World Wide Web (2024) 27:63
https://doi.org/10.1007/s11280-024-01302-2

Use of prompt-based learning for code-mixed and code-switched text classification

Pasindu Udawatta[1] · **Indunil Udayangana**[1] · **Chathulanka Gamage**[1] · **Ravi Shekhar**[2] · **Surangika Ranathunga**[3]

Received: 10 April 2024 / Revised: 18 August 2024 / Accepted: 20 August 2024 /
Published online: 9 September 2024

Abstract
Code-mixing and code-switching (CMCS) are prevalent phenomena observed in social media conversations and various other modes of communication. When developing applications such as sentiment analysers and hate-speech detectors that operate on this social media data, CMCS text poses challenges. Recent studies have demonstrated that prompt-based learning of pre-trained language models outperforms full fine-tuning across various tasks. Despite the growing interest in classifying CMCS text, the effectiveness of prompt-based learning for the task remains unexplored. This paper presents an extensive exploration of prompt-based learning for CMCS text classification and the first comprehensive analysis of the impact of the script on classifying CMCS text. Our study reveals that the performance in classifying CMCS text is significantly influenced by the inclusion of multiple scripts and the intensity of code-mixing. In response, we introduce a novel method, *Dynamic+AdapterPrompt*, which employs distinct models for each script, integrated with adapters. While DynamicPrompt captures the script-specific representation of the text, AdapterPrompt emphasizes capturing the task-oriented functionality. Our experiments on Sinhala-English, Kannada-English, and Hindi-English datasets for sentiment classification, hate-speech detection, and humour detection tasks show that our method outperforms strong fine-tuning baselines and basic prompting strategies.

Keywords Code-mixing · Code-switching · Prompt-based learning · Pre-trained language models · XLM-R · Text classification · Language script · Adapters · Sinhala · Kannada · Hindi

1 Introduction

Code-mixing involves borrowing words from one language and incorporating them into another without affecting the context [1, 2]. Code-switching, or language alternation, occurs when individuals alternate between two or more languages within a single conversation or situation [3]. In the context of code-mixed and code-switched (CMCS) text, we distinguish

Extended author information available on the last page of the article

two subtypes: (1) text comprising words that alternate between two languages, and (2) text transitioning from one script to another by substituting letters in a predictable manner, known as Transliteration [4].

Code-mixing and code-switching are intricate phenomena of linguistic behaviour, characterized by the intentional or spontaneous alternation of languages within a single discourse. Another characteristic of CMCS data is lexical borrowing, where words or phrases from one language are used in another. Grammatical hybridity [5], a distinct feature of CMCS, results in blending grammatical structures from different languages. Furthermore, CMCS is influenced by linguistic, social, and cultural constraints, leading to a specific contextual framework.

CMCS is commonly observed in online conversations. A thorough understanding of CMCS data is pivotal for effective communication, advertising, sentiment analysis, and fostering inclusivity across language boundaries. However, the inherent characteristics of CMCS data introduce unique challenges to NLP systems. In particular, the inclusion of multiple scripts and lexical patterns and the potential misidentification of transliterated tokens pose challenges even to modern Natural Language Processing (NLP) systems when processing such text. These challenges are particularly pronounced when working with low-resource languages [6, 7].

In recent years, the domain of NLP has witnessed remarkable advancements, notably propelled by the emergence of pre-trained language models (PLMs) [8, 9]. These PLMs have been trained on extensive datasets, preserving a task-agnostic stance regarding the specific tasks for which they will be later used. To leverage the extensive knowledge embedded in PLMs for diverse NLP tasks, the PLM has to be fine-tuned with task-specific data [10]. This "pre-train and fine-tune" paradigm has been able to activate and harness the comprehensive knowledge within PLMs, leading to very promising results across various downstream tasks such as text classification and named entity recognition [10, 11]. On the negative side, this paradigm faces challenges due to the disparity between pre-training and fine-tuning objectives, leading to inefficiencies in utilizing PLMs across diverse tasks, as they may be unstable in low-resource settings, and less transferable to new tasks after fine-tuning [10–13].

Prompt-based learning has recently been demonstrated to yield promising results compared to full fine-tuning of PLMs for many downstream tasks [13], even in low-resource scenarios [14]. This paradigm involves redefining downstream tasks using textual prompts, encompassing both prompt engineering and answer engineering [11]. In contrast to fine-tuning, prompt-based learning leverages the existing knowledge of PLMs by redefining downstream tasks as pre-training objectives [10, 11, 15]. This removes the need for extensive parameter updates in PLMs, thus preserving their transferability across various tasks. Prompt-based learning has been extended to incorporate pre-trained multilingual language models (PMLMs) as well, enabling experimentation in languages beyond English [16–18].

Existing research on CMCS text classification mainly focuses on the full fine-tuning of PMLMs for downstream tasks [6, 19]. On the other hand, while prompt-based learning has shown success over full fine-tuning for monolingual text, its application to CMCS data has not been explored. Given that prompt-based learning relies on textual prompts, designing effective prompts for CMCS text remains an open question. In other words, a prompt formulated in one language might not be suitable for effectively classifying CMCS data. The absence of multilingual prompts poses a challenge in inducing knowledge from PMLMs effectively, and the potential misidentification of transliterated tokens adds further complexity to accurate classification. These challenges are even more pronounced for low-resource languages. Therefore, addressing these unique challenges is crucial for advancing CMCS text classification through prompt-based learning.

In this study, we focus on prompt-based learning for CMCS text classification. To the best of our knowledge, we believe that we are the first to explore prompt-based learning for CMCS text classification. Therefore, we first delve into the challenges surrounding CMCS text classification and the intricacies introduced by the presence of multiple scripts within a single text. Our experiments unveil that the performance of prompt-based CMCS text classification is influenced by the inclusion of multiple scripts and the intensity of code-mixing.

In response to the aforementioned challenges, we propose a novel methodology named *Dynamic+AdapterPrompt*. This approach employs distinct models for each script to generate script-specific representations by considering the script of the input sentence (DynamicPrompt). Additionally, it effectively captures task-specific representations necessary for the respective CMCS classification tasks through the utilization of adapters (AdapterPrompt). This combined approach leverages the benefits of both adapters and dynamic script considerations.

We have conducted extensive experiments across Sinhala-English, Kannada-English, and Hindi-English datasets, for the tasks of sentiment classification, hate-speech detection, and humour detection. It is noteworthy that Sinhala and Kannada are categorized as low-resource languages [20]. The outcomes demonstrate that our novel approach, Dynamic+AdapterPrompt, outperforms the existing methodologies: full fine-tuning, adapter-based fine-tuning, and conventional prompt-based learning techniques.

To summarize, the key contributions of this paper are as follows:

- We present an extensive study on prompt-based learning for CMCS text classification and the first comprehensive exploration of the impact of the script on CMCS text classification.
- We introduce a novel prompt tuning approach for CMCS text classification termed *Dynamic+AdapterPrompt* that provides script-specific and task-specific representations, to address the intricacies introduced by the inclusion of multiple scripts in CMCS data.

2 Related work

In this section, we delve into three key areas: prompt-based learning, adapter-based fine-tuning of PLMs, and the challenges and advancements in CMCS text classification.

2.1 Prompt-based learning

Until recently, full fine-tuning, also known as vanilla fine-tuning, was the predominant method for adapting PLMs to downstream tasks [13, 21, 22]. In full fine-tuning, all the parameters of the PLM are trained for an underlying downstream task, which demands a significant amount of computational resources. Full fine-tuning also struggles in fully exploiting the linguistic knowledge acquired during pre-training, due to the disparity between the objectives of pre-training and fine-tuning stages [10, 23, 24]. While pre-training typically encompasses self-supervised tasks such as masked language modelling, full fine-tuning has to use task-specific training objectives (e.g. classification, sequence labelling, or generation). Prompt-based learning aims to bridge this gap between pre-training and fine-tuning objectives. In other words, prompt-based learning reformulates downstream tasks to be similar to training objectives used during PLM pre-training [11]. For encoder-based models that use a masked language modelling objective, one such reformulation technique is to convert the downstream task into a cloze-style format, as illustrated in Figure 1.

