCC--NapQet1A_FCYeF95BxJZklvfGZQXiWqfQ==
  </responseHeader>
  <requestHeader class="java.lang.String">Connection: keep-alive
  Content-Length: 0
  Host: slack.com
  User-Agent: Apache-HttpClient/4.5.5 (Java/1.8.0_131)
  </requestHeader>
  <responseData class="java.lang.String">{
    "ok":true,
    "channel":"CCGRWTRKQ",
    "ts":"1535352570.000200"
  }
  </responseData>
  <cookies class="java.lang.String"></cookies>
  <method class="java.lang.String">POST</method>
  <queryString class="java.lang.String"></queryString>
  <java.net.URL>https://slack.com/api/chat.delete?token=xoxp
      -415149719555-416618363702-422804431936-b30a935a430030bec72e399b896b0bcc&
      channel=CCGRWTRKQ&ts=1535352570.000200</java.net.URL>
</httpSample>
```

Appendix B

Training Data

B.1 Attributes Summary

B.1.1 Attributes for GHTraffic

Attribute	Distinct
RequestMethod	6
ResponseStatusCode	8
RequestUriSchema	1
RequestUriHost	1
RequestUriPathToken1	1
RequestUriPathToken2	1
RequestUriPathToken3	1
RequestUriPathToken4	1
RequestUriPathToken5	1794
RequestUriPathToken6	2
RequestUriQueryToken1	1
RequestUriQueryToken2	1
RequestUriQueryToken3	1
RequestUriQueryToken4	1
RequestUriFragmentToken1	1
RequestUriFragmentToken2	1
HasRequestPayload	2
HasValidRequestPayload	2
HasAuthorisationToken	2
RequestHeader_Accept	1
RequestHeader_Content-Length	29
RequestHeader_Content-Type	2
RequestHeader_Host	1
RequestHeader_User-Agent	5
ResponseHeader_Access-Control-Allow-Origin	1
ResponseHeader_Access-Control-Expose-Headers	1
ResponseHeader_Cache-Control	2
ResponseHeader_Content-Length	39
ResponseHeader_Content-Type	1
ResponseHeader_Date	32210
ResponseHeader_ETag	7633
ResponseHeader_Last-Modified	1024
ResponseHeader_Location	1794
ResponseHeader_Server	1
ResponseHeader_Vary	2
ResponseHeader_X-Accepted-OAuth-Scopes	3
ResponseHeader_X-GitHub-Media-Type	1

ResponseHeader_X-GitHub-Request-Id	32215
ResponseHeader_X-OAuth-Scopes	2
RequestBody_state	2
ResponseBody_assignee.avatar_url	20
ResponseBody_assignee.events_url	11
ResponseBody_assignee.followers_url	11
ResponseBody_assignee.following_url	11
ResponseBody_assignee.gists_url	11
ResponseBody_assignee.gravatar_id	1
ResponseBody_assignee.html_url	11
ResponseBody_assignee.id	11
ResponseBody_assignee.login	11
ResponseBody_assignee.organizations_url	11
ResponseBody_assignee.received_events_url	11
ResponseBody_assignee.repos_url	11
ResponseBody_assignee.site_admin	2
ResponseBody_assignee.starred_url	11
ResponseBody_assignee.subscriptions_url	11
ResponseBody_assignee.type	2
ResponseBody_assignee.url	11
ResponseBody_assignees.avatar_url	20
ResponseBody_assignees.events_url	11
ResponseBody_assignees.followers_url	11
ResponseBody_assignees.following_url	11
ResponseBody_assignees.gists_url	11
ResponseBody_assignees.gravatar_id	1
ResponseBody_assignees.html_url	11
ResponseBody_assignees.id	11
ResponseBody_assignees.login	11
ResponseBody_assignees.organizations_url	11
ResponseBody_assignees.received_events_url	11
ResponseBody_assignees.repos_url	11
ResponseBody_assignees.site_admin	2
ResponseBody_assignees.starred_url	11
ResponseBody_assignees.subscriptions_url	11
ResponseBody_assignees.type	2
ResponseBody_assignees.url	11
ResponseBody_closed_at	1059
ResponseBody_closed_by.avatar_url	4
ResponseBody_closed_by.events_url	4
ResponseBody_closed_by.followers_url	4
ResponseBody_closed_by.following_url	4
ResponseBody_closed_by.gists_url	4
ResponseBody_closed_by.gravatar_id	1
ResponseBody_closed_by.html_url	4
ResponseBody_closed_by.id	4
ResponseBody_closed_by.login	4
ResponseBody_closed_by.organizations_url	4
ResponseBody_closed_by.received_events_url	4
ResponseBody_closed_by.repos_url	4
ResponseBody_closed_by.site_admin	2
ResponseBody_closed_by.starred_url	4
ResponseBody_closed_by.subscriptions_url	4
ResponseBody_closed_by.type	2
ResponseBody_closed_by.url	4
ResponseBody_comments	46
ResponseBody_created_at	1792
ResponseBody_documentation_url	6
ResponseBody_html_url	1794
ResponseBody_id	1794

ResponseBody_locked	3
ResponseBody_message	6
ResponseBody_milestone.closed_at	15
ResponseBody_milestone.closed_issues	13
ResponseBody_milestone.created_at	16
ResponseBody_milestone.creator.avatar_url	3
ResponseBody_milestone.creator.events_url	2
ResponseBody_milestone.creator.followers_url	2
ResponseBody_milestone.creator.following_url	2
ResponseBody_milestone.creator.gists_url	2
ResponseBody_milestone.creator.gravatar_id	1
ResponseBody_milestone.creator.html_url	2
ResponseBody_milestone.creator.id	2
ResponseBody_milestone.creator.login	2
ResponseBody_milestone.creator.organizations_url	2
ResponseBody_milestone.creator.received_events_url	2
ResponseBody_milestone.creator.repos_url	2
ResponseBody_milestone.creator.site_admin	2
ResponseBody_milestone.creator.starred_url	2
ResponseBody_milestone.creator.subscriptions_url	2
ResponseBody_milestone.creator.type	2
ResponseBody_milestone.creator.url	2
ResponseBody_milestone.description	1
ResponseBody_milestone.due_on	2
ResponseBody_milestone.html_url	11
ResponseBody_milestone.id	16
ResponseBody_milestone.labels_url	16
ResponseBody_milestone.number	16
ResponseBody_milestone.open_issues	4
ResponseBody_milestone.state	3
ResponseBody_milestone.title	16
ResponseBody_milestone.updated_at	20
ResponseBody_milestone.url	16
ResponseBody_number	1794
ResponseBody_state	3
ResponseBody_updated_at	2967
ResponseBody_url	1794
ResponseBody_user.avatar_url	137
ResponseBody_user.events_url	135
ResponseBody_user.followers_url	135
ResponseBody_user.following_url	135
ResponseBody_user.gists_url	135
ResponseBody_user.gravatar_id	1
ResponseBody_user.html_url	135
ResponseBody_user.id	134
ResponseBody_user.login	135
ResponseBody_user.organizations_url	135
ResponseBody_user.received_events_url	135
ResponseBody_user.repos_url	135
ResponseBody_user.site_admin	2
ResponseBody_user.starred_url	135
ResponseBody_user.subscriptions_url	135
ResponseBody_user.type	2
ResponseBody_user.url	135
HasImmediatePreviousTransaction	2
HasImmediatePreviousTransactionSucceeded	2
ImmediatelyPreviousStatusCode	9
ImmediatelyPreviousMethod	7
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate	1
HasURLInImmediatelyPreviousTransactionContainsATokenToRead	1

HasURLInImmediatelyPreviousTransactionContainsATokenToUpdate	1
HasURLInImmediatelyPreviousTransactionContainsATokenToDelete	1
HasSuccessfulCreateOperationOccurredBefore	2
HasSuccessfulReadOperationOccurredBefore	2
HasSuccessfulUpdateOperationOccurredBefore	2
HasSuccessfulDeleteOperationOccurredBefore	2

B.1.2 Attributes for Twitter

Attribute	Distinct
RequestMethod	2
ResponseStatusCode	2
RequestUriSchema	1
RequestUriHost	1
RequestUriPathToken1	1
RequestUriPathToken2	1
RequestUriPathToken3	3
RequestUriPathToken4	538
RequestUriPathToken5	1
RequestUriPathToken6	1
RequestUriQueryToken1	1053
RequestUriQueryToken2	1
RequestUriQueryToken3	1
RequestUriQueryToken4	1
RequestUriFragmentToken1	1
RequestUriFragmentToken2	1
HasRequestPayload	1
HasValidRequestPayload	1
HasAuthorisationToken	1
RequestHeader_Connection	1
RequestHeader_Content-Length	10
RequestHeader_Content-Type	2
RequestHeader_Host	1
RequestHeader_User-Agent	1
ResponseHeader_cache-control	1
ResponseHeader_content-disposition	1
ResponseHeader_content-length	17
ResponseHeader_content-type	1
ResponseHeader_date	19517
ResponseHeader_expires	1
ResponseHeader_last-modified	19511
ResponseHeader_pragma	1
ResponseHeader_server	1
ResponseHeader_status	2
ResponseHeader_strict-transport-security	1
ResponseHeader_x-connection-hash	3688
ResponseHeader_x-content-type-options	1
ResponseHeader_x-frame-options	1
ResponseHeader_x-rate-limit-limit	2
ResponseHeader_x-rate-limit-remaining	901
ResponseHeader_x-rate-limit-reset	27
ResponseHeader_x-response-time	278
ResponseHeader_x-transaction	26053
ResponseHeader_x-twitter-response-tags	1
ResponseHeader_x-xss-protection	1
ResponseBody_contributors	2
ResponseBody_coordinates	2
ResponseBody_created_at	517
ResponseBody_entities.hashtags	2

ResponseBody_entities.symbols	2
ResponseBody_entities.urls	2
ResponseBody_entities.user_mentions	2
ResponseBody_errors.code	2
ResponseBody_errors.message	2
ResponseBody_favorite_count	2
ResponseBody_favorited	2
ResponseBody_geo	2
ResponseBody_id	868
ResponseBody_id_str	868
ResponseBody_in_reply_to_screen_name	2
ResponseBody_in_reply_to_status_id	2
ResponseBody_in_reply_to_status_id_str	2
ResponseBody_in_reply_to_user_id	2
ResponseBody_in_reply_to_user_id_str	2
ResponseBody_is_quote_status	2
ResponseBody_lang	30
ResponseBody_place	2
ResponseBody_retweet_count	2
ResponseBody_retweeted	2
ResponseBody_text	853
ResponseBody_truncated	2
ResponseBody_user.contributors_enabled	2
ResponseBody_user.created_at	2
ResponseBody_user.default_profile	2
ResponseBody_user.default_profile_image	2
ResponseBody_user.description	1
ResponseBody_user.favourites_count	2
ResponseBody_user.follow_request_sent	2
ResponseBody_user.followers_count	2
ResponseBody_user.following	2
ResponseBody_user.friends_count	2
ResponseBody_user.geo_enabled	2
ResponseBody_user.has_extended_profile	2
ResponseBody_user.id	2
ResponseBody_user.id_str	2
ResponseBody_user.is_translation_enabled	2
ResponseBody_user.is_translator	2
ResponseBody_user.lang	2
ResponseBody_user.listed_count	2
ResponseBody_user.location	2
ResponseBody_user.name	2
ResponseBody_user.notifications	2
ResponseBody_user.profile_background_color	2
ResponseBody_user.profile_background_image_url	2
ResponseBody_user.profile_background_image_url_https	2
ResponseBody_user.profile_background_tile	2
ResponseBody_user.profile_banner_url	2
ResponseBody_user.profile_image_url	2
ResponseBody_user.profile_image_url_https	2
ResponseBody_user.profile_link_color	2
ResponseBody_user.profile_sidebar_border_color	2
ResponseBody_user.profile_sidebar_fill_color	2
ResponseBody_user.profile_text_color	2
ResponseBody_user.profile_use_background_image	2
ResponseBody_user.protected	2
ResponseBody_user.screen_name	2
ResponseBody_user.statuses_count	423
ResponseBody_user.time_zone	2
ResponseBody_user.translator_type	2

ResponseBody_user.url	2
ResponseBody_user.utc_offset	2
ResponseBody_user.verified	2
HasImmediatePreviousTransaction	2
HasImmediatePreviousTransactionSucceeded	2
ImmediatelyPreviousStatusCode	3
ImmediatelyPreviousMethod	3
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate	2
HasURLInImmediatelyPreviousTransactionContainsATokenToRead	2
HasURLInImmediatelyPreviousTransactionContainsATokenToUpdate	1
HasURLInImmediatelyPreviousTransactionContainsATokenToDelete	2
HasSuccessfulCreateOperationOccurredBefore	2
HasSuccessfulReadOperationOccurredBefore	2
HasSuccessfulUpdateOperationOccurredBefore	1
HasSuccessfulDeleteOperationOccurredBefore	2

B.1.3 Attributes for Google Tasks

Attribute	Distinct
RequestMethod	4
ResponseStatusCode	4
RequestUriSchema	1
RequestUriHost	1
RequestUriPathToken1	1
RequestUriPathToken2	1
RequestUriPathToken3	1
RequestUriPathToken4	1
RequestUriPathToken5	1
RequestUriPathToken6	1122
RequestUriQueryToken1	1
RequestUriQueryToken2	1
RequestUriQueryToken3	1
RequestUriQueryToken4	1
RequestUriFragmentToken1	1
RequestUriFragmentToken2	1
HasRequestPayload	2
HasValidRequestPayload	2
HasAuthorisationToken	1
RequestHeader_Connection	1
RequestHeader_Content-Length	15
RequestHeader_Content-Type	2
RequestHeader_Host	1
RequestHeader_User-Agent	1
ResponseHeader_Accept-Ranges	2
ResponseHeader_Alt-Svc	1
ResponseHeader_Cache-Control	3
ResponseHeader_Content-Length	10
ResponseHeader_Content-Type	2
ResponseHeader_Date	4161
ResponseHeader_ETag	883
ResponseHeader_Expires	2341
ResponseHeader_Pragma	2
ResponseHeader_Server	1
ResponseHeader_Transfer-Encoding	2
ResponseHeader_Vary	2
ResponseHeader_X-Content-Type-Options	2
ResponseHeader_X-Frame-Options	2
ResponseHeader_X-XSS-Protection	2
ResponseBody_error.code	3

ResponseBody_error.errors.domain	2
ResponseBody_error.errors.message	3
ResponseBody_error.errors.reason	3
ResponseBody_error.message	3
ResponseBody_etag	1612
ResponseBody_id	1125
ResponseBody_kind	2
ResponseBody_selfLink	1125
ResponseBody_title	1606
ResponseBody_updated	1590
HasImmediatePreviousTransaction	2
HasImmediatePreviousTransactionSucceeded	2
ImmediatelyPreviousResponseStatusCode	5
ImmediatelyPreviousRequestMethod	5
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate	1
HasURLInImmediatelyPreviousTransactionContainsATokenToRead	1
HasURLInImmediatelyPreviousTransactionContainsATokenToUpdate	1
HasURLInImmediatelyPreviousTransactionContainsATokenToDelete	1
HasSuccessfulCreateOperationOccurredBefore	2
HasSuccessfulReadOperationOccurredBefore	2
HasSuccessfulUpdateOperationOccurredBefore	2
HasSuccessfulDeleteOperationOccurredBefore	2

B.1.4 Attributes for Slack

Attribute	Distinct
RequestMethod	1
ResponseStatusCode	1
RequestUriSchema	1
RequestUriHost	1
RequestUriPathToken1	1
RequestUriPathToken2	3
RequestUriPathToken3	1
RequestUriPathToken4	1
RequestUriPathToken5	1
RequestUriPathToken6	1
RequestUriQueryToken1	1
RequestUriQueryToken2	1
RequestUriQueryToken3	7614
RequestUriQueryToken4	6421
RequestUriFragmentToken1	1
RequestUriFragmentToken2	1
HasRequestPayload	1
HasValidRequestPayload	1
HasAuthorisationToken	1
RequestHeader_Connection	1
RequestHeader_Content-Length	13
RequestHeader_Content-Type	2
RequestHeader_Host	1
RequestHeader_User-Agent	1
ResponseHeader_Access-Control-Allow-Origin	1
ResponseHeader_Cache-Control	1
ResponseHeader_Connection	1
ResponseHeader_Content-Type	1
ResponseHeader_Date	11977
ResponseHeader_Expires	1
ResponseHeader_Pragma	1
ResponseHeader_Referrer-Policy	1
ResponseHeader_Server	1

ResponseHeader_Strict-Transport-Security	1
ResponseHeader_Transfer-Encoding	1
ResponseHeader_Vary	1
ResponseHeader_Via	47
ResponseHeader_X-Accepted-OAuth-Scopes	1
ResponseHeader_X-Amz-Cf-Id	17422
ResponseHeader_X-Cache	1
ResponseHeader_X-Content-Type-Options	1
ResponseHeader_X-OAuth-Scopes	1
ResponseHeader_X-Slack-Backend	1
ResponseHeader_X-Slack-Exp	1
ResponseHeader_X-Slack-Req-Id	17422
ResponseHeader_X-Via	126
ResponseHeader_X-XSS-Protection	1
ResponseHeader_x-slack-router	2
ResponseBody_channel	2
ResponseBody_error	2
ResponseBody_message.bot_id	2
ResponseBody_message.edited.ts	50
ResponseBody_message.edited.user	2
ResponseBody_message.text	5321
ResponseBody_message.ts	3986
ResponseBody_message.type	2
ResponseBody_message.user	2
ResponseBody_ok	2
ResponseBody_text	1340
ResponseBody_ts	3986
HasImmediatePreviousTransaction	2
HasImmediatePreviousTransactionSucceeded	2
ImmediatelyPreviousResponseStatusCode	2
ImmediatelyPreviousRequestMethod	2
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate	2
HasURLInImmediatelyPreviousTransactionContainsATokenToRead	1
HasURLInImmediatelyPreviousTransactionContainsATokenToUpdate	2
HasURLInImmediatelyPreviousTransactionContainsATokenToDelete	2
HasSuccessfulCreateOperationOccurredBefore	2
HasSuccessfulReadOperationOccurredBefore	1
HasSuccessfulUpdateOperationOccurredBefore	2
HasSuccessfulDeleteOperationOccurredBefore	2

B.2 Sample ARFF Files

B.2.1 ARFF File for GHTraffic

```
@relation ghtraffic

@attribute RequestMethod {HEAD,DELETE,POST,GET,PUT,PATCH}
@attribute ResponseStatusCode numeric
@attribute RequestUriSchema {not-exist}
@attribute RequestUriHost {not-exist}
@attribute RequestUriPathToken1 {repos}
@attribute RequestUriPathToken2 {google}
@attribute RequestUriPathToken3 {guava}
.
.
.
@attribute HasRequestPayload {false,true}
@attribute HasValidRequestPayload {false,true}
@attribute HasAuthorisationToken {true,false}
@attribute RequestHeader_Accept {*/*}
.
```

```
.
.
@attribute ResponseHeader_Access-Control-Allow-Origin {*}
@attribute ResponseHeader_Access-Control-Expose-Headers {'ETag, X-OAuth-Scopes, X
    -Accepted-OAuth-Scopes'}
@attribute ResponseHeader_Cache-Control {not-exist,'private, max-age=60'}
.
.
.
@attribute ResponseBody_locked {not-exist,false,true}
.
.
.
@attribute HasImmediatePreviousTransaction {false,true}
@attribute HasImmediatePreviousTransactionSucceeded {false,true}
@attribute ImmediatelyPreviousStatusCode {not-exist
    ,404,400,201,200,401,204,500,422}
@attribute ImmediatelyPreviousMethod {not-exist,PATCH,DELETE,GET,POST,HEAD,PUT}
@attribute HasURLInImmediatelyPreviousTransactionContainsATokenToCreate {false}
@attribute HasURLInImmediatelyPreviousTransactionContainsATokenToRead {false}
@attribute HasURLInImmediatelyPreviousTransactionContainsATokenToUpdate {false}
@attribute HasURLInImmediatelyPreviousTransactionContainsATokenToDelete {false}
@attribute HasSuccessfulCreateOperationOccurredBefore {false,true}
@attribute HasSuccessfulReadOperationOccurredBefore {false,true}
@attribute HasSuccessfulUpdateOperationOccurredBefore {false,true}
@attribute HasSuccessfulDeleteOperationOccurredBefore {false,true}

@data
PATCH,404,not-exist,not-exist,repos,google,guava,issues,591,not-exist,not-exist,
    not-exist,not-exist,not-exist,not-exist,not-exist,true,true,true,*/*,55,'
    application/json; charset=utf-8',api.github.com,'Mozilla/4.0 (compatible;
    MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322; .NET CLR 1.0.3705)',*,'ETag, X-
    OAuth-Scopes, X-Accepted-OAuth-Scopes',not-exist,111,'application/json;
    charset=utf-8','Wed, 30 Oct 2013 20:20:20 GMT',not-exist,not-exist,not-exist,
    GitHub.com,not-exist,repo,'github.v3; format=json',72
    D9:FC4D9:8A69082:7281141:84B33D38,public_repo,closed,not-exist,not-exist,not-
    exist,not-exist,not-exist,not-exist,not-exist,not-exist,not-exist,not-exist,
    not-exist,not-exist,not-exist,not-exist,not-exist,not-exist,not-exist,not-
    exist,not-exist,not-exist,not-exist,not-exist,not-exist,not-exist,not-exist,
    not-exist,not-exist,not-exist,not-exist,not-exist,not-exist,not-exist,not-
    exist,not-exist,not-exist,not-exist,not-exist,not-exist,not-exist,not-exist,
    not-exist,not-exist,not-exist,not-exist,not-exist,not-exist,not-exist,not-
    exist,not-exist,not-exist,not-exist,not-exist,not-exist,not-exist,https://
    developer.github.com/v3,not-exist,not-exist,not-exist,'Not Found',not-exist,
    not-exist,not-exist,not-exist,not-exist,not-exist,not-exist,not-exist,not-
    exist,not-exist,not-exist,not-exist,not-exist,not-exist,not-exist,not-exist,
    not-exist,not-exist,not-exist,not-exist,not-exist,not-exist,not-exist,not-
    exist,not-exist,not-exist,not-exist,not-exist,not-exist,not-exist,not-exist,
    not-exist,not-exist,not-exist,not-exist,not-exist,not-exist,not-exist,not-
    exist,not-exist,not-exist,not-exist,not-exist,not-exist,not-exist,not-exist,
    not-exist,not-exist,not-exist,not-exist,not-exist,not-exist,false,false,not-
    exist,not-exist,false,false,false,false,false,false,false,false
.
.
.
```

B.2.2 ARFF File for Twitter

```
@relation twitter

@attribute RequestMethod {POST,GET}
@attribute ResponseStatusCode numeric
@attribute RequestUriSchema {https}
@attribute RequestUriHost {api.twitter.com}
@attribute RequestUriPathToken1 numeric
@attribute RequestUriPathToken2 {statuses}
@attribute RequestUriPathToken3 {update.json,show.json,destroy}
.
.
.
@attribute HasRequestPayload {false}
@attribute HasValidRequestPayload {false}
```

```
@attribute HasAuthorisationToken {true}
@attribute RequestHeader_Connection {keep-alive}

.
.
.
@attribute ResponseHeader_pragma {no-cache}
@attribute ResponseHeader_server {tsa_l}
@attribute ResponseHeader_status {'200 OK','404 Not Found'}
.
.
.
@attribute ResponseBody_errors.code {not-exist,144}
@attribute ResponseBody_errors.message {not-exist,'No status found with that ID.'
    }
.
.
.
@attribute ResponseBody_user.id string
@attribute ResponseBody_user.id_str string
.
.
.
@attribute ResponseBody_user.verified {false,not-exist}
@attribute HasImmediatePreviousTransaction {false,true}
@attribute HasImmediatePreviousTransactionSucceeded {false,true}
@attribute ImmediatelyPreviousStatusCode {not-exist,200,404}
@attribute ImmediatelyPreviousMethod {not-exist,POST,GET}
@attribute HasURLInImmediatelyPreviousTransactionContainsATokenToCreate {false,
    true}
@attribute HasURLInImmediatelyPreviousTransactionContainsATokenToRead {false,true
    }
@attribute HasURLInImmediatelyPreviousTransactionContainsATokenToUpdate {false}
@attribute HasURLInImmediatelyPreviousTransactionContainsATokenToDelete {false,
    true}
@attribute HasSuccessfulCreateOperationOccurredBefore {false,true}
@attribute HasSuccessfulReadOperationOccurredBefore {false,true}
@attribute HasSuccessfulUpdateOperationOccurredBefore {false}
@attribute HasSuccessfulDeleteOperationOccurredBefore {false,true}

@data
POST,200,https,api.twitter.com,1.1,statuses,update.json,not-exist,not-exist,not-
    exist,status=WeZxBzRvxI,not-exist,not-exist,not-exist,not-exist,not-exist,
    false,false,true,keep-alive,17,application/x-www-form-urlencoded,api.twitter.
    com,'Apache-HttpClient/4.5.5 (Java/1.8.0_131)','no-cache, no-store, must-
    revalidate, pre-check=0, post-check=0','attachment; filename=json.json',2240,
    application/json;charset=utf-8,'Tue, 12 Jun 2018 02:44:20 GMT','Tue, 31 Mar
    1981 05:00:00 GMT','Tue, 12 Jun 2018 02:44:20 GMT',no-cache,tsa_l,'200 OK',
    max-age=631138519,e33e2c761417500910572dfc5cf2aff9,nosniff,SAMEORIGIN,not-
    exist,not-exist,not-exist,299,003a698700d60804,BouncerCompliant,'1; mode=
    block; report=https://twitter.com/i/xss_report',null,null,'Tue Jun 12 02
    :44:20 +0000 2018',[],[],[],[],not-exist,not-exist,0,false,null
    ,1006366562146611200,1006366562146611200,null,null,null,null,null,false,eu,
    null,0,false,WeZxBzRvxI,false,false,'Wed Mar 07 09:41:33 +0000 2012',false,
    false,?,64,false,185,false,249,true,false,517417816,517417816,false,false,en
    ,3,Kurunegala,'Thilini Bhagya',false,1A1B1F,http://abs.twimg.com/images/
    themes/theme9/bg.gif,https://abs.twimg.com/images/themes/theme9/bg.gif,false,
    https://pbs.twimg.com/profile_banners/517417816/1399047954,http://pbs.twimg.
    com/profile_images/950100192048562176/LKr7Ay21_normal.jpg,https://pbs.twimg.
    com/profile_images/950100192048562176/LKr7Ay21_normal.jpg,3E4547,FFFFFF
    ,252429,666666,true,false,bhagyasl,495,null,none,http://t.co/sQduiwqJiy,null,
    false,false,false,not-exist,not-exist,false,false,false,false,false,false,
    false,false
```

B.2.3 ARFF File for Google Tasks

```
@relation google tasks

@attribute RequestMethod {POST,GET,PATCH,DELETE}
@attribute ResponseStatusCode numeric
@attribute RequestUriSchema {https}
@attribute RequestUriHost {www.googleapis.com}
```

```
@attribute RequestUriPathToken1 {tasks}
@attribute RequestUriPathToken2 {v1}
@attribute RequestUriPathToken3 {users}
.
.
.
@attribute HasRequestPayload {true,false}
@attribute HasValidRequestPayload {true,false}
@attribute HasAuthorisationToken {true}
@attribute RequestHeader_Connection {keep-alive}
@attribute RequestHeader_Content-Length string
@attribute RequestHeader_Content-Type {application/json,not-exist}
@attribute RequestHeader_Host {www.googleapis.com}
.
.
.
@attribute ResponseHeader_Pragma {no-cache,not-exist}
@attribute ResponseHeader_Server {GSE}
.
.
.
@attribute ResponseBody_error.code {not-exist,404,503}
@attribute ResponseBody_error.errors.domain {not-exist,global}
.
.
.
@attribute ResponseBody_kind {tasks#taskList,not-exist}
.
.
.
@attribute HasImmediatePreviousTransaction {false,true}
@attribute HasImmediatePreviousTransactionSucceeded {false,true}
@attribute ImmediatelyPreviousResponseStatusCode {not-exist,200,204,404,503}
@attribute ImmediatelyPreviousRequestMethod {not-exist,POST,GET,PATCH,DELETE}
@attribute HasURLInImmediatelyPreviousTransactionContainsATokenToCreate {false}
@attribute HasURLInImmediatelyPreviousTransactionContainsATokenToRead {false}
@attribute HasURLInImmediatelyPreviousTransactionContainsATokenToUpdate {false}
@attribute HasURLInImmediatelyPreviousTransactionContainsATokenToDelete {false}
@attribute HasSuccessfulCreateOperationOccurredBefore {false,true}
@attribute HasSuccessfulReadOperationOccurredBefore {false,true}
@attribute HasSuccessfulUpdateOperationOccurredBefore {false,true}
@attribute HasSuccessfulDeleteOperationOccurredBefore {false,true}

@data
POST,200,https,www.googleapis.com,tasks,v1,users,@me,lists,not-exist,not-exist,
    not-exist,not-exist,not-exist,not-exist,not-exist,true,true,true,keep-alive
    ,24,application/json,www.googleapis.com,'Apache-HttpClient/4.5.5 (Java/1.8.0
    _131)',none,'quic=:443; ma=2592000; v=44,43,39,35','no-cache, no-store, max-
    age=0, must-revalidate',not-exist,'application/json; charset=UTF-8','Tue, 28
    Aug 2018 01:30:29 GMT',not-exist,'Mon, 01 Jan 1990 00:00:00 GMT',no-cache,GSE
    ,chunked,'Origin,Accept-Encoding',nosniff,SAMEORIGIN,'1; mode=block',not-
    exist,not-exist,not-exist,not-exist,not-exist,FhCqMAsBrrKDkDLKevwtJykQ9I8/
    e9GywRc-x6tOAMrNyueWJEEHNx8,
    MTcyMDU1OTI5NDQ0MDQ0ODg3NjI6NTkxMjg3NTgxMjE4NDQyMzow,tasks#taskList,https://
    www.googleapis.com/tasks/v1/users/@me/lists/
    MTcyMDU1OTI5NDQ0MDQ0ODg3NjI6NTkxMjg3NTgxMjE4NDQyMzow,glIijN,2018-08-28
    T01:30:28.000Z,false,false,not-exist,not-exist,false,false,false,false,false,
    false,false,false
.
.
```

B.2.4 ARFF File for Slack

```
@relation slack

@attribute RequestMethod {POST}
@attribute ResponseStatusCode numeric
@attribute RequestUriSchema {https}
@attribute RequestUriHost {slack.com}
@attribute RequestUriPathToken1 {api}
@attribute RequestUriPathToken2 {chat.update,chat.delete,chat.postMessage}
@attribute RequestUriPathToken3 {not-exist}
```

```
.
.
.
@attribute HasRequestPayload {false}
@attribute HasValidRequestPayload {false}
@attribute HasAuthorisationToken {true}
@attribute RequestHeader_Connection {keep-alive}
@attribute RequestHeader_Content-Length numeric
@attribute RequestHeader_Content-Type {application/x-www-form-urlencoded,not-
    exist}
@attribute RequestHeader_Host {slack.com}
.
.
.
@attribute ResponseHeader_Pragma {no-cache}
@attribute ResponseHeader_Referrer-Policy {no-referrer}
@attribute ResponseHeader_Server {Apache}
.
.
.
@attribute ResponseHeader_Cache-Control {'private, no-cache, no-store, must-
    revalidate'}
.
.
.
@attribute ResponseBody_channel {not-exist,CCGRWTRKQ}
@attribute ResponseBody_error {message_not_found,not-exist}
@attribute ResponseBody_message.bot_id {not-exist,BCEPNCQDN}
.
.
.
@attribute ResponseBody_message.type {message,not-exist}
@attribute ResponseBody_message.user {UC8J6APLN,not-exist}
@attribute ResponseBody_ok {true,false}
.
.
.
@attribute HasImmediatePreviousTransaction {false,true}
@attribute HasImmediatePreviousTransactionSucceeded {false,true}
@attribute ImmediatelyPreviousResponseStatusCode {not-exist,200}
@attribute ImmediatelyPreviousRequestMethod {not-exist,POST}
@attribute HasURLInImmediatelyPreviousTransactionContainsATokenToCreate {false,
    true}
@attribute HasURLInImmediatelyPreviousTransactionContainsATokenToRead {false}
@attribute HasURLInImmediatelyPreviousTransactionContainsATokenToUpdate {false,
    true}
@attribute HasURLInImmediatelyPreviousTransactionContainsATokenToDelete {false,
    true}
@attribute HasSuccessfulCreateOperationOccurredBefore {false,true}
@attribute HasSuccessfulReadOperationOccurredBefore {false}
@attribute HasSuccessfulUpdateOperationOccurredBefore {false,true}
@attribute HasSuccessfulDeleteOperationOccurredBefore {false,true}