Prompt-based learning primarily comprises three key components: the prompt, the PLM, and the verbalizer [11]. As depicted in Figure 1, prompt engineering involves the selection of a prompt template for a downstream task. Early research used manually designed human-readable prompts, referred to as manual or discrete prompts [22, 23, 25]. Subsequent studies have shifted focus towards soft prompts, also known as continuous prompts, which are optimized during training for specific downstream tasks [22, 23, 25]. Answer engineering refers to the selection of the verbalizer. The verbalizer is the component that maps the predicted mask token of the PLM into the intended label [26] as illustrated in Figure 1. Verbalizers that are human-readable are denoted as discrete verbalizers, whereas soft verbalizers undergo optimization during the training process. Several studies have explored designing suitable verbalizers for downstream tasks, utilizing both discrete and soft tokens [12, 16, 27]. The primary aim of these studies has been to broaden the coverage of the answer space of the verbalizer for each respective label. The effectiveness of this pipeline is significantly determined by prompt engineering and answer engineering [28].

2.2 Adapter-based fine-tuning of PLMs

Adapters are compact trainable modules that can be integrated into transformer layers. They provide a lightweight fine-tuning alternative to the full fine-tuning approach [29]. Houlsby [29] and Peiffer [30] are the two adapter architectures that are commonly used. The key distinction between these two is that the Houlsby adapter employs two down- and up-projection modules, whereas the Pfeiffer adapter utilizes only one module. Adapters can be generally categorized into two categories: task adapters, which learn task-specific representations, and language adapters, which learn language-specific representations [30]. Typically, language adapters are used in conjunction with task adapters [6, 31]. Extensive research has been conducted on adapters as a parameter-efficient fine-tuning method for various tasks. In Rathnayake et al. [6], Sinhala-English CMCS text classification was performed employing different combinations of adapters, yielding improved results compared to full fine-tuning with minimal parameter updates. Moreover, Rücklć et al. [32] demonstrated the benefits of adapters beyond lightweight fine-tuning. They observed a minimal impact on task performance when adapters were dropped from the lower layers of the PLM.

The application of adapters has proven to be beneficial for prompt-based learning as well. Karimi Mahabadi et al. [15] introduced a few-shot learning method utilizing a masked language modelling objective, and leveraged task-specific adapters as a prompt-free strategy. Their experimental results showcased the effectiveness of this technique in comparison to manual and soft prompts.

Smaller language models face difficulties with soft prompts, as discussed by Shah et al. [33]. Li and Liang [22], Reynolds and McDonell [34] suggest that, as the model size

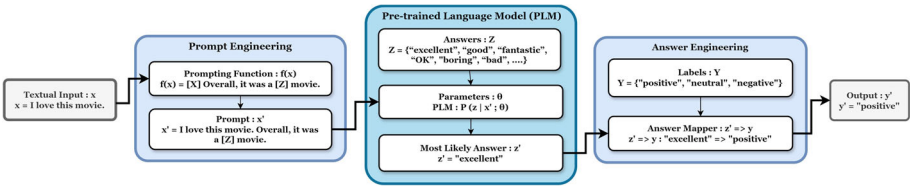

Figure 1 Prompt-Based Learning

increases, the performance gap between prompt-based approaches and fine-tuning narrows, indicating that larger models tend to benefit more from fine-tuning. To enhance smaller language models' effectiveness, Shah et al. [33] suggest using adapters in combination with soft prompts. Their novel approach shows promise in optimizing smaller models, achieving up to 98% of the performance of full fine-tuning.

2.3 CMCS text classification

Classifying CMCS text poses a significant challenge in NLP, largely due to the scarcity of annotated datasets, particularly in the context of low-resource languages. Despite these challenges, studies have been made in developing manually annotated CMCS text classification datasets for low-resource languages [2, 6, 19, 35, 36]. A range of Deep Learning (DL) approaches has been employed for classifying CMCS data. For instance, Chathuranga and Ranathunga [37], Kamble and Joshi [38] utilized techniques such as capsule networks, LSTM, and BiLSTM for CMCS text classification.

Currently, state-of-the-art performance in CMCS text classification is achieved using PMLMs [4, 6, 19, 31, 39–43]. However, Zhang et al. [44] showed that PMLMs are not perfectly code-switching compatible. When no training examples are provided (zero-shot), the observed performance of PLMs for CMCS-related tasks shows that these models are less effective compared to models that have been specifically trained for a task. Additionally, they exhibit limited learning capabilities in few-shot settings. Table 1 provides a summary of different PMLM approaches for CMCS text classification.

3 Datasets

We select three publicly available CMCS datasets. These cover low-resource languages (Sinhala, Kannada), as well as Hindi, a high-resource language [20]. They exhibit different levels of code-mixing and have been annotated for various classification tasks.

The first dataset [6] includes CMCS sentences in Sinhala and English languages. This dataset has been annotated for sentiment classification, humour detection, and hate-speech detection tasks. The second dataset [46] consists of Kannada and English CMCS content and has annotations for sentiment analysis and hate-speech detection. The Hindi-English dataset[1] contains CMCS content written in the Latin script, which has been annotated for the humour detection task. Each language possesses its unique script for writing (Latin for English, Sinhala for Sinhala, Kannada for Kannada, and Devanagari for Hindi).

Altogether, these corpora exhibit six distinct CMCS variations with respect to the script used in training instances, as shown in Table 2. In the first two variants, the text is exclusively composed in one language, employing characters from the same language. Conversely, in the next two variants, the text is written in one language, utilizing characters from a different language. The fifth variant comprises sentences that alternate between languages, with each sentence written in the script corresponding to the language, while the last variant involves sentences that blend elements from two or more of the aforementioned types.

To better analyze the extent of language mixing, we systematically classify sentences in each corpus based on the percentage of characters from each script[2] as outlined in

[1] https://github.com/amsuhane/Humour-Detection-in-English-Hindi-Code-Mixed-Text

[2] Implementation of script-based categorization: https://github.com/Lingua-UoM-CSE/script-identifier

Table 1 Related Work in CMCS Text Classification

Year	Model	Approach / Methodology
2020	mBERT, XLM	A data augmentation framework has been proposed to generate multilingual code-switched data for fine-tuning PMLMs [45].
2021	BERT-based models	Performed full fine-tuning of various transformer models for word-level language identification (WLLI) [43].
2021	mBERT, XLM-R, HME	The effectiveness of PMLMs and meta-embeddings on code-switching tasks has been evaluated [42].
2021	ULMFiT, mBERT, XLM-R, etc.	Conducted full fine-tuning on a custom transliterated dataset, generated using pseudo-labelling, for offensive language identification [4].
2022	BERT, RoBERTa, CharBERT, DistilBERT, XLM-R, etc.	Compared machine learning and deep learning techniques with PLMs [19].
2022	XLM-R	Adapter-based fine-tuning for code-mixed text classification, including methods such as sequential and parallel stacking of adapters. [6].
2023	mBERT, XLM-R, IndicBERT, etc.	Explored the impact of the pre-trained languages when fine-tuning PMLMs with code-mixed data [39].
2023	mBERT, XLM-R, mDeBERTa, etc.	A comprehensive empirical analysis of PMLMs on code-switching tasks, using zero or few-shot prompting and full fine-tuning [44].
2023	BERT-based models	Proposed a language augmentation-based pipelined approach for BERT-based models applied to Hindi-English code-mixed data [40].
2024	XLM-R	Developed an AdapterFusion-based Multi-Task Learning (MTL) model for CMCS text classification [31].
2024	mBERT, XLM-R	Investigated how PLMs handle code-switched text across three dimensions: detection, syntax, and semantics [41].