@data
POST,200,https,slack.com,api,chat.postMessage,not-exist,not-exist,not-exist,not-
    exist,token=xoxp-415149719555-416618363702-422804431936-
    b30a935a430030bec72e399b896b0bcc,channel=CCGRWTRKQ,text=GhyIQ,not-exist,not-
    exist,not-exist,false,false,true,keep-alive,111,application/x-www-form-
    urlencoded,slack.com,'Apache-HttpClient/4.5.5 (Java/1.8.0_131)',*,'private,
    no-cache, no-store, must-revalidate',keep-alive,'application/json; charset=
    utf-8','Mon, 27 Aug 2018 08:06:54 GMT','Mon, 26 Jul 1997 05:00:00 GMT',no-
    cache,no-referrer,Apache,'max-age=31536000; includeSubDomains; preload',
    chunked,Accept-Encoding,'1.1 60287558223e50f42a557f7dddcf3385.cloudfront.net
    (CloudFront)',chat:write:user,awo-
    YAvO7c1B1qfB9GvcaUZykX59IRx3wRuXIQYnU9XUVCRoGhqlWA==,'Miss from cloudfront',
    nosniff,'identify,channels:write,chat:write:user',h,1,bf11c89e-1412-4457-9b03
    -2755d06a95a5,haproxy-www-sv7g,0,p,CCGRWTRKQ,not-exist,BCEPNCQDN,not-exist,
    not-exist,GhyIQ,1535357214.000300,message,UC8J6APLN,true,not-exist
    ,1535357214.000300,false,false,not-exist,not-exist,false,false,false,false,
    false,false,false,false
.
.
.
```

B.3 Sub-Datasets of GHTraffic, Twitter, and Slack

Transactions Per Method on Sub-Datasets

Method	**GHTraffic**	**Twitter**	**Slack**
POST	629	2,307	5,245
PATCH	555	-	-
GET	357	218	-
HEAD	274	-	-
PUT	394	-	-
DELETE	221	-	-

Transactions Per Response Code on Sub-Datasets

Response Code	**GHTraffic**	**Twitter**	**Slack**
200	472	1,022	5,245
201	354	-	-
204	191	-	-
400	238	-	-
401	280	-	-
404	829	1,503	-
422	56	-	-
500	10	-	-

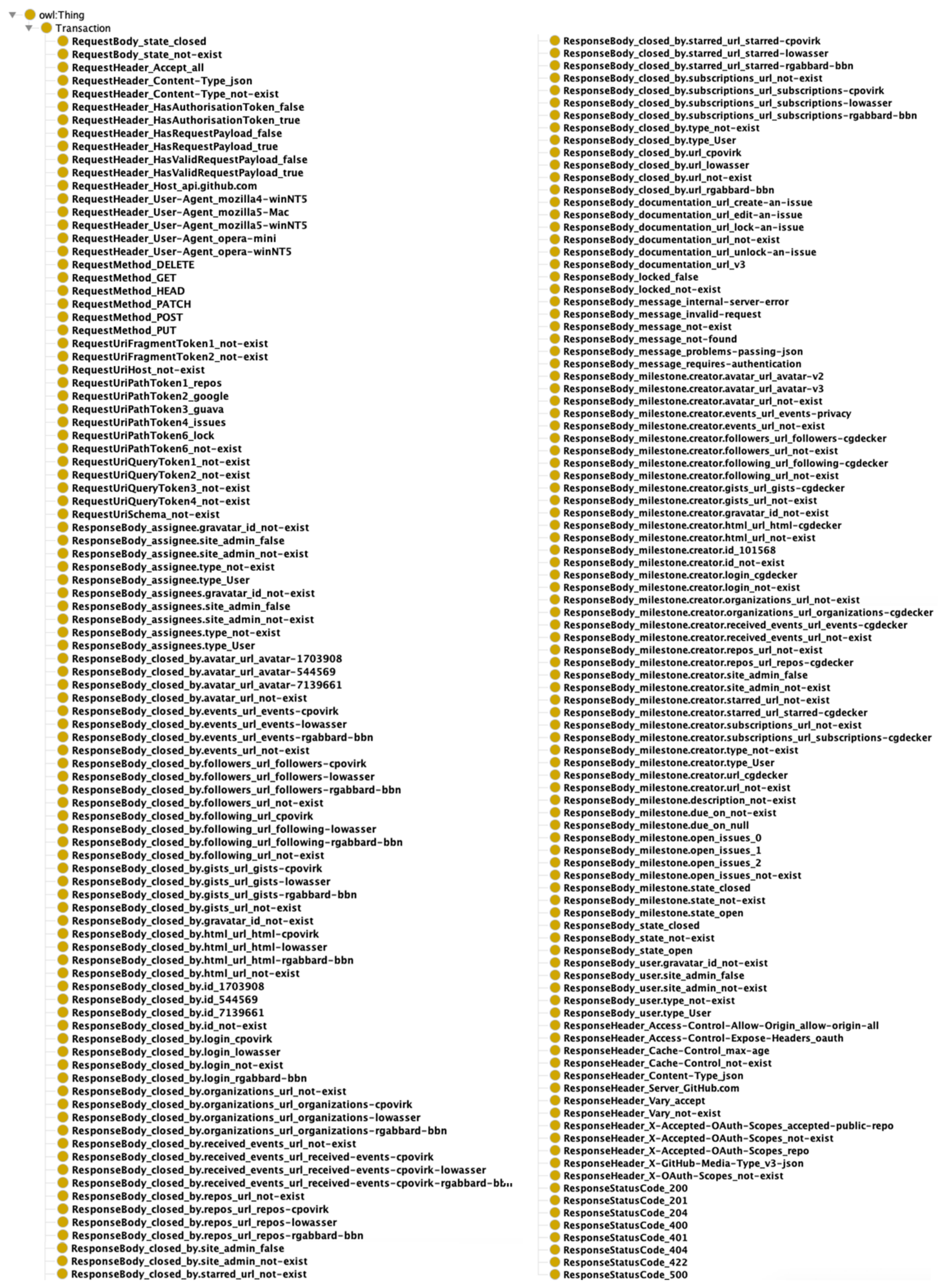
Accessing Sub-Datasets

- `https://zenodo.org/record/4008239/files/sub-ghtraffic-S-2.0.0.zip`
- `https://zenodo.org/record/4008239/files/sub-twitter-1.0.0.zip`
- `https://zenodo.org/record/4008239/files/sub-slack-1.0.0.zip`

B.4 Sample OWL Knowledge Bases

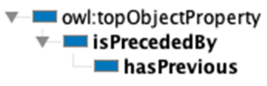
B.4.1 GHTraffic Knowledge Base Visualised in Protege

Class Hierarchy

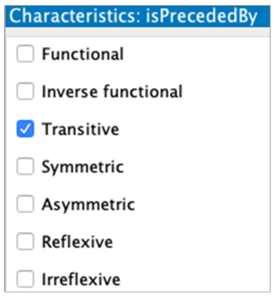


```
owl:Thing
  Transaction
    RequestBody_state_closed
    RequestBody_state_not-exist
    RequestHeader_Accept_all
    RequestHeader_Content-Type_json
    RequestHeader_Content-Type_not-exist
    RequestHeader_HasAuthorisationToken_false
    RequestHeader_HasAuthorisationToken_true
    RequestHeader_HasRequestPayload_false
    RequestHeader_HasRequestPayload_true
    RequestHeader_HasValidRequestPayload_false
    RequestHeader_HasValidRequestPayload_true
    RequestHeader_Host_api.github.com
    RequestHeader_User-Agent_mozilla4-winNT5
    RequestHeader_User-Agent_mozilla5-Mac
    RequestHeader_User-Agent_mozilla5-winNT5
    RequestHeader_User-Agent_opera-mini
    RequestHeader_User-Agent_opera-winNT5
    RequestMethod_DELETE
    RequestMethod_GET
    RequestMethod_HEAD
    RequestMethod_PATCH
    RequestMethod_POST
    RequestMethod_PUT
    RequestUriFragmentToken1_not-exist
    RequestUriFragmentToken2_not-exist
    RequestUriHost_not-exist
    RequestUriPathToken1_repos
    RequestUriPathToken2_google
    RequestUriPathToken3_guava
    RequestUriPathToken4_issues
    RequestUriPathToken6_lock
    RequestUriPathToken6_not-exist
    RequestUriQueryToken1_not-exist
    RequestUriQueryToken2_not-exist
    RequestUriQueryToken3_not-exist
    RequestUriQueryToken4_not-exist
    RequestUriSchema_not-exist
    ResponseBody_assignee.gravatar_id_not-exist
    ResponseBody_assignee.site_admin_false
    ResponseBody_assignee.site_admin_not-exist
    ResponseBody_assignee.type_not-exist
    ResponseBody_assignee.type_User
    ResponseBody_assignees.gravatar_id_not-exist
    ResponseBody_assignees.site_admin_false
    ResponseBody_assignees.site_admin_not-exist
    ResponseBody_assignees.type_not-exist
    ResponseBody_assignees.type_User
    ResponseBody_closed_by.avatar_url_avatar-1703908
    ResponseBody_closed_by.avatar_url_avatar-544569
    ResponseBody_closed_by.avatar_url_avatar-7139661
    ResponseBody_closed_by.avatar_url_not-exist
    ResponseBody_closed_by.events_url_events-cpovirk
    ResponseBody_closed_by.events_url_events-lowasser
    ResponseBody_closed_by.events_url_events-rgabbard-bbn
    ResponseBody_closed_by.events_url_not-exist
    ResponseBody_closed_by.followers_url_followers-cpovirk
    ResponseBody_closed_by.followers_url_followers-lowasser
    ResponseBody_closed_by.followers_url_followers-rgabbard-bbn
    ResponseBody_closed_by.followers_url_not-exist
    ResponseBody_closed_by.following_url_cpovirk
    ResponseBody_closed_by.following_url_following-lowasser
    ResponseBody_closed_by.following_url_following-rgabbard-bbn
    ResponseBody_closed_by.following_url_not-exist
    ResponseBody_closed_by.gists_url_gists-cpovirk
    ResponseBody_closed_by.gists_url_gists-lowasser
    ResponseBody_closed_by.gists_url_gists-rgabbard-bbn
    ResponseBody_closed_by.gists_url_not-exist
    ResponseBody_closed_by.gravatar_id_not-exist
    ResponseBody_closed_by.html_url_html-cpovirk
    ResponseBody_closed_by.html_url_html-lowasser
    ResponseBody_closed_by.html_url_html-rgabbard-bbn
    ResponseBody_closed_by.html_url_not-exist
    ResponseBody_closed_by.id_1703908
    ResponseBody_closed_by.id_544569
    ResponseBody_closed_by.id_7139661
    ResponseBody_closed_by.id_not-exist
    ResponseBody_closed_by.login_cpovirk
    ResponseBody_closed_by.login_lowasser
    ResponseBody_closed_by.login_not-exist
    ResponseBody_closed_by.login_rgabbard-bbn
    ResponseBody_closed_by.organizations_url_not-exist
    ResponseBody_closed_by.organizations_url_organizations-cpovirk
    ResponseBody_closed_by.organizations_url_organizations-lowasser
    ResponseBody_closed_by.organizations_url_organizations-rgabbard-bbn
    ResponseBody_closed_by.received_events_url_not-exist
    ResponseBody_closed_by.received_events_url_received-events-cpovirk
    ResponseBody_closed_by.received_events_url_received-events-cpovirk-lowasser
    ResponseBody_closed_by.received_events_url_received-events-cpovirk-rgabbard-bb
    ResponseBody_closed_by.repos_url_not-exist
    ResponseBody_closed_by.repos_url_repos-cpovirk
    ResponseBody_closed_by.repos_url_repos-lowasser
    ResponseBody_closed_by.repos_url_repos-rgabbard-bbn
    ResponseBody_closed_by.site_admin_false
    ResponseBody_closed_by.site_admin_not-exist
    ResponseBody_closed_by.starred_url_not-exist
    ResponseBody_closed_by.starred_url_starred-cpovirk
    ResponseBody_closed_by.starred_url_starred-lowasser
    ResponseBody_closed_by.starred_url_starred-rgabbard-bbn
    ResponseBody_closed_by.subscriptions_url_not-exist
    ResponseBody_closed_by.subscriptions_url_subscriptions-cpovirk
    ResponseBody_closed_by.subscriptions_url_subscriptions-lowasser
    ResponseBody_closed_by.subscriptions_url_subscriptions-rgabbard-bbn
    ResponseBody_closed_by.type_not-exist
    ResponseBody_closed_by.type_User
    ResponseBody_closed_by.url_cpovirk
    ResponseBody_closed_by.url_lowasser
    ResponseBody_closed_by.url_not-exist
    ResponseBody_closed_by.url_rgabbard-bbn
    ResponseBody_documentation_url_create-an-issue
    ResponseBody_documentation_url_edit-an-issue
    ResponseBody_documentation_url_lock-an-issue
    ResponseBody_documentation_url_not-exist
    ResponseBody_documentation_url_unlock-an-issue
    ResponseBody_documentation_url_v3
    ResponseBody_locked_false
    ResponseBody_locked_not-exist
    ResponseBody_message_internal-server-error
    ResponseBody_message_invalid-request
    ResponseBody_message_not-exist
    ResponseBody_message_not-found
    ResponseBody_message_problems-passing-json
    ResponseBody_message_requires-authentication
    ResponseBody_milestone.creator.avatar_url_avatar-v2
    ResponseBody_milestone.creator.avatar_url_avatar-v3
    ResponseBody_milestone.creator.avatar_url_not-exist
    ResponseBody_milestone.creator.events_url_events-privacy
    ResponseBody_milestone.creator.events_url_not-exist
    ResponseBody_milestone.creator.followers_url_followers-cgdecker
    ResponseBody_milestone.creator.followers_url_not-exist
    ResponseBody_milestone.creator.following_url_following-cgdecker
    ResponseBody_milestone.creator.following_url_not-exist
    ResponseBody_milestone.creator.gists_url_gists-cgdecker
    ResponseBody_milestone.creator.gists_url_not-exist
    ResponseBody_milestone.creator.gravatar_id_not-exist
    ResponseBody_milestone.creator.html_url_html-cgdecker
    ResponseBody_milestone.creator.html_url_not-exist
    ResponseBody_milestone.creator.id_101568
    ResponseBody_milestone.creator.id_not-exist
    ResponseBody_milestone.creator.login_cgdecker
    ResponseBody_milestone.creator.login_not-exist
    ResponseBody_milestone.creator.organizations_url_not-exist
    ResponseBody_milestone.creator.organizations_url_organizations-cgdecker
    ResponseBody_milestone.creator.received_events_url_events-cgdecker
    ResponseBody_milestone.creator.received_events_url_not-exist
    ResponseBody_milestone.creator.repos_url_not-exist
    ResponseBody_milestone.creator.repos_url_repos-cgdecker
    ResponseBody_milestone.creator.site_admin_false
    ResponseBody_milestone.creator.site_admin_not-exist
    ResponseBody_milestone.creator.starred_url_not-exist
    ResponseBody_milestone.creator.starred_url_starred-cgdecker
    ResponseBody_milestone.creator.subscriptions_url_not-exist
    ResponseBody_milestone.creator.subscriptions_url_subscriptions-cgdecker
    ResponseBody_milestone.creator.type_not-exist
    ResponseBody_milestone.creator.type_User
    ResponseBody_milestone.creator.url_cgdecker
    ResponseBody_milestone.creator.url_not-exist
    ResponseBody_milestone.description_not-exist
    ResponseBody_milestone.due_on_not-exist
    ResponseBody_milestone.due_on_null
    ResponseBody_milestone.open_issues_0
    ResponseBody_milestone.open_issues_1
    ResponseBody_milestone.open_issues_2
    ResponseBody_milestone.open_issues_not-exist
    ResponseBody_milestone.state_closed
    ResponseBody_milestone.state_not-exist
    ResponseBody_milestone.state_open
    ResponseBody_state_closed
    ResponseBody_state_not-exist
    ResponseBody_state_open
    ResponseBody_user.gravatar_id_not-exist
    ResponseBody_user.site_admin_false
    ResponseBody_user.site_admin_not-exist
    ResponseBody_user.type_not-exist
    ResponseBody_user.type_User
    ResponseHeader_Access-Control-Allow-Origin_allow-origin-all
    ResponseHeader_Access-Control-Expose-Headers_oauth
    ResponseHeader_Cache-Control_max-age
    ResponseHeader_Cache-Control_not-exist
    ResponseHeader_Content-Type_json
    ResponseHeader_Server_GitHub.com
    ResponseHeader_Vary_accept
    ResponseHeader_Vary_not-exist
    ResponseHeader_X-Accepted-OAuth-Scopes_accepted-public-repo
    ResponseHeader_X-Accepted-OAuth-Scopes_not-exist
    ResponseHeader_X-Accepted-OAuth-Scopes_repo
    ResponseHeader_X-GitHub-Media-Type_v3-json
    ResponseHeader_X-OAuth-Scopes_not-exist
    ResponseStatusCode_200
    ResponseStatusCode_201
    ResponseStatusCode_204
    ResponseStatusCode_400
    ResponseStatusCode_401
    ResponseStatusCode_404
    ResponseStatusCode_422
    ResponseStatusCode_500
```

Object Property Hierarchy



- owl:topObjectProperty
  - isPrecededBy
    - hasPrevious



Characteristics: isPrecededBy

Functional

Inverse functional

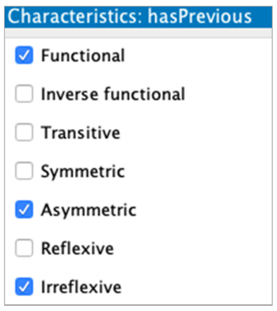
[✓ Transitive]

Symmetric

Asymmetric

Reflexive

Irreflexive



Characteristics: hasPrevious

[✓ Functional]

Inverse functional

Transitive

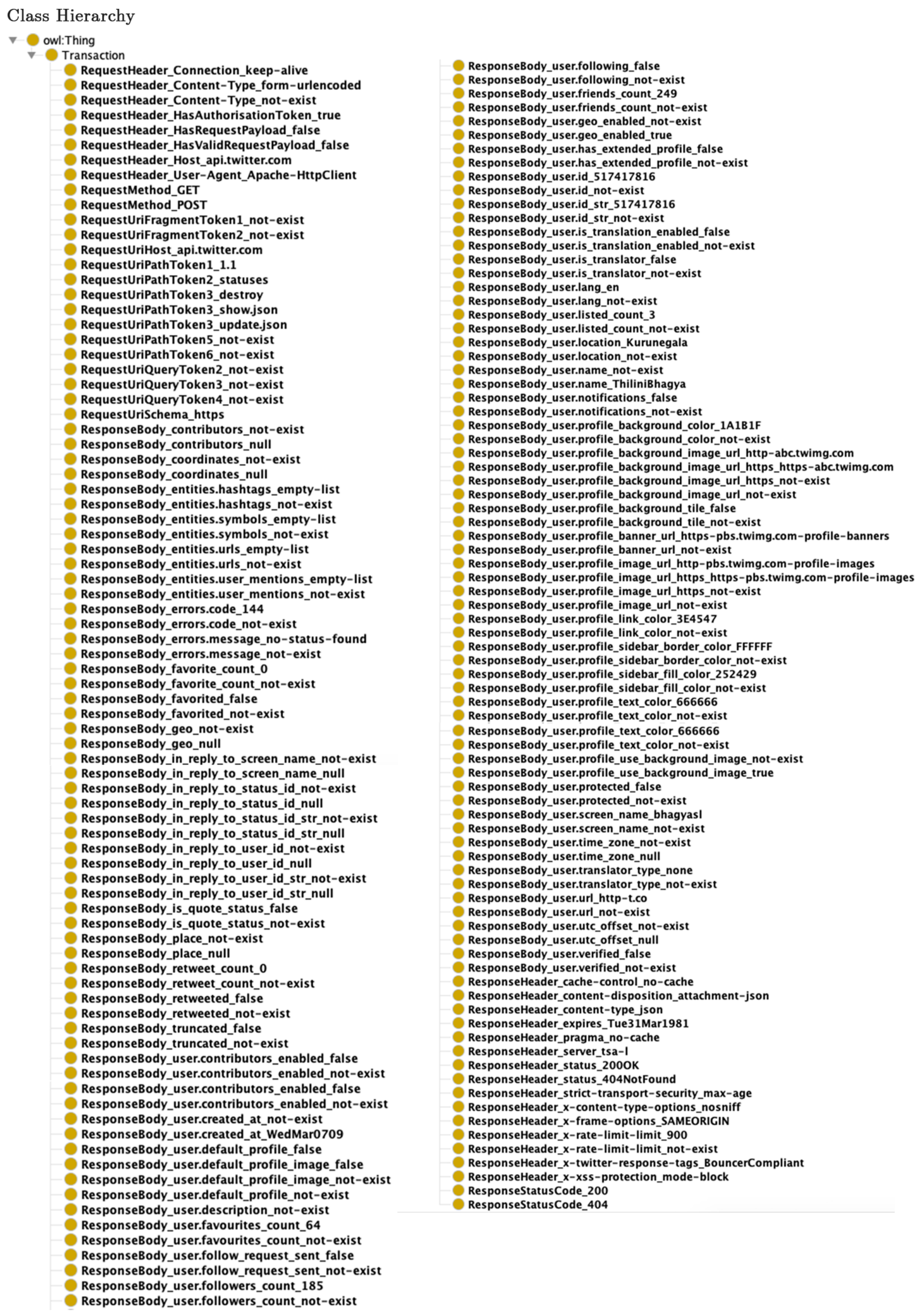
Symmetric

[✓ Asymmetric]

Reflexive

[✓ Irreflexive]

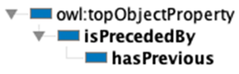
B.4.2 Twitter Knowledge Base Visualised in Protege

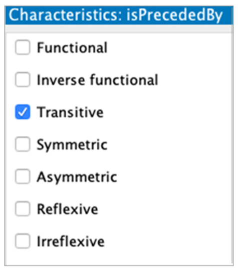


Class Hierarchy

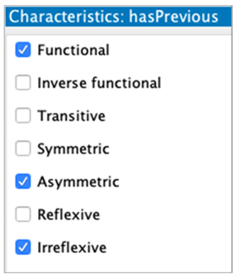
- owl:Thing
  - Transaction
    - RequestHeader_Connection_keep-alive
    - RequestHeader_Content-Type_form-urlencoded
    - RequestHeader_Content-Type_not-exist
    - RequestHeader_HasAuthorisationToken_true
    - RequestHeader_HasRequestPayload_false
    - RequestHeader_HasValidRequestPayload_false
    - RequestHeader_Host_api.twitter.com
    - RequestHeader_User-Agent_Apache-HttpClient
    - RequestMethod_GET
    - RequestMethod_POST
    - RequestUriFragmentToken1_not-exist
    - RequestUriFragmentToken2_not-exist
    - RequestUriHost_api.twitter.com
    - RequestUriPathToken1_1.1
    - RequestUriPathToken2_statuses
    - RequestUriPathToken3_destroy
    - RequestUriPathToken3_show.json
    - RequestUriPathToken3_update.json
    - RequestUriPathToken5_not-exist
    - RequestUriPathToken6_not-exist
    - RequestUriQueryToken2_not-exist
    - RequestUriQueryToken3_not-exist
    - RequestUriQueryToken4_not-exist
    - RequestUriSchema_https
    - ResponseBody_contributors_not-exist
    - ResponseBody_contributors_null
    - ResponseBody_coordinates_not-exist
    - ResponseBody_coordinates_null
    - ResponseBody_entities.hashtags_empty-list
    - ResponseBody_entities.hashtags_not-exist
    - ResponseBody_entities.symbols_empty-list
    - ResponseBody_entities.symbols_not-exist
    - ResponseBody_entities.urls_empty-list
    - ResponseBody_entities.urls_not-exist
    - ResponseBody_entities.user_mentions_empty-list
    - ResponseBody_entities.user_mentions_not-exist
    - ResponseBody_errors.code_144
    - ResponseBody_errors.code_not-exist
    - ResponseBody_errors.message_no-status-found
    - ResponseBody_errors.message_not-exist
    - ResponseBody_favorite_count_0
    - ResponseBody_favorite_count_not-exist
    - ResponseBody_favorited_false
    - ResponseBody_favorited_not-exist
    - ResponseBody_geo_not-exist
    - ResponseBody_geo_null
    - ResponseBody_in_reply_to_screen_name_not-exist
    - ResponseBody_in_reply_to_screen_name_null
    - ResponseBody_in_reply_to_status_id_not-exist
    - ResponseBody_in_reply_to_status_id_null
    - ResponseBody_in_reply_to_status_id_str_not-exist
    - ResponseBody_in_reply_to_status_id_str_null
    - ResponseBody_in_reply_to_user_id_not-exist
    - ResponseBody_in_reply_to_user_id_null
    - ResponseBody_in_reply_to_user_id_str_not-exist
    - ResponseBody_in_reply_to_user_id_str_null
    - ResponseBody_is_quote_status_false
    - ResponseBody_is_quote_status_not-exist
    - ResponseBody_place_not-exist
    - ResponseBody_place_null
    - ResponseBody_retweet_count_0
    - ResponseBody_retweet_count_not-exist
    - ResponseBody_retweeted_false
    - ResponseBody_retweeted_not-exist
    - ResponseBody_truncated_false
    - ResponseBody_truncated_not-exist
    - ResponseBody_user.contributors_enabled_false
    - ResponseBody_user.contributors_enabled_not-exist
    - ResponseBody_user.contributors_enabled_false
    - ResponseBody_user.contributors_enabled_not-exist
    - ResponseBody_user.created_at_not-exist
    - ResponseBody_user.created_at_WedMar0709
    - ResponseBody_user.default_profile_false
    - ResponseBody_user.default_profile_image_false
    - ResponseBody_user.default_profile_image_not-exist
    - ResponseBody_user.default_profile_not-exist
    - ResponseBody_user.description_not-exist
    - ResponseBody_user.favourites_count_64
    - ResponseBody_user.favourites_count_not-exist
    - ResponseBody_user.follow_request_sent_false
    - ResponseBody_user.follow_request_sent_not-exist
    - ResponseBody_user.followers_count_185
    - ResponseBody_user.followers_count_not-exist
    - ResponseBody_user.following_false
    - ResponseBody_user.following_not-exist
    - ResponseBody_user.friends_count_249
    - ResponseBody_user.friends_count_not-exist
    - ResponseBody_user.geo_enabled_not-exist
    - ResponseBody_user.geo_enabled_true
    - ResponseBody_user.has_extended_profile_false
    - ResponseBody_user.has_extended_profile_not-exist
    - ResponseBody_user.id_517417816
    - ResponseBody_user.id_not-exist
    - ResponseBody_user.id_str_517417816
    - ResponseBody_user.id_str_not-exist
    - ResponseBody_user.is_translation_enabled_false
    - ResponseBody_user.is_translation_enabled_not-exist
    - ResponseBody_user.is_translator_false
    - ResponseBody_user.is_translator_not-exist
    - ResponseBody_user.lang_en
    - ResponseBody_user.lang_not-exist
    - ResponseBody_user.listed_count_3
    - ResponseBody_user.listed_count_not-exist
    - ResponseBody_user.location_Kurunegala
    - ResponseBody_user.location_not-exist
    - ResponseBody_user.name_not-exist
    - ResponseBody_user.name_ThiliniBhagya
    - ResponseBody_user.notifications_false
    - ResponseBody_user.notifications_not-exist
    - ResponseBody_user.profile_background_color_1A1B1F
    - ResponseBody_user.profile_background_color_not-exist
    - ResponseBody_user.profile_background_image_url_http-abc.twimg.com
    - ResponseBody_user.profile_background_image_url_https_https-abc.twimg.com
    - ResponseBody_user.profile_background_image_url_https_not-exist
    - ResponseBody_user.profile_background_image_url_not-exist
    - ResponseBody_user.profile_background_tile_false
    - ResponseBody_user.profile_background_tile_not-exist
    - ResponseBody_user.profile_banner_url_https-pbs.twimg.com-profile-banners
    - ResponseBody_user.profile_banner_url_not-exist
    - ResponseBody_user.profile_image_url_http-pbs.twimg.com-profile-images
    - ResponseBody_user.profile_image_url_https_https-pbs.twimg.com-profile-images
    - ResponseBody_user.profile_image_url_https_not-exist
    - ResponseBody_user.profile_image_url_not-exist
    - ResponseBody_user.profile_link_color_3E4547
    - ResponseBody_user.profile_link_color_not-exist
    - ResponseBody_user.profile_sidebar_border_color_FFFFFF
    - ResponseBody_user.profile_sidebar_border_color_not-exist
    - ResponseBody_user.profile_sidebar_fill_color_252429
    - ResponseBody_user.profile_sidebar_fill_color_not-exist
    - ResponseBody_user.profile_text_color_666666
    - ResponseBody_user.profile_text_color_not-exist
    - ResponseBody_user.profile_text_color_666666
    - ResponseBody_user.profile_text_color_not-exist
    - ResponseBody_user.profile_use_background_image_not-exist
    - ResponseBody_user.profile_use_background_image_true
    - ResponseBody_user.protected_false
    - ResponseBody_user.protected_not-exist
    - ResponseBody_user.screen_name_bhagyasl
    - ResponseBody_user.screen_name_not-exist
    - ResponseBody_user.time_zone_not-exist
    - ResponseBody_user.time_zone_null
    - ResponseBody_user.translator_type_none
    - ResponseBody_user.translator_type_not-exist
    - ResponseBody_user.url_http-t.co
    - ResponseBody_user.url_not-exist
    - ResponseBody_user.utc_offset_not-exist
    - ResponseBody_user.utc_offset_null
    - ResponseBody_user.verified_false
    - ResponseBody_user.verified_not-exist
    - ResponseHeader_cache-control_no-cache
    - ResponseHeader_content-disposition_attachment-json
    - ResponseHeader_content-type_json
    - ResponseHeader_expires_Tue31Mar1981
    - ResponseHeader_pragma_no-cache
    - ResponseHeader_server_tsa-l
    - ResponseHeader_status_200OK
    - ResponseHeader_status_404NotFound
    - ResponseHeader_strict-transport-security_max-age
    - ResponseHeader_x-content-type-options_nosniff
    - ResponseHeader_x-frame-options_SAMEORIGIN
    - ResponseHeader_x-rate-limit-limit_900
    - ResponseHeader_x-rate-limit-limit_not-exist
    - ResponseHeader_x-twitter-response-tags_BouncerCompliant
    - ResponseHeader_x-xss-protection_mode-block
    - ResponseStatusCode_200
    - ResponseStatusCode_404

Object Property Hierarchy


- owl:topObjectProperty
  - **isPrecededBy**
    - **hasPrevious**


Characteristics: isPrecededBy

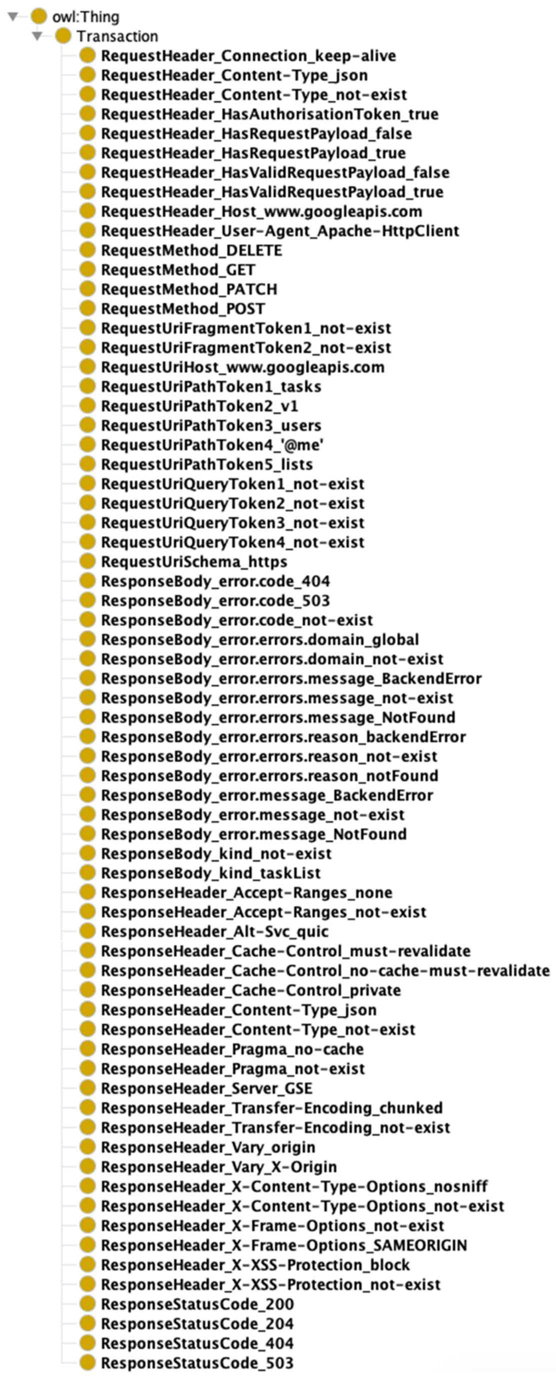
- [ ] Functional
- [ ] Inverse functional
- [✓] Transitive
- [ ] Symmetric
- [ ] Asymmetric
- [ ] Reflexive
- [ ] Irreflexive


Characteristics: hasPrevious

- [✓] Functional
- [ ] Inverse functional
- [ ] Transitive
- [ ] Symmetric
- [✓] Asymmetric
- [ ] Reflexive
- [✓] Irreflexive

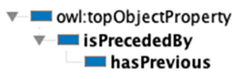
B.4.3 Google Tasks Knowledge Base Visualised in Protege

Class Hierarchy

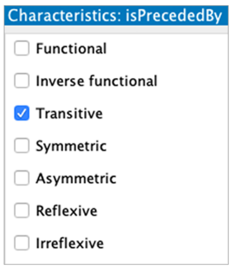


- owl:Thing
  - Transaction
    - RequestHeader_Connection_keep-alive
    - RequestHeader_Content-Type_json
    - RequestHeader_Content-Type_not-exist
    - RequestHeader_HasAuthorisationToken_true
    - RequestHeader_HasRequestPayload_false
    - RequestHeader_HasRequestPayload_true
    - RequestHeader_HasValidRequestPayload_false
    - RequestHeader_HasValidRequestPayload_true
    - RequestHeader_Host_www.googleapis.com
    - RequestHeader_User-Agent_Apache-HttpClient
    - RequestMethod_DELETE
    - RequestMethod_GET
    - RequestMethod_PATCH
    - RequestMethod_POST
    - RequestUriFragmentToken1_not-exist
    - RequestUriFragmentToken2_not-exist
    - RequestUriHost_www.googleapis.com
    - RequestUriPathToken1_tasks
    - RequestUriPathToken2_v1
    - RequestUriPathToken3_users
    - RequestUriPathToken4_'@me'
    - RequestUriPathToken5_lists
    - RequestUriQueryToken1_not-exist
    - RequestUriQueryToken2_not-exist
    - RequestUriQueryToken3_not-exist
    - RequestUriQueryToken4_not-exist
    - RequestUriSchema_https
    - ResponseBody_error.code_404
    - ResponseBody_error.code_503
    - ResponseBody_error.code_not-exist
    - ResponseBody_error.errors.domain_global
    - ResponseBody_error.errors.domain_not-exist
    - ResponseBody_error.errors.message_BackendError
    - ResponseBody_error.errors.message_not-exist
    - ResponseBody_error.errors.message_NotFound
    - ResponseBody_error.errors.reason_backendError
    - ResponseBody_error.errors.reason_not-exist
    - ResponseBody_error.errors.reason_notFound
    - ResponseBody_error.message_BackendError
    - ResponseBody_error.message_not-exist
    - ResponseBody_error.message_NotFound
    - ResponseBody_kind_not-exist
    - ResponseBody_kind_taskList
    - ResponseHeader_Accept-Ranges_none
    - ResponseHeader_Accept-Ranges_not-exist
    - ResponseHeader_Alt-Svc_quic
    - ResponseHeader_Cache-Control_must-revalidate
    - ResponseHeader_Cache-Control_no-cache-must-revalidate
    - ResponseHeader_Cache-Control_private
    - ResponseHeader_Content-Type_json
    - ResponseHeader_Content-Type_not-exist
    - ResponseHeader_Pragma_no-cache
    - ResponseHeader_Pragma_not-exist
    - ResponseHeader_Server_GSE
    - ResponseHeader_Transfer-Encoding_chunked
    - ResponseHeader_Transfer-Encoding_not-exist
    - ResponseHeader_Vary_origin
    - ResponseHeader_Vary_X-Origin
    - ResponseHeader_X-Content-Type-Options_nosniff
    - ResponseHeader_X-Content-Type-Options_not-exist
    - ResponseHeader_X-Frame-Options_not-exist
    - ResponseHeader_X-Frame-Options_SAMEORIGIN
    - ResponseHeader_X-XSS-Protection_block
    - ResponseHeader_X-XSS-Protection_not-exist
    - ResponseStatusCode_200
    - ResponseStatusCode_204
    - ResponseStatusCode_404
    - ResponseStatusCode_503

Object Property Hierarchy

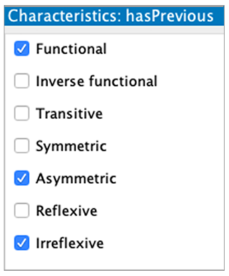


- owl:topObjectProperty
  - isPrecededBy
    - hasPrevious



Characteristics: isPrecededBy

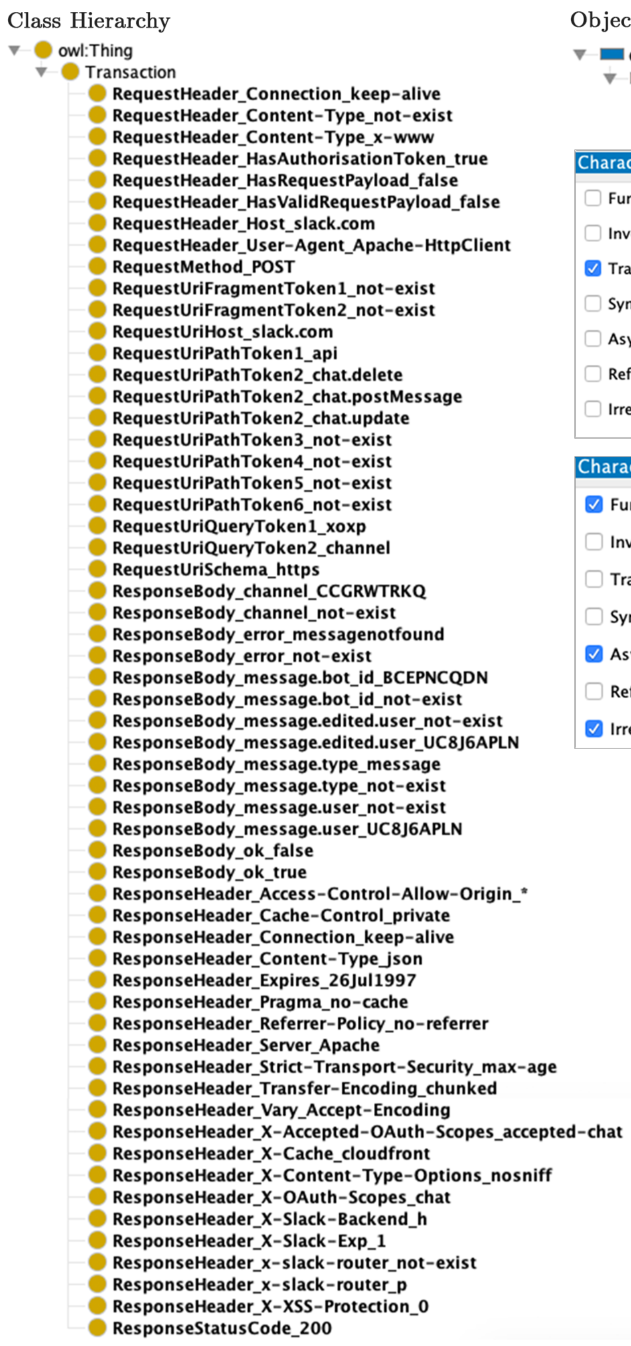
- [ ] Functional
- [ ] Inverse functional
- [✓] Transitive
- [ ] Symmetric
- [ ] Asymmetric
- [ ] Reflexive
- [ ] Irreflexive



Characteristics: hasPrevious

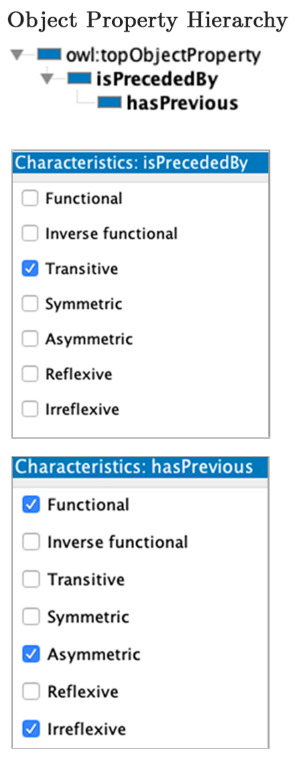
- [✓] Functional
- [ ] Inverse functional
- [ ] Transitive
- [ ] Symmetric
- [✓] Asymmetric
- [ ] Reflexive
- [✓] Irreflexive

B.4.4 Slack Knowledge Base Visualised in Protege



Class Hierarchy

- owl:Thing
  - Transaction
    - RequestHeader_Connection_keep-alive
    - RequestHeader_Content-Type_not-exist
    - RequestHeader_Content-Type_x-www
    - RequestHeader_HasAuthorisationToken_true
    - RequestHeader_HasRequestPayload_false
    - RequestHeader_HasValidRequestPayload_false
    - RequestHeader_Host_slack.com
    - RequestHeader_User-Agent_Apache-HttpClient
    - RequestMethod_POST
    - RequestUriFragmentToken1_not-exist
    - RequestUriFragmentToken2_not-exist
    - RequestUriHost_slack.com
    - RequestUriPathToken1_api
    - RequestUriPathToken2_chat.delete
    - RequestUriPathToken2_chat.postMessage
    - RequestUriPathToken2_chat.update
    - RequestUriPathToken3_not-exist
    - RequestUriPathToken4_not-exist
    - RequestUriPathToken5_not-exist
    - RequestUriPathToken6_not-exist
    - RequestUriQueryToken1_xoxp
    - RequestUriQueryToken2_channel
    - RequestUriSchema_https
    - ResponseBody_channel_CCGRWTRKQ
    - ResponseBody_channel_not-exist
    - ResponseBody_error_messagenotfound
    - ResponseBody_error_not-exist
    - ResponseBody_message.bot_id_BCEPNCQDN
    - ResponseBody_message.bot_id_not-exist
    - ResponseBody_message.edited.user_not-exist
    - ResponseBody_message.edited.user_UC8J6APLN
    - ResponseBody_message.type_message
    - ResponseBody_message.type_not-exist
    - ResponseBody_message.user_not-exist
    - ResponseBody_message.user_UC8J6APLN
    - ResponseBody_ok_false
    - ResponseBody_ok_true
    - ResponseHeader_Access-Control-Allow-Origin_*
    - ResponseHeader_Cache-Control_private
    - ResponseHeader_Connection_keep-alive
    - ResponseHeader_Content-Type_json
    - ResponseHeader_Expires_26Jul1997
    - ResponseHeader_Pragma_no-cache
    - ResponseHeader_Referrer-Policy_no-referrer
    - ResponseHeader_Server_Apache
    - ResponseHeader_Strict-Transport-Security_max-age
    - ResponseHeader_Transfer-Encoding_chunked
    - ResponseHeader_Vary_Accept-Encoding
    - ResponseHeader_X-Accepted-OAuth-Scopes_accepted-chat
    - ResponseHeader_X-Cache_cloudfront
    - ResponseHeader_X-Content-Type-Options_nosniff
    - ResponseHeader_X-OAuth-Scopes_chat
    - ResponseHeader_X-Slack-Backend_h
    - ResponseHeader_X-Slack-Exp_1
    - ResponseHeader_x-slack-router_not-exist
    - ResponseHeader_x-slack-router_p
    - ResponseHeader_X-XSS-Protection_0
    - ResponseStatusCode_200



Object Property Hierarchy

- owl:topObjectProperty
  - isPrecededBy
    - hasPrevious

Characteristics: isPrecededBy

- ☐ Functional
- ☐ Inverse functional
- ☑ Transitive
- ☐ Symmetric
- ☐ Asymmetric
- ☐ Reflexive
- ☐ Irreflexive

Characteristics: hasPrevious

- ☑ Functional
- ☐ Inverse functional
- ☐ Transitive
- ☐ Symmetric
- ☑ Asymmetric
- ☐ Reflexive
- ☑ Irreflexive

Appendix C

Results

C.1 Sample C4.5 Trees

C.1.1 Trees from C4.5 on GHTraffic

ResponseStatusCode