Algorithm 1. Opting to examine sentences at the character level, as opposed to the word level, enables us to capture finer details of language mixing. In CMCS sentences, particularly in informal communication, individual words can seamlessly blend characters from multiple scripts.

A threshold of 100% is considered, implying that if all characters in a sentence belong to one script, it is categorized under that script; otherwise, it is labelled as a mixed-script sentence. The following examples illustrate this algorithm:

- *Latin Script Example*:

  Sentence: “Mama wibhagaya samath una” (Sinhala written in Latin script)
  Script Label: Latin (100% of characters are in Latin script)

- *[Other] Script Example*:

  Sentence: “මම විභාගය සමත් උනා” (Sinhala written in Sinhala script)
  Script Label: Sinhala (100% of characters are in Sinhala script)

Table 2 Variations of CMCS Data Across Datasets

Type	Script	Si-En	Ka-En	Hi-En	Code-Mixing Intensity
Solely in English language	Latin	✓	✓	✓	0%
Solely in [other] language	[other]	✓	✓		0%
In [other] language using English letters (CM)	Latin	✓	✓	✓	100%
In English language using [other] letters (CM)	[other]	✓	✓		100%
Language alternation between English and [other] (CS)	Mixed	✓	✓		0 - 100%
A blend of two or more of the aforementioned types	Latin, [other], or Mixed	✓	✓	✓	0 - 100%

Note: Si-En, Ka-En, and Hi-En denote Sinhala-English, Kannada-English, and Hindi-English, respectively. Code-mixing intensity measures the extent to which multiple languages are integrated within a single sentence or discourse. The term [other] represents the script of the language combined with English in the CMCS context

Algorithm 1 Instance Classification Based on Script. The term [other] represents the script of the language combined with English in the CMCS context.

```
for all instances in corpus do
    Initialize counters for total characters, Latin script characters, and [other] script characters
    for all tokens in instance do
        if token is tagged as irrelevant or only numbers then
            continue
        end if
        Modify the token by removing symbols and numbers
        for all characters in token do
            if character is [other] then
                Increment [other] character count
            else
                Increment Latin character count
            end if
        end for
        Calculate percentages of Latin and [other] characters
        Classify the token based on percentages
    end for
end for
```

- *Mixed-Script Example*:

  Sentence: “I passed the විභාගය”
  Script Label: Mixed (a combination of Latin and Sinhala script characters)

As shown in Table 2, note that the Hindi-English dataset does not have content in Devanagari script - instead, Hindi words have been written in Latin script. Comprehensive statistics for all three datasets are provided in the Appendix A.

4 Baselines

For our experiments, we use a random baseline (assigning class labels to instances without any predetermined criteria) and a majority/minority class baseline (where classification is based on the majority or minority class), along with three additional baselines associated with PMLMs. As mentioned earlier, only full fine-tuning and adapter-based fine-tuning of PMLMs have been employed for CMCS text classification [6]. Therefore we use these two techniques as our baselines. Basic prompting entails training artificial tokens while maintaining the frozen state of the PMLM. Since prompt-based learning has not been attempted previously for CMCS text classification, we utilize Soft Prompt + Soft Verbalizer as our baseline for prompting.

4.1 Full fine-tuning (Full FT)

We train the PLM by updating all parameters, including the task-dependent sequence classification head added on top, as proposed by Devlin et al. [9]. Throughout this process, the PLM weights are adjusted using task-specific data, which facilitates the learning of task-specific representations. We fine-tune the PLM separately for each downstream task (single-task fine-tuning).

4.2 Adapter-based fine-tuning (A-B FT)

We integrate randomly initialized adapters into the PLM. During the fine-tuning phase, we specifically train the introduced adapter parameters while keeping the original PLM parameters frozen. For each downstream classification task, we train distinct sets of adapters. We experiment with both Houlsby [29] and Pfeiffer [30] adapter architectures.

4.3 Prompt-based learning with soft prompt + soft verbalizer (SP+SV)

We employ soft prompt (SP), which comprises artificial token embeddings, and soft verbalizer (SV), which consists of artificial tokens in label words, with the PLM, as proposed by Hambardzumyan et al. [27]. SP and SV replace traditional discrete tokens with artificial ones. During the training phase, we fine-tune SP and SV while keeping the PLM parameters frozen.

5 Experimental setup

PMLMs excel in CMCS text classification by leveraging their contextual understanding and transfer learning capabilities. Their multilingual proficiency, derived from diverse training datasets, enables effective handling of language variations within the same text. In this study, we utilize the XLM-RoBERTa-base (XLM-R) [8] model as the PMLM for our experiments. This choice is motivated because the XLM-R model has been pre-trained on a range of languages, including the languages considered in our study. Moreover, it proves to be well-suited for our work, particularly within the constraints of a resource-efficient computing infrastructure. For full fine-tuning and adapter-based fine-tuning, we employ the code released by Rathnayake et al. [6]. We implement all the prompt-based learning models using the

OpenPrompt[3] [26] framework, which supports Hugging Face Transformers[4] and is built upon the PyTorch framework[5]. For adapter-based implementations within OpenPrompt, we utilize the Adapter-Transformers[6] library, which is built on Hugging Face Transformers.

The datasets specified in Section 3 are partitioned into training, validation, and testing subsets in a stratified manner, with respective proportions of 80%, 10%, and 10% (statistics are provided in Appendix A). As suggested by Rathnayake et al. [6], we employ Random Oversampling (ROS) to address the class imbalance issue of the hate-speech detection task within the Sinhala-English CMCS dataset. Given the pronounced class imbalance in these datasets, the Macro F1-Score is opted as our primary evaluation metric, as it facilitates a more consistent and reliable comparison.

All models are tested across three different seeds (8, 42, 77), and the average results are reported. The maximum sequence length for the input sentence is set at 128. We conduct each experiment for 20 epochs with a batch size of 32. Early stopping is employed in the experiments with a patience of 5 epochs. An evaluation is conducted at the end of each epoch, and the best-performing model is chosen for testing. We use the Adam optimizer as the gradient optimizer, paired with a linear learning rate scheduler. Additionally, a grid search is conducted for hyperparameter tuning to boost the performance of each model. The optimized hyperparameters for each experiment are delineated in Appendix B. All the experiments are conducted using NVIDIA Tesla P100 GPU machines on the Kaggle[7] platform.

6 Impact of script variation and code-mixing intensity on CMCS text classification

The variations in CMCS data, as outlined in Table 2, underscore the unique properties and characteristics inherent to CMCS data. To better understand the complexities in handling CMCS text, let's consider an illustrative example: "I passed the විභාගය". This sentence illustrates a classic instance of Sinhala-English code-mixing. The word "විභාගය" is a Sinhala term for "examination". When entirely transliterated into the Latin script, it might read: "I passed the wibhagaya".

For a model primarily trained on English data, the term විභාගය might be unfamiliar. Conversely, for a model with extensive Sinhala training, the transliterated version "wibhagaya" might pose confusion.

To explore the influence of scripts on CMCS text classification, we conduct training on baseline models using the training set outlined in Section 5, which encompasses training samples from all scripts. Table 3 illustrates the script-wise results of this experiment. In the Sinhala-English context, despite the Latin script demonstrating the best performance in full fine-tuning, the Sinhala script outperforms the other two scripts in adapter-based fine-tuning and SP+SV. Conversely, in the Kannada-English context, the Latin script yields the highest performance in full fine-tuning and SP+SV, while the Mixed script excels in adapter-based fine-tuning. The sentences in the Kannada script exhibit the lowest results. It is evident that in both CMCS contexts, significant performance variations exist based on the script of the training instance.