```
HasSuccessfulCreateOperationOccurredBefore = false
|   HasValidRequestPayload = true
|   |   RequestMethod = PATCH: 404 (153.0)
|   |   RequestMethod = DELETE: 201 (0.0)
|   |   RequestMethod = GET: 201 (0.0)
|   |   RequestMethod = POST
|   |   |   HasAuthorisationToken = true: 201 (354.0)
|   |   |   HasAuthorisationToken = false: 404 (122.0)
|   |   RequestMethod = HEAD: 201 (0.0)
|   |   RequestMethod = PUT: 201 (0.0)
|   HasValidRequestPayload = false
|   |   HasRequestPayload = true: 400 (138.0)
|   |   HasRequestPayload = false
|   |   |   RequestMethod = PATCH: 404 (0.0)
|   |   |   RequestMethod = DELETE: 404 (109.0)
|   |   |   RequestMethod = GET: 404 (161.0/10.0)
|   |   |   RequestMethod = POST: 422 (15.0)
|   |   |   RequestMethod = HEAD: 404 (171.0)
|   |   |   RequestMethod = PUT: 404 (123.0)
HasSuccessfulCreateOperationOccurredBefore = true
|   HasAuthorisationToken = true
|   |   RequestUriPathToken6 = not-exist
|   |   |   HasRequestPayload = true
|   |   |   |   HasValidRequestPayload = true: 200 (173.0)
|   |   |   |   HasValidRequestPayload = false: 400 (100.0)
|   |   |   HasRequestPayload = false
|   |   |   |   RequestMethod = PATCH: 422 (41.0)
|   |   |   |   RequestMethod = DELETE: 200 (0.0)
|   |   |   |   RequestMethod = GET: 200 (196.0)
|   |   |   |   RequestMethod = POST: 200 (0.0)
|   |   |   |   RequestMethod = HEAD: 200 (103.0)
|   |   |   |   RequestMethod = PUT: 200 (0.0)
|   |   RequestUriPathToken6 = lock: 204 (191.0)
|   HasAuthorisationToken = false: 401 (280.0)
```

ResponseHeader_Cache-Control

```
RequestUriPathToken6 = not-exist
|   HasAuthorisationToken = true
|   |   HasValidRequestPayload = true
|   |   |   RequestMethod = PATCH
|   |   |   |   HasImmediatePreviousTransactionSucceeded = false: not-exist
    (153.0)
```

```
|   |   |   |   HasImmediatePreviousTransactionSucceeded = true: private, max-age
    =60 (173.0)
|   |   |   RequestMethod = DELETE: private, max-age=60 (0.0)
|   |   |   RequestMethod = GET: private, max-age=60 (0.0)
|   |   |   RequestMethod = POST: private, max-age=60 (354.0)
|   |   |   RequestMethod = HEAD: private, max-age=60 (0.0)
|   |   |   RequestMethod = PUT: private, max-age=60 (0.0)
|   |   HasValidRequestPayload = false
|   |   |   HasSuccessfulCreateOperationOccurredBefore = false: not-exist (485.0)
|   |   |   HasSuccessfulCreateOperationOccurredBefore = true
|   |   |   |   RequestMethod = PATCH: not-exist (141.0)
|   |   |   |   RequestMethod = DELETE: private, max-age=60 (0.0)
|   |   |   |   RequestMethod = GET: private, max-age=60 (196.0)
|   |   |   |   RequestMethod = POST: private, max-age=60 (0.0)
|   |   |   |   RequestMethod = HEAD: private, max-age=60 (103.0)
|   |   |   |   RequestMethod = PUT: private, max-age=60 (0.0)
|   HasAuthorisationToken = false: not-exist (210.0)
RequestUriPathToken6 = lock: not-exist (615.0)
```

ResponseHeader_Vary

```
RequestUriPathToken6 = not-exist
|   HasAuthorisationToken = true
|   |   HasValidRequestPayload = true
|   |   |   RequestMethod = PATCH
|   |   |   |   HasImmediatePreviousTransactionSucceeded = false: not-exist
    (153.0)
|   |   |   |   HasImmediatePreviousTransactionSucceeded = true: Accept,
    Authorization, Cookie (173.0)
|   |   |   RequestMethod = DELETE: Accept, Authorization, Cookie (0.0)
|   |   |   RequestMethod = GET: Accept, Authorization, Cookie (0.0)
|   |   |   RequestMethod = POST: Accept, Authorization, Cookie (354.0)
|   |   |   RequestMethod = HEAD: Accept, Authorization, Cookie (0.0)
|   |   |   RequestMethod = PUT: Accept, Authorization, Cookie (0.0)
|   |   HasValidRequestPayload = false
|   |   |   HasSuccessfulCreateOperationOccurredBefore = false: not-exist (485.0)
|   |   |   HasSuccessfulCreateOperationOccurredBefore = true
|   |   |   |   RequestMethod = PATCH: not-exist (141.0)
|   |   |   |   RequestMethod = DELETE: Accept, Authorization, Cookie (0.0)
|   |   |   |   RequestMethod = GET: Accept, Authorization, Cookie (196.0)
|   |   |   |   RequestMethod = POST: Accept, Authorization, Cookie (0.0)
|   |   |   |   RequestMethod = HEAD: Accept, Authorization, Cookie (103.0)
|   |   |   |   RequestMethod = PUT: Accept, Authorization, Cookie (0.0)
|   HasAuthorisationToken = false: not-exist (210.0)
RequestUriPathToken6 = lock: not-exist (615.0)
```

ResponseHeader_X-Accepted-OAuth-Scopes

```
HasAuthorisationToken = true
|   HasSuccessfulCreateOperationOccurredBefore = false
|   |   HasValidRequestPayload = true
|   |   |   RequestMethod = PATCH: repo (153.0)
|   |   |   RequestMethod = DELETE: public_repo, repo (0.0)
|   |   |   RequestMethod = GET: public_repo, repo (0.0)
|   |   |   RequestMethod = POST: public_repo, repo (354.0)
|   |   |   RequestMethod = HEAD: public_repo, repo (0.0)
|   |   |   RequestMethod = PUT: public_repo, repo (0.0)
|   |   HasValidRequestPayload = false: repo (717.0)
|   HasSuccessfulCreateOperationOccurredBefore = true
|   |   RequestMethod = PATCH
|   |   |   HasValidRequestPayload = true: public_repo, repo (173.0)
|   |   |   HasValidRequestPayload = false: repo (141.0)
|   |   RequestMethod = DELETE: public_repo, repo (48.0)
|   |   RequestMethod = GET: public_repo, repo (196.0)
|   |   RequestMethod = POST: public_repo, repo (0.0)
|   |   RequestMethod = HEAD: public_repo, repo (103.0)
|   |   RequestMethod = PUT: public_repo, repo (143.0)
HasAuthorisationToken = false: not-exist (402.0)
```

ResponseHeader_X-OAuth-Scopes

```
HasAuthorisationToken = true: public_repo (2028.0)
HasAuthorisationToken = false: not-exist (402.0)
```

ResponseBody_assignee.site_admin

```
: not-exist (2430.0/116.0)
```

ResponseBody_assignee.type

```
: not-exist (2430.0/116.0)
```

ResponseBody_assignees.site_admin

```
: not-exist (2430.0/95.0)
```

ResponseBody_assignees.type

```
: not-exist (2430.0/95.0)
```

ResponseBody_closed_by.avatar_url

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.events_url

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.followers_url

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.following_url

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.gists_url

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.html_url

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.id

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.login

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.organizations_url

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.received_events_url

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.repos_url

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.site_admin

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.starred_url

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.subscriptions_url

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.type

```
: not-exist (2430.0/8.0)
```

ResponseBody_documentation_url

```
HasAuthorisationToken = true
|   HasSuccessfulCreateOperationOccurredBefore = false
|   |   RequestMethod = PATCH: https://developer.github.com/v3 (153.0)
|   |   RequestMethod = DELETE: https://developer.github.com/v3 (109.0)
|   |   RequestMethod = GET: https://developer.github.com/v3 (161.0/10.0)
|   |   RequestMethod = POST
|   |   |   HasValidRequestPayload = true: not-exist (354.0)
|   |   |   HasValidRequestPayload = false: https://developer.github.com/v3/
    issues/#create-an-issue (153.0)
|   |   RequestMethod = HEAD: not-exist (171.0)
|   |   RequestMethod = PUT: https://developer.github.com/v3 (123.0)
|   HasSuccessfulCreateOperationOccurredBefore = true
|   |   RequestMethod = PATCH
|   |   |   HasValidRequestPayload = true: not-exist (173.0)
|   |   |   HasValidRequestPayload = false: https://developer.github.com/v3/
    issues/#edit-an-issue (141.0)
|   |   RequestMethod = DELETE: not-exist (48.0)
|   |   RequestMethod = GET: not-exist (196.0)
|   |   RequestMethod = POST: not-exist (0.0)
|   |   RequestMethod = HEAD: not-exist (103.0)
|   |   RequestMethod = PUT: not-exist (143.0)
HasAuthorisationToken = false
|   RequestMethod = PATCH: https://developer.github.com/v3/issues/#edit-an-issue
    (88.0)
|   RequestMethod = DELETE: https://developer.github.com/v3/issues/#unlock-an-
    issue (64.0)
|   RequestMethod = GET: https://developer.github.com/v3/issues/#lock-an-issue
    (0.0)
|   RequestMethod = POST: https://developer.github.com/v3/issues/#create-an-issue
     (122.0)
|   RequestMethod = HEAD: https://developer.github.com/v3/issues/#lock-an-issue
    (0.0)
|   RequestMethod = PUT: https://developer.github.com/v3/issues/#lock-an-issue
    (128.0)
```

ResponseBody_locked

```
HasValidRequestPayload = true
|   HasAuthorisationToken = true
|   |   RequestMethod = PATCH
|   |   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (153.0)
|   |   |   HasImmediatePreviousTransactionSucceeded = true: false (173.0)
|   |   RequestMethod = DELETE: false (0.0)
|   |   RequestMethod = GET: false (0.0)
|   |   RequestMethod = POST: false (354.0)
|   |   RequestMethod = HEAD: false (0.0)
|   |   RequestMethod = PUT: false (0.0)
|   HasAuthorisationToken = false: not-exist (122.0)
HasValidRequestPayload = false
|   RequestMethod = PATCH: not-exist (229.0)
|   RequestMethod = DELETE: not-exist (221.0)
|   RequestMethod = GET
|   |   HasSuccessfulCreateOperationOccurredBefore = false: not-exist (161.0)
|   |   HasSuccessfulCreateOperationOccurredBefore = true: false (196.0)
|   RequestMethod = POST: not-exist (153.0)
|   RequestMethod = HEAD: not-exist (274.0)
|   RequestMethod = PUT: not-exist (394.0)
```

ResponseBody_message

```
HasAuthorisationToken = true
|   HasSuccessfulCreateOperationOccurredBefore = false
|   |   RequestMethod = PATCH: Not Found (153.0)
|   |   RequestMethod = DELETE: Not Found (109.0)
|   |   RequestMethod = GET: Not Found (161.0/10.0)
|   |   RequestMethod = POST
|   |   |   HasValidRequestPayload = true: not-exist (354.0)
|   |   |   HasValidRequestPayload = false
|   |   |   |   HasRequestPayload = true: Problems parsing JSON (138.0)
|   |   |   |   HasRequestPayload = false: Invalid request (15.0)
|   |   RequestMethod = HEAD: not-exist (171.0)
|   |   RequestMethod = PUT: Not Found (123.0)
|   HasSuccessfulCreateOperationOccurredBefore = true
|   |   HasRequestPayload = true
|   |   |   HasValidRequestPayload = true: not-exist (173.0)
|   |   |   HasValidRequestPayload = false: Problems parsing JSON (100.0)
|   |   HasRequestPayload = false
|   |   |   RequestMethod = PATCH: Invalid request (41.0)
|   |   |   RequestMethod = DELETE: not-exist (48.0)
|   |   |   RequestMethod = GET: not-exist (196.0)
|   |   |   RequestMethod = POST: not-exist (0.0)
|   |   |   RequestMethod = HEAD: not-exist (103.0)
|   |   |   RequestMethod = PUT: not-exist (143.0)
HasAuthorisationToken = false
|   HasValidRequestPayload = true: Not Found (122.0)
|   HasValidRequestPayload = false: Requires authentication (280.0)
```

ResponseBody_milestone.creator.avatar_url

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.events_url

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.followers_url

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.following_url

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.gists_url

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.html_url

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.id

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.login

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.organizations_url

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.received_events_url

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.repos_url

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.site_admin

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.starred_url

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.subscriptions_url

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.type

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.url

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.due_on

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.open_issues

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.state

```
: not-exist (2430.0/112.0)
```

ResponseBody_state

```
RequestBody_state = closed
|   HasImmediatePreviousTransactionSucceeded = false: not-exist (173.0)
|   HasImmediatePreviousTransactionSucceeded = true
|   |   HasValidRequestPayload = true: closed (173.0)
|   |   HasValidRequestPayload = false: not-exist (68.0)
RequestBody_state = not-exist
|   HasValidRequestPayload = true
|   |   HasAuthorisationToken = true: open (354.0)
|   |   HasAuthorisationToken = false: not-exist (122.0)
|   HasValidRequestPayload = false
|   |   RequestMethod = PATCH: not-exist (141.0)
|   |   RequestMethod = DELETE: not-exist (221.0)
|   |   RequestMethod = GET
|   |   |   HasSuccessfulCreateOperationOccurredBefore = false: not-exist (161.0)
|   |   |   HasSuccessfulCreateOperationOccurredBefore = true
|   |   |   |   HasSuccessfulUpdateOperationOccurredBefore = false: open
    (84.0/2.0)
|   |   |   |   HasSuccessfulUpdateOperationOccurredBefore = true
|   |   |   |   |   ImmediatelyPreviousStatusCode = not-exist: open (0.0)
|   |   |   |   |   ImmediatelyPreviousStatusCode = 404: open (0.0)
|   |   |   |   |   ImmediatelyPreviousStatusCode = 400
|   |   |   |   |   |   HasSuccessfulReadOperationOccurredBefore = false: closed
    (5.0/1.0)
|   |   |   |   |   |   HasSuccessfulReadOperationOccurredBefore = true: open
    (3.0/1.0)
|   |   |   |   |   ImmediatelyPreviousStatusCode = 201: open (0.0)
|   |   |   |   |   ImmediatelyPreviousStatusCode = 200: closed (39.0/7.0)
|   |   |   |   |   ImmediatelyPreviousStatusCode = 401: open (16.0/6.0)
|   |   |   |   |   ImmediatelyPreviousStatusCode = 204
|   |   |   |   |   |   ImmediatelyPreviousMethod = not-exist: open (0.0)
|   |   |   |   |   |   ImmediatelyPreviousMethod = PATCH: open (0.0)
|   |   |   |   |   |   ImmediatelyPreviousMethod = DELETE: closed (4.0/1.0)
|   |   |   |   |   |   ImmediatelyPreviousMethod = GET: open (0.0)
|   |   |   |   |   |   ImmediatelyPreviousMethod = POST: open (0.0)
|   |   |   |   |   |   ImmediatelyPreviousMethod = HEAD: open (0.0)
|   |   |   |   |   |   ImmediatelyPreviousMethod = PUT: open (41.0/7.0)
|   |   |   |   |   ImmediatelyPreviousStatusCode = 500: open (0.0)
|   |   |   |   |   ImmediatelyPreviousStatusCode = 422: open (4.0/1.0)
|   |   RequestMethod = POST: not-exist (153.0)
|   |   RequestMethod = HEAD: not-exist (274.0)
|   |   RequestMethod = PUT: not-exist (394.0)
```

ResponseBody_user.site_admin

```
HasValidRequestPayload = true
|   HasAuthorisationToken = true
|   |   RequestMethod = PATCH
|   |   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (153.0)
|   |   |   HasImmediatePreviousTransactionSucceeded = true: false (173.0)
|   |   RequestMethod = DELETE: false (0.0)
|   |   RequestMethod = GET: false (0.0)
|   |   RequestMethod = POST: false (354.0)
|   |   RequestMethod = HEAD: false (0.0)
|   |   RequestMethod = PUT: false (0.0)
|   HasAuthorisationToken = false: not-exist (122.0)
HasValidRequestPayload = false
|   RequestMethod = PATCH: not-exist (229.0)
|   RequestMethod = DELETE: not-exist (221.0)
|   RequestMethod = GET
|   |   HasSuccessfulCreateOperationOccurredBefore = false: not-exist (161.0)
|   |   HasSuccessfulCreateOperationOccurredBefore = true: false (196.0)
|   RequestMethod = POST: not-exist (153.0)
|   RequestMethod = HEAD: not-exist (274.0)
|   RequestMethod = PUT: not-exist (394.0)
```

ResponseBody_user.type

```
HasValidRequestPayload = true
|   HasAuthorisationToken = true
|   |   RequestMethod = PATCH
|   |   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (153.0)
|   |   |   HasImmediatePreviousTransactionSucceeded = true: User (173.0)
|   |   RequestMethod = DELETE: User (0.0)
|   |   RequestMethod = GET: User (0.0)
|   |   RequestMethod = POST: User (354.0)
|   |   RequestMethod = HEAD: User (0.0)
|   |   RequestMethod = PUT: User (0.0)
|   HasAuthorisationToken = false: not-exist (122.0)
HasValidRequestPayload = false
|   RequestMethod = PATCH: not-exist (229.0)
|   RequestMethod = DELETE: not-exist (221.0)
|   RequestMethod = GET
|   |   HasSuccessfulCreateOperationOccurredBefore = false: not-exist (161.0)
|   |   HasSuccessfulCreateOperationOccurredBefore = true: User (196.0)
|   RequestMethod = POST: not-exist (153.0)
|   RequestMethod = HEAD: not-exist (274.0)
|   RequestMethod = PUT: not-exist (394.0)
```

C.1.2 Trees from C4.5 on Twitter

ResponseStatusCode

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: 200 (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: 404 (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: 404 (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: 404 (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: 200 (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: 200
    (547.0)
```

ResponseHeader_status

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: 200 OK (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: 404 Not Found (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: 404 Not Found (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: 404 Not Found (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: 200 OK (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: 200 OK
    (547.0)
```

ResponseHeader_x-rate-limit-limit

```
RequestMethod = POST: not-exist (20445.0)
RequestMethod = GET: 900 (5608.0)
```

ResponseBody_contributors

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: null (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: null (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: null
    (547.0)
```

ResponseBody_coordinates

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: null (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: null (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: null
    (547.0)
```

ResponseBody_entities.hashtags

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: [] (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: [] (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: []
    (547.0)
```

ResponseBody_entities.symbols

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: [] (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: [] (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: []
    (547.0)
```

ResponseBody_entities.urls

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: [] (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: [] (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: []
    (547.0)
```

ResponseBody_entities.user_mentions

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: [] (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: [] (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: []
    (547.0)
```

ResponseBody_errors.code

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: not-exist (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: 144 (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: 144 (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: 144 (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: not-exist (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: not-
    exist (547.0)
```

ResponseBody_errors.message

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: not-exist (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: No status found with
    that ID. (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: No status found with that ID.
    (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: No status found with that ID.
    (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: not-exist (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: not-
    exist (547.0)
```

ResponseBody_favorite_count

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: 0 (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: 0 (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: 0
    (547.0)
```

ResponseBody_favorited

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: false (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: false (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: false
    (547.0)
```

ResponseBody_geo

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: null (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: null (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: null
    (547.0)
```

ResponseBody_in_reply_to_screen_name

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: null (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: null (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: null
    (547.0)
```

ResponseBody_in_reply_to_status_id

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: null (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: null (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: null
    (547.0)
```

ResponseBody_in_reply_to_status_id_str

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: null (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: null (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: null
    (547.0)
```

ResponseBody_in_reply_to_user_id

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: null (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: null (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: null
    (547.0)
```

ResponseBody_in_reply_to_user_id_str

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: null (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: null (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: null
    (547.0)
```

ResponseBody_is_quote_status

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: false (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: false (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: false
    (547.0)
```

ResponseBody_place

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: null (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: null (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: null
    (547.0)
```

ResponseBody_retweet_count

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: 0 (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: 0 (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: 0
    (547.0)
```

ResponseBody_retweeted

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: false (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: false (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: false
    (547.0)
```

ResponseBody_truncated

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: false (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: false (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: false
    (547.0)
```

ResponseBody_user.contributors_enabled

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: false (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: false (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: false
    (547.0)
```

ResponseBody_user.created_at

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: Wed Mar 07 09
    :41:33 +0000 2012 (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: Wed Mar 07 09:41:33 +0000 2012
    (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: Wed Mar
    07 09:41:33 +0000 2012 (547.0)
```

ResponseBody_user.default_profile

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: false (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: false (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: false
    (547.0)
```

ResponseBody_user.default_profile_image

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: false (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: false (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: false
    (547.0)
```

ResponseBody_user.favourites_count

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: 64 (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: 64 (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: 64
    (547.0)
```

ResponseBody_user.follow_request_sent

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: false (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: false (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: false
    (547.0)
```

ResponseBody_user.followers_count

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: 185 (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: 185 (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: 185
    (547.0)
```

ResponseBody_user.following

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: false (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: false (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: false
    (547.0)
```

ResponseBody_user.friends_count

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: 249 (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: 249 (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: 249
    (547.0)
```

ResponseBody_user.geo_enabled

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: true (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: true (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: true
    (547.0)
```

ResponseBody_user.has_extended_profile

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: false (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: false (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: false
    (547.0)
```

ResponseBody_user.id

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: 517417816 (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: 517417816 (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true:
    517417816 (547.0)
```

ResponseBody_user.id_str

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: 517417816 (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: 517417816 (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true:
    517417816 (547.0)
```

ResponseBody_user.is_translation_enabled

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: false (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: false (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: false
    (547.0)
```

ResponseBody_user.is_translator

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: false (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: false (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: false
    (547.0)
```

ResponseBody_user.lang

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: en (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: en (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: en
    (547.0)
```

ResponseBody_user.listed_count

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: 3 (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: 3 (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: 3
    (547.0)
```

ResponseBody_user.location

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: Kurunegala
    (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: Kurunegala (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true:
    Kurunegala (547.0)
```

ResponseBody_user.name

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: Thilini Bhagya
    (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: Thilini Bhagya (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: Thilini
    Bhagya (547.0)
```

ResponseBody_user.notifications

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: false (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: false (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: false
    (547.0)
```

ResponseBody_user.profile_background_color

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: 1A1B1F (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: 1A1B1F (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: 1A1B1F
    (547.0)
```

ResponseBody_user.profile_background_image_url

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: http://abs.twimg.
    com/images/themes/theme9/bg.gif (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: http://abs.twimg.com/images/themes/
    theme9/bg.gif (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: http://
    abs.twimg.com/images/themes/theme9/bg.gif (547.0)
```

ResponseBody_user.profile_background_image_url_https

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: https://abs.twimg
    .com/images/themes/theme9/bg.gif (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: https://abs.twimg.com/images/themes/
    theme9/bg.gif (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: https://
    abs.twimg.com/images/themes/theme9/bg.gif (547.0)
```

ResponseBody_user.profile_background_tile

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: false (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: false (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: false
    (547.0)
```

ResponseBody_user.profile_banner_url

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: https://pbs.twimg
    .com/profile_banners/517417816/1399047954 (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: https://pbs.twimg.com/
    profile_banners/517417816/1399047954 (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: https://
    pbs.twimg.com/profile_banners/517417816/1399047954 (547.0)
```

ResponseBody_user.profile_image_url

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: http://pbs.twimg.
    com/profile_images/950100192048562176/LKr7Ay21_normal.jpg (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: http://pbs.twimg.com/profile_images
    /950100192048562176/LKr7Ay21_normal.jpg (101.0)
```

```
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: http://
    pbs.twimg.com/profile_images/950100192048562176/LKr7Ay21_normal.jpg (547.0)
```