[3] https://github.com/thunlp/OpenPrompt

[4] https://github.com/huggingface/transformers

[5] https://pytorch.org

[6] https://github.com/adapter-hub/adapter-transformers

[7] https://www.kaggle.com

Table 3 Results by Script obtained through training using the entire training dataset: Sentiment Classification

	Script	Full FT	A-B FT[1]	SP+SV
Sinhala-English	Latin	54.63	54.08	53.67
	Sinhala	52.47	60.65	63.85
	Mixed	46.82	54.36	55.81
Kannada-English	Latin	48.98	47.89	48.84
	Kannada	31.56	33.04	30.20
	Mixed	40.32	48.14	43.39

Note: The evaluation metric employed is Macro F1
[1]A-B FT stands for Adapter-Based Fine-Tuning

To delve further into the impact of script on CMCS text classification, we create distinct training sets by considering the language script of the training instances. These training sets are employed to train the models, each focusing on a single script, to investigate the impact of the training script. For both the Sinhala-English and Kannada-English datasets, we first create separate training sets based on script type, resulting in distinct portions for each script (e.g., for the Sinhala-English corpus, we have portions for Latin script only, Sinhala script only, and mixed script). The first experiment involves selecting a subset of training data from each script-based portion, ensuring that the size of each subset equals 10% of the total training data.

We then expand our analysis to include a larger subset, comprising 20% of the overall training data for each script category. In this phase, 20% of the training data is selected from the Latin script instances and another 20% from a combined subset of Sinhala/Kannada and mixed script instances, because the sentences in Sinhala/Kannada and mixed scripts individually constitute less than 20% of the overall training data. Each subset is stratified based on the label distribution of the task, ensuring a balanced representation of task labels within each script category. Subsequently, we train the MPLM, utilizing the aforementioned training subsets.

The test dataset utilized in all experiments is as described in Section 5. Note that the Hindi-English dataset, consisting solely of Latin script instances, is excluded from this particular experiment. Also, note that this analysis is focused exclusively on the sentiment classification task.

Table 4 depicts that in the Sinhala-English context, training on Latin-script sentences leads to optimal results across all three baselines. With the 10% training dataset proportion, full fine-tuning shows similar outcomes when trained on either Sinhala-script or mixed-script sentences. With adapter-based fine-tuning, training on mixed-script sentences yields better performance compared to training on Sinhala-script sentences, whereas SP+SV shows better results with Sinhala-script compared to the mixed script. At a 20% training dataset proportion, training with a combination of Sinhala and mixed-script sentences results in lower performance compared to training with Latin-script sentences.

In the Kannada-English context, with the 10% training dataset proportion, training with mixed-script sentences excels over Latin and Kannada scripts in full fine-tuning and adapter-based fine-tuning, while SP+SV is most effective with Latin-script sentences. Training on Kannada-script sentences results in the lowest performance across all baselines. At a 20% training dataset proportion, the patterns between Latin-script and Kannada+mixed-script training resemble those at the 10% level.

Table 4 Script-Based Analysis: Sentiment Classification

	Proportion	Trained Script	Full FT	A-B FT[1]	SP+SV
Sinhala-English	10%	Latin	49.04	52.38	48.60
		Sinhala	43.11	46.51	45.15
		Mixed	43.87	49.78	40.92
	20%	Latin	49.75	51.57	49.68
		Sinhala + Mixed	45.33	48.85	46.68
Kannada-English	10%	Latin	38.14	37.92	41.84
		Kannada	28.08	34.27	33.73
		Mixed	41.77	40.01	40.39
	20%	Latin	41.44	40.66	42.81
		Kannada + Mixed	42.59	41.86	39.92

Note: The evaluation metric employed is Macro F1
[1] A-B FT stands for Adapter-Based Fine-Tuning

The prompt-based learning baseline, SP+SV, reaches its highest performance with Latin-script training in both Sinhala-English and Kannada-English contexts. It can be observed that the performance of full fine-tuning, adapter-based fine-tuning, and SP+SV for both datasets exhibit significant fluctuations based on the training script.

Revisiting our example, when trained on Latin-script sentences, the model might proficiently classify the sentence "I passed the wibhagaya" due to the dominance of Latin content. However, the original sentence, "I passed the විභාගය", which blends Sinhala and Latin scripts, might be more challenging, primarily contingent on the model's familiarity with Sinhala characters.

Based on the discernible findings from the aforementioned experiments, it becomes evident that performance disparities are contingent on the script of the input sentence. Thus, a conclusive inference is drawn: *CMCS text classification performance is significantly influenced by the inclusion of multiple scripts and the degree of code-mixing intensity.*

7 Optimizing prompt-based learning through script-based adaptations

To address the limitations observed when employing soft prompts with small PLMs [33] as mentioned in Section 2.2, we conduct experiments with adapters in the context of prompt-based learning, referred to as *AdapterPrompt*. As elaborated in Section 6, the effectiveness of prompt-based learning for CMCS text classification depends on the script of the input text. To address this dependency, we believe that instead of using the same soft prompt and soft verbalizer for inputs of different scripts, dynamically determining the prompt and verbalizer based on the input script, could enhance the model's capability. To achieve this, we propose *DynamicPrompt*. Finally, we combine DynamicPrompt with adapters forming a fusion of DynamicPrompt and AdapterPrompt, to leverage the strengths of both and present *Dynamic+AdapterPrompt*.

7.1 AdapterPrompt

As mentioned in Section 2.2, when utilizing soft prompts, the effectiveness of small pre-trained language models such as *XLM-R* diminishes, thereby reducing the efficacy of prompt-based learning [33]. To mitigate this, we utilize AdapterPrompt, while preserving the static state of the PLM parameters.

In AdapterPrompt, we integrate adapters with the SP+SV model to classify CMCS data, as depicted in Figure 2. Instead of solely querying the PLM using soft prompts as in the SP+SV approach, we incorporate task adapters into the PLM. This enhancement augments the task-specific representation for the underlying task.

We experiment with the two commonly used adapter architectures, Houlsby [29] and Peiffer [30], integrating both into SP+SV models. Additionally, employing the adapter-dropping technique [32], we progressively remove adapters, starting from higher layers of the PLM as illustrated in Figure 3. This iterative process aims to identify the optimal set of adapters necessary to effectively acquire the task-specific representations for the respective task associated with the adapters.

7.2 DynamicPrompt

Our DynamicPrompt approach consists of separate SP+SV models[8], each optimized for a specific script category. Each model is trained exclusively with sentences from its respective script, yielding script-specific representations. While the PLM remains frozen, all SP+SV models share this common PLM.

The script of the input sentence is programmatically determined by the script identifier, as shown in Figure 4, based on the percentage of characters from each script, as elaborated in Section 3. A threshold of 100% is applied, meaning that if all characters in a sentence belong to one script, it is categorized under that script; otherwise, it is labelled as a mixed-script sentence.

Based on the identified script, the corresponding SP+SV model is selected dynamically. We then concatenate the soft prompt with the input sentence and feed it into the PLM, which predicts the masked token based on the surrounding context. Subsequently, the soft verbalizer maps the predicted answer tokens to the corresponding label using soft answer tokens.

7.3 Dynamic+AdapterPrompt

By combining the aforementioned approaches, we propose a novel prompt-based learning methodology termed *Dynamic+AdapterPrompt*. With the introduction of Dynamic+Adapter Prompt, we train separate SP+SV models, each augmented with adapters, for each script category. Each model is exclusively trained on sentences corresponding to its designated script, thereby providing a script-specific representation for enhanced classification.

The frozen PLM serves as the backbone shared across all the SP+SV models, with adapters being integrated into the PLM to encapsulate task-specific functionality. This strategy effectively capitalizes on the inherent strengths of both DynamicPrompt and AdapterPrompt.

Two architectural variants of Dynamic+AdapterPrompt can be implemented, employing distinct methods for integrating adapters to the PLM. For both variants, separate SP+SV

[8] Each SP+SV model consists of SP and SV components, equivalent to the baseline model described in Section 4.3.