ResponseBody_user.profile_image_url_https

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: https://pbs.twimg
    .com/profile_images/950100192048562176/LKr7Ay21_normal.jpg (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: https://pbs.twimg.com/profile_images
    /950100192048562176/LKr7Ay21_normal.jpg (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: https://
    pbs.twimg.com/profile_images/950100192048562176/LKr7Ay21_normal.jpg (547.0)
```

ResponseBody_user.profile_link_color

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: 3E4547 (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: 3E4547 (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: 3E4547
    (547.0)
```

ResponseBody_user.profile_sidebar_border_color

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: FFFFFF (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: FFFFFF (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: FFFFFF
    (547.0)
```

ResponseBody_user.profile_sidebar_fill_color

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: 252429 (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: 252429 (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: 252429
    (547.0)
```

ResponseBody_user.profile_text_color

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: 666666 (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: 666666 (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: 666666
    (547.0)
```

ResponseBody_user.profile_use_background_image

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: true (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: true (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: true
    (547.0)
```

ResponseBody_user.protected

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: false (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: false (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: false
    (547.0)
```

ResponseBody_user.screen_name

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: bhagyasl (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: bhagyasl (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: bhagyasl
     (547.0)
```

ResponseBody_user.time_zone

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: null (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: null (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: null
    (547.0)
```

ResponseBody_user.translator_type

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: none (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: none (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: none
    (547.0)
```

ResponseBody_user.url

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: http://t.co/
    sQduiwqJiy (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: http://t.co/sQduiwqJiy (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: http://t
    .co/sQduiwqJiy (547.0)
```

ResponseBody_user.utc_offset

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: null (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: null (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: null
    (547.0)
```

ResponseBody_user.verified

```
RequestHeader_Content-Type = application/x-www-form-urlencoded: false (867.0)
RequestHeader_Content-Type = not-exist
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)
|   |   HasImmediatePreviousTransactionSucceeded = true
|   |   |   ImmediatelyPreviousMethod = not-exist: not-exist (0.0)
|   |   |   ImmediatelyPreviousMethod = POST: not-exist (513.0)
|   |   |   ImmediatelyPreviousMethod = GET: false (101.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true: false
    (547.0)
```

C.1.3 Trees from C4.5 on Google Tasks

ResponseStatusCode

```
HasSuccessfulDeleteOperationOccurredBefore = false
|   RequestMethod = POST: 200 (1124.0)
|   RequestMethod = GET: 200 (1005.0/3.0)
|   RequestMethod = PATCH: 200 (485.0/4.0)
|   RequestMethod = DELETE: 204 (606.0)
HasSuccessfulDeleteOperationOccurredBefore = true: 404 (1482.0/4.0)
```

ResponseHeader_Accept-Ranges

```
HasRequestPayload = true: none (2206.0)
HasRequestPayload = false
|   HasSuccessfulDeleteOperationOccurredBefore = false: not-exist (1611.0/3.0)
|   HasSuccessfulDeleteOperationOccurredBefore = true: none (885.0)
```

ResponseHeader_Cache-Control

```
HasSuccessfulDeleteOperationOccurredBefore = false
|   RequestMethod = POST: no-cache, no-store, max-age=0, must-revalidate (1124.0)
|   RequestMethod = GET: private, max-age=0, must-revalidate, no-transform
    (1005.0/3.0)
|   RequestMethod = PATCH: no-cache, no-store, max-age=0, must-revalidate
    (485.0/4.0)
|   RequestMethod = DELETE: no-cache, no-store, max-age=0, must-revalidate
    (606.0)
HasSuccessfulDeleteOperationOccurredBefore = true: private, max-age=0 (1482.0)
```

ResponseHeader_Content-Type

```
RequestMethod = POST: application/json; charset=UTF-8 (1124.0)
RequestMethod = GET: application/json; charset=UTF-8 (1367.0)
RequestMethod = PATCH: application/json; charset=UTF-8 (1082.0)
RequestMethod = DELETE
|   HasSuccessfulDeleteOperationOccurredBefore = false: not-exist (606.0)
|   HasSuccessfulDeleteOperationOccurredBefore = true: application/json; charset=
    UTF-8 (523.0)
```

ResponseHeader_Pragma

```
HasSuccessfulDeleteOperationOccurredBefore = false
|   RequestMethod = POST: no-cache (1124.0)
|   RequestMethod = GET: not-exist (1005.0)
|   RequestMethod = PATCH: no-cache (485.0/4.0)
|   RequestMethod = DELETE: no-cache (606.0)
HasSuccessfulDeleteOperationOccurredBefore = true: not-exist (1482.0)
```

ResponseHeader_Transfer-Encoding

```
HasRequestPayload = true: chunked (2206.0)
HasRequestPayload = false
|   HasSuccessfulDeleteOperationOccurredBefore = false: not-exist (1611.0/3.0)
|   HasSuccessfulDeleteOperationOccurredBefore = true: chunked (885.0)
```

ResponseHeader_Vary

```
HasRequestPayload = true: Origin,Accept-Encoding (2206.0)
HasRequestPayload = false
|   HasSuccessfulDeleteOperationOccurredBefore = false: X-Origin (1611.0/3.0)
|   HasSuccessfulDeleteOperationOccurredBefore = true: Origin,Accept-Encoding
    (885.0)
```

ResponseHeader_X-Content-Type-Options

```
RequestMethod = POST: nosniff (1124.0)
RequestMethod = GET: nosniff (1367.0)
RequestMethod = PATCH: nosniff (1082.0)
RequestMethod = DELETE
|   HasSuccessfulDeleteOperationOccurredBefore = false: not-exist (606.0)
|   HasSuccessfulDeleteOperationOccurredBefore = true: nosniff (523.0)
```

ResponseHeader_X-Frame-Options

```
RequestMethod = POST: SAMEORIGIN (1124.0)
RequestMethod = GET: SAMEORIGIN (1367.0)
RequestMethod = PATCH: SAMEORIGIN (1082.0)
RequestMethod = DELETE
|   HasSuccessfulDeleteOperationOccurredBefore = false: not-exist (606.0)
|   HasSuccessfulDeleteOperationOccurredBefore = true: SAMEORIGIN (523.0)
```

ResponseHeader_X-XSS-Protection

```
RequestMethod = POST: 1; mode=block (1124.0)
RequestMethod = GET: 1; mode=block (1367.0)
RequestMethod = PATCH: 1; mode=block (1082.0)
RequestMethod = DELETE
|   HasSuccessfulDeleteOperationOccurredBefore = false: not-exist (606.0)
|   HasSuccessfulDeleteOperationOccurredBefore = true: 1; mode=block (523.0)
```

ResponseBody_error.code

```
HasSuccessfulDeleteOperationOccurredBefore = false: not-exist (3220.0/7.0)
HasSuccessfulDeleteOperationOccurredBefore = true: 404 (1482.0/4.0)
```

ResponseBody_error.errors.domain

```
HasSuccessfulDeleteOperationOccurredBefore = false: not-exist (3220.0/7.0)
HasSuccessfulDeleteOperationOccurredBefore = true: global (1482.0)
```

ResponseBody_error.errors.message

```
HasSuccessfulDeleteOperationOccurredBefore = false: not-exist (3220.0/7.0)
HasSuccessfulDeleteOperationOccurredBefore = true: NotFound (1482.0/4.0)
```

ResponseBody_error.errors.reason

```
HasSuccessfulDeleteOperationOccurredBefore = false: not-exist (3220.0/7.0)
HasSuccessfulDeleteOperationOccurredBefore = true: notFound (1482.0/4.0)
```

ResponseBody_error.message

```
HasSuccessfulDeleteOperationOccurredBefore = false: not-exist (3220.0/7.0)
HasSuccessfulDeleteOperationOccurredBefore = true: NotFound (1482.0/4.0)
```

ResponseBody_kind

```
HasSuccessfulDeleteOperationOccurredBefore = false
|   RequestMethod = POST: tasks#taskList (1124.0)
|   RequestMethod = GET: tasks#taskList (1005.0/3.0)
|   RequestMethod = PATCH: tasks#taskList (485.0/4.0)
|   RequestMethod = DELETE: not-exist (606.0)
HasSuccessfulDeleteOperationOccurredBefore = true: not-exist (1482.0)
```

C.1.4 Trees from C4.5 on Slack

ResponseHeader_x-slack-router

```
: p (17422.0/6359.0)
```

ResponseBody_channel

```
HasSuccessfulDeleteOperationOccurredBefore = false
|   HasURLInImmediatelyPreviousTransactionContainsATokenToUpdate = false
|   |   RequestUriPathToken2 = chat.update
|   |   |   HasImmediatePreviousTransaction = false: not-exist (309.0)
|   |   |   HasImmediatePreviousTransaction = true: CCGRWTRKQ (1292.0)
|   |   RequestUriPathToken2 = chat.delete
|   |   |   HasImmediatePreviousTransaction = false: not-exist (206.0)
|   |   |   HasImmediatePreviousTransaction = true: CCGRWTRKQ (1826.0)
|   |   RequestUriPathToken2 = chat.postMessage: CCGRWTRKQ (3985.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToUpdate = true
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (335.0)
|   |   HasImmediatePreviousTransactionSucceeded = true: CCGRWTRKQ (540.0)
HasSuccessfulDeleteOperationOccurredBefore = true: not-exist (8929.0)
```

ResponseBody_error

```
HasSuccessfulDeleteOperationOccurredBefore = false
|   HasURLInImmediatelyPreviousTransactionContainsATokenToUpdate = false
|   |   RequestUriPathToken2 = chat.update
|   |   |   HasImmediatePreviousTransaction = false: message_not_found (309.0)
|   |   |   HasImmediatePreviousTransaction = true: not-exist (1292.0)
|   |   RequestUriPathToken2 = chat.delete
|   |   |   HasImmediatePreviousTransaction = false: message_not_found (206.0)
|   |   |   HasImmediatePreviousTransaction = true: not-exist (1826.0)
|   |   RequestUriPathToken2 = chat.postMessage: not-exist (3985.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToUpdate = true
|   |   HasImmediatePreviousTransactionSucceeded = false: message_not_found
    (335.0)
|   |   HasImmediatePreviousTransactionSucceeded = true: not-exist (540.0)
HasSuccessfulDeleteOperationOccurredBefore = true: message_not_found (8929.0)
```

ResponseBody_message.bot_id

```
HasSuccessfulDeleteOperationOccurredBefore = false
|   RequestUriPathToken2 = chat.update
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (392.0)
|   |   HasImmediatePreviousTransactionSucceeded = true: BCEPNCQDN (1341.0)
|   RequestUriPathToken2 = chat.delete: not-exist (2775.0)
|   RequestUriPathToken2 = chat.postMessage: BCEPNCQDN (3985.0)
HasSuccessfulDeleteOperationOccurredBefore = true: not-exist (8929.0)
```

ResponseBody_message.edited.user

```
HasURLInImmediatelyPreviousTransactionContainsATokenToUpdate = false: not-exist
    (12564.0)
HasURLInImmediatelyPreviousTransactionContainsATokenToUpdate = true
|   HasImmediatePreviousTransactionSucceeded = false: not-exist (4318.0)
|   HasImmediatePreviousTransactionSucceeded = true
|   |   RequestUriPathToken2 = chat.update: UC8J6APLN (49.0)
|   |   RequestUriPathToken2 = chat.delete: not-exist (491.0)
|   |   RequestUriPathToken2 = chat.postMessage: not-exist (0.0)
```

ResponseBody_message.type

```
HasSuccessfulDeleteOperationOccurredBefore = false
|   RequestUriPathToken2 = chat.update
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (392.0)
|   |   HasImmediatePreviousTransactionSucceeded = true: message (1341.0)
|   RequestUriPathToken2 = chat.delete: not-exist (2775.0)
|   RequestUriPathToken2 = chat.postMessage: message (3985.0)
HasSuccessfulDeleteOperationOccurredBefore = true: not-exist (8929.0)
```

ResponseBody_message.user

```
HasSuccessfulDeleteOperationOccurredBefore = false
|   RequestUriPathToken2 = chat.update
|   |   HasImmediatePreviousTransactionSucceeded = false: not-exist (392.0)
|   |   HasImmediatePreviousTransactionSucceeded = true: UC8J6APLN (1341.0)
|   RequestUriPathToken2 = chat.delete: not-exist (2775.0)
|   RequestUriPathToken2 = chat.postMessage: UC8J6APLN (3985.0)
HasSuccessfulDeleteOperationOccurredBefore = true: not-exist (8929.0)
```

ResponseBody_ok

```
HasSuccessfulDeleteOperationOccurredBefore = false
|   HasURLInImmediatelyPreviousTransactionContainsATokenToUpdate = false
|   |   RequestUriPathToken2 = chat.update
|   |   |   HasImmediatePreviousTransaction = false: false (309.0)
|   |   |   HasImmediatePreviousTransaction = true: true (1292.0)
|   |   RequestUriPathToken2 = chat.delete
|   |   |   HasImmediatePreviousTransaction = false: false (206.0)
|   |   |   HasImmediatePreviousTransaction = true: true (1826.0)
|   |   RequestUriPathToken2 = chat.postMessage: true (3985.0)
|   HasURLInImmediatelyPreviousTransactionContainsATokenToUpdate = true
|   |   HasImmediatePreviousTransactionSucceeded = false: false (335.0)
|   |   HasImmediatePreviousTransactionSucceeded = true: true (540.0)
HasSuccessfulDeleteOperationOccurredBefore = true: false (8929.0)
```

C.2 Sample RIPPER Rulesets

C.2.1 Rules from RIPPER on GHTraffic

ResponseStatusCode

```
(RequestMethod = PATCH) and
(HasRequestPayload = false) => ResponseStatusCode=422 (41.0/0.0)

(RequestMethod = POST) and
(HasRequestPayload = false) => ResponseStatusCode=422 (15.0/0.0)

(RequestUriPathToken6 = lock) and
(HasSuccessfulCreateOperationOccurredBefore = true) and
(HasAuthorisationToken = true) => ResponseStatusCode=204 (191.0/0.0)

(HasRequestPayload = true) and
(HasValidRequestPayload = false) and
(HasAuthorisationToken = true) => ResponseStatusCode=400 (238.0/0.0)

(HasAuthorisationToken = false) and
(HasValidRequestPayload = false) => ResponseStatusCode=401 (280.0/0.0)

(RequestMethod = POST) and
(HasAuthorisationToken = true) => ResponseStatusCode=201 (354.0/0.0)

(HasSuccessfulCreateOperationOccurredBefore = true) => ResponseStatusCode=200
    (472.0/0.0)

=> ResponseStatusCode=404 (839.0/10.0)
```

ResponseHeader_Cache-Control

```
(HasValidRequestPayload = true) and
(HasAuthorisationToken = true) and
(RequestMethod = POST) => ResponseHeader_Cache-Control=private, max-age=60
    (354.0/0.0)

(HasSuccessfulCreateOperationOccurredBefore = true) and
(RequestUriPathToken6 = not-exist) and (HasRequestPayload = false) and
(RequestMethod = GET) => ResponseHeader_Cache-Control=private, max-age=60
    (196.0/0.0)

(ImmediatelyPreviousStatusCode = 201) and
(HasValidRequestPayload = true) => ResponseHeader_Cache-Control=private, max-age
    =60 (173.0/0.0)

(RequestMethod = HEAD) and
(HasSuccessfulCreateOperationOccurredBefore = true) => ResponseHeader_Cache-
    Control=private, max-age=60 (103.0/0.0)

=> ResponseHeader_Cache-Control=not-exist (1604.0/0.0)
```

ResponseHeader_Vary

```
(HasValidRequestPayload = true) and
(HasAuthorisationToken = true) and
(RequestMethod = POST) => ResponseHeader_Vary=Accept, Authorization, Cookie
    (354.0/0.0)

(HasSuccessfulCreateOperationOccurredBefore = true) and
(RequestUriPathToken6 = not-exist)
and (HasRequestPayload = false) and
(RequestMethod = GET) => ResponseHeader_Vary=Accept, Authorization, Cookie
    (196.0/0.0)

(ImmediatelyPreviousStatusCode = 201) and
(HasValidRequestPayload = true) => ResponseHeader_Vary=Accept, Authorization,
    Cookie (173.0/0.0)
```

```
(RequestMethod = HEAD) and
(HasSuccessfulCreateOperationOccurredBefore = true) => ResponseHeader_Vary=Accept
    , Authorization, Cookie (103.0/0.0)

=> ResponseHeader_Vary=not-exist (1604.0/0.0)
```

ResponseHeader_X-Accepted-OAuth-Scopes

```
(HasAuthorisationToken = false) => ResponseHeader_X-Accepted-OAuth-Scopes=not-
    exist (402.0/0.0)

(HasSuccessfulCreateOperationOccurredBefore = false) and
(HasValidRequestPayload = false) => ResponseHeader_X-Accepted-OAuth-Scopes=repo
    (717.0/0.0)

(RequestMethod = PATCH) and
(HasImmediatePreviousTransactionSucceeded = false) => ResponseHeader_X-Accepted-
    OAuth-Scopes=repo (194.0/0.0)

(RequestMethod = PATCH) and
(HasValidRequestPayload = false) => ResponseHeader_X-Accepted-OAuth-Scopes=repo
    (100.0/0.0)

=> ResponseHeader_X-Accepted-OAuth-Scopes=public_repo, repo (1017.0/0.0)
```

ResponseHeader_X-OAuth-Scopes

```
(HasAuthorisationToken = false) => ResponseHeader_X-OAuth-Scopes=not-exist
    (402.0/0.0)

=> ResponseHeader_X-OAuth-Scopes=public_repo (2028.0/0.0)
```

ResponseBody_assignee.site_admin

```
 => ResponseBody_assignee.site_admin=not-exist (2430.0/116.0)
```

ResponseBody_assignee.type

```
 => ResponseBody_assignee.type=not-exist (2430.0/116.0)
```

ResponseBody_assignees.site_admin

```
 => ResponseBody_assignees.site_admin=not-exist (2430.0/95.0)
```

ResponseBody_assignees.type

```
 => ResponseBody_assignees.type=not-exist (2430.0/95.0)
```

ResponseBody_closed_by.avatar_url

```
 => ResponseBody_closed_by.avatar_url=not-exist (2430.0/8.0)
```

ResponseBody_closed_by.events_url

```
 => ResponseBody_closed_by.events_url=not-exist (2430.0/8.0)
```

ResponseBody_closed_by.followers_url

```
 => ResponseBody_closed_by.followers_url=not-exist (2430.0/8.0)
```

ResponseBody_closed_by.following_url

```
 => ResponseBody_closed_by.following_url=not-exist (2430.0/8.0)
```

ResponseBody_closed_by.gists_url

```
 => ResponseBody_closed_by.gists_url=not-exist (2430.0/8.0)
```

ResponseBody_closed_by.html_url

```
 => ResponseBody_closed_by.html_url=not-exist (2430.0/8.0)
```

ResponseBody_closed_by.id

```
 => ResponseBody_closed_by.id=not-exist (2430.0/8.0)
```

ResponseBody_closed_by.login

```
 => ResponseBody_closed_by.login=not-exist (2430.0/8.0)
```

ResponseBody_closed_by.organizations_url

```
 => ResponseBody_closed_by.organizations_url=not-exist (2430.0/8.0)
```

ResponseBody_closed_by.received_events_url

```
 => ResponseBody_closed_by.received_events_url=not-exist (2430.0/8.0)
```

ResponseBody_closed_by.repos_url

```
 => ResponseBody_closed_by.repos_url=not-exist (2430.0/8.0)
```

ResponseBody_closed_by.site_admin

```
=> ResponseBody_closed_by.repos_url=not-exist (2430.0/8.0)
```

ResponseBody_closed_by.starred_url

```
 => ResponseBody_closed_by.starred_url=not-exist (2430.0/8.0)
```

ResponseBody_closed_by.subscriptions_url

```
 => ResponseBody_closed_by.subscriptions_url=not-exist (2430.0/8.0)
```

ResponseBody_closed_by.type

```
 => ResponseBody_closed_by.type=not-exist (2430.0/8.0)
```

ResponseBody_closed_by.url

```
 => ResponseBody_closed_by.url=not-exist (2430.0/8.0)
```

ResponseBody_documentation_url

```
(RequestMethod = DELETE) and
(HasAuthorisationToken = false) => ResponseBody_documentation_url=https://
    developer.github.com/v3/issues/#unlock-an-issue (64.0/0.0)

(HasAuthorisationToken = false) and
(RequestMethod = PUT) => ResponseBody_documentation_url=https://developer.github.
    com/v3/issues/#lock-an-issue (128.0/0.0)

(RequestMethod = PATCH) and
(HasValidRequestPayload = false) => ResponseBody_documentation_url=https://
    developer.github.com/v3/issues/#edit-an-issue (229.0/0.0)

(RequestMethod = POST) and
```

```
(HasValidRequestPayload = false) => ResponseBody_documentation_url=https://
    developer.github.com/v3/issues/#create-an-issue (153.0/0.0)

(HasAuthorisationToken = false) => ResponseBody_documentation_url=https://
    developer.github.com/v3/issues/#create-an-issue (122.0/0.0)

(HasSuccessfulCreateOperationOccurredBefore = false) and
(RequestUriPathToken6 = lock) => ResponseBody_documentation_url=https://developer
    .github.com/v3 (232.0/0.0)

(HasSuccessfulCreateOperationOccurredBefore = false) and
(RequestMethod = PATCH) => ResponseBody_documentation_url=https://developer.
    github.com/v3 (153.0/0.0)

(RequestMethod = GET) and
(HasSuccessfulCreateOperationOccurredBefore = false) =>
    ResponseBody_documentation_url=https://developer.github.com/v3 (161.0/10.0)

=> ResponseBody_documentation_url=not-exist (1188.0/0.0)
```

ResponseBody_locked

```
(HasValidRequestPayload = true) and
(HasAuthorisationToken = true) and
(RequestMethod = POST) => ResponseBody_locked=false (354.0/0.0)

(ImmediatelyPreviousStatusCode = 201) and
(HasValidRequestPayload = true) => ResponseBody_locked=false (173.0/0.0)

(RequestMethod = GET) and
(HasSuccessfulCreateOperationOccurredBefore = true) => ResponseBody_locked=false
    (196.0/0.0)

=> ResponseBody_locked=not-exist (1707.0/0.0)
```

ResponseBody_message

```
(RequestMethod = PATCH) and
(HasRequestPayload = false) => ResponseBody_message=Invalid request (41.0/0.0)

(RequestMethod = POST) and
(HasRequestPayload = false) => ResponseBody_message=Invalid request (15.0/0.0)

(HasRequestPayload = true) and
(HasValidRequestPayload = false) and
(HasAuthorisationToken = true) => ResponseBody_message=Problems parsing JSON
    (238.0/0.0)

(HasAuthorisationToken = false) and
(HasValidRequestPayload = false) => ResponseBody_message=Requires authentication
    (280.0/0.0)

(HasSuccessfulCreateOperationOccurredBefore = false) and
(RequestUriPathToken6 = lock) => ResponseBody_message=Not Found (232.0/0.0)

(HasSuccessfulCreateOperationOccurredBefore = false) and
(RequestMethod = PATCH) => ResponseBody_message=Not Found (153.0/0.0)
(HasAuthorisationToken = false) => ResponseBody_message=Not Found (122.0/0.0)

(RequestMethod = GET) and
(HasSuccessfulCreateOperationOccurredBefore = false) => ResponseBody_message=Not
    Found (161.0/10.0)

=> ResponseBody_message=not-exist (1188.0/0.0)
```

ResponseBody_milestone.creator.avatar_url

```
 => ResponseBody_milestone.creator.avatar_url=not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.events_url

```
=> ResponseBody_milestone.creator.events_url=not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.followers_url

```
=> ResponseBody_milestone.creator.followers_url=not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.following_url

```
=> ResponseBody_milestone.creator.following_url=not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.gists_url

```
=> ResponseBody_milestone.creator.gists_url=not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.html_url

```
=> ResponseBody_milestone.creator.html_url=not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.id

```
=> ResponseBody_milestone.creator.id=not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.login

```
=> ResponseBody_milestone.creator.login=not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.organizations_url

```
=> ResponseBody_milestone.creator.organizations_url=not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.received_events_url

```
=> ResponseBody_milestone.creator.received_events_url=not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.repos_url

```
=> ResponseBody_milestone.creator.repos_url=not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.site_admin

```
=> ResponseBody_milestone.creator.site_admin=not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.starred_url

```
=> ResponseBody_milestone.creator.starred_url=not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.subscriptions_url

```
=> ResponseBody_milestone.creator.subscriptions_url=not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.type

```
=> ResponseBody_milestone.creator.type=not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.url

```
=> ResponseBody_milestone.creator.url=not-exist (2430.0/112.0)
```

ResponseBody_milestone.due_on

```
 => ResponseBody_milestone.due_on=not-exist (2430.0/112.0)
```

ResponseBody_milestone.open_issues

```
 => ResponseBody_milestone.open_issues=not-exist (2430.0/112.0)
```

ResponseBody_milestone.state

```
 => ResponseBody_milestone.state=not-exist (2430.0/112.0)
```

ResponseBody_state

```
(ImmediatelyPreviousStatusCode = 201) and
(HasValidRequestPayload = true) => ResponseBody_state=closed (173.0/0.0)

(RequestMethod = GET) and
(HasSuccessfulUpdateOperationOccurredBefore = true) and
(ImmediatelyPreviousStatusCode = 200) and
(ImmediatelyPreviousMethod = PATCH) => ResponseBody_state=closed (22.0/0.0)

(RequestMethod = GET) and
(HasSuccessfulUpdateOperationOccurredBefore = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_state=closed (11.0/3.0)

(RequestMethod = GET) and
(HasSuccessfulUpdateOperationOccurredBefore = true) and
(HasImmediatePreviousTransactionSucceeded = false) and
(HasSuccessfulReadOperationOccurredBefore = false) => ResponseBody_state=closed
    (16.0/7.0)

(RequestMethod = GET) and
(HasSuccessfulUpdateOperationOccurredBefore = true) and
(ImmediatelyPreviousMethod = DELETE) => ResponseBody_state=closed (4.0/1.0)

(RequestMethod = POST) and
(HasValidRequestPayload = true) and
(HasAuthorisationToken = true) => ResponseBody_state=open (354.0/0.0)

(RequestMethod = GET) and
(HasSuccessfulCreateOperationOccurredBefore = true) => ResponseBody_state=open
    (143.0/14.0)

=> ResponseBody_state=not-exist (1707.0/0.0)
```

ResponseBody_user.site_admin

```
(HasValidRequestPayload = true) and
(HasAuthorisationToken = true) and
(RequestMethod = POST) => ResponseBody_user.site_admin=false (354.0/0.0)

(ImmediatelyPreviousStatusCode = 201) and
(HasValidRequestPayload = true) => ResponseBody_user.site_admin=false (173.0/0.0)

(RequestMethod = GET) and
(HasSuccessfulCreateOperationOccurredBefore = true) => ResponseBody_user.
    site_admin=false (196.0/0.0)

=> ResponseBody_user.site_admin=not-exist (1707.0/0.0)
```

ResponseBody_user.type

```
(HasValidRequestPayload = true) and
(HasAuthorisationToken = true) and
(RequestMethod = POST) => ResponseBody_user.type=User (354.0/0.0)

(ImmediatelyPreviousStatusCode = 201) and
```

```
(HasValidRequestPayload = true) => ResponseBody_user.type=User (173.0/0.0)

(RequestMethod = GET) and
(HasSuccessfulCreateOperationOccurredBefore = true) => ResponseBody_user.type=
    User (196.0/0.0)

=> ResponseBody_user.type=not-exist (1707.0/0.0)
```

C.2.2 Rules from RIPPER on Twitter

ResponseStatusCode

```
(RequestUriPathToken3 = update.json) => ResponseStatusCode=200 (867.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false) =>
    ResponseStatusCode=200 (648.0/0.0)

=> ResponseStatusCode=404 (24538.0/0.0)
```

ResponseHeader_status

```
(RequestUriPathToken3 = update.json) => ResponseHeader_status=200 OK (867.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false) =>
    ResponseHeader_status=200 OK (648.0/0.0)

=> ResponseHeader_status=404 Not Found (24538.0/0.0)
```

ResponseHeader_x-rate-limit-limit

```
(RequestMethod = GET) => ResponseHeader_x-rate-limit-limit=900 (5608.0/0.0)

=> ResponseHeader_x-rate-limit-limit=not-exist (20445.0/0.0)
```

ResponseBody_contributors

```
(RequestUriPathToken3 = update.json) => ResponseBody_contributors=null
    (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_contributors=null (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_contributors=null (101.0/0.0)

=> ResponseBody_contributors=not-exist (24538.0/0.0)
```

ResponseBody_coordinates

```
(RequestUriPathToken3 = update.json) => ResponseBody_coordinates=null (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_coordinates=null (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_coordinates=null (101.0/0.0)

=> ResponseBody_coordinates=not-exist (24538.0/0.0)
```

ResponseBody_entities.hashtags

```
(RequestUriPathToken3 = update.json) => ResponseBody_entities.hashtags=[]
    (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_entities.hashtags=[] (547.0/0.0)
```

```
(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_entities.hashtags=[]
    (101.0/0.0)

=> ResponseBody_entities.hashtags=not-exist (24538.0/0.0)
```

ResponseBody_entities.symbols

```
(RequestUriPathToken3 = update.json) => ResponseBody_entities.symbols=[]
    (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_entities.symbols=[] (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_entities.symbols=[] (101.0/0.0)

=> ResponseBody_entities.symbols=not-exist (24538.0/0.0)
```

ResponseBody_entities.urls

```
(RequestUriPathToken3 = update.json) => ResponseBody_entities.urls=[] (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_entities.urls=[] (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_entities.urls=[] (101.0/0.0)

=> ResponseBody_entities.urls=not-exist (24538.0/0.0)
```

ResponseBody_entities.user_mentions

```
(RequestUriPathToken3 = update.json) => ResponseBody_entities.user_mentions=[]
    (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_entities.user_mentions=[] (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_entities.user_mentions=[]
    (101.0/0.0)

=> ResponseBody_entities.user_mentions=not-exist (24538.0/0.0)
```

ResponseBody_errors.code

```
(RequestUriPathToken3 = update.json) => ResponseBody_errors.code=not-exist
    (867.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false) =>
    ResponseBody_errors.code=not-exist (648.0/0.0)

=> ResponseBody_errors.code=144 (24538.0/0.0)
```

ResponseBody_errors.message

```
(RequestUriPathToken3 = update.json) => ResponseBody_errors.message=not-exist
    (867.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false) =>
    ResponseBody_errors.message=not-exist (648.0/0.0)

=> ResponseBody_errors.message=No status found with that ID. (24538.0/0.0)
```

ResponseBody_favorite_count

```
(RequestUriPathToken3 = update.json) => ResponseBody_favorite_count=0 (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_favorite_count=0 (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_favorite_count=0 (101.0/0.0)

=> ResponseBody_favorite_count=not-exist (24538.0/0.0)
```

ResponseBody_favorited

```
(RequestUriPathToken3 = update.json) => ResponseBody_favorited=false (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_favorited=false (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_favorited=false (101.0/0.0)

=> ResponseBody_favorited=not-exist (24538.0/0.0)
```

ResponseBody_geo

```
(RequestUriPathToken3 = update.json) => ResponseBody_geo=null (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_geo=null (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_geo=null (101.0/0.0)

=> ResponseBody_geo=not-exist (24538.0/0.0)
```

ResponseBody_in_reply_to_screen_name

```
(RequestUriPathToken3 = update.json) => ResponseBody_in_reply_to_screen_name=null
     (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_in_reply_to_screen_name=null (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_in_reply_to_screen_name=null
    (101.0/0.0)

=> ResponseBody_in_reply_to_screen_name=not-exist (24538.0/0.0)
```

ResponseBody_in_reply_to_status_id

```
(RequestUriPathToken3 = update.json) => ResponseBody_in_reply_to_status_id=null
    (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_in_reply_to_status_id=null (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_in_reply_to_status_id=null
    (101.0/0.0)

=> ResponseBody_in_reply_to_status_id=not-exist (24538.0/0.0)
```

ResponseBody_in_reply_to_status_id_str

```
(RequestUriPathToken3 = update.json) => ResponseBody_in_reply_to_status_id_str=
    null (867.0/0.0)
```

```
(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_in_reply_to_status_id_str=null (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_in_reply_to_status_id_str=null
    (101.0/0.0)

=> ResponseBody_in_reply_to_status_id_str=not-exist (24538.0/0.0)
```

ResponseBody_in_reply_to_user_id

```
(RequestUriPathToken3 = update.json) => ResponseBody_in_reply_to_user_id=null
    (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_in_reply_to_user_id=null (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_in_reply_to_user_id=null
    (101.0/0.0)

=> ResponseBody_in_reply_to_user_id=not-exist (24538.0/0.0)
```

ResponseBody_in_reply_to_user_id_str

```
(RequestUriPathToken3 = update.json) => ResponseBody_in_reply_to_user_id_str=null
     (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_in_reply_to_user_id_str=null (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_in_reply_to_user_id_str=null
    (101.0/0.0)

=> ResponseBody_in_reply_to_user_id_str=not-exist (24538.0/0.0)
```

ResponseBody_is_quote_status

```
(RequestUriPathToken3 = update.json) => ResponseBody_is_quote_status=false
    (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_is_quote_status=false (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_is_quote_status=false
    (101.0/0.0)

=> ResponseBody_is_quote_status=not-exist (24538.0/0.0)
```

ResponseBody_place

```
(RequestUriPathToken3 = update.json) => ResponseBody_place=null (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_place=null (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_place=null (101.0/0.0)

=> ResponseBody_place=not-exist (24538.0/0.0)
```

ResponseBody_retweet_count

```
(RequestUriPathToken3 = update.json) => ResponseBody_retweet_count=0 (867.0/0.0)
```

```
(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_retweet_count=0 (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_retweet_count=0 (101.0/0.0)

=> ResponseBody_retweet_count=not-exist (24538.0/0.0)
```

ResponseBody_retweeted