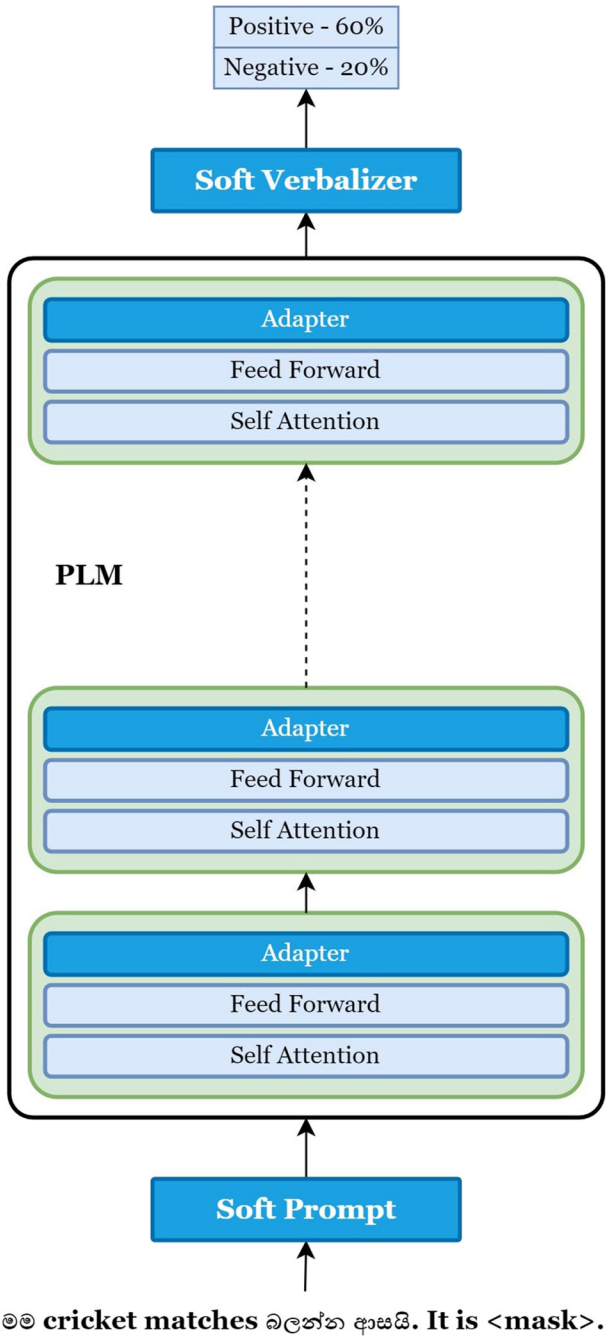

Figure 2 AdapterPrompt without Adapter Dropping

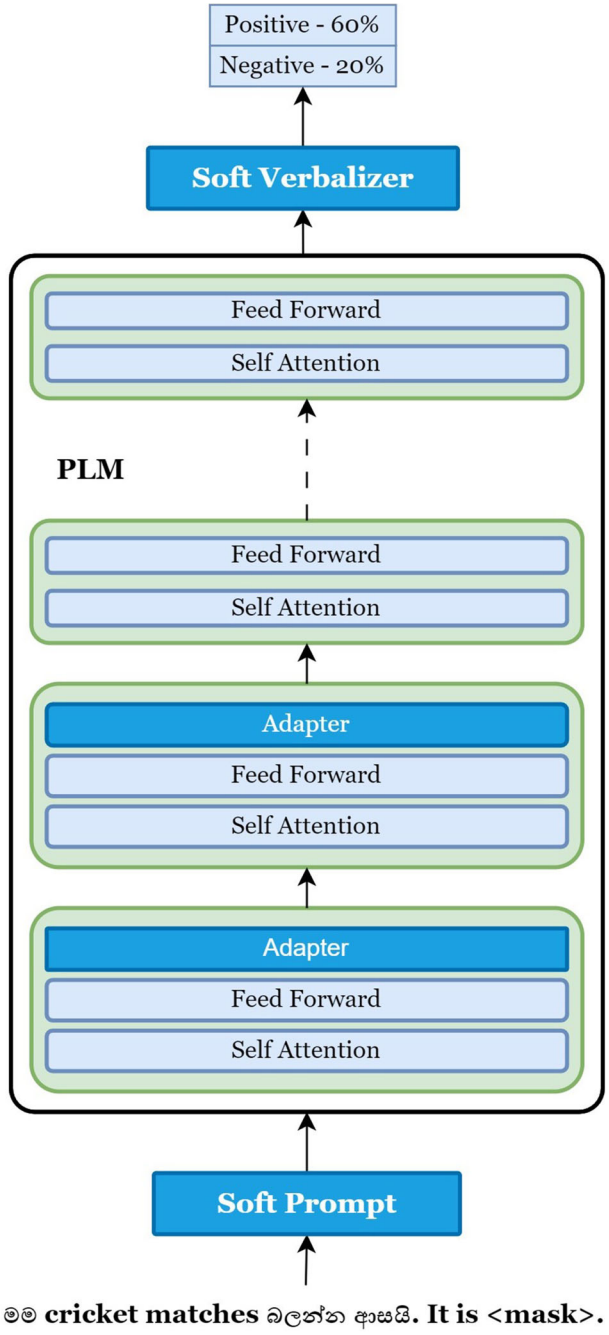

Figure 3 AdapterPrompt with Adapter Dropping

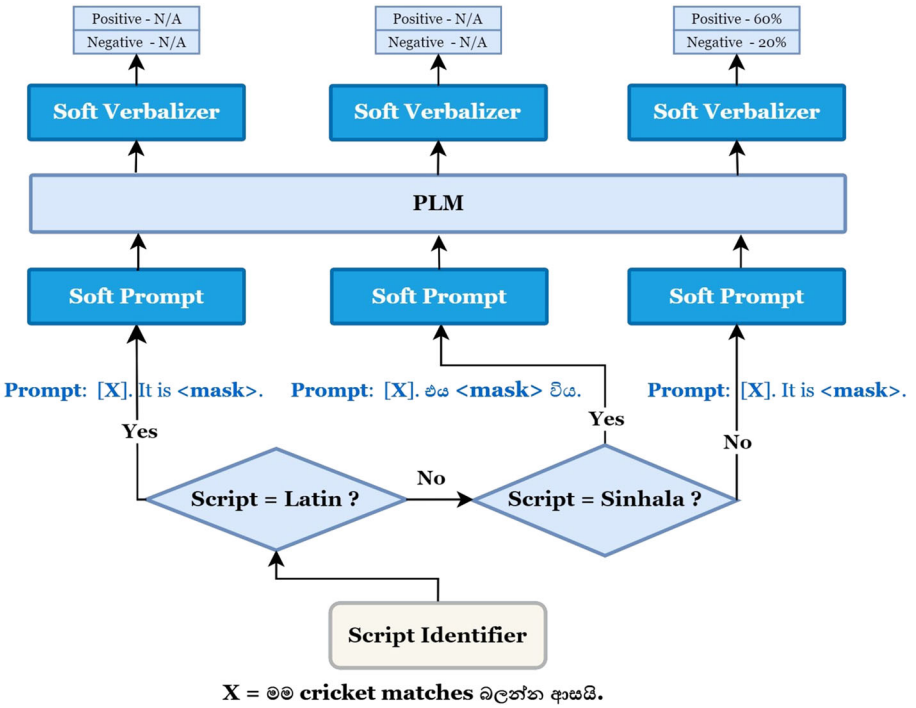

Figure 4 DynamicPrompt Architecture: A Sinhala-English Example. The input sentence translates into English as "I like to watch cricket matches"

models are employed for each script category. These variants are described in the following two sub-sections.