```
(RequestUriPathToken3 = update.json) => ResponseBody_retweeted=false (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_retweeted=false (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_retweeted=false (101.0/0.0)

=> ResponseBody_retweeted=not-exist (24538.0/0.0)
```

ResponseBody_truncated

```
(RequestUriPathToken3 = update.json) => ResponseBody_truncated=false (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_truncated=false (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_truncated=false (101.0/0.0)

=> ResponseBody_truncated=not-exist (24538.0/0.0)
```

ResponseBody_user.contributors_enabled

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.contributors_enabled=
    false (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.contributors_enabled=false (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.contributors_enabled=false
     (101.0/0.0)

=> ResponseBody_user.contributors_enabled=not-exist (24538.0/0.0)
```

ResponseBody_user.created_at

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.created_at=Wed Mar 07
    09:41:33 +0000 2012 (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.created_at=Wed Mar 07 09:41:33 +0000 2012 (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.created_at=Wed Mar 07 09
    :41:33 +0000 2012 (101.0/0.0)

=> ResponseBody_user.created_at=not-exist (24538.0/0.0)
```

ResponseBody_user.default_profile

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.default_profile=false
    (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.default_profile=false (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
```

```
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.default_profile=false
    (101.0/0.0)

=> ResponseBody_user.default_profile=not-exist (24538.0/0.0)
```

ResponseBody_user.default_profile_image

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.default_profile_image=
    false (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.default_profile_image=false (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.default_profile_image=
    false (101.0/0.0)

=> ResponseBody_user.default_profile_image=not-exist (24538.0/0.0)
```

ResponseBody_user.favourites_count

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.favourites_count=64
    (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.favourites_count=64 (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.favourites_count=64
    (101.0/0.0)

=> ResponseBody_user.favourites_count=not-exist (24538.0/0.0)
```

ResponseBody_user.follow_request_sent

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.follow_request_sent=
    false (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.follow_request_sent=false (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.follow_request_sent=false
    (101.0/0.0)

=> ResponseBody_user.follow_request_sent=not-exist (24538.0/0.0)
```

ResponseBody_user.followers_count

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.followers_count=185
    (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.followers_count=185 (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.followers_count=185
    (101.0/0.0)

=> ResponseBody_user.followers_count=not-exist (24538.0/0.0)
```

ResponseBody_user.following

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.following=false
    (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.following=false (547.0/0.0)
```

```
(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.following=false
    (101.0/0.0)

=> ResponseBody_user.following=not-exist (24538.0/0.0)
```

ResponseBody_user.friends_count

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.friends_count=249
    (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.friends_count=249 (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.friends_count=249
    (101.0/0.0)

=> ResponseBody_user.friends_count=not-exist (24538.0/0.0)
```

ResponseBody_user.geo_enabled

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.geo_enabled=true
    (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.geo_enabled=true (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.geo_enabled=true
    (101.0/0.0)

=> ResponseBody_user.geo_enabled=not-exist (24538.0/0.0)
```

ResponseBody_user.has_extended_profile

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.has_extended_profile=
    false (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.has_extended_profile=false (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.has_extended_profile=false
     (101.0/0.0)

=> ResponseBody_user.has_extended_profile=not-exist (24538.0/0.0)
```

ResponseBody_user.id

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.id=517417816
    (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.id=517417816 (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.id=517417816 (101.0/0.0)

=> ResponseBody_user.id=not-exist (24538.0/0.0)
```

ResponseBody_user.id_str

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.id_str=517417816
    (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.id_str=517417816 (547.0/0.0)
```

```
(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.id_str=517417816
    (101.0/0.0)

=> ResponseBody_user.id_str=not-exist (24538.0/0.0)
```

ResponseBody_user.is_translation_enabled

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.is_translation_enabled=
    false (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.is_translation_enabled=false (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.is_translation_enabled=
    false (101.0/0.0)

=> ResponseBody_user.is_translation_enabled=not-exist (24538.0/0.0)
```

ResponseBody_user.is_translator

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.is_translator=false
    (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.is_translator=false (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.is_translator=false
    (101.0/0.0)

=> ResponseBody_user.is_translator=not-exist (24538.0/0.0)
```

ResponseBody_user.lang

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.lang=en (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.lang=en (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.lang=en (101.0/0.0)

=> ResponseBody_user.lang=not-exist (24538.0/0.0)
```

ResponseBody_user.listed_count

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.listed_count=3
    (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.listed_count=3 (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.listed_count=3 (101.0/0.0)

=> ResponseBody_user.listed_count=not-exist (24538.0/0.0)
```

ResponseBody_user.location

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.location=Kurunegala
    (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.location=Kurunegala (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
```

```
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.location=Kurunegala
    (101.0/0.0)

=> ResponseBody_user.location=not-exist (24538.0/0.0)
```

ResponseBody_user.name

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.name=Thilini Bhagya
    (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.name=Thilini Bhagya (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.name=Thilini Bhagya
    (101.0/0.0)

=> ResponseBody_user.name=not-exist (24538.0/0.0)
```

ResponseBody_user.notifications

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.notifications=false
    (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.notifications=false (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.notifications=false
    (101.0/0.0)

=> ResponseBody_user.notifications=not-exist (24538.0/0.0)
```

ResponseBody_user.profile_background_color

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.
    profile_background_color=1A1B1F (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.profile_background_color=1A1B1F (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.profile_background_color=1
    A1B1F (101.0/0.0)

=> ResponseBody_user.profile_background_color=not-exist (24538.0/0.0)
```

ResponseBody_user.profile_background_image_url

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.
    profile_background_image_url=http://abs.twimg.com/images/themes/theme9/bg.gif
     (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.profile_background_image_url=http://abs.twimg.com/images/
    themes/theme9/bg.gif (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.
    profile_background_image_url=http://abs.twimg.com/images/themes/theme9/bg.gif
     (101.0/0.0)

=> ResponseBody_user.profile_background_image_url=not-exist (24538.0/0.0)
```

ResponseBody_user.profile_background_image_url_https

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.
    profile_background_image_url_https=https://abs.twimg.com/images/themes/theme9
    /bg.gif (867.0/0.0)
```

```
(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.profile_background_image_url_https=https://abs.twimg.com/
    images/themes/theme9/bg.gif (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.
    profile_background_image_url_https=https://abs.twimg.com/images/themes/theme9
    /bg.gif (101.0/0.0)

=> ResponseBody_user.profile_background_image_url_https=not-exist (24538.0/0.0)
```

ResponseBody_user.profile_background_tile

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.profile_background_tile
    =false (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.profile_background_tile=false (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.profile_background_tile=
    false (101.0/0.0)

=> ResponseBody_user.profile_background_tile=not-exist (24538.0/0.0)
```

ResponseBody_user.profile_banner_url

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.profile_banner_url=
    https://pbs.twimg.com/profile_banners/517417816/1399047954 (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.profile_banner_url=https://pbs.twimg.com/profile_banners
    /517417816/1399047954 (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.profile_banner_url=https:
    //pbs.twimg.com/profile_banners/517417816/1399047954 (101.0/0.0)

=> ResponseBody_user.profile_banner_url=not-exist (24538.0/0.0)
```

ResponseBody_user.profile_image_url

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.profile_image_url=http:
    //pbs.twimg.com/profile_images/950100192048562176/LKr7Ay21_normal.jpg
    (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.profile_image_url=http://pbs.twimg.com/profile_images
    /950100192048562176/LKr7Ay21_normal.jpg (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.profile_image_url=http://
    pbs.twimg.com/profile_images/950100192048562176/LKr7Ay21_normal.jpg
    (101.0/0.0)

=> ResponseBody_user.profile_image_url=not-exist (24538.0/0.0)
```

ResponseBody_user.profile_image_url_https

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.profile_image_url_https
    =https://pbs.twimg.com/profile_images/950100192048562176/LKr7Ay21_normal.jpg
    (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.profile_image_url_https=https://pbs.twimg.com/
    profile_images/950100192048562176/LKr7Ay21_normal.jpg (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
```

```
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.profile_image_url_https=
    https://pbs.twimg.com/profile_images/950100192048562176/LKr7Ay21_normal.jpg
    (101.0/0.0)

=> ResponseBody_user.profile_image_url_https=not-exist (24538.0/0.0)
```

ResponseBody_user.profile_link_color

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.profile_link_color=3
    E4547 (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.profile_link_color=3E4547 (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.profile_link_color=3E4547
    (101.0/0.0)

=> ResponseBody_user.profile_link_color=not-exist (24538.0/0.0)
```

ResponseBody_user.profile_sidebar_border_color

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.
    profile_sidebar_border_color=FFFFFF (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.profile_sidebar_border_color=FFFFFF (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.
    profile_sidebar_border_color=FFFFFF (101.0/0.0)

=> ResponseBody_user.profile_sidebar_border_color=not-exist (24538.0/0.0)
```

ResponseBody_user.profile_sidebar_fill_color

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.
    profile_sidebar_fill_color=252429 (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.profile_sidebar_fill_color=252429 (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.profile_sidebar_fill_color
    =252429 (101.0/0.0)

=> ResponseBody_user.profile_sidebar_fill_color=not-exist (24538.0/0.0)
```

ResponseBody_user.profile_text_color

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.profile_text_color
    =666666 (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.profile_text_color=666666 (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.profile_text_color=666666
    (101.0/0.0)

=> ResponseBody_user.profile_text_color=not-exist (24538.0/0.0)
```

ResponseBody_user.profile_use_background_image

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.
    profile_use_background_image=true (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.profile_use_background_image=true (547.0/0.0)
```

```
(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.
    profile_use_background_image=true (101.0/0.0)

=> ResponseBody_user.profile_use_background_image=not-exist (24538.0/0.0)
```

ResponseBody_user.protected

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.protected=false
    (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.protected=false (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.protected=false
    (101.0/0.0)

=> ResponseBody_user.protected=not-exist (24538.0/0.0)
```

ResponseBody_user.screen_name

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.screen_name=bhagyasl
    (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.screen_name=bhagyasl (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.screen_name=bhagyasl
    (101.0/0.0)

=> ResponseBody_user.screen_name=not-exist (24538.0/0.0)
```

ResponseBody_user.time_zone

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.time_zone=null
    (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.time_zone=null (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.time_zone=null (101.0/0.0)

=> ResponseBody_user.time_zone=not-exist (24538.0/0.0)
```

ResponseBody_user.translator_type

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.translator_type=none
    (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.translator_type=none (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.translator_type=none
    (101.0/0.0)

=> ResponseBody_user.translator_type=not-exist (24538.0/0.0)
```

ResponseBody_user.url

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.url=http://t.co/
    sQduiwqJiy (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.url=http://t.co/sQduiwqJiy (547.0/0.0)
```

```
(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.url=http://t.co/sQduiwqJiy
     (101.0/0.0)

=> ResponseBody_user.url=not-exist (24538.0/0.0)
```

ResponseBody_user.utc_offset

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.utc_offset=null
    (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.utc_offset=null (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.utc_offset=null
    (101.0/0.0)

=> ResponseBody_user.utc_offset=not-exist (24538.0/0.0)
```

ResponseBody_user.verified

```
(RequestUriPathToken3 = update.json) => ResponseBody_user.verified=false
    (867.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_user.verified=false (547.0/0.0)

(HasImmediatePreviousTransactionSucceeded = true) and
(ImmediatelyPreviousMethod = GET) => ResponseBody_user.verified=false (101.0/0.0)

=> ResponseBody_user.verified=not-exist (24538.0/0.0)
```

C.2.3 Rules from RIPPER on Google Tasks

ResponseStatusCode

```
(RequestMethod = DELETE) and
(HasSuccessfulDeleteOperationOccurredBefore = false) => ResponseStatusCode=204
    (606.0/0.0)

(HasSuccessfulDeleteOperationOccurredBefore = true) => ResponseStatusCode=404
    (1482.0/4.0)

=> ResponseStatusCode=200 (2614.0/7.0)
```

ResponseHeader_Accept-Ranges

```
(ImmediatelyPreviousResponseStatusCode = 200) and
(HasRequestPayload = false) => ResponseHeader_Accept-Ranges=not-exist
    (1606.0/3.0)

(ImmediatelyPreviousResponseStatusCode = 503) and
(HasSuccessfulDeleteOperationOccurredBefore = false) => ResponseHeader_Accept-
    Ranges=not-exist (6.0/1.0)

=> ResponseHeader_Accept-Ranges=none (3090.0/0.0)
```

ResponseHeader_Cache-Control

```
(RequestMethod = GET) and
(HasSuccessfulDeleteOperationOccurredBefore = false) => ResponseHeader_Cache-
    Control=private, max-age=0, must-revalidate, no-transform (1005.0/3.0)

(HasSuccessfulDeleteOperationOccurredBefore = true) => ResponseHeader_Cache-
    Control=private, max-age=0 (1482.0/0.0)
```

```
=> ResponseHeader_Cache-Control=no-cache, no-store, max-age=0, must-revalidate
    (2215.0/4.0)
```

ResponseHeader_Content-Type

```
(RequestMethod = DELETE) and
(HasSuccessfulDeleteOperationOccurredBefore = false) => ResponseHeader_Content-
    Type=not-exist (606.0/0.0)

=> ResponseHeader_Content-Type=application/json; charset=UTF-8 (4096.0/0.0)
```

ResponseHeader_Transfer-Encoding

```
(RequestMethod = POST) => ResponseHeader_Pragma=no-cache (1124.0/0.0)

(HasSuccessfulDeleteOperationOccurredBefore = false) and
(RequestMethod = DELETE) => ResponseHeader_Pragma=no-cache (606.0/0.0)

(RequestMethod = PATCH) and
(HasSuccessfulDeleteOperationOccurredBefore = false) => ResponseHeader_Pragma=no-
    cache (485.0/4.0)

=> ResponseHeader_Pragma=not-exist (2487.0/0.0)
```

ResponseHeader_Vary

```
(ImmediatelyPreviousResponseStatusCode = 200) and
(HasRequestPayload = false) => ResponseHeader_Vary=X-Origin (1606.0/3.0)

(ImmediatelyPreviousResponseStatusCode = 503) and
(HasRequestPayload = false) and (HasSuccessfulDeleteOperationOccurredBefore =
    false) => ResponseHeader_Vary=X-Origin (5.0/0.0)

=> ResponseHeader_Vary=Origin,Accept-Encoding (3091.0/0.0)
```

ResponseHeader_X-Content-Type-Options

```
(RequestMethod = DELETE) and
(HasSuccessfulDeleteOperationOccurredBefore = false) => ResponseHeader_X-Content-
    Type-Options=not-exist (606.0/0.0)

=> ResponseHeader_X-Content-Type-Options=nosniff (4096.0/0.0)
```

ResponseHeader_X-Frame-Options

```
(RequestMethod = DELETE) and
(HasSuccessfulDeleteOperationOccurredBefore = false) => ResponseHeader_X-Frame-
    Options=not-exist (606.0/0.0)

=> ResponseHeader_X-Frame-Options=SAMEORIGIN (4096.0/0.0)
```

ResponseHeader_X-XSS-Protection

```
(RequestMethod = DELETE) and
(HasSuccessfulDeleteOperationOccurredBefore = false) => ResponseHeader_X-XSS-
    Protection=not-exist (606.0/0.0)

=> ResponseHeader_X-XSS-Protection=1; mode=block (4096.0/0.0)
```

ResponseBody_error.code

```
(HasSuccessfulDeleteOperationOccurredBefore = true) => ResponseBody_error.code
    =404 (1482.0/4.0)

=> ResponseBody_error.code=not-exist (3220.0/7.0)
```

ResponseBody_error.errors.domain

```
(HasSuccessfulDeleteOperationOccurredBefore = true) => ResponseBody_error.errors.
    domain=global (1482.0/0.0)

=> ResponseBody_error.errors.domain=not-exist (3220.0/7.0)
```

ResponseBody_error.errors.message

```
(HasSuccessfulDeleteOperationOccurredBefore = true) => ResponseBody_error.errors.
    message=NotFound (1482.0/4.0)

=> ResponseBody_error.errors.message=not-exist (3220.0/7.0)
```

ResponseBody_error.errors.reason

```
(HasSuccessfulDeleteOperationOccurredBefore = true) => ResponseBody_error.errors.
    reason=notFound (1482.0/4.0)

=> ResponseBody_error.errors.reason=not-exist (3220.0/7.0)
```

ResponseBody_error.message

```
(HasSuccessfulDeleteOperationOccurredBefore = true) => ResponseBody_error.message
    =NotFound (1482.0/4.0)

=> ResponseBody_error.message=not-exist (3220.0/7.0)
```

ResponseBody_kind

```
(HasSuccessfulDeleteOperationOccurredBefore = true) => ResponseBody_kind=not-
    exist (1482.0/0.0)

(RequestMethod = DELETE) => ResponseBody_kind=not-exist (606.0/0.0)

=> ResponseBody_kind=tasks#taskList (2614.0/7.0)
```

C.2.4 Rules from RIPPER on Slack

ResponseHeader_x-slack-router

```
 => ResponseHeader_x-slack-router=p (5243.0/1975.0)
```

ResponseBody_channel

```
(HasSuccessfulDeleteOperationOccurredBefore = false) and
(RequestUriPathToken2 = chat.postMessage) => ResponseBody_channel=CCGRWTRKQ
    (1152.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_channel=CCGRWTRKQ (726.0/0.0)

(HasSuccessfulDeleteOperationOccurredBefore = false) and
(HasImmediatePreviousTransactionSucceeded = true) => ResponseBody_channel=
    CCGRWTRKQ (235.0/0.0)

=> ResponseBody_channel=not-exist (3130.0/1.0)
```

ResponseBody_error

```
(HasSuccessfulDeleteOperationOccurredBefore = false) and
(RequestUriPathToken2 = chat.postMessage) => ResponseBody_error=not-exist
    (1152.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) =>
    ResponseBody_error=not-exist (726.0/0.0)
```

```
(HasSuccessfulDeleteOperationOccurredBefore = false) and
(HasImmediatePreviousTransactionSucceeded = true) => ResponseBody_error=not-exist
     (235.0/0.0)

=> ResponseBody_error=message_not_found (3130.0/1.0)
```

ResponseBody_message.bot_id

```
(RequestUriPathToken2 = chat.postMessage) => ResponseBody_message.bot_id=
    BCEPNCQDN (1152.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) and
(RequestUriPathToken2 = chat.update) => ResponseBody_message.bot_id=BCEPNCQDN
    (224.0/0.0)

(HasSuccessfulDeleteOperationOccurredBefore = false) and
(RequestUriPathToken2 = chat.update) and
(HasImmediatePreviousTransactionSucceeded = true) => ResponseBody_message.bot_id=
    BCEPNCQDN (11.0/0.0)

=> ResponseBody_message.bot_id=not-exist (3856.0/1.0)
```

ResponseBody_message.edited.user

```
(HasURLInImmediatelyPreviousTransactionContainsATokenToUpdate = true) and
(HasImmediatePreviousTransactionSucceeded = true) and
(RequestUriPathToken2 = chat.update) => ResponseBody_message.edited.user=
    UC8J6APLN (11.0/0.0)

=> ResponseBody_message.edited.user=not-exist (5232.0/0.0)
```

ResponseBody_message.type

```
(RequestUriPathToken2 = chat.postMessage) => ResponseBody_message.type=message
    (1152.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) and
(RequestUriPathToken2 = chat.update) => ResponseBody_message.type=message
    (224.0/0.0)

(HasSuccessfulDeleteOperationOccurredBefore = false) and
(RequestUriPathToken2 = chat.update) and
(HasImmediatePreviousTransactionSucceeded = true) => ResponseBody_message.type=
    message (11.0/0.0)

=> ResponseBody_message.type=not-exist (3856.0/1.0)
```

ResponseBody_message.user

```
(RequestUriPathToken2 = chat.postMessage) => ResponseBody_message.user=UC8J6APLN
    (1152.0/0.0)

(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) and
(RequestUriPathToken2 = chat.update) => ResponseBody_message.user=UC8J6APLN
    (224.0/0.0)

(HasSuccessfulDeleteOperationOccurredBefore = false) and
(RequestUriPathToken2 = chat.update) and
(HasImmediatePreviousTransactionSucceeded = true) => ResponseBody_message.user=
    UC8J6APLN (11.0/0.0)

=> ResponseBody_message.user=not-exist (3856.0/1.0)
```

ResponseBody_ok

```
(RequestUriPathToken2 = chat.postMessage) => ResponseBody_message.user=UC8J6APLN
    (1152.0/0.0)
```

```
(HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = true) and
(RequestUriPathToken2 = chat.update) => ResponseBody_message.user=UC8J6APLN
    (224.0/0.0)

(HasSuccessfulDeleteOperationOccurredBefore = false) and
(RequestUriPathToken2 = chat.update) and
(HasImmediatePreviousTransactionSucceeded = true) => ResponseBody_message.user=
    UC8J6APLN (11.0/0.0)

=> ResponseBody_message.user=not-exist (3856.0/1.0)
```

C.3 Sample PART Rulesets

C.3.1 Rules from PART on GHTraffic

ResponseStatusCode

```
HasSuccessfulCreateOperationOccurredBefore = false AND
HasValidRequestPayload = false AND
HasRequestPayload = false AND
RequestMethod = HEAD: 404 (171.0)

HasSuccessfulCreateOperationOccurredBefore = false AND
HasValidRequestPayload = true AND
RequestMethod = POST AND
HasAuthorisationToken = true: 201 (354.0)

HasSuccessfulCreateOperationOccurredBefore = false AND
HasRequestPayload = false AND
RequestMethod = GET: 404 (161.0/10.0)

HasSuccessfulCreateOperationOccurredBefore = false AND
HasValidRequestPayload = true: 404 (275.0)

HasAuthorisationToken = false: 401 (280.0)

RequestUriPathToken6 = lock AND
HasSuccessfulCreateOperationOccurredBefore = false: 404 (232.0)

RequestUriPathToken6 = lock: 204 (191.0)

HasSuccessfulCreateOperationOccurredBefore = false AND
HasRequestPayload = true: 400 (138.0)

HasRequestPayload = false AND
HasSuccessfulCreateOperationOccurredBefore = true AND
RequestMethod = GET: 200 (196.0)

HasValidRequestPayload = true: 200 (173.0)

HasRequestPayload = false AND
RequestMethod = HEAD: 200 (103.0)

HasRequestPayload = true: 400 (100.0)

: 422 (56.0)
```

ResponseHeader_Cache-Control

```
RequestUriPathToken6 = lock: not-exist (615.0)

HasAuthorisationToken = true AND
HasValidRequestPayload = true AND
RequestMethod = POST: private, max-age=60 (354.0)

HasSuccessfulCreateOperationOccurredBefore = false: not-exist (760.0)

HasAuthorisationToken = true AND
RequestMethod = GET: private, max-age=60 (196.0)

HasValidRequestPayload = false AND
RequestMethod = PATCH: not-exist (229.0)

: private, max-age=60 (276.0)
```

ResponseHeader_Vary

```
RequestUriPathToken6 = lock: not-exist (615.0)

HasAuthorisationToken = true AND
```

```
HasValidRequestPayload = true AND
RequestMethod = POST: Accept, Authorization, Cookie (354.0)

HasSuccessfulCreateOperationOccurredBefore = false: not-exist (760.0)

HasAuthorisationToken = true AND
RequestMethod = GET: Accept, Authorization, Cookie (196.0)

HasValidRequestPayload = false AND
RequestMethod = PATCH: not-exist (229.0)

: Accept, Authorization, Cookie (276.0)
```

ResponseHeader_X-Accepted-OAuth-Scopes

```
HasAuthorisationToken = false: not-exist (402.0)

HasSuccessfulCreateOperationOccurredBefore = true AND
RequestMethod = GET: public_repo, repo (196.0)

HasValidRequestPayload = true AND
RequestMethod = POST: public_repo, repo (354.0)

HasSuccessfulCreateOperationOccurredBefore = false: repo (870.0)

RequestUriPathToken6 = lock: public_repo, repo (191.0)

HasValidRequestPayload = true: public_repo, repo (173.0)

RequestMethod = PATCH: repo (141.0)

: public_repo, repo (103.0)
```

ResponseHeader_X-OAuth-Scopes

```
HasAuthorisationToken = true: public_repo (2028.0)

: not-exist (402.0)
```

ResponseBody_assignee.site_admin

```
: not-exist (2430.0/116.0)
```

ResponseBody_assignee.type

```
ResponseBody\_assignee.type
```

ResponseBody_assignee.site_admin

```
: not-exist (2430.0/95.0)
```

ResponseBody_assignees.type

```
: not-exist (2430.0/95.0)
```

ResponseBody_closed_by.avatar_url

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.events_url

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.followers_url

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.following_url

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.gists_url

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.html_url

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.id

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.login

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.organizations_url

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.received_events_url

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.repos_url

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.site_admin

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.starred_url

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.subscriptions_url

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.type

```
: not-exist (2430.0/8.0)
```

ResponseBody_closed_by.url

```
: not-exist (2430.0/8.0)
```

ResponseBody_documentation_url

```
HasAuthorisationToken = true AND
HasSuccessfulCreateOperationOccurredBefore = true AND
RequestMethod = GET: not-exist (196.0)

HasAuthorisationToken = true AND
HasSuccessfulCreateOperationOccurredBefore = true AND
RequestUriPathToken6 = lock: not-exist (191.0)
```

```
HasAuthorisationToken = true AND
RequestUriPathToken6 = not-exist AND
RequestMethod = POST AND
HasValidRequestPayload = true: not-exist (354.0)

HasSuccessfulCreateOperationOccurredBefore = false AND
RequestMethod = POST: https://developer.github.com/v3/issues/#create-an-issue
    (275.0)

HasAuthorisationToken = true AND
HasSuccessfulCreateOperationOccurredBefore = false AND
RequestMethod = HEAD: not-exist (171.0)

HasSuccessfulCreateOperationOccurredBefore = false: https://developer.github.com/
    v3 (546.0/10.0)

RequestUriPathToken6 = lock AND
RequestMethod = PUT: https://developer.github.com/v3/issues/#lock-an-issue
    (128.0)

RequestUriPathToken6 = not-exist AND
HasValidRequestPayload = false AND
RequestMethod = PATCH: https://developer.github.com/v3/issues/#edit-an-issue
    (229.0)

RequestUriPathToken6 = not-exist: not-exist (276.0)

: https://developer.github.com/v3/issues/#unlock-an-issue (64.0)
```

ResponseBody_locked

```
HasValidRequestPayload = false AND
RequestMethod = PUT: not-exist (394.0)

HasValidRequestPayload = false AND
HasSuccessfulCreateOperationOccurredBefore = false: not-exist (594.0)

HasAuthorisationToken = true AND
RequestMethod = POST: false (354.0)

RequestMethod = PATCH AND
HasValidRequestPayload = false: not-exist (229.0)

HasSuccessfulCreateOperationOccurredBefore = false: not-exist (275.0)

RequestMethod = GET: false (196.0)

HasRequestPayload = false: not-exist (215.0)

: false (173.0)
```

ResponseBody_message

```
HasAuthorisationToken = false AND
HasValidRequestPayload = false: Requires authentication (280.0)

HasSuccessfulCreateOperationOccurredBefore = true AND
HasRequestPayload = false AND
RequestMethod = GET: not-exist (196.0)

HasSuccessfulCreateOperationOccurredBefore = true AND
HasRequestPayload = false AND
RequestMethod = PUT: not-exist (143.0)

HasSuccessfulCreateOperationOccurredBefore = true AND
HasRequestPayload = false AND
RequestMethod = HEAD: not-exist (103.0)

HasSuccessfulCreateOperationOccurredBefore = true AND
HasValidRequestPayload = true: not-exist (173.0)

HasSuccessfulCreateOperationOccurredBefore = true AND
```

```
RequestMethod = PATCH AND
HasRequestPayload = true: Problems parsing JSON (100.0)

RequestMethod = HEAD: not-exist (171.0)

HasValidRequestPayload = true AND
RequestMethod = POST AND
HasAuthorisationToken = true: not-exist (354.0)

HasSuccessfulCreateOperationOccurredBefore = false AND
HasRequestPayload = false AND
RequestMethod = GET: Not Found (161.0/10.0)

HasSuccessfulCreateOperationOccurredBefore = false AND
HasValidRequestPayload = true: Not Found (275.0)

RequestUriPathToken6 = lock AND
HasSuccessfulCreateOperationOccurredBefore = false: Not Found (232.0)

RequestUriPathToken6 = not-exist AND
HasRequestPayload = true: Problems parsing JSON (138.0)

RequestUriPathToken6 = not-exist: Invalid request (56.0)

: not-exist (48.0)
```

ResponseBody_milestone.creator.avatar_url

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.events_url

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.followers_url

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.following_url

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.gists_url

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.html_url

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.id

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.login

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.organizations_url

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.received_events_url

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.repos_url

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.site_admin

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.starred_url

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.subscriptions_url

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.type

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.creator.url

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.due_on

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.open_issues

```
: not-exist (2430.0/112.0)
```

ResponseBody_milestone.state

```
: not-exist (2430.0/112.0)
```

ResponseBody_state

```
RequestBody_state = not-exist AND
HasValidRequestPayload = false AND
RequestMethod = PUT: not-exist (394.0)

RequestBody_state = closed AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (173.0)

RequestBody_state = closed AND
HasValidRequestPayload = true: closed (173.0)

HasValidRequestPayload = false AND
HasSuccessfulCreateOperationOccurredBefore = false: not-exist (594.0)

HasAuthorisationToken = true AND
RequestMethod = POST: open (354.0)

RequestMethod = PATCH: not-exist (209.0)

RequestMethod = POST: not-exist (122.0)

RequestMethod = DELETE: not-exist (112.0)

RequestMethod = HEAD: not-exist (103.0)

HasSuccessfulUpdateOperationOccurredBefore = false: open (84.0/2.0)

ImmediatelyPreviousStatusCode = 200 AND
```

```
HasSuccessfulReadOperationOccurredBefore = false: closed (22.0)

ImmediatelyPreviousMethod = PUT: open (51.0/11.0)

ImmediatelyPreviousMethod = GET: closed (11.0/3.0)

HasSuccessfulReadOperationOccurredBefore = true: open (18.0/6.0)

: closed (10.0/3.0)
```

ResponseBody_user.site_admin

```
HasValidRequestPayload = false AND
RequestMethod = PUT: not-exist (394.0)

HasValidRequestPayload = false AND
HasSuccessfulCreateOperationOccurredBefore = false: not-exist (594.0)

HasAuthorisationToken = true AND
RequestMethod = POST: false (354.0)

RequestMethod = PATCH AND
HasValidRequestPayload = false: not-exist (229.0)

HasSuccessfulCreateOperationOccurredBefore = false: not-exist (275.0)

RequestMethod = GET: false (196.0)

HasRequestPayload = false: not-exist (215.0)

: false (173.0)
```

ResponseBody_user.type

```
HasValidRequestPayload = false AND
RequestMethod = PUT: not-exist (394.0)

HasValidRequestPayload = false AND
HasSuccessfulCreateOperationOccurredBefore = false: not-exist (594.0)

HasAuthorisationToken = true AND
RequestMethod = POST: User (354.0)

RequestMethod = PATCH AND
HasValidRequestPayload = false: not-exist (229.0)

HasSuccessfulCreateOperationOccurredBefore = false: not-exist (275.0)

RequestMethod = GET: User (196.0)

HasRequestPayload = false: not-exist (215.0)

: User (173.0)
```

C.3.2 Rules from PART on Twitter

ResponseStatusCode

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: 404 (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: 200
    (1515.0)
```

ResponseHeader_status

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
```

```
HasImmediatePreviousTransactionSucceeded = false: 404 Not Found (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: 200 OK
    (1515.0)
```

ResponseHeader_x-rate-limit-limit

```
RequestMethod = POST: not-exist (20445.0)

: 900 (5608.0)
```

ResponseBody_contributors

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: null
    (1515.0)

: not-exist (513.0)
```

ResponseBody_coordinates

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: null
    (1515.0)

: not-exist (513.0)
```

ResponseBody_entities.hashtags

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: [] (1515.0)

: not-exist (513.0)
```

ResponseBody_entities.symbols

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: [] (1515.0)

: not-exist (513.0)
```

ResponseBody_entities.urls

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: [] (1515.0)

: not-exist (513.0)
```

ResponseBody_entities.user_mentions

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: [] (1515.0)

: not-exist (513.0)
```

ResponseBody_errors.code

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: 144 (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: not-exist
    (1515.0)

: 144 (513.0)
```

ResponseBody_errors.message

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: No status found with that ID.
    (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: not-exist
    (1515.0)

: No status found with that ID. (513.0)
```

ResponseBody_favorite_count

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: 0 (1515.0)

: not-exist (513.0)
```

ResponseBody_favorited

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: false
    (1515.0)

: not-exist (513.0)
```

ResponseBody_geo

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: null
    (1515.0)

: not-exist (513.0)
```

ResponseBody_in_reply_to_screen_name

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: null
    (1515.0)

: not-exist (513.0)
```

ResponseBody_in_reply_to_status_id

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: null
    (1515.0)

: not-exist (513.0)
```

ResponseBody_in_reply_to_status_id_str

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: null
    (1515.0)

: not-exist (513.0)
```

ResponseBody_in_reply_to_user_id

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: null
    (1515.0)

: not-exist (513.0)
```

ResponseBody_in_reply_to_user_id_str

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: null
    (1515.0)

: not-exist (513.0)
```

ResponseBody_is_quote_status

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: false
    (1515.0)

: not-exist (513.0)
```

ResponseBody_place

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: null
    (1515.0)

: not-exist (513.0)
```

ResponseBody_retweet_count

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: 0 (1515.0)

: not-exist (513.0)
```

ResponseBody_retweeted

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: false
    (1515.0)

: not-exist (513.0)
```

ResponseBody_truncated

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: false
    (1515.0)

: not-exist (513.0)
```

ResponseBody_user.contributors_enabled

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: false
    (1515.0)

: not-exist (513.0)
```

ResponseBody_user.created_at

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: Wed Mar 07
    09:41:33 +0000 2012 (1515.0)

: not-exist (513.0)
```

ResponseBody_user.default_profile

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: false
    (1515.0)

: not-exist (513.0)
```

ResponseBody_user.default_profile_image

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: false
    (1515.0)

: not-exist (513.0)
```

ResponseBody_user.favourites_count

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: 64 (1515.0)

: not-exist (513.0)
```

ResponseBody_user.follow_request_sent

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: false
    (1515.0)

: not-exist (513.0)
```

ResponseBody_user.followers_count

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: 185
    (1515.0)

: not-exist (513.0)
```

ResponseBody_user.following

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: false
    (1515.0)

: not-exist (513.0)
```

ResponseBody_user.friends_count

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: 249
    (1515.0)

: not-exist (513.0)
```

ResponseBody_user.geo_enabled

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: true
    (1515.0)

: not-exist (513.0)
```

ResponseBody_user.has_extended_profile

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: false
    (1515.0)

: not-exist (513.0)
```

ResponseBody_user.id

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: 517417816
    (1515.0)

: not-exist (513.0)
```

ResponseBody_user.id_str

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: 517417816
    (1515.0)

: not-exist (513.0)
```

ResponseBody_user.is_translation_enabled

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: false
    (1515.0)

: not-exist (513.0)
```

ResponseBody_user.is_translator

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: false
    (1515.0)

: not-exist (513.0)
```

ResponseBody_user.lang

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: en (1515.0)

: not-exist (513.0)
```

ResponseBody_user.listed_count

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: 3 (1515.0)

: not-exist (513.0)
```

ResponseBody_user.location

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: Kurunegala
    (1515.0)

: not-exist (513.0)
```

ResponseBody_user.name

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: Thilini
    Bhagya (1515.0)

: not-exist (513.0)
```

ResponseBody_user.notifications

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: false
    (1515.0)

: not-exist (513.0)
```

ResponseBody_user.profile_background_color

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: 1A1B1F
    (1515.0)

: not-exist (513.0)
```

ResponseBody_user.profile_background_image_url

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: http://abs.
    twimg.com/images/themes/theme9/bg.gif (1515.0)

: not-exist (513.0)
```

ResponseBody_user.profile_background_image_url_https

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: https://abs
    .twimg.com/images/themes/theme9/bg.gif (1515.0)

: not-exist (513.0)
```

ResponseBody_user.profile_background_tile

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: false
    (1515.0)

: not-exist (513.0)
```

ResponseBody_user.profile_banner_url

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: https://pbs
    .twimg.com/profile_banners/517417816/1399047954 (1515.0)

: not-exist (513.0)
```

ResponseBody_user.profile_image_url

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: http://pbs.
    twimg.com/profile_images/950100192048562176/LKr7Ay21_normal.jpg (1515.0)

: not-exist (513.0)
```

ResponseBody_user.profile_image_url_https

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: https://pbs
    .twimg.com/profile_images/950100192048562176/LKr7Ay21_normal.jpg (1515.0)

: not-exist (513.0)
```

ResponseBody_user.profile_link_color

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: 3E4547
    (1515.0)

: not-exist (513.0)
```

ResponseBody_user.profile_sidebar_border_color

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: FFFFFF
    (1515.0)

: not-exist (513.0)
```

ResponseBody_user.profile_sidebar_fill_color

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: 252429
    (1515.0)

: not-exist (513.0)
```

ResponseBody_user.profile_text_color

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: 666666
    (1515.0)

: not-exist (513.0)
```

ResponseBody_user.profile_use_background_image

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: true
    (1515.0)

: not-exist (513.0)
```

ResponseBody_user.protected

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: false
    (1515.0)

: not-exist (513.0)
```

ResponseBody_user.screen_name

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: bhagyasl
    (1515.0)

: not-exist (513.0)
```

ResponseBody_user.time_zone

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: null
    (1515.0)

: not-exist (513.0)
```

ResponseBody_user.translator_type

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: none
    (1515.0)

: not-exist (513.0)
```

ResponseBody_user.url

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: http://t.co
    /sQduiwqJiy (1515.0)

: not-exist (513.0)
```

ResponseBody_user.utc_offset

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: null
    (1515.0)

: not-exist (513.0)
```

ResponseBody_user.verified

```
RequestHeader_Content-Type = not-exist AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasImmediatePreviousTransactionSucceeded = false: not-exist (24025.0)

HasURLInImmediatelyPreviousTransactionContainsATokenToDelete = false: false
    (1515.0)

: not-exist (513.0)
```

C.3.3 Rules from PART on Google Tasks

ResponseStatusCode

```
HasSuccessfulDeleteOperationOccurredBefore = true: 404 (1482.0/4.0)

RequestMethod = POST: 200 (1124.0)

RequestMethod = GET: 200 (1005.0/3.0)

RequestMethod = DELETE: 204 (606.0)

: 200 (485.0/4.0)
```

ResponseHeader_Accept-Ranges

```
HasRequestPayload = true: none (2206.0)

HasSuccessfulDeleteOperationOccurredBefore = false: not-exist (1611.0/3.0)

: none (885.0)
```

ResponseHeader_Cache-Control

```
HasSuccessfulDeleteOperationOccurredBefore = true: private, max-age=0 (1482.0)

RequestMethod = POST: no-cache, no-store, max-age=0, must-revalidate (1124.0)

RequestMethod = GET: private, max-age=0, must-revalidate, no-transform
    (1005.0/3.0)

: no-cache, no-store, max-age=0, must-revalidate (1091.0/4.0)
```

ResponseHeader_Content-Type

```
RequestMethod = GET: application/json; charset=UTF-8 (1367.0)

HasRequestPayload = true: application/json; charset=UTF-8 (2206.0)

HasSuccessfulDeleteOperationOccurredBefore = false: not-exist (606.0)

: application/json; charset=UTF-8 (523.0)
```

ResponseHeader_Pragma

```
HasSuccessfulDeleteOperationOccurredBefore = true: not-exist (1482.0)

RequestMethod = POST: no-cache (1124.0)

RequestMethod = GET: not-exist (1005.0)

: no-cache (1091.0/4.0)
```

ResponseHeader_Transfer-Encoding

```
HasRequestPayload = true: chunked (2206.0)

HasSuccessfulDeleteOperationOccurredBefore = false: not-exist (1611.0/3.0)

: chunked (885.0)
```

ResponseHeader_Vary

```
HasRequestPayload = true: Origin,Accept-Encoding (2206.0)

HasSuccessfulDeleteOperationOccurredBefore = false: X-Origin (1611.0/3.0)

: Origin,Accept-Encoding (885.0)
```

ResponseHeader_X-Content-Type-Options

```
RequestMethod = GET: nosniff (1367.0)

HasRequestPayload = true: nosniff (2206.0)

HasSuccessfulDeleteOperationOccurredBefore = false: not-exist (606.0)

: nosniff (523.0)
```

ResponseHeader_X-Frame-Options

```
RequestMethod = GET: SAMEORIGIN (1367.0)

HasRequestPayload = true: SAMEORIGIN (2206.0)

HasSuccessfulDeleteOperationOccurredBefore = false: not-exist (606.0)

: SAMEORIGIN (523.0)
```

ResponseHeader_X-XSS-Protection

```
RequestMethod = GET: 1; mode=block (1367.0)

HasRequestPayload = true: 1; mode=block (2206.0)

HasSuccessfulDeleteOperationOccurredBefore = false: not-exist (606.0)

: 1; mode=block (523.0)
```

ResponseBody_error.code

```
HasSuccessfulDeleteOperationOccurredBefore = false: not-exist (3220.0/7.0)

: 404 (1482.0/4.0)
```

ResponseBody_error.errors.domain

```
HasSuccessfulDeleteOperationOccurredBefore = false: not-exist (3220.0/7.0)

: global (1482.0)
```

ResponseBody_error.errors.message

```
HasSuccessfulDeleteOperationOccurredBefore = false: not-exist (3220.0/7.0)

: NotFound (1482.0/4.0)
```

ResponseBody_error.errors.reason

```
HasSuccessfulDeleteOperationOccurredBefore = false: not-exist (3220.0/7.0)

: notFound (1482.0/4.0)
```

ResponseBody_error.message

```
HasSuccessfulDeleteOperationOccurredBefore = false: not-exist (3220.0/7.0)

: NotFound (1482.0/4.0)
```

ResponseBody_kind

```
HasSuccessfulDeleteOperationOccurredBefore = true: not-exist (1482.0)

RequestMethod = POST: tasks#taskList (1124.0)

RequestMethod = GET: tasks#taskList (1005.0/3.0)

RequestMethod = DELETE: not-exist (606.0)

: tasks#taskList (485.0/4.0)
```

C.3.4 Rules from PART on Slack

ResponseHeader_x-slack-router

```
: p (5243.0/1975.0)
```

ResponseBody_channel

```
HasSuccessfulDeleteOperationOccurredBefore = true: not-exist (2648.0)

HasImmediatePreviousTransactionSucceeded = false AND
RequestUriPathToken2 = chat.postMessage: CCGRWTRKQ (1152.0)

HasImmediatePreviousTransactionSucceeded = true: CCGRWTRKQ (961.0)

: not-exist (482.0/1.0)
```

ResponseBody_error

```
HasSuccessfulDeleteOperationOccurredBefore = true: message_not_found (2648.0)

HasImmediatePreviousTransactionSucceeded = false AND
RequestUriPathToken2 = chat.postMessage: not-exist (1152.0)

HasImmediatePreviousTransactionSucceeded = true: not-exist (961.0)

: message_not_found (482.0/1.0)
```

ResponseBody_message.bot_id

```
HasImmediatePreviousTransaction = true AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasSuccessfulDeleteOperationOccurredBefore = true: not-exist (2648.0)

RequestHeader_Content-Type = application/x-www-form-urlencoded AND
RequestUriPathToken2 = chat.postMessage: BCEPNCQDN (1152.0)

RequestUriPathToken2 = chat.delete: not-exist (983.0)

HasImmediatePreviousTransactionSucceeded = true: BCEPNCQDN (235.0)

: not-exist (225.0/1.0)
```

ResponseBody_message.edited.user

```
HasURLInImmediatelyPreviousTransactionContainsATokenToUpdate = false: not-exist
    (3766.0)

HasImmediatePreviousTransactionSucceeded = false: not-exist (1242.0)

RequestUriPathToken2 = chat.delete: not-exist (224.0)

: UC8J6APLN (11.0)
```

ResponseBody_message.type

```
HasImmediatePreviousTransaction = true AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasSuccessfulDeleteOperationOccurredBefore = true: not-exist (2648.0)

RequestHeader_Content-Type = application/x-www-form-urlencoded AND
RequestUriPathToken2 = chat.postMessage: message (1152.0)

RequestUriPathToken2 = chat.delete: not-exist (983.0)

HasImmediatePreviousTransactionSucceeded = true: message (235.0)

: not-exist (225.0/1.0)
```

ResponseBody_message.user

```
HasImmediatePreviousTransaction = true AND
HasURLInImmediatelyPreviousTransactionContainsATokenToCreate = false AND
HasSuccessfulDeleteOperationOccurredBefore = true: not-exist (2648.0)

RequestHeader_Content-Type = application/x-www-form-urlencoded AND
RequestUriPathToken2 = chat.postMessage: UC8J6APLN (1152.0)

RequestUriPathToken2 = chat.delete: not-exist (983.0)

HasImmediatePreviousTransactionSucceeded = true: UC8J6APLN (235.0)

: not-exist (225.0/1.0)
```

ResponseBody_ok

```
HasSuccessfulDeleteOperationOccurredBefore = true: false (2648.0)

HasImmediatePreviousTransactionSucceeded = false AND
RequestUriPathToken2 = chat.postMessage: true (1152.0)

HasImmediatePreviousTransactionSucceeded = true: true (961.0)

: false (482.0/1.0)
```

C.4 Sample OCEL Class Expressions

C.4.1 Class Expressions from OCEL on GHTraffic

ResponseStatusCode_200

```
(RequestHeader_HasValidRequestPayload_true or
(RequestUriPathToken6_not-exist and
(not (RequestMethod_PATCH)))) and
(isPrecededBy some ResponseStatusCode_201)
```

ResponseStatusCode_204

```
RequestHeader_HasAuthorisationToken_true and
RequestUriPathToken6_lock and
(isPrecededBy some ResponseStatusCode_201)
```

ResponseStatusCode_400

```
RequestHeader_HasAuthorisationToken_true and
RequestHeader_HasRequestPayload_true and
RequestHeader_HasValidRequestPayload_false
```

ResponseStatusCode_401

```
RequestHeader_HasAuthorisationToken_false
```

ResponseStatusCode_404

```
RequestHeader_HasAuthorisationToken_true and
((RequestBody_state_closed and
(hasPrevious some ResponseHeader_Vary_not-exist)) or
(RequestHeader_Content-Type_not-exist and
(not (RequestMethod_PATCH))))
```

ResponseStatusCode_422

```
RequestHeader_HasRequestPayload_false and
RequestMethod_PATCH
```

ResponseHeader_Cache-Control_max-age

```
(RequestHeader_HasValidRequestPayload_true or
(RequestUriPathToken6_not-exist and
(not (RequestMethod_PATCH)))) and
(isPrecededBy some ResponseStatusCode_201)
```

ResponseHeader_Cache-Control_not-exist

```
(RequestHeader_HasValidRequestPayload_false and
(not (RequestMethod_HEAD))) or
(hasPrevious some (RequestUriPathToken6_not-exist and
ResponseHeader_Vary_not-exist))
```

ResponseHeader_Vary_accept

```
(RequestHeader_HasValidRequestPayload_true or
(RequestUriPathToken6_not-exist and
(not (RequestMethod_PATCH)))) and
(isPrecededBy some ResponseStatusCode_201)
```

ResponseHeader_Vary_not-exist

```
(RequestHeader_HasValidRequestPayload_false and
(not (RequestMethod_HEAD))) or
(hasPrevious some (RequestUriPathToken6_not-exist and
ResponseHeader_Vary_not-exist))
```

ResponseHeader_X-Accepted-OAuth-Scopes_not-exist

```
RequestHeader_HasAuthorisationToken_false
```

ResponseHeader_X-Accepted-OAuth-Scopes_accepted-public-repo

```
RequestHeader_HasAuthorisationToken_true and
(RequestHeader_HasValidRequestPayload_true or
(not (RequestMethod_PATCH))) and
(isPrecededBy some ResponseStatusCode_201)
```

ResponseHeader_X-Accepted-OAuth-Scopes_repo

```
RequestHeader_HasAuthorisationToken_true
```

ResponseHeader_X-OAuth-Scopes_not-exist

```
RequestHeader_HasAuthorisationToken_false
```

ResponseHeader_X-OAuth-Scopes_public-repo

```
RequestHeader_HasAuthorisationToken_true
```

ResponseBody_assignee.site_admin_false

```
(RequestHeader_HasValidRequestPayload_true or
RequestMethod_GET) and
(not (RequestUriPathToken6_lock)) and
(isPrecededBy some ResponseBody_assignees.type_User)
```

ResponseBody_assignee.site_admin_false

```
RequestUriPathToken6_not-exist and
((RequestHeader_HasValidRequestPayload_true and
(isPrecededBy some ResponseBody_assignees.type_User)) or
(RequestMethod_GET and
(isPrecededBy some ResponseBody_assignees.type_User)))
```

ResponseBody_assignee.site_admin_not-exist

```
RequestHeader_HasValidRequestPayload_false or
(hasPrevious some ResponseBody_assignees.type_not-exist)
```

ResponseBody_assignee.type_not-exist

```
RequestHeader_HasValidRequestPayload_false or
(hasPrevious some ResponseBody_assignees.type_not-exist)
```

ResponseBody_assignee.type_User

```
RequestUriPathToken6_not-exist and
((RequestHeader_HasValidRequestPayload_true and
(isPrecededBy some ResponseBody_assignees.type_User)) or
(RequestMethod_GET and
(isPrecededBy some ResponseBody_assignees.type_User)))
```

ResponseBody_assignees.site_admin_false

```
(not (RequestHeader_HasValidRequestPayload_false)) and
(isPrecededBy some ResponseBody_assignees.type_User)
```

ResponseBody_assignees.site_admin_not-exist

```
RequestHeader_HasValidRequestPayload_false or
(hasPrevious some ResponseBody_assignees.type_not-exist)
```

ResponseBody_assignees.type_not-exist

```
RequestHeader_HasValidRequestPayload_false or
(hasPrevious some ResponseBody_assignees.type_not-exist)
```

ResponseBody_assignees.type_User

```
(not (RequestHeader_HasValidRequestPayload_false)) and
(isPrecededBy some ResponseBody_assignees.type_User)
```

ResponseBody_locked_false

```
RequestUriPathToken6_not-exist and
(RequestHeader_HasValidRequestPayload_true or
RequestMethod_GET) and
(isPrecededBy some ResponseStatusCode_201)
```

ResponseBody_locked_not-exist

```
RequestHeader_HasValidRequestPayload_false or
(RequestBody_state_closed and
(hasPrevious some ResponseHeader_Vary_not-exist))
```

ResponseBody_milestone.creator.avatar_url_avatar-v3

```
RequestUriPathToken6_not-exist and
((RequestHeader_HasValidRequestPayload_true and
(isPrecededBy some ResponseBody_milestone.state_closed)) or
(RequestMethod_GET and
(isPrecededBy some ResponseBody_milestone.state_closed)))
```

ResponseBody_milestone.creator.avatar_url_not-exist

```
RequestHeader_HasValidRequestPayload_false or
(hasPrevious some ResponseBody_milestone.state_not-exist)
```

ResponseBody_milestone.creator.events_url_events-privacy

```
RequestUriPathToken6_not-exist and
((RequestHeader_HasValidRequestPayload_true and
(isPrecededBy some ResponseBody_milestone.state_closed)) or
(RequestMethod_GET and
(isPrecededBy some ResponseBody_milestone.state_closed)))
```

ResponseBody_milestone.creator.events_url_not-exist

```
RequestHeader_HasValidRequestPayload_false or
(hasPrevious some ResponseBody_milestone.state_not-exist)
```

ResponseBody_milestone.creator.followers_url_followers-cgdecker

```
RequestUriPathToken6_not-exist and
((RequestHeader_HasValidRequestPayload_true and
(isPrecededBy some ResponseBody_milestone.state_closed)) or
(RequestMethod_GET and
(isPrecededBy some ResponseBody_milestone.state_closed)))
```

ResponseBody_milestone.creator.followers_url_not-exist

```
RequestHeader_HasValidRequestPayload_false or
(hasPrevious some ResponseBody_milestone.state_not-exist)
```

ResponseBody_milestone.creator.following_url_following-cgdecker

```
RequestUriPathToken6_not-exist and
((RequestHeader_HasValidRequestPayload_true and
(isPrecededBy some ResponseBody_milestone.state_closed)) or
(RequestMethod_GET and
(isPrecededBy some ResponseBody_milestone.state_closed)))
```

ResponseBody_milestone.creator.following_url_not-exist

```
RequestHeader_HasValidRequestPayload_false or
(hasPrevious some ResponseBody_milestone.state_not-exist)
```

ResponseBody_milestone.creator.gists_url_gists-cgdecker

```
RequestUriPathToken6_not-exist and
((RequestHeader_HasValidRequestPayload_true and
(isPrecededBy some ResponseBody_milestone.state_closed)) or
(RequestMethod_GET and
(isPrecededBy some ResponseBody_milestone.state_closed)))
```

ResponseBody_milestone.creator.gists_url_not-exist

```
RequestHeader_HasValidRequestPayload_false or
(hasPrevious some ResponseBody_milestone.state_not-exist)
```

ResponseBody_milestone.creator.html_url_html-cgdecker

```
RequestUriPathToken6_not-exist and
((RequestHeader_HasValidRequestPayload_true and
(isPrecededBy some ResponseBody_milestone.state_closed)) or
(RequestMethod_GET and
(isPrecededBy some ResponseBody_milestone.state_closed)))
```

ResponseBody_milestone.creator.html_url_not-exist

```
RequestHeader_HasValidRequestPayload_false or
(hasPrevious some ResponseBody_milestone.state_not-exist)
```

ResponseBody_milestone.creator.id_101568

```
RequestUriPathToken6_not-exist and
((RequestHeader_HasValidRequestPayload_true and
(isPrecededBy some ResponseBody_milestone.state_closed)) or
(RequestMethod_GET and
(isPrecededBy some ResponseBody_milestone.state_closed)))
```

ResponseBody_milestone.creator.id_not-exist

```
RequestHeader_HasValidRequestPayload_false or
(hasPrevious some ResponseBody_milestone.state_not-exist)
```

ResponseBody_milestone.creator.login_cgdecker

```
RequestUriPathToken6_not-exist and
((RequestHeader_HasValidRequestPayload_true and
(isPrecededBy some ResponseBody_milestone.state_closed)) or
(RequestMethod_GET and
(isPrecededBy some ResponseBody_milestone.state_closed)))
```

ResponseBody_milestone.creator.login_not-exist

```
RequestHeader_HasValidRequestPayload_false or
(hasPrevious some ResponseBody_milestone.state_not-exist)
```

ResponseBody_milestone.creator.organizations_url_not-exist

```
RequestHeader_HasValidRequestPayload_false or
(hasPrevious some ResponseBody_milestone.state_not-exist)
```

ResponseBody_milestone.creator.organizations_url_organizations-cgdecker

```
RequestUriPathToken6_not-exist and
((RequestHeader_HasValidRequestPayload_true and
(isPrecededBy some ResponseBody_milestone.state_closed)) or
(RequestMethod_GET and
(isPrecededBy some ResponseBody_milestone.state_closed)))
```

ResponseBody_milestone.creator.received_events_url_events-cgdecker

```
RequestUriPathToken6_not-exist and
((RequestHeader_HasValidRequestPayload_true and
(isPrecededBy some ResponseBody_milestone.state_closed)) or
(RequestMethod_GET and
(isPrecededBy some ResponseBody_milestone.state_closed)))
```

ResponseBody_milestone.creator.received_events_url_not-exist

```
RequestHeader_HasValidRequestPayload_false or
(hasPrevious some ResponseBody_milestone.state_not-exist)
```

ResponseBody_milestone.creator.repos_url_not-exist

```
RequestHeader_HasValidRequestPayload_false or
(hasPrevious some ResponseBody_milestone.state_not-exist)
```

ResponseBody_milestone.creator.repos_url_repos-cgdecker

```
RequestUriPathToken6_not-exist and
((RequestHeader_HasValidRequestPayload_true and
(isPrecededBy some ResponseBody_milestone.state_closed)) or
(RequestMethod_GET and
(isPrecededBy some ResponseBody_milestone.state_closed)))
```

ResponseBody_milestone.creator.site_admin_false

```
RequestUriPathToken6_not-exist and
((RequestHeader_HasValidRequestPayload_true and
(isPrecededBy some ResponseBody_milestone.state_closed)) or
(RequestMethod_GET and
(isPrecededBy some ResponseBody_milestone.state_closed)))
```

ResponseBody_milestone.creator.site_admin_not-exist

```
RequestHeader_HasValidRequestPayload_false or
(hasPrevious some ResponseBody_milestone.state_not-exist)
```

ResponseBody_milestone.creator.starred_url_not-exist

```
RequestHeader_HasValidRequestPayload_false or
(hasPrevious some ResponseBody_milestone.state_not-exist)
```

ResponseBody_milestone.creator.starred_url_starred-cgdecker

```
RequestUriPathToken6_not-exist and
((RequestHeader_HasValidRequestPayload_true and
(isPrecededBy some ResponseBody_milestone.state_closed)) or
(RequestMethod_GET and
(isPrecededBy some ResponseBody_milestone.state_closed)))
```

ResponseBody_milestone.creator.subscriptions_url_not-exist

```
RequestHeader_HasValidRequestPayload_false or
(hasPrevious some ResponseBody_milestone.state_not-exist)
```

ResponseBody_milestone.creator.subscriptions_url_subscriptions-cgdecker

```
RequestUriPathToken6_not-exist and
((RequestHeader_HasValidRequestPayload_true and
(isPrecededBy some ResponseBody_milestone.state_closed)) or
(RequestMethod_GET and
(isPrecededBy some ResponseBody_milestone.state_closed)))
```

ResponseBody_milestone.creator.type_not-exist

```
RequestHeader_HasValidRequestPayload_false or
(hasPrevious some ResponseBody_milestone.state_not-exist)
```

ResponseBody_milestone.creator.type_User

```
RequestUriPathToken6_not-exist and
((RequestHeader_HasValidRequestPayload_true and
(isPrecededBy some ResponseBody_milestone.state_closed)) or
(RequestMethod_GET and
(isPrecededBy some ResponseBody_milestone.state_closed)))
```

ResponseBody_milestone.creator.url_creator-cgdecker

```
RequestUriPathToken6_not-exist and
((RequestHeader_HasValidRequestPayload_true and
(isPrecededBy some ResponseBody_milestone.state_closed)) or
(RequestMethod_GET and
(isPrecededBy some ResponseBody_milestone.state_closed)))
```

ResponseBody_milestone.creator.url_not-exist

```
RequestHeader_HasValidRequestPayload_false or
(hasPrevious some ResponseBody_milestone.state_not-exist)
```

ResponseBody_milestone.due_on_not-exist

```
RequestHeader_HasValidRequestPayload_false or
(hasPrevious some ResponseBody_milestone.state_not-exist)
```

ResponseBody_milestone.due_on_null

```
RequestUriPathToken6_not-exist and
((RequestHeader_HasValidRequestPayload_true and
(isPrecededBy some ResponseBody_milestone.state_closed)) or
(RequestMethod_GET and
(isPrecededBy some ResponseBody_milestone.state_closed)))
```

ResponseBody_milestone.open_issues_0

```
(RequestHeader_HasValidRequestPayload_true or
RequestMethod_GET) and
(not (RequestUriPathToken6_lock)) and
(isPrecededBy some ResponseBody_milestone.open_issues_0)
```

ResponseBody_milestone.open_issues_1

```
RequestHeader_HasValidRequestPayload_true and
(isPrecededBy some ResponseBody_milestone.open_issues_1)
```

ResponseBody_milestone.open_issues_not-exist

```
RequestHeader_HasValidRequestPayload_false or
(hasPrevious some ResponseBody_milestone.state_not-exist)
```

ResponseBody_milestone.state_closed

```
RequestUriPathToken6_not-exist and
((RequestHeader_HasValidRequestPayload_true and
(isPrecededBy some ResponseBody_milestone.state_closed)) or
(RequestMethod_GET and
(isPrecededBy some ResponseBody_milestone.state_closed)))
```

ResponseBody_milestone.state_not-exist

```
RequestHeader_HasValidRequestPayload_false or
(hasPrevious some ResponseBody_milestone.state_not-exist)
```

ResponseBody_state_closed

```
RequestUriPathToken6_not-exist and
((RequestHeader_HasValidRequestPayload_true and
(isPrecededBy some ResponseStatusCode_201)) or
(RequestMethod_GET and
(isPrecededBy some ResponseBody_state_closed)))
```

ResponseBody_state_not-exist

```
RequestHeader_HasValidRequestPayload_false or
(RequestBody_state_closed and
(hasPrevious some ResponseHeader_Vary_not-exist))
```

ResponseBody_state_open

```
RequestMethod_GET and
RequestUriPathToken6_not-exist and
(not (RequestMethod_PATCH)) and
(hasPrevious some (RequestBody_state_not-exist and
(hasPrevious some (not (ResponseBody_state_closed)))))
```

ResponseBody_user.site_admin_false

```
(RequestHeader_HasValidRequestPayload_true or
RequestMethod_GET) and
(not (RequestUriPathToken6_lock)) and
(isPrecededBy some ResponseStatusCode_201)
```

ResponseBody_user.site_admin_not-exist

```
RequestHeader_HasValidRequestPayload_false or
(RequestBody_state_closed and
(hasPrevious some ResponseHeader_Vary_not-exist))
```

ResponseBody_user.type_not-exist

```
RequestHeader_HasValidRequestPayload_false or
(RequestBody_state_closed and
(hasPrevious some ResponseHeader_Vary_not-exist))
```

ResponseBody_user.type_User

```
RequestHeader_HasAuthorisationToken_true and
RequestUriPathToken6_not-exist and
(RequestHeader_HasValidRequestPayload_true or
RequestMethod_GET) and
(isPrecededBy some ResponseStatusCode_201)
```

ResponseBody_message_invalid-request

```
RequestHeader_HasRequestPayload_false and
RequestMethod_PATCH
```

ResponseBody_message_not-exist

```
RequestHeader_HasAuthorisationToken_true and
((RequestHeader_HasRequestPayload_false and
(not (RequestMethod_PATCH))) or
(RequestHeader_HasValidRequestPayload_true and
(isPrecededBy some ResponseStatusCode_201)))
```

ResponseBody_message_not-found

```
RequestHeader_HasAuthorisationToken_true and
(not (RequestMethod_HEAD)) and
(hasPrevious some (ResponseHeader_Vary_not-exist and
(ResponseBody_documentation_url_create-an-issue or
ResponseStatusCode_404)))
```

ResponseBody_message_problems-passing-json

```
RequestHeader_HasAuthorisationToken_true and
RequestHeader_HasRequestPayload_true and
RequestHeader_HasValidRequestPayload_false
```

ResponseBody_message_requires-authentication

```
RequestHeader_HasAuthorisationToken_false
```

ResponseBody_documentation_url_edit-an-issue

```
RequestHeader_HasValidRequestPayload_false and
RequestMethod_PATCH
```

ResponseBody_documentation_url_lock-an-issue

```
RequestHeader_HasAuthorisationToken_false and
RequestMethod_PUT
```

ResponseBody_documentation_url_not-exist

```
RequestHeader_HasAuthorisationToken_true and
(RequestHeader_HasValidRequestPayload_true or
(RequestHeader_HasRequestPayload_false and
(not (RequestMethod_PATCH)))) and
(isPrecededBy some ResponseStatusCode_201)
```

ResponseBody_documentation_url_unlock-an-issue

```
RequestHeader_HasAuthorisationToken_false and
RequestMethod_DELETE
```

ResponseBody_documentation_url_v3

```
(RequestHeader_HasValidRequestPayload_false or
(RequestBody_state_closed and
(hasPrevious some ResponseHeader_Vary_not-exist))) and
(not (RequestMethod_HEAD))
```

C.4.2 Class Expressions from OCEL on Twitter

ResponseStatusCode_200

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseStatusCode_404

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseHeader_status_200OK

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseHeader_status_404NotFound

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseHeader_x-rate-limit-limit_900

```
RequestMethod_GET
```

ResponseHeader_x-rate-limit-limit_not-exist

```
RequestMethod_POST
```

ResponseBody_contributors_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_contributors_null

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_coordinates_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_coordinates_null

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_entities.hashtags_empty-list

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_entities.hashtags_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_entities.symbols_empty-list

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_entities.symbols_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_entities.urls_empty-list

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_entities.urls_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_entities.user_mentions_empty-list

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_entities.user_mentions_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_errors.code_144

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_errors.code_not-exist

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_errors.message_no-status-found

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_errors.message_not-exist

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_favorite_count_0

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_favorite_count_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_favorited_false

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_favorited_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_geo_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_geo_null

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_in_reply_to_screen_name_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_in_reply_to_screen_name_null

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_in_reply_to_status_id_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_in_reply_to_status_id_null

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_in_reply_to_status_id_str_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_in_reply_to_status_id_str_null

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_in_reply_to_user_id_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_in_reply_to_user_id_null

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_in_reply_to_user_id_str_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_in_reply_to_user_id_str_null

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_is_quote_status_false

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_is_quote_status_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_place_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_place_null

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_retweet_count_0

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_retweet_count_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_retweeted_false

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_retweeted_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_truncated_false

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_truncated_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.contributors_enabled_false

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.contributors_enabled_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.created_at_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.created_at_WedMar0709

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.default_profile_false

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.default_profile_image_false

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.default_profile_image_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.default_profile_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.favourites_count_64

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.favourites_count_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.follow_request_sent_false

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.follow_request_sent_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.followers_count_185

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.followers_count_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.following_false

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.following_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.friends_count_249

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.friends_count_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.geo_enabled_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.geo_enabled_true

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.has_extended_profile_false

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.has_extended_profile_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.id_517417816

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.id_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.id_str_517417816

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.id_str_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.is_translation_enabled_false

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.is_translation_enabled_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.is_translator_false

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.is_translator_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.lang_en

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.lang_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.listed_count_3

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.listed_count_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.location_Kurunegala

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.location_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.name_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.name_ThiliniBhagya

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.notifications_false

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.notifications_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.profile_background_color_1A1B1F

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.profile_background_color_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.profile_background_image_url_http-abc.twimg.com

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.profile_background_image_url_https_https-abc.twimg.com

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.profile_background_image_url_https_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.profile_background_image_url_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.profile_background_tile_false

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.profile_background_tile_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.profile_banner_url_https-pbs.twimg.com-profile-banners

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.profile_banner_url_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.profile_image_url_http-pbs.twimg.com-profile-images

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.profile_image_url_https_https-pbs.twimg.com-profile-images

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.profile_image_url_https_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.profile_image_url_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.profile_link_color_3E4547

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200
and (not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.profile_link_color_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.profile_sidebar_border_color_FFFFFF

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.profile_sidebar_border_color_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.profile_sidebar_fill_color_252429

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.profile_sidebar_fill_color_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.profile_text_color_666666

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.profile_text_color_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.profile_use_background_image_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.profile_use_background_image_true

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.protected_false

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.protected_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.screen_name_bhagyasl

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.screen_name_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.time_zone_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.time_zone_null

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.translator_type_none

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.translator_type_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.url_http-t.co

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.url_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.utc_offset_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

ResponseBody_user.utc_offset_null

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.verified_false

```
RequestHeader_Content-Type_form-urlencoded or
(RequestHeader_Content-Type_not-exist and
(hasPrevious some (ResponseStatusCode_200 and
(not (RequestUriPathToken3_destroy)))))
```