7.3.1 Dynamic+AdapterPrompt with shared adapters setting

As illustrated in Figure 5, we employ adapters that are shared across all SP+SV models. This means that each SP+SV model in Dynamic+AdapterPrompt shares the same set of adapters with the PLM. The goal of this approach is to allow the models to leverage common task-specific functionality through shared adapters while simultaneously benefiting from the script-specific representation provided by the separate SP+SV models.

7.3.2 Dynamic+AdapterPrompt with distinct adapters setting

As depicted in Figure 6, we integrate distinct adapters for each SP+SV model. This approach involves using a common PLM across all SP+SV models, but each SP+SV model employs a separate set of adapters that are not shared among them. When the script of input is identified, the set of adapters relevant to the script is activated along with the corresponding SP+SV model. The objective of this approach is to facilitate fine-tuning and adaptation that are specific to the characteristics of each script category.

In our experiments, we explore the first architectural variant, employing separate SP+SV models for each script category, while applying shared adapters across all models. Subsequently, in an ablation study detailed in Section 8.3, we investigate the second variant to

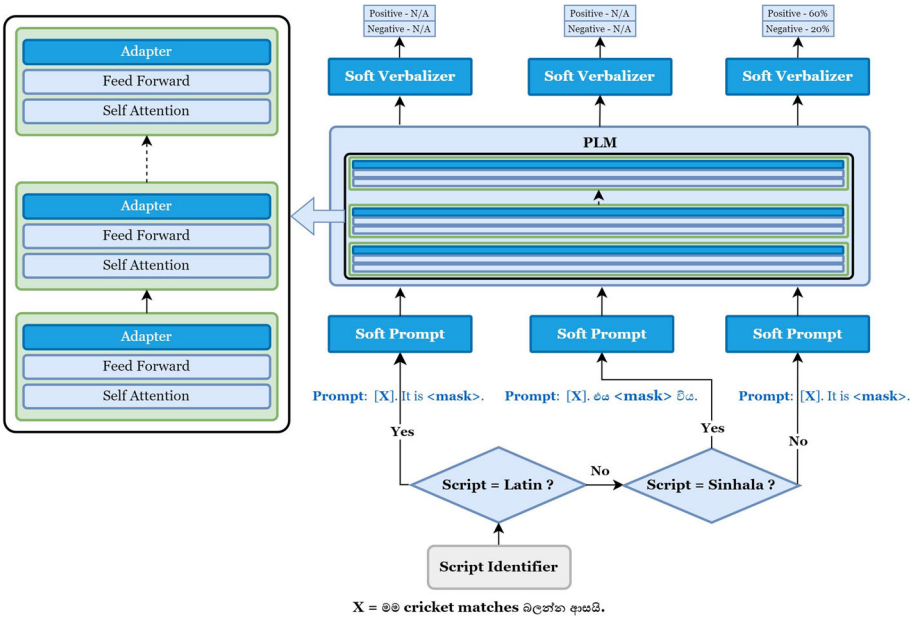

Figure 5 Dynamic+AdapterPrompt Architecture with Shared Adapters Setting: A Sinhala-English Example. The input sentence translates into English as "I like to watch cricket matches"

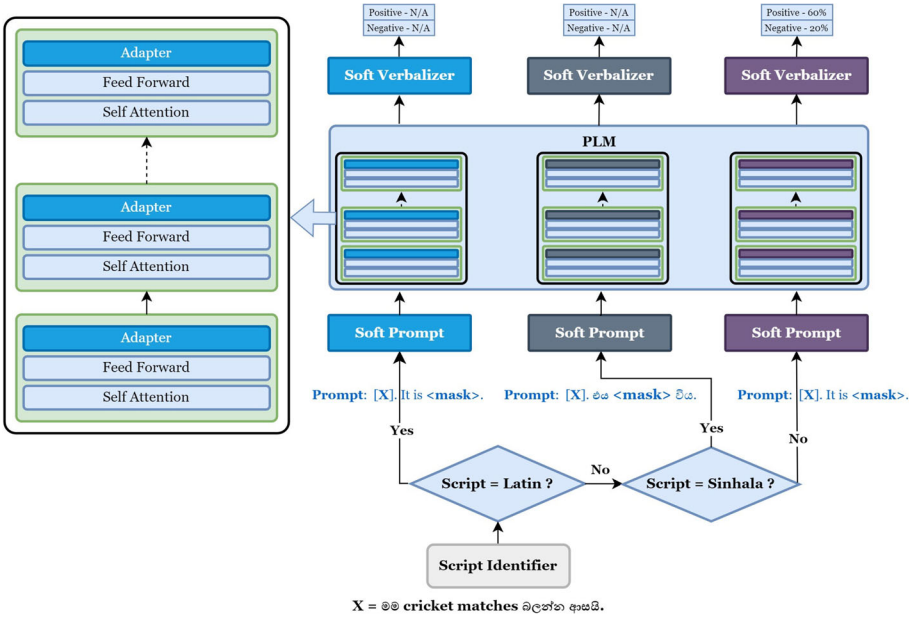

Figure 6 Dynamic+AdapterPrompt Architecture with Distinct Adapters Setting: A Sinhala-English Example. The input sentence translates into English as "I like to watch cricket matches"

compare the effectiveness of both architectures and to determine the impact and viability of such architectural variants. Within these variations, we experiment with the Houlsby adapter architecture.

8 Evaluation and analysis

In this section, we evaluate the effectiveness of the proposed Dynamic+AdapterPrompt approach compared to the baselines. Subsequently, we conduct two ablation studies, with a particular focus on the sentiment classification task in the Sinhala-English context: (1) a script-based analysis to examine the impact of different scripts, and (2) a comparative study to explore the effectiveness of the two adapter integration architectures within Dynamic+AdapterPrompt - shared adapters and script-wise adapters. Following these analyses, an error analysis is conducted to identify common issues related to misclassified sentences within the sentiment classification task for the Sinhala-English context using the Dynamic+AdapterPrompt approach. The training and test datasets utilized in all experiments are as outlined in Section 5. Throughout this section, Ac. denotes Accuracy, while Precision (Pr.), Recall (Re.), and F1 correspond to macro averages.

8.1 Overall evaluation

We first conduct a detailed study to determine the optimal adapter architecture for prompt-based learning in CMCS text classification. In the results for the Sinhala-English sentiment analysis task, as depicted in Appendix C, Houlsby architecture, with activated layers from 0-10, yielded the highest performance in AdapterPrompt. Therefore, the results reported in Tables 5, 6, and 7 are with Houlsby adapter architecture.

In the baseline evaluation presented in Tables 5, 6, and 7, for the majority class, minority class, and random baselines, it is observed that the random baseline outperformed the majority and minority class baselines. However, all three of these baselines exhibit significantly lower performance in comparison to the baselines associated with PLMs.