ResponseBody_user.verified_not-exist

```
(RequestUriPathToken3_show.json and
(isPrecededBy some ResponseStatusCode_404)) or
(isPrecededBy some RequestUriPathToken3_destroy)
```

C.4.3 Class Expressions from OCEL on Google Tasks

ResponseStatusCode_200

```
(not (RequestMethod_DELETE)) and
(hasPrevious some ResponseStatusCode_200)
```

ResponseStatusCode_204

```
RequestMethod_DELETE and
(hasPrevious some ResponseStatusCode_200)
```

ResponseStatusCode_404

```
isPrecededBy some ResponseStatusCode_204
```

ResponseStatusCode_503

```
(not (RequestMethod_DELETE)) and
(not (RequestMethod_PATCH)) and
(hasPrevious some (RequestMethod_DELETE and
(hasPrevious some (hasPrevious some (hasPrevious some RequestMethod_PATCH)))))
```

ResponseHeader_Accept-Ranges_none

```
RequestHeader_HasValidRequestPayload_true or
(isPrecededBy some ResponseStatusCode_204)
```

ResponseHeader_Accept-Ranges_not-exist

```
RequestHeader_HasValidRequestPayload_false and
(hasPrevious some ResponseStatusCode_200)
```

ResponseHeader_Alt-Svc_quic

ResponseHeader_Cache-Control_must-revalidate

```
RequestMethod_GET and
(hasPrevious some ResponseStatusCode_200)
```

ResponseHeader_Cache-Control_no-cache

```
RequestHeader_HasValidRequestPayload_true or
(RequestMethod_DELETE and
(hasPrevious some ResponseStatusCode_200))
```

ResponseHeader_Cache-Control_private

```
isPrecededBy some ResponseStatusCode_204
```

ResponseHeader_Content-Type_json

```
(not (RequestMethod_DELETE)) or
(isPrecededBy some ResponseStatusCode_204)
```

ResponseHeader_Content-Type_not-exist

```
RequestMethod_DELETE and
(hasPrevious some ResponseStatusCode_200)
```

ResponseHeader_Pragma_no-cache

```
RequestHeader_HasValidRequestPayload_true or
(RequestMethod_DELETE and
(hasPrevious some ResponseStatusCode_200))
```

ResponseHeader_Pragma_not-exist

```
RequestMethod_GET or
(isPrecededBy some ResponseHeader_Vary_X-Origin)
```

ResponseHeader_Server_GSE

ResponseHeader_Transfer-Encoding_chunked

```
RequestHeader_HasValidRequestPayload_true or
(isPrecededBy some ResponseStatusCode_204)
```

ResponseHeader_Transfer-Encoding_not-exist

```
RequestHeader_HasValidRequestPayload_false and
(hasPrevious some ResponseStatusCode_200)
```

ResponseHeader_Vary_origin

```
RequestHeader_HasValidRequestPayload_true or
(isPrecededBy some ResponseStatusCode_204)
```

ResponseHeader_Vary_X-Origin

```
RequestHeader_HasValidRequestPayload_false and
(hasPrevious some ResponseStatusCode_200)
```

ResponseHeader_X-Content-Type-Options_nosniff

```
(not (RequestMethod_DELETE)) or
(isPrecededBy some ResponseStatusCode_204)
```

ResponseHeader_X-Content-Type-Options_not-exist

```
RequestMethod_DELETE and
(hasPrevious some ResponseStatusCode_200)
```

ResponseHeader_X-Frame-Options_not-exist

```
RequestMethod_DELETE and
(hasPrevious some ResponseStatusCode_200)
```

ResponseHeader_X-Frame-Options_SAMEORIGIN

```
(not (RequestMethod_DELETE)) or
(isPrecededBy some ResponseStatusCode_204)
```

ResponseHeader_X-XSS-Protection_block

```
(not (RequestMethod_DELETE)) or
(isPrecededBy some ResponseStatusCode_204)
```

ResponseHeader_X-XSS-Protection_not-exist

```
RequestMethod_DELETE and
(hasPrevious some ResponseStatusCode_200)
```

ResponseBody_error.code_404

```
isPrecededBy some ResponseStatusCode_204
```

ResponseBody_error.code_503

```
(not (RequestMethod_DELETE)) and
(not (RequestMethod_PATCH)) and
(hasPrevious some (RequestMethod_DELETE and
(hasPrevious some (hasPrevious some (hasPrevious some RequestMethod_PATCH)))))
```

ResponseBody_error.code_not-exist

```
(RequestHeader_Content-Type_json and
(not (RequestMethod_PATCH))) or
(hasPrevious some ResponseStatusCode_200)
```

ResponseBody_error.errors.domain_global

```
isPrecededBy some ResponseStatusCode_204
```

ResponseBody_error.errors.domain_not-exist

```
(RequestHeader_Content-Type_json and
(not (RequestMethod_PATCH))) or
(hasPrevious some ResponseStatusCode_200)
```

ResponseBody_error.errors.message_BackendError

```
(not (RequestMethod_DELETE)) and
(not (RequestMethod_PATCH)) and
(hasPrevious some (RequestMethod_DELETE and
(hasPrevious some (hasPrevious some (hasPrevious some RequestMethod_PATCH)))))
```

ResponseBody_error.errors.message_not-exist

```
(RequestMethod_POST and
(not (RequestMethod_PATCH))) or
(hasPrevious some ResponseStatusCode_200)
```

ResponseBody_error.errors.message_NotFound

```
isPrecededBy some ResponseStatusCode_204
```

ResponseBody_error.errors.reason_backendError

```
(not (RequestMethod_DELETE)) and
(not (RequestMethod_PATCH)) and
(hasPrevious some (RequestMethod_DELETE and
(hasPrevious some (hasPrevious some (hasPrevious some RequestMethod_PATCH)))))
```

ResponseBody_error.errors.reason_not-exist

```
(RequestMethod_POST and
(not (RequestMethod_PATCH))) or
(hasPrevious some ResponseStatusCode_200)
```

ResponseBody_error.errors.reason_notFound

```
isPrecededBy some ResponseStatusCode_204
```

ResponseBody_error.message_BackendError

```
(not (RequestMethod_DELETE)) and
(not (RequestMethod_PATCH)) and
(hasPrevious some (RequestMethod_DELETE and
(hasPrevious some (hasPrevious some (hasPrevious some RequestMethod_PATCH)))))
```

ResponseBody_error.message_not-exist

```
(RequestMethod_POST and
(not (RequestMethod_PATCH))) or
(hasPrevious some ResponseStatusCode_200)
```

ResponseBody_error.message_NotFound

```
isPrecededBy some ResponseStatusCode_204
```

C.4.4 Class Expressions from OCEL on Slack

ResponseHeader_x-slack-router_not-exist

```
Thing
```

ResponseHeader_x-slack-router_p

```
Thing
```

ResponseBody_channel_CCGRWTRKQ

```
RequestUriPathToken2_chat.postMessage or
((not (RequestUriPathToken2_chat.update)) and
(hasPrevious some ResponseBody_ok_true))
```

ResponseBody_channel_not-exist

```
not (RequestUriPathToken2_chat.postMessage)
```

ResponseBody_error_messagenotfound

```
not (RequestUriPathToken2_chat.postMessage)
```

ResponseBody_error_not-exist

```
RequestUriPathToken2_chat.postMessage or
((not (RequestUriPathToken2_chat.update)) and
(hasPrevious some ResponseBody_ok_true))
```

ResponseBody_message.bot_id_BCEPNCQDN

```
RequestHeader_Content-Type_x-www and
(not (RequestUriPathToken2_chat.update))
```

ResponseBody_message.bot_id_not-exist

```
(not (RequestUriPathToken2_chat.postMessage))
```

ResponseBody_message.type_message

```
RequestHeader_Content-Type_x-www and
(not (RequestUriPathToken2_chat.update))
```

ResponseBody_message.type_not-exist

```
not (RequestUriPathToken2_chat.postMessage)
```

ResponseBody_message.user_not-exist

```
not (RequestUriPathToken2_chat.postMessage)
```

ResponseBody_message.user_UC8J6APLN

```
RequestHeader_Content-Type_x-www and
(not (RequestUriPathToken2_chat.update))
```

ResponseBody_ok_false

```
not (RequestUriPathToken2_chat.postMessage)
```

ResponseBody_ok_true

```
RequestUriPathToken2_chat.postMessage or
((not (RequestUriPathToken2_chat.update)) and
(hasPrevious some ResponseBody_ok_true))
```

C.5 2-Fold Cross Validation of Attribute-Based Learning Algorithms

C.5.1 2-Fold Cross Validation Results on GHTraffic

Target	C4.5					RIPPER				PART			
	Accuracy	Precision	Recall	Leaves	Size	Accuracy	Precision	Recall	Rules	Accuracy	Precision	Recall	Rules
ResponseStatusCode	0.9858	0.9948	0.8893	38	51	0.9858	0.9948	0.8893	9	0.9858	0.9948	0.8893	23
ResponseHeader_Cache-Control	1.0000	1.0000	1.0000	16	23	1.0000	1.0000	1.0000	5	1.0000	1.0000	1.0000	8
ResponseHeader_Vary	1.0000	1.0000	1.0000	16	23	1.0000	1.0000	1.0000	5	1.0000	1.0000	1.0000	8
ResponseHeader_X-Accepted-OAuth-Scopes	1.0000	1.0000	1.0000	15	21	1.0000	1.0000	1.0000	7	1.0000	1.0000	1.0000	7
ResponseHeader_X-OAuth-Scopes	1.0000	1.0000	1.0000	2	3	1.0000	1.0000	1.0000	2	1.0000	1.0000	1.0000	2
ResponseBody_assignee.site_admin	0.9634	0.4815	0.5000	1	1	0.9634	0.4815	0.5000	1	0.9634	0.4815	0.5000	1
ResponseBody_assignee.type	0.9634	0.4815	0.5000	1	1	0.9634	0.4815	0.5000	1	0.9634	0.4815	0.5000	1
ResponseBody_assignees.site_admin	0.9825	0.4915	0.5000	1	1	0.9825	0.4915	0.5000	1	0.9825	0.4915	0.5000	1
ResponseBody_assignees.type	0.9825	0.4915	0.5000	1	1	0.9825	0.4915	0.5000	1	0.9825	0.4915	0.5000	1
ResponseBody_closed_by.avatar_url	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.events_url	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.followers_url	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.following_url	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.gists_url	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.html_url	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.id	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.login	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.organizations_url	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.received_events_url	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.repos_url	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.site_admin	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.starred_url	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.subscriptions_url	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.type	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_closed_by.url	0.9998	0.2500	0.2500	1	1	0.9998	0.2500	0.2500	1	0.9998	0.2500	0.2500	1
ResponseBody_documentation_url	0.9856	0.9897	0.9937	34	43	0.9840	0.8217	0.9933	17	0.9856	0.9897	0.9937	23
ResponseBody_locked	0.9999	0.6663	0.6667	16	23	0.9999	0.6663	0.6667	4	0.9999	0.6663	0.6667	8
ResponseBody_message	0.9855	0.9433	0.8602	32	43	0.9855	0.8602	0.8602	10	0.9855	0.9433	0.8602	24
ResponseBody_milestone.creator.avatar_url	0.9654	0.3217	0.3333	1	1	0.9652	0.4050	0.3340	1	0.9652	0.4050	0.3340	1
ResponseBody_milestone.creator.events_url	0.9654	0.3217	0.3333	1	1	0.9654	0.4825	0.5000	1	0.9654	0.4825	0.5000	1
ResponseBody_milestone.creator.followers_url	0.9654	0.3217	0.3333	1	1	0.9654	0.4825	0.5000	1	0.9654	0.4825	0.5000	1
ResponseBody_milestone.creator.following_url	0.9654	0.3217	0.3333	1	1	0.9654	0.4825	0.5000	1	0.9654	0.4825	0.5000	1
ResponseBody_milestone.creator.gists_url	0.9654	0.3217	0.3333	1	1	0.9654	0.4825	0.5000	1	0.9654	0.4825	0.5000	1

Continued on next page

Table C.1 – continued from previous page

Target	**C4.5**					**RIPPER**				**PART**			
	Accuracy	**Precision**	**Recall**	**Leaves**	**Size**	**Accuracy**	**Precision**	**Recall**	**Rules**	**Accuracy**	**Precision**	**Recall**	**Rules**
ResponseBody_milestone.creator.html_url	0.9654	0.3217	0.3333	1	1	0.9654	0.4825	0.5000	1	0.9654	0.4825	0.5000	1
ResponseBody_milestone.creator.id	0.9654	0.3217	0.3333	1	1	0.9654	0.4825	0.5000	1	0.9654	0.4825	0.5000	1
ResponseBody_milestone.creator.login	0.9654	0.3217	0.3333	1	1	0.9654	0.4825	0.5000	1	0.9654	0.4825	0.5000	1
ResponseBody_milestone.creator.organizations_url	0.9654	0.3217	0.3333	1	1	0.9654	0.4825	0.5000	1	0.9654	0.4825	0.5000	1
ResponseBody_milestone.creator.received_events_url	0.9654	0.3217	0.3333	1	1	0.9654	0.4825	0.5000	1	0.9654	0.4825	0.5000	1
ResponseBody_milestone.creator.repos_url	0.9654	0.3217	0.3333	1	1	0.9654	0.4825	0.5000	1	0.9654	0.4825	0.5000	1
ResponseBody_milestone.creator.site_admin	0.9654	0.3217	0.3333	1	1	0.9654	0.4825	0.5000	1	0.9654	0.4825	0.5000	1
ResponseBody_milestone.creator.starred_url	0.9654	0.3217	0.3333	1	1	0.9654	0.4825	0.5000	1	0.9654	0.4825	0.5000	1
ResponseBody_milestone.creator.subscriptions_url	0.9654	0.3217	0.3333	1	1	0.9654	0.4825	0.5000	1	0.9654	0.4825	0.5000	1
ResponseBody_milestone.creator.type	0.9654	0.3217	0.3333	1	1	0.9654	0.4825	0.5000	1	0.9654	0.4825	0.5000	1
ResponseBody_milestone.creator.url	0.9654	0.3217	0.3333	1	1	0.9654	0.4825	0.5000	1	0.9654	0.4825	0.5000	1
ResponseBody_milestone.due_on	0.9654	0.3217	0.3333	1	1	0.9654	0.4825	0.5000	1	0.9654	0.4825	0.5000	1
ResponseBody_milestone.open_issues	0.9654	0.2413	0.2500	1	1	0.9654	0.2413	0.2500	1	0.9654	0.2413	0.2412	1
ResponseBody_milestone.state	0.9654	0.3217	0.3333	1	1	0.9654	0.4825	0.3333	1	0.9654	0.3217	0.2500	1
ResponseBody_state	0.9790	0.9350	0.9273	13	21	0.9790	0.9350	0.9273	5	0.9787	0.9317	0.9260	14
ResponseBody_user.site_admin	1.0000	1.0000	1.0000	16	23	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	8
ResponseBody_user.type	1.0000	1.0000	1.0000	16	23	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	8
Mean	0.9833	0.4501	0.4517	5.0600	6.7200	0.9833	0.5015	0.5051	2.2200	0.9833	0.5032	0.5032	3.4400
Standard Deviation	0.0164	0.2826	0.2741	9.1035	12.4672	0.0164	0.2586	0.2615	2.9918	0.0164	0.2676	0.2631	5.8034

C.5.2 2-Fold Cross Validation Results on Twitter

Target	C4.5					RIPPER				PART			
	Accuracy	Precision	Recall	Leaves	Size	Accuracy	Precision	Recall	Rules	Accuracy	Precision	Recall	Rules
ResponseStatusCode	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	3	1.0000	1.0000	1.0000	3
ResponseHeader_status	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	3	1.0000	1.0000	1.0000	3
ResponseHeader_x-rate-limit-limit	1.0000	1.0000	1.0000	2	3	1.0000	1.0000	1.0000	2	1.0000	1.0000	1.0000	2
ResponseBody_contributors	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_coordinates	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_entities.hashtags	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_entities.symbols	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_entities.urls	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_entities.user_mentions	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_errors.code	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_errors.message	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_favorite_count	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_favorited	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_geo	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_in_reply_to_screen_name	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_in_reply_to_status_id	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_in_reply_to_status_id_str	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_in_reply_to_user_id	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_in_reply_to_user_id_str	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_is_quote_status	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_place	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_retweet_count	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_retweeted	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_truncated	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.contributors_enabled	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.created_at	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.default_profile	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.default_profile_image	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.favourites_count	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.follow_request_sent	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.followers_count	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.following	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.friends_count	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.geo_enabled	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.has_extended_profile	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.id	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.id_str	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3

Continued on next page

Table C.2 – continued from previous page

Target	C4.5					RIPPER				PART			
	Accuracy	**Precision**	**Recall**	**Leaves**	**Size**	**Accuracy**	**Precision**	**Recall**	**Rules**	**Accuracy**	**Precision**	**Recall**	**Rules**
ResponseBody_user.is_translation_enabled	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.is_translator	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.lang	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.listed_count	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.location	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.name	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.notifications	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.profile_background_color	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.profile_background_image_url	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.profile_background_image_url_https	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.profile_background_tile	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.profile_banner_url	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.profile_image_url	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.profile_image_url_https	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.profile_link_color	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.profile_sidebar_border_color	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.profile_sidebar_fill_color	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.profile_text_color	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.profile_use_background_image	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.protected	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.screen_name	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.time_zone	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.translator_type	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.url	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.utc_offset	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
ResponseBody_user.verified	1.0000	1.0000	1.0000	6	10	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	3
Mean	1.0000	1.0000	1.0000	5.9365	9.8889	1.0000	1.0000	1.0000	3.9365	1.0000	1.0000	1.0000	2.9841
Standard Deviation	0.0000	0.0000	0.0000	0.5040	0.8819	0.0000	0.0000	0.0000	0.3044	0.0000	0.0000	0.0000	0.1260

C.5.3 2-Fold Cross Validation Results on Google Tasks

Target	**C4.5**					**RIPPER**				**PART**			
	Accuracy	**Precision**	**Recall**	**Leaves**	**Size**	**Accuracy**	**Precision**	**Recall**	**Rules**	**Accuracy**	**Precision**	**Recall**	**Rules**
ResponseStatusCode	0.9977	0.7485	0.7500	5	7	0.9977	0.7485	0.7500	3	0.9977	0.7485	0.7500	5
ResponseHeader_Accept-Ranges	0.9994	0.9990	0.9995	3	5	0.9983	0.9980	0.9985	3	0.9994	0.9990	0.9995	3
ResponseHeader_Cache-Control	0.9985	0.9983	0.9983	5	7	0.9983	0.9983	0.9980	3	0.9985	0.9983	0.9983	4
ResponseHeader_Content-Type	1.0000	1.0000	1.0000	5	7	1.0000	1.0000	1.0000	2	1.0000	1.0000	1.0000	4
ResponseHeader_Pragma	0.9991	0.9990	0.9990	5	7	0.9989	0.9990	0.9990	4	0.9991	0.9990	0.9990	4
ResponseHeader_Transfer-Encoding	0.9994	0.9990	0.9995	3	5	0.9985	0.9980	0.9990	3	0.9994	0.9990	0.9995	3
ResponseHeader_Vary	0.9994	0.9990	0.9995	3	5	0.9983	0.9980	0.9985	3	0.9994	0.9990	0.9995	3
ResponseHeader_X-Content-Type-Options	1.0000	1.0000	1.0000	5	7	1.0000	1.0000	1.0000	2	1.0000	1.0000	1.0000	4
ResponseHeader_X-Frame-Options	1.0000	1.0000	1.0000	5	7	1.0000	1.0000	1.0000	2	1.0000	1.0000	1.0000	4
ResponseHeader_X-XSS-Protection	1.0000	1.0000	1.0000	5	7	1.0000	1.0000	1.0000	2	1.0000	1.0000	1.0000	4
ResponseBody_error.code	0.9977	0.6650	0.6667	2	3	0.9977	0.6650	0.6667	2	0.9977	0.6650	0.6667	2
ResponseBody_error.errors.domain	0.9985	0.9990	0.9975	2	3	0.9985	0.9990	0.9975	2	0.9985	0.9990	0.9975	2
ResponseBody_error.errors.message	0.9977	0.6650	0.6667	2	3	0.9977	0.6650	0.6667	2	0.9977	0.6650	0.6667	2
ResponseBody_error.errors.reason	0.9977	0.6650	0.6667	2	3	0.9977	0.6650	0.6667	2	0.9977	0.6650	0.6667	2
ResponseBody_error.message	0.9977	0.6650	0.6667	2	3	0.9977	0.6650	0.6667	2	0.9977	0.6650	0.6667	2
ResponseBody_kind	0.9985	0.9985	0.9985	5	7	0.9985	0.9985	0.9985	3	0.9985	0.9985	0.9985	5
Mean	0.9988	0.9000	0.9005	3.6875	5.3750	0.9986	0.8998	0.9004	2.5000	0.9988	0.9000	0.9005	3.3125
Standard Deviation	0.0009	0.1532	0.1524	1.4009	1.8212	0.0009	0.1531	0.1523	0.6325	0.0009	0.1532	0.1524	1.0782

C.5.4 2-Fold Cross Validation Results on Slack

Target	C4.5					RIPPER				PART			
	Accuracy	**Precision**	**Recall**	**Leaves**	**Size**	**Accuracy**	**Precision**	**Recall**	**Rules**	**Accuracy**	**Precision**	**Recall**	**Rules**
ResponseHeader_x-slack-router	0.6350	0.3175	0.5000	1	1	0.6350	0.3175	0.5000	1	0.6344	0.4715	0.9990	1
ResponseBody_channel	1.0000	1.0000	1.0000	8	14	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	4
ResponseBody_error	1.0000	1.0000	1.0000	8	14	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	4
ResponseBody_message.bot_id	1.0000	1.0000	1.0000	5	8	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	5
ResponseBody_message.edited.user	1.0000	1.0000	1.0000	5	8	1.0000	1.0000	1.0000	2	1.0000	1.0000	1.0000	4
ResponseBody_message.type	1.0000	1.0000	1.0000	5	8	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	5
ResponseBody_message.user	1.0000	1.0000	1.0000	5	8	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	5
ResponseBody_ok	1.0000	1.0000	1.0000	8	14	1.0000	1.0000	1.0000	4	1.0000	1.0000	1.0000	4
Mean	0.9544	0.9147	0.9375	5.6250	9.3750	0.9544	0.9147	0.9375	3.3750	0.9543	0.9339	0.9999	4.0000
Standard Deviation	0.1290	0.2413	0.1768	2.3868	4.5020	0.1290	0.2413	0.1768	1.1877	0.1293	0.1869	0.0004	1.3093

C.6 T-Test Results

T-Test Results Comparing Means Obtained for Predictive Accuracy

		Pairs					
Dataset	**Criterion**	**C4.5 - RIPPER**	**C4.5 - PART**	**RIPPER - PART**	**C4.5 - OCEL**	**RIPPER - OCEL**	**PART - OCEL**
GHTraffic	t-value	0.00848	0	-0.00848	2.17839	2.17638	2.17839
	p-value	0.496627	0.5	0.496627	0.015597	.015674	0.015597
	Significant	No	No	No	Yes	Yes	Yes
Twitter	t-value	0	0	0	11.59104	11.59104	11.59104
	p-value	0.5	0.5	0.5	<.00001	<.00001	<.00001
	Significant	No	No	No	Yes	Yes	Yes
Google Tasks	t-value	0.68699	0	-0.68699	2.2488	2.22899	2.2488
	p-value	0.248682	0.5	0.248682	0.014353	0.015039	0.014353
	Significant	No	No	No	Yes	Yes	Yes
Slack	t-value	0	0	0	-13.75928	-13.75928	-13.75928
	p-value	0.5	0.5	0.5	<.00001	<.00001	<.00001
	Significant	No	No	No	Yes	Yes	Yes

T-Test Results Comparing Means Obtained for Precision

		Pairs					
Dataset	**Criterion**	**C4.5 - RIPPER**	**C4.5 - PART**	**RIPPER - PART**	**C4.5 - OCEL**	**RIPPER - OCEL**	**PART - OCEL**
GHTraffic	t-value	0.00036	0	-0.00036	-12.94095	-12.94273	-12.94095
	p-value	0.499857	0.5	0.499857	<.00001	-12.94273	<.00001
	Significant	No	No	No	Yes	Yes	Yes
Twitter	t-value	0	0	0	7.77081	7.77081	7.77081
	p-value	0.5	0.5	0.5	<.00001	<.00001	<.00001
	Significant	No	No	No	Yes	Yes	Yes
Google Tasks	t-value	0.00462	0	-0.00462	0.85041	0.8476	0.85041
	p-value	0.498173	0.5	0.498173	0.199462	0.200237	.199462.
	Significant	No	No	No	No	No	No
Slack	t-value	0	0	0	-13.77105	-13.77105	-13.77105
	p-value	0.5	0.5	0.5	<.00001	<.00001	<.00001
	Significant	No	No	No	Yes	Yes	Yes

T-Test Results Comparing Means Obtained for Recall

		Pairs					
Dataset	**Criterion**	**C4.5 - RIPPER**	**C4.5 - PART**	**RIPPER - PART**	**C4.5 - OCEL**	**RIPPER - OCEL**	**PART - OCEL**
GHTraffic	t-value	-0.00176	0	0.00176	-16.95517	-16.94109	-16.95517
	p-value	-0.00176	0.5	0.00176	<.00001	<.00001	<.00001
	Significant	No	No	No	Yes	Yes	Yes
Twitter	t-value	0	0	0	7.77081	7.77081	7.77081
	p-value	0.5	0.5	0.5	<.00001	<.00001	<.00001
	Significant	No	No	No	Yes	Yes	Yes
Google Tasks	t-value	0.00441	0	-0.00441	0.35745	0.35479	0.35745
	p-value	0.498256	0.5	0.498256	0.361088	0.362079	0.361088
	Significant	No	No	No	No	No	No
Slack	t-value	0	0	0	-2137.2136	-2137.21356	-2137.2136
	p-value	0.5	0.5	0.5	<.00001	<.00001	<.00001
	Significant	not	No	No	Yes	Yes	Yes

T-Test Results Comparing Means Obtained for Model Size

		Pairs					
Dataset	**Criterion**	**C4.5 - RIPPER**	**C4.5 - PART**	**RIPPER - PART**	**C4.5 - OCEL**	**RIPPER - OCEL**	**PART - OCEL**
GHTraffic	t-value	2.462	1.74405	-1.22665	-1.08149	-8.74435	-5.79662
	p-value	0.007782	0.042143	0.111446	0.140733	<.00001	<.00001
	Significant	Yes	Yes	No	No	Yes	Yes
Twitter	t-value	50.6393	61.51829	22.94374	3.11962	-35.96353	-42.84684
	p-value	<.00001	<.00001	<.00001	0.001049	<.00001	<.00001
	Significant	Yes	Yes	Yes	Yes	Yes	Yes
Google Tasks	t-value	5.96515	3.89812	-2.6	-1.38617	-4.13035	-3.3393
	p-value	<.00001	0.000252	0.007164	0.85749	0.000065	0.000771
	Significant	Yes	Yes	Yes	No	Yes	Yes
Slack	t-value	3.64486	3.24256	-1	3.86725	-0.20093	0.43369
	p-value	0.001326	0.00295	0.167141	0.00048	0.42139	0.334577
	Significant	Yes	Yes	No	Yes	No	No

Appendix D

List of Publications

Part of this thesis was published in the following peer-reviewed conferences:

1. **Bhagya, T.**, Dietrich, J., Guesgen, H., & Versteeg, S. (2018, July). GHTraffic: A dataset for reproducible research in service-oriented computing. In *2018 IEEE International Conference on Web Services (ICWS)* (pp. 123-130). IEEE.

   Presents the GHTraffic dataset described in Chapter 4.

2. **Bhagya, T.**, Dietrich, J., & Guesgen, H. (2019, December). Generating Mock Skeletons for Lightweight Web-Service Testing. In *2019 26th Asia-Pacific Software Engineering Conference (APSEC)* (pp. 181-188). IEEE.

   Presents the experiments on attribute-based learning algorithms and the results described in Chapter 5 and Chapter 6.

In addition, the plan is to submit the following journal article on the basis of some of the other contributions made by this thesis:

3. **Bhagya, T.**, Dietrich, J., & Guesgen, H. Description Logic Class Expression Learning Applied to HTTP Mock Skeletons Generation. *IEEE Transactions on Services Computing.*

   Presents the experiments on OCEL description logic learning algorithm and the results described in Chapter 5 and Chapter 6. There is also a comparison of the description logic learning results with attribute-based learning outcomes reported in Paper No. 2.

Bibliography

[1] E. Christensen et al., “Web services description language (WSDL) 1.1,” 2001, accessed 16 Aug. 2020. [Online]. Available: https://w3.org/TR/wsdl

[2] D. Box et al., “Simple object access protocol (SOAP) 1.1,” 2000, accessed 16 Aug. 2020. [Online]. Available: https://w3.org/TR/2000/NOTE-SOAP-20000508

[3] R. T. Fielding and R. N. Taylor, Architectural styles and the design of network-based software architectures. University of California, Irvine Doctoral dissertation, 2000.

[4] G. Canfora and M. Di Penta, “Testing services and service-centric systems: Challenges and opportunities,” It Professional, 2006.

[5] S. Freeman, T. Mackinnon, N. Pryce, and J. Walnes, “Mock roles, not objects,” in Proc. OOPSLA’04. ACM, 2004.

[6] J. Michelsen and J. English, “What is service virtualization?” in Service Virtualization. Springer, 2012.

[7] ——, Service Virtualization: Reality Is Overrated. Apress, 2012.

[8] “CA Service Virtualization,” accessed 16 Aug. 2020. [Online]. Available: https://ca.com/us/products/ca-service-virtualization.html

[9] “Parasoft Virtualize,” accessed 16 Aug. 2020. [Online]. Available: https://parasoft.com/products/virtualize

[10] “Wiremock,” accessed 16 Aug. 2020. [Online]. Available: http://wiremock.org

[11] “Hoverfly,” accessed 16 Aug. 2020. [Online]. Available: https://hoverfly.io

[12] M. Du, J.-G. Schneider, C. Hine, J. Grundy, and S. Versteeg, “Generating service models by trace subsequence substitution,” in Proc. QoSA’13, 2013.

[13] M. Du, S. Versteeg, J.-G. Schneider, J. Han, and J. Grundy, “Interaction traces mining for efficient system responses generation,” ACM SIGSOFT Software Engineering Notes, 2015.

[14] S. C. Versteeg et al., “Opaque service virtualisation: a practical tool for emulating endpoint systems,” in Proceedings ICSE ’16. ACM, 2016.

[15] S. Versteeg, M. Du, J. Bird, J.-G. Schneider, J. Grundy, and J. Han, "Enhanced playback of automated service emulation models using entropy analysis," in Proc. CSED'16. IEEE, 2016.

[16] M. A. Hossain, S. Versteeg, J. Han, M. A. Kabir, J. Jiang, and J.-G. Schneider, "Mining accurate message formats for service apis," in Proc. SANER'18. IEEE, 2018.

[17] M. A. Hossain, "Discovering context dependent service models for stateful service virtualization," 2020.

[18] H. F. Eniser, A. Sen, and S. O. Polat, "Fancymock: creating virtual services from transactions," in Proc. SAC'18, 2018.

[19] H. F. Enişer and A. Sen, "Testing service oriented architectures using stateful service visualization via machine learning," in Proc. AST'18, 2018.

[20] ——, "Virtualization of stateful services via machine learning," Software Quality Journal, 2019.

[21] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Machine learning: An artificial intelligence approach. Springer Science & Business Media, 2013.

[22] A. Holzinger, "From machine learning to explainable ai," in Proc. DISA'18. IEEE, 2018.

[23] C. Molnar, G. Casalicchio, and B. Bischl, "Interpretable machine learning–a brief history, state-of-the-art and challenges," arXiv preprint arXiv:2010.09337, 2020.

[24] "Mockito," accessed 16 Aug. 2020. [Online]. Available: https://site.mockito.org

[25] W. Lam, P. Godefroid, S. Nath, A. Santhiar, and S. Thummalapenta, "Root causing flaky tests in a large-scale industrial setting," in Proc. ISSTA'19, 2019.

[26] J. R. Quinlan, C4.5: programs for machine learning. Elsevier, 2014.

[27] W. W. Cohen, "Fast effective rule induction," in Machine learning Proc.'95. Elsevier, 1995.

[28] E. Frank and I. H. Witten, "Generating accurate rule sets without global optimization," 1998.

[29] J. Lehmann, Learning OWL class expressions. IOS Press, 2010.

[30] I. H. Witten, E. Frank, and M. A. Hall, Data Mining Practical Machine Learning Tools and Techniques Third Edition. Morgan Kaufmann, 2017.

[31] C. Zhang, C. Liu, X. Zhang, and G. Almpanidis, "An up-to-date comparison of state-of-the-art classification algorithms," Expert Systems with Applications, 2017.

[32] J. Lehmann, S. Auer, L. Bühmann, and S. Tramp, “Class expression learning for ontology engineering,” Journal of Web Semantics, 2011.

[33] C. Collberg and T. A. Proebsting, “Repeatability in computer systems research,” Communications of the ACM, 2016.

[34] R. D. Peng, “Reproducible research in computational science,” Science, 2011.

[35] P. Kruchten, The rational unified process: an introduction. Addison-Wesley Professional, 2004.

[36] A. D. Birrell and B. J. Nelson, “Implementing remote procedure calls,” ACM Transactions on Computer Systems (TOCS), 1984.

[37] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon, “Design and implementation of the sun network filesystem,” in Proc. of the Summer USENIX conference, 1985.

[38] M. Kong, T. H. Dineen, P. J. Leach, E. A. Martin, N. W. Mishkin, J. N. Pato, and G. L. Wyant, Network computing system reference manual. Prentice-Hall, Inc., 1990.