Notably, the SP+SV approach outperforms the XLM-R full fine-tuning in all cases, except in the Hindi-English context. It achieves superior or competitive results compared

Table 5 Overall Results: Sentiment Classification

	Sinhala-English				Kannada-English			
	Ac.	Pr.	Re.	F1	Ac.	Pr.	Re.	F1
Majority Class	63.61	15.90	25.00	19.44	45.33	9.07	20.00	12.48
Minority Class	0.89	0.22	25.00	0.44	9.34	1.87	20.00	3.42
Random	48.47	24.57	24.54	24.55	26.97	18.53	18.59	18.52
Full Fine-Tuning	78.40	54.29	52.65	53.12	71.05	71.99	70.93	48.14
Adapter-Based FT	78.80	55.41	55.21	55.12	62.15	50.31	48.17	48.62
SP+SV	77.98	57.20	54.85	55.71	62.94	51.23	49.21	48.97
AdapterPrompt	78.85	59.59	62.31	**60.67**	61.08	50.01	51.10	**50.16**
DynamicPrompt	77.25	54.73	53.48	53.97	62.04	49.35	48.23	47.85
Dynamic+AdapterPrompt	78.66	61.70	58.64	59.76	64.43	51.76	48.69	49.11

Table 6 Overall Results: Hate-Speech Detection

	Sinhala-English				Kannada-English			
	Ac.	Pr.	Re.	F1	Ac.	Pr.	Re.	F1
Majority Class	90.68	30.23	33.33	31.70	73.84	36.92	50.00	42.47
Minority Class	2.59	0.86	33.33	1.68	26.16	13.08	50.00	20.74
Random	82.27	31.43	31.37	31.40	61.29	49.07	49.17	49.08
Full Fine-Tuning	92.50	59.87	52.66	55.09	78.68	73.38	72.07	72.03
Adapter-Based FT	92.01	70.24	61.96	65.44	77.90	71.72	72.28	71.88
SP+SV	89.94	60.18	56.46	57.60	80.82	75.68	71.98	73.40
AdapterPrompt	93.20	73.15	60.30	64.62	80.88	75.83	72.24	73.58
DynamicPrompt	89.37	56.50	52.70	54.30	77.60	70.90	69.28	69.94
Dynamic+AdapterPrompt	92.16	66.68	66.12	**65.93**	80.71	75.21	72.71	**73.72**

to adapter-based fine-tuning, except for hate-speech detection in the Sinhala-English context. The enhanced performance of SP+SV over full fine-tuning can be traced back to the substantial discrepancy in objectives during the pre-training and fine-tuning phases within the fine-tuning paradigm, which hinders the full exploitation of knowledge within PLMs, as we discussed in Section 2.1.

AdapterPrompt consistently demonstrates superiority over the baseline results in sentiment classification, hate-speech detection, and humour detection as shown in Tables 5, 6, and 7 across all language contexts, except hate-speech detection in the Sinhala-English context. This observation is aligned with previous research that highlights the effectiveness of integrating adapters into the PLM within the fine-tuning paradigm for improving CMCS text classification [6]. Importantly, our findings reiterate this trend, emphasizing that even within the prompt-based learning paradigm, the integration of adapters into the PLM results in performance improvements. This strategic use of adapters significantly enhanced the SP+SV approach, showcasing a substantial improvement in the model's understanding of specific task intricacies by providing task-specific representations.

Table 7 Overall Results: Humour Detection

	Sinhala-English				Hindi-English			
	Ac.	Pr.	Re.	F1	Ac.	Pr.	Re.	F1
Majority Class	90.31	45.16	50.00	47.45	50.88	25.44	50.00	33.72
Minority Class	9.69	4.84	50.00	8.83	49.12	24.56	50.00	32.94
Random	82.79	49.99	50.04	50.01	50.10	50.07	50.06	50.04
Full Fine-Tuning	93.07	87.09	68.45	73.85	71.05	71.99	70.93	70.68
Adapter-Based FT	92.70	80.52	73.93	76.67	71.05	71.73	70.89	70.73
SP+SV	92.26	78.97	73.00	75.41	69.69	70.27	69.81	69.53
AdapterPrompt	93.05	82.48	73.78	77.18	72.22	72.47	72.18	**72.11**
DynamicPrompt	91.89	78.61	68.25	71.87	–	–	–	–
Dynamic+AdapterPrompt	92.01	77.14	78.43	**77.76**	–	–	–	–

DynamicPrompt exhibits lower performance compared to the baselines in all language contexts, except for sentiment classification in the Sinhala-English context. Notably, DynamicPrompt yields inferior results compared to AdapterPrompt across all tasks and language contexts. In Section 8.2, we delve into an in-depth analysis of the models' performance across different script categories.

Despite the lower performance observed with DynamicPrompt, the Dynamic+AdapterPrompt approach outperforms the results of DynamicPrompt and AdapterPrompt in the majority of tasks across all language contexts (except for sentiment classification). This improvement can be attributed to the adapters' ability to learn task-specific representations, while the soft prompt and soft verbalizer within each model acquire script-specific knowledge for classification. This underscores the proficiency of the combined approach in adeptly addressing challenges intrinsic to both script and task across various language combinations. Note that in the context of humour detection in Hindi-English, the DynamicPrompt and Dynamic+AdapterPrompt techniques are not explored, primarily because the dataset is entirely in the Latin script.

In summary, the SP+SV approach demonstrates superior or competitive results compared to XLM-R full fine-tuning and adapter-based fine-tuning for most tasks, with only a few exceptions. AdapterPrompt consistently outperforms baseline results, showcasing the effectiveness of integrating adapters into the PLM within the prompt-based learning paradigm. DynamicPrompt alone exhibits lower overall performance. However, the combination of DynamicPrompt and AdapterPrompt, Dynamic+AdapterPrompt, emerges as the most effective strategy. This underscores the benefits of leveraging both script-based prompts and adapters to address the intricacies of script and task variations in CMCS text classification.

8.2 Script-based analysis

Section 6 unveiled a significant variance in performance, particularly in the context of prompt-based learning, depending on the script of the input sentence. In this ablation study, we further analyze the impact of the script, employing the sentiment classification task in the Sinhala-English context as a case study.

Table 8 presents the script-wise results of the sentiment classification task for Sinhala-English, further highlighting the discrepancy introduced by the inclusion of multiple scripts. Despite the integration of adapters into SP+SV (AdapterPrompt), this variance persists. This can be ascribed to the adapters providing only task-specific representations, which may not fully rectify the script-related disparities. However, it is noteworthy that AdapterPrompt has demonstrated notable effectiveness when the input is in a single script.

Although DynamicPrompt has a relatively lower performance, it has reduced the script's influence on CMCS text classification, as demonstrated in Table 8. This method is effective because DynamicPrompt provides script-specific representations, making it more robust against script variations compared to fine-tuning, SP+SV, and AdapterPrompt. This underscores its contribution to addressing challenges related to including multiple scripts in CMCS text classification.

When considering Dynamic+AdapterPrompt, the outcomes indicate that the integration of adapters led to a variance in performance, similar to the observations in AdapterPrompt. However, it narrows the gap between Latin and Mixed scripts compared to SP+SV and AdapterPrompt, due to the script-specific representations provided by DynamicPrompt.

Table 8 Results by Script: Sentiment Classification for Sinhala-English

Method	Script	Ac.	Pr.	Re.	F1
Full FT	Latin	79.18	56.00	54.18	54.63
	Sinhala	76.27	53.15	52.91	52.47
	Mixed	76.41	48.74	46.04	46.28
Adapter-Based FT	Latin	79.18	54.01	54.38	54.08
	Sinhala	76.63	67.13	57.89	60.65
	Mixed	78.80	55.22	54.72	54.36
SP+SV	Latin	78.19	53.69	53.69	53.67
	Sinhala	76.08	74.37	58.84	63.85
	Mixed	78.63	56.89	55.24	55.81
AdapterPrompt	Latin	78.67	60.31	60.37	**60.05**
	Sinhala	79.89	75.31	68.64	**71.00**
	Mixed	76.58	56.81	55.45	55.82
DynamicPrompt	Latin	78.15	53.45	53.00	53.19
	Sinhala	75.91	57.57	53.46	54.31
	Mixed	74.01	53.77	53.07	52.80
Dynamic+AdapterPrompt	Latin	79.08	58.45	62.38	59.89
	Sinhala	76.81	81.19	65.24	70.03
	Mixed	77.43	66.89	58.29	**60.51**

In conclusion, DynamicPrompt helps reduce script variations, and AdapterPrompt enhances performance with task-specific representations. The combination, Dynamic+Adapter Prompt, achieves even better results by leveraging the strengths of both approaches.