[39] H. W. Lockhart Jr, OSF DCE: guide to developing distributed applications. McGraw-Hill, Inc., 1994.

[40] D. Krafzig, K. Banke, and D. Slama, Enterprise SOA: service-oriented architecture best practices. Prentice Hall Professional, 2005.

[41] S. Vinoski, “Corba: integrating diverse applications within distributed heterogeneous environments,” IEEE Communications magazine, 1997.

[42] R. Sessions, COM and DCOM: Microsoft’s vision for distributed objects. John Wiley & Sons, Inc., 1997.

[43] P. E. Chung, Y. Huang, S. Yajnik, D. Liang, J. C. Shih, C.-Y. Wang, and Y.-M. Wang, “Dcom and corba side by side, step by step, and layer by layer,” C++ Report, 1998.

[44] B. Downing-Troy, Java RMI: remote method invocation. IDG, 1998.

[45] M. Henning, “The rise and fall of corba,” Queue, 2006.

[46] C. Szyperski, D. Gruntz, and S. Murer, Component software: beyond object-oriented programming. Pearson Education, 2002.

[47] O. C. F. RFP, “Common business objects and business object facility,” Tech. Rep., 1996.

[48] V. Matena and M. Hapner, “Enterprise javabeans tm,” Sun Microsystems, 1997.

[49] N. Wang, D. C. Schmidt, and C. O’Ryan, “Overview of the corba component model,” in Component-based software engineering: putting the pieces together, 2001.

[50] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, F. Yergeau et al., “Extensible markup language (xml) 1.0,” 2000.

[51] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, “Web services,” in Web services. Springer, 2004.

[52] T. Bellwood, L. Clément, D. Ehnebuske, A. Hately, M. Hondo, Y. Husband, K. Januszewski, S. Lee, B. McKee, J. Munter et al., “The universal description, discovery and integration (uddi) specification,” Rapport technique, Comit OASIS, 2002.

[53] K. Gottschalk, S. Graham, H. Kreger, and J. Snell, “Introduction to web services architecture,” IBM systems Journal, 2002.

[54] E. Newcomer and G. Lomow, Understanding SOA with Web services. Addison-Wesley, 2005.

[55] M. P. Papazoglou, “Service-oriented computing: Concepts, characteristics and directions,” in Proc. WISE’03. IEEE, 2003.

[56] D. Crockford, “The application/json media type for javascript object notation (json),” RFC 4627, 2006.

[57] O. Alliance, “Osgi service platform,” Core Specification Release, 2007.

[58] R. S. Hall, K. Pauls, S. McCulloch, and D. Savage, “Osgi in action,” Creating Modular Applications in Java, 2011.

[59] J. Thönes, “Microservices,” IEEE software, 2015.

[60] S. Newman, Building microservices: designing fine-grained systems. ” O’Reilly Media, Inc.”, 2015.

[61] H. Haas and A. Brown, “Web services glossary,” W3C Working Group Note (11 February 2004), 2004.

[62] C. M. MacKenzie, K. Laskey, F. McCabe, P. F. Brown, R. Metz, and B. A. Hamilton, “Reference model for service oriented architecture 1.0,” OASIS standard, 2006.

[63] A. Brown, S. Johnston, and K. Kelly, “Using service-oriented architecture and component-based development to build web service applications,” Rational Software Corporation, 2002.

[64] J. Bih, “Service oriented architecture (soa) a new paradigm to implement dynamic e-business solutions,” ubiquity, 2006.

[65] G. J. Myers, C. Sandler, and T. Badgett, The art of software testing. John Wiley & Sons, 2011.

[66] B. Beizer, Software testing techniques. Dreamtech Press, 2003.

[67] R. V. Binder, “Testing object-oriented software: a survey,” Software Testing, Verification and Reliability, 1996.

[68] B. Meyer, Object-oriented software construction. Prentice hall New York, 1988.

[69] C. D. Turner and D. J. Robson, “The state-based testing of object-oriented programs,” in Proc. ICSM’93. IEEE, 1993.

[70] I. Jacobson, Object-oriented software engineering: a use case driven approach. Pearson Education India, 1993.

[71] M. J. Harrold, J. D. McGregor, and K. J. Fitzpatrick, “Incremental testing of object-oriented class structures,” in Proc. ICSE’92, 1992.

[72] P. C. Jorgensen and C. Erickson, “Object-oriented integration testing,” Communications of the ACM, 1994.

[73] G. D. Everett and R. McLeod Jr, Software testing: testing across the entire software development life cycle. John Wiley & Sons, 2007.

[74] J. S. Hartmann and C. Imoberdorf, “System and method for functional testing of distributed, component-based software,” 2003.

[75] K. Beck, Test-driven development: by example. Addison-Wesley Professional, 2003.

[76] ——, Extreme programming explained: embrace change. addison-wesley professional, 2000.

[77] T. Mackinnon, S. Freeman, and P. Craig, “Endo-testing: unit testing with mock objects,” 2000.

[78] “EasyMock,” accessed 16 Aug. 2020. [Online]. Available: http://easymock.org

[79] “Moq,” accessed 16 Aug. 2020. [Online]. Available: https://github.com/moq/moq

[80] “Mocker,” accessed 16 Aug. 2020. [Online]. Available: https://labix.org/mocker

[81] J. Bloomberg, “Web services testing: Beyond soap,” ZapThink LLC, Sep, 2002.

[82] “JMeter,” accessed 16 Aug. 2020. [Online]. Available: https://jmeter.apache.org

[83] “SoapUI,” accessed 16 Aug. 2020. [Online]. Available: https://soapui.org

[84] S. J. Russell and P. Norvig, Artificial intelligence: a modern approach. Pearson Education Limited,, 2016.

[85] A. L. Samuel, “Some studies in machine learning using the game of checkers,” IBM Journal of research and development, 1959.

[86] B. Kosko, Neural networks and fuzzy systems: A dynamical systems approach to machine intelligence., 1992, no. QA76. 76. E95 K86.

[87] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." Psychological review, 1958.

[88] M. Minsky and S. Papert, "Perceptrons: An essay in computational geometry," MIT Press., 1969.

[89] T. Cover and P. Hart, "Nearest neighbor pattern classification," IEEE transactions on information theory, 1967.

[90] P. J. Werbos, "Applications of advances in nonlinear sensitivity analysis," in System modeling and optimization. Springer, 1982.

[91] J. R. Quinlan, "Learning efficient classification procedures and their application to chess end games," in Machine learning. Springer, 1983.

[92] T. K. Ho, "Random decision forests," in Proc. ICDAR'95. IEEE, 1995.

[93] Y. Freund and L. Mason, "The alternating decision tree learning algorithm," in icml, 1999.

[94] L. Breiman, Classification and regression trees. Routledge, 2017.

[95] R. S. Michalski, "A theory and methodology of inductive learning," in Machine learning. Springer, 1983.

[96] R. S. Michalski, I. Mozetic, J. Hong, and N. Lavrac, "The multi-purpose incremental learning system aq15 and its testing application to three medical domains," Proc. AAAI'86, 1986.

[97] P. Clark and R. Boswell, "Rule induction with cn2: Some recent improvements," in European Working Session on Learning. Springer, 1991.

[98] R. C. Holte, "Very simple classification rules perform well on most commonly used datasets," Machine learning, 1993.

[99] R. Kohavi, "The power of decision tables," in European conference on machine learning. Springer, 1995.

[100] J. R. Quinlan, "Learning logical definitions from relations," Machine learning, 1990.

[101] S. Muggleton, C. Feng et al., Efficient induction of logic programs. Citeseer, 1990.

[102] S. Muggleton, "Inductive logic programming," New generation computing, 1991.

[103] S. Muggleton and L. De Raedt, "Inductive logic programming: Theory and methods," The Journal of Logic Programming, 1994.

[104] A. Srinivasan, "A learning engine for proposing hypotheses (aleph)," 1999.

[105] C. Cortes and V. Vapnik, "Support-vector networks," Machine learning, 1995.

[106] F. V. Jensen et al., An introduction to Bayesian networks. UCL press London, 1996.

[107] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg, "Feature hashing for large scale multitask learning," in Proc. ICML'09, 2009.

[108] M. Hoffman, F. R. Bach, and D. M. Blei, "Online learning for latent dirichlet allocation," in advances in neural information processing systems, 2010.

[109] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," Journal of machine learning research, 2011.

[110] L. Breiman, "Random forests," Machine learning, 2001.

[111] S. Menard, Applied logistic regression analysis. Sage, 2002.

[112] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web," Scientific american, 2001.

[113] L. Bühmann, J. Lehmann, and P. Westphal, "Dl-learner a framework for inductive learning on the semantic web," Journal of Web Semantics, 2016.

[114] L. Bühmann, J. Lehmann, P. Westphal, and S. Bin, "Dl-learner structured machine learning on semantic web data," in Proc. WWW '18, 2018.

[115] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," nature, 2015.

[116] I. Bratko, B. Cestnik, and I. Kononenko, "Attribute-based learning," AI Commun., 1996.

[117] H. Boström, "Covering vs. divide-and-conquer for top-down induction of logic programs," in IJCAI, 1995.

[118] J. R. Quinlan, "Generating production rules from decision trees." in ijcai. Citeseer, 1987.

[119] ——, "Induction of decision trees," Machine learning, 1986.

[120] J. Fürnkranz, "Separate-and-conquer rule learning," Artificial Intelligence Review, 1999.

[121] R. L. Rivest, "Learning decision lists," Machine learning, 1987.

[122] J. Fürnkranz and G. Widmer, "Incremental reduced error pruning," in Machine Learning Proceedings 1994. Elsevier, 1994.

[123] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," ACM SIGKDD explorations newsletter.

[124] S.-H. Nienhuys-Cheng and R. De Wolf, Foundations of inductive logic programming. Springer Science & Business Media, 1997.

[125] F. Baader, D. Calvanese, D. McGuinness, P. Patel-Schneider, D. Nardi et al., The description logic handbook: Theory, implementation and applications. Cambridge university press, 2003.

[126] D. L. McGuinness, F. Van Harmelen et al., "Owl web ontology language overview," W3C recommendation, 2004.

[127] B. Motik, P. F. Patel-Schneider, B. Parsia, C. Bock, A. Fokoue, P. Haase, R. Hoekstra, I. Horrocks, A. Ruttenberg, U. Sattler et al., "Owl 2 web ontology language: Structural specification and functional-style syntax," W3C recommendation, 2009.

[128] B. L. Richards and R. J. Mooney, "Automated refinement of first-order horn-clause domain theories," Machine Learning, 1995.

[129] M. Horridge and P. F. Patel-Schneider, "Manchester syntax for owl 1.1." in OWLED (Spring). Citeseer, 2008.

[130] J. Lehmann, "Dl-learner: learning concepts in description logics," Journal of Machine Learning Research, 2009.

[131] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: A practical owl-dl reasoner," Journal of Web Semantics, 2007.

[132] T. Berners-Lee, R. Fielding, and H. Frystyk, "Hypertext transfer protocol–http/1.0," 1996.

[133] R. T. Fielding et al., "Hypertext transfer protocol–HTTP/1.1 (RFC2616)," 1999, accessed 16 Aug. 2020. [Online]. Available: https://tools.ietf.org/html/rfc2616

[134] M. Belshe, R. Peon, and M. Thomson, "Hypertext transfer protocol version 2 (http/2)," 2015.

[135] D. Crocker, "Rfc0822: Standard for the format of arpa internet text messages," 1982.

[136] N. Freed and N. Borenstein, "Multipurpose internet mail extensions (mime) part one: Format of internet message bodies," 1996.

[137] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering–a systematic literature review," Information and software technology, 2009.

[138] M. R. Marri, T. Xie, N. Tillmann, J. De Halleux, and W. Schulte, "An empirical study of testing file-system-dependent software with mock objects," in Proc. AST'09.

[139] "Jmock," accessed 16 Aug. 2020. [Online]. Available: http://jmock.org

[140] S. Mostafa and X. Wang, “An empirical study on the usage of mocking frameworks in software testing,” in Proc. QSIC’14. IEEE, 2014.

[141] N. Tillmann and W. Schulte, “Unit tests reloaded: Parameterized unit testing with symbolic execution,” IEEE software, 2006.

[142] ——, “Parameterized unit tests,” ACM SIGSOFT Software Engineering Notes, 2005.

[143] M. Achenbach and K. Ostermann, “Testing object-oriented programs using dynamic aspects and non-determinism,” in Proc. ECOOP’10, 2010.

[144] N. Tillmann and W. Schulte, “Mock-object generation with behavior,” in Proc. ASE’06. IEEE, 2006.

[145] S. J. Galler, A. Maller, and F. Wotawa, “Automatically extracting mock object behavior from design by contract specification for test data generation,” in Proc. AST’10, 2010.

[146] M. Islam and C. Csallner, “Generating test cases for programs that are coded against interfaces and annotations,” ACM Transactions on Software Engineering and Methodology (TOSEM), 2014.

[147] N. Alshahwan, Y. Jia, K. Lakhotia, G. Fraser, D. Shuler, and P. Tonella, “Automock: Automated synthesis of a mock environment for test case generation,” in Proc. Dagstuhl Seminar. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2010.

[148] A. Arcuri, G. Fraser, and R. Just, “Private api access and functional mocking in automated unit test generation,” in Proc. ICST’17. IEEE, 2017.

[149] H. Samimi, R. Hicks, A. Fogel, and T. Millstein, “Declarative mocking,” in Proc. ISSTA’13, 2013.

[150] F. Solms and L. Marshall, “Contract-based mocking for services-oriented development,” in Proc. SAICSIT’16, 2016.

[151] D. Saff, S. Artzi, J. H. Perkins, and M. D. Ernst, “Automatic test factoring for java,” in Proc. ASE’05, 2005.

[152] S. Joshi and A. Orso, “Scarpe: A technique and tool for selective capture and replay of program executions,” in Proc. ICSM’07. IEEE, 2007.

[153] B. Pasternak, S. Tyszberowicz, and A. Yehudai, “Genutest: a unit test and mock aspect generation tool,” International journal on software tools for technology transfer, 2009.

[154] S. Mani, V. S. Sinha, S. Sinha, P. Dhoolia, D. Mukherjee, and S. Chakraborty, “Efficient testing of service-oriented applications using semantic service stubs,” in Proc. ICWS’09. IEEE, 2009.

[155] N. Ashikhmin, G. Radchenko, and A. Tchernykh, “Raml-based mock service generator for microservice applications testing,” in Russian Supercomputing Days. Springer, 2017.

[156] D. Merkel, “Docker: lightweight linux containers for consistent development and deployment,” Linux journal, 2014.

[157] H. Reza and D. Van Gilst, “A framework for testing restful web services,” in Proc. ITNG’10. IEEE, 2010.

[158] A. Soni, V. Ranga, and S. Jadhav, “Mockresta generic approach for automated mock framework for rest apis generation,” in Inventive Communication and Computational Technologies. Springer, 2020.

[159] G. V. Hattangadi, “A practitioners approach to successfully implementing service virtualization,” Infosys White Paper, 2011.

[160] M. Luke and G. Hodgkinson, “Use service virtualization to remove testing bottlenecks,” IBM White Paper, 2013.

[161] “Key capabilities of a service virtualization solution,” CA Technologies White Paper, 2014.

[162] D. Subbiah, B. Arulmozhi, and H. Maruthamuthu, “Constraint free testing using service virtualization,” International Journal of Computer Applications, 2014.

[163] S. Upadhyay, H. Mukherjee, and A. A. Acharya, “Continuous testing of service-oriented applications using service virtualization.”

[164] J. C. Bezdek and R. J. Hathaway, “Vat: A tool for visual assessment of (cluster) tendency,” in Proc. IJCNN’02. IEEE, 2002.

[165] L. Wang and T. Jiang, “On the complexity of multiple sequence alignment,” Journal of computational biology, 1994.

[166] S. B. Needleman and C. D. Wunsch, “A general method applicable to the search for similarities in the amino acid sequence of two proteins,” Journal of molecular biology, 1970.

[167] A. W. Biermann and J. A. Feldman, “On the synthesis of finite-state machines from samples of their behavior,” IEEE transactions on Computers, 1972.

[168] G. Canfora and M. Di Penta, “Service-oriented architectures testing: A survey,” in Software Engineering. Springer, 2007.

[169] M. Bozkurt, M. Harman, Y. Hassoun et al., “Testing web services: A survey,” Department of Computer Science, Kings College London, Tech. Rep. TR-10-01, 2010.

[170] A. Sharma, T. D. Hellmann, and F. Maurer, “Testing of web services-a systematic mapping,” in Proc. SERVICES’12. IEEE, 2012.

[171] M. Bozkurt, M. Harman, and Y. Hassoun, “Testing and verification in service-oriented architecture: a survey,” Software Testing, Verification and Reliability, 2013.

[172] A. Kumar and M. Singh, "An empirical study on testing of soa based services," International Journal of Information Technology and Computer Science, 2015.

[173] G. Saleem, F. Azam, M. U. Younus, N. Ahmed, and L. Yong, "Quality assurance of web services: A systematic literature review," in Proc. ICCC'16. IEEE, 2016.

[174] I. Ghani, W. M. Wan-Kadir, and A. Mustafa, "Comparative evaluation of state-of-the-art approaches in web service testing," TEST Engineering & Management, 2020.

[175] X. Bai, W. Dong, W.-T. Tsai, and Y. Chen, "Wsdl-based automatic test case generation for web services testing," in Proc. SOSE'05. IEEE, 2005.

[176] E. Martin, S. Basu, and T. Xie, "Automated robustness testing of web services," in Proc. SOAWS'06. Citeseer, 2006.

[177] H. M. Sneed and S. Huang, "Wsdltest-a tool for testing web services," in Proc. WSE'06. IEEE, 2006.

[178] C. Ma, C. Du, T. Zhang, F. Hu, and X. Cai, "Wsdl-based automated test data generation for web service," in Proc. CSSE'08. IEEE, 2008.

[179] S. Hanna and M. Munro, "Fault-based web services testing," in Proc. ITNG'08. IEEE, 2008.

[180] C. Bartolini, A. Bertolino, E. Marchetti, and A. Polini, "Ws-taxi: A wsdl-based testing tool for web services," in Proc. ICST'09. IEEE, 2009.

[181] Y. Li, Z.-a. Sun, and J.-Y. Fang, "Generating an automated test suite by variable strength combinatorial testing for web services," Journal of computing and information technology, 2016.

[182] N. El Ioini, A. Sillitti, and G. Succi, "Using rules for web service client side testing," in Proc. SERVICES'13. IEEE, 2013.

[183] S. K. Chakrabarti and P. Kumar, "Test-the-rest: An approach to testing restful web-services," in Proc. COMPUTATIONWORLD'09. IEEE, 2009.

[184] M. J. Hadley, "Web application description language (wadl)," 2006.

[185] S. K. Chakrabarti and R. Rodriquez, "Connectedness testing of restful web-services," in Proc. ISEC'10, 2010.

[186] H. Ed-Douibi, J. L. C. Izquierdo, and J. Cabot, "Automatic generation of test cases for rest apis: a specification-based approach," in Proc. EDOC'18. IEEE, 2018.

[187] T. Fertig and P. Braun, "Model-driven testing of restful apis," in Proc. WWW'15, 2015.

[188] P. V. P. Pinheiro, A. T. Endo, and A. Simao, "Model-based testing of restful web services using uml protocol state machines," in Proc. SAST'13, 2013.

[189] P. Lamela Seijas, H. Li, and S. Thompson, “Towards property-based testing of restful web services,” in Proceedings of the twelfth ACM SIGPLAN workshop on Erlang, 2013.

[190] S. Karlsson, A. Causevic, and D. Sundmark, “Quickrest: Property-based test generation of openapi-described restful apis,” arXiv preprint arXiv:1912.09686, 2019.

[191] S. Segura, J. A. Parejo, J. Troya, and A. Ruiz-Cortés, “Metamorphic testing of restful web apis,” IEEE Transactions on Software Engineering, 2017.

[192] M. Höschele and A. Zeller, “Mining input grammars from dynamic taints,” in Proc. ASE’16. IEEE, 2016.

[193] V. Atlidakis, P. Godefroid, and M. Polishchuk, “Restler: Stateful rest api fuzzing,” in Proc. ICSE’19. IEEE, 2019.

[194] A. Arcuri, “Restful api automated test case generation with evomaster,” ACM Transactions on Software Engineering and Methodology (TOSEM), 2019.

[195] M. Zhang, B. Marculescu, and A. Arcuri, “Resource-based test case generation for restful web services,” in Proc. GECCO’19, 2019.

[196] V. H. Durelli, R. S. Durelli, S. S. Borges, A. T. Endo, M. M. Eler, D. R. Dias, and M. P. Guimaraes, “Machine learning applied to software testing: A systematic mapping study,” IEEE Transactions on Reliability, 2019.

[197] M. Noorian, E. Bagheri, and W. Du, “Machine learning-based software testing: Towards a classification framework.” in SEKE, 2011.

[198] L. C. Briand, “Novel applications of machine learning in software testing,” in Proc. QSIC’08. IEEE, 2008.

[199] F. Bergadano and D. Gunetti, “Testing by means of inductive program learning,” ACM Transactions on Software Engineering and Methodology, 1996.

[200] J. Sant, A. Souter, and L. Greenwald, “An exploration of statistical models for automated test case generation,” in Proc. WODA’05, 2005.

[201] M. A. Ahmed and I. Hermadi, “Ga-based multiple paths test data generator,” Computers & Operations Research, 2008.

[202] J. Wegener, A. Baresel, and H. Sthamer, “Evolutionary test environment for automatic structural testing,” Information and software technology, 2001.

[203] A. Rosenfeld, O. Kardashov, and O. Zang, “Automation of android applications functional testing using machine learning activities classification,” in Proc. MOBILESoft’18, 2018.

[204] S. Chen, Z. Chen, Z. Zhao, B. Xu, and Y. Feng, “Using semi-supervised clustering to improve regression test selection techniques,” in Proc. ICST’11. IEEE, 2011.

[205] L. C. Briand, Y. Labiche, Z. Bawar, and N. T. Spido, “Using machine learning to refine category-partition test specifications and test suites,” Information and Software Technology, 2009.

[206] L. C. Briand, Y. Labiche, and Z. Bawar, “Using machine learning to refine black-box test specifications and test suites,” in Proc. QSIC’08. IEEE, 2008.

[207] A. von Mayrhauser, C. Anderson, and R. Mraz, “Using a neural network to predict test case effectiveness,” in Proc. AESS’95. IEEE, 1995.

[208] R. Gove and J. Faytong, “Identifying infeasible gui test cases using support vector machines and induced grammars,” in Proc. ICSTW’11. IEEE, 2011.

[209] F. Wang, L.-W. Yao, and J.-H. Wu, “Intelligent test oracle construction for reactive systems without explicit specifications,” in Proc. DASC’11. IEEE, 2011.

[210] D. Agarwal, D. E. Tamir, M. Last, and A. Kandel, “A comparative study of artificial neural networks and info-fuzzy networks as automated oracles in software testing,” IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 2012.

[211] A. Singhal, A. Bansal et al., “Generation of test oracles using neural network and decision tree model,” in Proc. CONFLUENCE’14. IEEE, 2014.

[212] M. Vanmali, M. Last, and A. Kandel, “Using a neural network in the software testing process,” International Journal of Intelligent Systems, 2002.

[213] S. R. Shahamiri, W. M. W. Kadir, and S. Ibrahim, “A single-network ann-based oracle to verify logical software modules,” in Proc. ICSTE’10. IEEE, 2010.

[214] S. R. Shahamiri, W. M. N. W. Kadir, S. Ibrahim, and S. Z. M. Hashim, “An automated framework for software test oracle,” Information and Software Technology, 2011.

[215] S. R. Shahamiri, W. M. Wan-Kadir, S. Ibrahim, and S. Z. M. Hashim, “Artificial neural networks as multi-networks automated test oracle,” Automated Software Engineering, 2012.

[216] F. Gholami, N. Attar, H. Haghighi, M. V. Asl, M. Valueian, and S. Mohamadyari, “A classifier-based test oracle for embedded software,” in Proc. RTEST’18. IEEE, 2018.

[217] A. K. Monsefi, B. Zakeri, S. Samsam, and M. Khashehchi, “Performing software test oracle based on deep neural network with fuzzy inference system,” in Proc. TopHPC’19. Springer, 2019.

[218] F. Hewson, J. Dietrich, and S. Marsland, “Performance regression testing on the java virtual machine using statistical test oracles,” in Proc. ASWEC’15. IEEE, 2015.

[219] U. Kanewala, J. M. Bieman, and A. Ben-Hur, "Predicting metamorphic relations for testing scientific software: a machine learning approach using graph kernels," Software testing, verification and reliability, 2016.

[220] T. J. Cheatham, J. P. Yoo, and N. J. Wahl, "Software testing: a machine learning experiment," in Proc. CSC'95, 1995.

[221] L. C. Briand, Y. Labiche, and X. Liu, "Using machine learning to support debugging with tarantula," in Proc. ISSRE'07. IEEE, 2007.

[222] W. E. Wong and Y. Qi, "Bp neural network-based effective fault localization," International Journal of Software Engineering and Knowledge Engineering, 2009.

[223] P. Godefroid, H. Peleg, and R. Singh, "Learn&fuzz: Machine learning for input fuzzing," in Proc. ASE'17. IEEE, 2017.

[224] M. Z. Nasrabadi, S. Parsa, and A. Kalaee, "Format-aware learn&fuzz: Deep test data generation for efficient fuzzing," arXiv preprint arXiv:1812.09961, 2018.

[225] Y. Wang, Z. Wu, Q. Wei, and Q. Wang, "Neufuzz: Efficient fuzzing with deep neural network," IEEE Access, 2019.

[226] R. Fan and Y. Chang, "Machine learning for black-box fuzzing of network protocols," in Proc. ICICS'17. Springer, 2017.

[227] L. Cheng, Y. Zhang, Y. Zhang, C. Wu, Z. Li, Y. Fu, and H. Li, "Optimizing seed inputs in fuzzing with machine learning," in Proc. ICSE-Companion'19. IEEE, 2019.

[228] Z. Hu, J. Shi, Y. Huang, J. Xiong, and X. Bu, "Ganfuzz: a gan-based industrial network protocol fuzzing framework," in Proc. CF'18, 2018.

[229] C. Cummins, P. Petoumenos, A. Murray, and H. Leather, "Compiler fuzzing through deep learning," in Proc. ISSTA'18, 2018.

[230] M. Sablotny, B. S. Jensen, and C. W. Johnson, "Recurrent neural networks for fuzz testing web browsers," in Proc. ICISC'18. Springer, 2018.

[231] M. Rajpal, W. Blum, and R. Singh, "Not all bytes are equal: Neural byte sieve for fuzzing," arXiv preprint arXiv:1711.04596, 2017.

[232] W. Gong, G. Zhang, and X. Zhou, "Learn to accelerate identifying new test cases in fuzzing," in Proc. SpaCCS'18. Springer, 2017.

[233] S. Krishnamurthi and J. Vitek, "The real software crisis: Repeatability as a core value," Communications of the ACM, 2015.

[234] "SPECweb2009," accessed 16 Aug. 2020. [Online]. Available: https://spec.org/web2009

[235] W. D. Smith, "TPC-W: Benchmarking an ecommerce solution," 2000.

[236] "RUBiS: Rice University Bidding System," 2013, accessed 16 Aug. 2020. [Online]. Available: http://rubis.ow2.org

[237] "DARPA Intrusion Detection Data Sets," 2000, accessed 16 Aug. 2020. [Online]. Available: https://ll.mit.edu/ideval/data

[238] C. Gimnez, A. P. Villegas, and G. Maranon, "CSIC 2010," 2010.

[239] S. M. Blackburn et al., "The DaCapo benchmarks: Java benchmarking development and analysis," in Proc. OOPSLA'06. ACM, 2006.

[240] E. Tempero et al., "The Qualitas Corpus: A curated collection of Java code for empirical studies," in Proc. APSEC'10. IEEE, 2010.

[241] J. Dietrich, H. Schole, L. Sui, and E. Tempero, "XCorpus–An executable Corpus of Java Programs," 2017.

[242] S. Bajracharya et al., "Sourcerer: a search engine for open source code supporting structure-based search," in Proc. OOPSLA '06. ACM, 2006.

[243] A. Frank, "UCI machine learning repository," 2010, accessed 16 Aug. 2020. [Online]. Available: http://archive.ics.uci.edu/ml/index.php

[244] G. Gousios, "The GHTorrent dataset and tool suite," in Proc. MSR'13, 2013.

[245] G. Gousios, M. Pinzger, and A. v. Deursen, "An exploratory study of the pull-based software development model," in Proceedings ICSE'14. ACM, 2014.

[246] B. Vasilescu, V. Filkov, and A. Serebrenik, "StackOverflow and GitHub: Associations between software development and crowdsourced knowledge," in Proceedings SocialCom'13. ASE/IEEE, 2013.

[247] P. P.-S. Chen, "The entity-relationship model–toward a unified view of data," TODS, 1976.

[248] A. Wright, "JSON Schema: A Media Type for Describing JSON Documents," Technical Report. Internet Engineering Task Force, Tech. Rep., 2016.

[249] F. Galiegue and K. Zyp, "JSON Schema: core definitions and terminology draft-zyp-json-schema-04," Working Draft, 2013.

[250] K. Boumillion and J. Levy, "Guava," 2010, accessed 16 Aug. 2020. [Online]. Available: https://github.com/google/guava

[251] "npm," accessed 16 Aug. 2020. [Online]. Available: https://github.com/npm/npm

[252] "Rails," accessed 16 Aug. 2020. [Online]. Available: https://github.com/rails/rails

[253] "The Moby Project," accessed 16 Aug. 2020. [Online]. Available: https://github.com/docker/docker

[254] "The Rust Programming Language," accessed 16 Aug. 2020. [Online]. Available: https://github.com/rust-lang/rust

[255] "AngularJS," accessed 16 Aug. 2020. [Online]. Available: https://github.com/angular/angular.js

[256] "Bootstrap," accessed 16 Aug. 2020. [Online]. Available: https://github.com/twbs/bootstrap

[257] "Kubernetes," accessed 16 Aug. 2020. [Online]. Available: https://github.com/kubernetes/kubernetes

[258] "Homebrew (Legacy)," accessed 16 Aug. 2020. [Online]. Available: https://github.com/Homebrew/homebrew

[259] "Symfony," accessed 16 Aug. 2020. [Online]. Available: https://github.com/symfony/symfony

[260] "Zenodo," accessed 16 Aug. 2020. [Online]. Available: https://zenodo.org

[261] J. M. Zhang, M. Harman, L. Ma, and Y. Liu, "Machine learning testing: Survey, landscapes and horizons," IEEE Transactions on Software Engineering, 2020.

[262] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," Science, 2015.

[263] A. K. Waljee, P. D. Higgins, and A. G. Singal, "A primer on predictive models," Clinical and translational gastroenterology, 2014.

[264] H. Bensusan and A. Kalousis, "Estimating the predictive accuracy of a classifier," in European Conference on Machine Learning. Springer, 2001.

[265] A. A. Freitas, "Comprehensible classification models: a position paper," ACM SIGKDD explorations newsletter, 2014.

[266] J. Huysmans, K. Dejaeger, C. Mues, J. Vanthienen, and B. Baesens, "An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models," Decision Support Systems, 2011.

[267] Z. C. Lipton, "The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery." Queue, 2018.

[268] P. H. Westfall and S. S. Young, Resampling-based multiple testing: Examples and methods for p-value adjustment. John Wiley & Sons, 1993.

[269] G. E. Box, W. H. Hunter, S. Hunter et al., Statistics for experimenters. John Wiley and sons New York.

[270] D. Semenick, "Tests and measurements: The t-test," Strength & Conditioning Journal, 1990.

[271] Q. Shen and J. Faraway, “An f test for linear models with functional responses,” Statistica Sinica, 2004.

[272] J. White, A. Yeats, and G. Skipworth, Tables for statisticians. Nelson Thornes, 1979.

MASSEY UNIVERSITY
TE KUNENGA KI PŪREHUROA
UNIVERSITY OF NEW ZEALAND

GRADUATE RESEARCH SCHOOL

STATEMENT OF CONTRIBUTION DOCTORATE WITH PUBLICATIONS/MANUSCRIPTS

We, the candidate and the candidate's Primary Supervisor, certify that all co-authors have consented to their work being included in the thesis and they have accepted the candidate's contribution as indicated below in the *Statement of Originality*.

Name of candidate:	Thilini Bhagya, Randunu Pathirannehelage
Name/title of Primary Supervisor:	Prof. Hans Guesgen
In which chapter is the manuscript /published work:	Chapter 4, 5 and 6

Please select one of the following three options:

[✓ The manuscript/published work is published or in press]

- Please provide the full reference of the Research Output:
  Bhagya, T., Dietrich, J., Guesgen, H., & Versteeg, S. (2018, July). GHTraffic: A dataset for reproducible research in service-oriented computing. In 2018 IEEE International Conference on Web Services (ICWS) (pp. 123-130). IEEE.
  Bhagya, T., Dietrich, J., & Guesgen, H. (2019, December). Generating Mock Skeletons for Lightweight Web-Service Testing. In 2019 26th Asia-Pacific Software Engineering Conference (APSEC) (pp. 181-188). IEEE.

The manuscript is currently under review for publication – please indicate:

- The name of the journal:
- The percentage of the manuscript/published work that was contributed by the candidate:
- Describe the contribution that the candidate has made to the manuscript/published work:

It is intended that the manuscript will be published, but it has not yet been submitted to a journal

Candidate's Signature:	Thilini Bhagya Digitally signed by Thilini Bhagya Date: 2020.09.14 11:48:33 +12'00'
Date:	14-Sep-2020
Primary Supervisor's Signature:	[illegible]
Date:	14-Sep-2020

This form should appear at the end of each thesis chapter/section/appendix submitted as a manuscript/ publication or collected as an appendix at the end of the thesis.