8.3 Evaluating the efficacy of the architecture in Dynamic+AdapterPrompt

In Section 7.3, we employ the shared adapter architecture; however, it is crucial to note that the script-wise adapter architecture remains a viable alternative. To explore the comparative effectiveness of these two adapter integration architectures, we conduct an ablation study, and the results are presented in Table 9. The superiority observed in the shared adapter architecture can be attributed to its capacity to develop a uniform task representation across all scripts. By sharing adapters across all SP+SV models, the shared adapters in the Dynamic+AdapterPrompt model can benefit from the knowledge obtained from the entire training dataset. Conversely, script-wise adapters are confined to learning solely from the samples of a specific script with which they are associated, resulting in a more circumscribed and script-dependent understanding. Consequently, in the Dynamic+AdapterPrompt, the shared adapter architecture enables the model to leverage a more extensive spectrum of training data, leading to a performance that is markedly superior to that of the script-wise adapter architecture.

Table 9 Dynamic+AdapterPrompt Results by Architecture: Sentiment Classification for Sinhala-English

Architecture	Ac.	Pr.	Re.	F1
Script-wise Adapters	78.88	60.92	58.02	58.86
Shared Adapters	78.66	61.70	58.64	**59.76**

8.4 Error analysis

To identify issues related to misclassified sentences, we conduct an error analysis on the Sinhala-English dataset in the sentiment classification task. For this analysis, we employ the Dynamic+AdapterPrompt approach for handling the intricacies introduced by the inclusion of multiple scripts in CMCS data. Based on our analysis, we have identified the following issues with misclassified sentences:

1. *Context-specific sentiment in an input sentence*: An input sentence may convey sentiment in a specific context that is not apparent to the model. For example, the sentence "Matews kiyanne poll buruwek" was labelled as Negative but predicted as Neutral. This is a Sinhala sentence written in Latin script, where "poll buruwek" means "idiot." The model may not understand this context, especially since the word "poll" has an entirely different meaning in English.
2. *Words highlighting polarity*: Words that emphasize the positive or negative polarity of a sentence can impact the prediction. For example, the sentence "Nidokin ara dilshan ain wena ekamy hoda..." was labelled as Negative but predicted as Positive. The word "hoda", a Sinhala term written in Latin script meaning "good", highlights the positive sentiment of the sentence.

9 Conclusion and future work

In this paper, we explore the potential of prompt-based learning for CMCS text classification, providing a thorough investigation into the impact of the script on classifying CMCS text. Our comprehensive experiments reveal that the effectiveness of prompt-based CMCS text classification is significantly affected by the inclusion of multiple scripts and the intensity of code-mixing. In light of these findings, we introduce a novel prompt-based tuning method named *Dynamic+AdapterPrompt*. We employ separate models for each script category, integrated with adapters to encapsulate the script-specific representation and the task-oriented functionality of CMCS text. The experimental results prove that our proposed method outperforms strong baselines across various CMCS contexts and text classification tasks. This underscores its robustness and efficiency in classifying CMCS text, particularly involving low-resource languages.

Our proposed approach, Dynamic+AdapterPrompt, is suitable for CMCS text where the scripts are distinguishable from each other (such as Sinhala and English). Replacing or adding script identification with language identification will be effective for CMCS contexts, where scripts are similar (such as in German and English). We leave that for future work. Additionally, our approach requires balanced data for each script in the dataset to yield optimal results. However, balanced datasets, especially in low-resource settings, are typically

challenging to obtain, potentially limiting the advancement of CMCS classification using this approach.

As part of our future work, we intend to delve into the application of multi-task learning for code-mixed text classification, leveraging prompt-based learning techniques. Future research could also validate the generalizability of Dynamic+AdapterPrompt by incorporating a wider range of language scripts and classification tasks. Additionally, exploring adapter fine-tuning variants, such as LoRA (Low-Ranked Adapters), in conjunction with the proposed approach could provide valuable insights. We have released our code to facilitate future research[9].

Appendix A: Dataset statistics

By stratified partitioning, the datasets described in Section 3 are divided into training, validation, and testing subsets, with corresponding allocations of 80%, 10%, and 10%, respectively, as depicted in Tables 10, 11, and 12.

Table 10 Label Distribution: Sinhala-English CMCS Dataset

Task	Label	Train	Val	Test	Total
Sentiment Classification	Positive	924	116	116	1,156
	Negative	2,913	364	364	3,641
	Neutral	6,877	860	860	8,597
	Conflict	100	12	12	124
	Latin	7,640	974	983	9,597
	Sinhala	1,544	190	189	1,923
	Mixed	1,630	188	178	1,996
Hate-Speech Detection	Hate-inducing	278	35	35	348
	Abusive	726	91	91	908
	Not offensive	9,810	1,226	1,226	12,262
	Latin	7,655	971	971	9,597
	Sinhala	1,566	175	199	1,940
	Mixed	1,591	206	182	1,979
Humour Detection	Humorous	1,044	130	131	1,305
	Non-humorous	9,770	1,222	1,221	12,213
	Latin	7,697	960	940	9,597
	Sinhala	1,535	184	204	1,923
	Mixed	1,581	207	208	1,996

[9] https://github.com/Lingua-UoM-CSE

Table 11 Label Distribution: Kannada-English CMCS Dataset

Task	Label	Train	Val	Test	Total
Sentiment Classification	Positive	2,633	329	329	3,291
	Negative	1,185	148	148	1,481
	Neutral	656	82	82	820
	Conflict	542	68	68	678
	Latin	4,232	547	528	5,307
	Kannada	818	93	102	1,013
	Mixed	767	88	98	953
Hate-Speech Detection	Not offensive	3,296	412	413	4,121
	Offensive	1,163	146	145	1,454
	Latin	2,886	375	362	3,623
	Kannada	806	93	107	1,006
	Mixed	767	90	89	946

Table 12 Label Distribution: Hindi-English CMCS Dataset

Task	Label	Train	Val	Test	Total
Humour Detection	0	1,345	168	168	1,681
	1	1,389	174	174	1,737
	Latin	2,734	342	342	3,418

Appendix B: Hyperparameters

Most of the hyperparameters used are set to the default values from Hugging Face Transformers, with other hyperparameters chosen based on practical considerations. Tables 13, 14, and 15 present the optimized hyperparameters for tasks in the Sinhala-English, Hindi-English, and Kannada-English CMCS contexts, respectively.

- *Batch Size and Maximum Length*: The maximum sequence length and batch size are selected to fit the capabilities of the NVIDIA Tesla P100 GPU machines.
- *Training Steps*: Experimental results indicate that 20 epochs of training are sufficient for the model to converge.
- *Learning Rate and Warmup Steps*: The learning rate and warmup steps are obtained through the grid search hyperparameter optimization technique to achieve the best Macro-F1 score on the validation dataset.

Table 13 Hyperparameters: Sinhala-English CMCS Experiments

Experiment			Sentiment Classification		Humour Detection		Hate-Speech Detection	
			Learning Rate	Warmup Steps	Learning Rate	Warmup Steps	Learning Rate	Warmup Steps
Full FT	PLM		5e-5	0	5e-5	0	5e-5	0
Adapter-Based FT	Adapter		5e-4	0	5e-4	0	5e-4	0
SP+SV	SP		1e-2	1250	1e-3	1250	1e-2	500
	SV		2e-2 | 2e-2	1250	4e-5 | 4e-4	1250	2e-2 | 2e-2	500
DynamicPrompt	Latin	SP	1e-3	1000	1e-3	1000	1e-3	1000
		SV	4e-5 | 4e-4	1000	4e-5| 4e-4	1000	4e-5| 4e-4	1000
	Sinhala	SP	1e-2	1000	1e-2	1000	1e-2	1000
		SV	2e-2 | 2e-2	1000	2e-2 | 2e-2	1000	2e-2 | 2e-2	1000
	Mixed	SP	1e-2	1000	1e-2	1000	1e-2	1000
		SV	2e-2 | 2e-2	1000	2e-2 | 2e-2	1000	2e-2 | 2e-2	1000
AdapterPrompt	SP		1e-2	1250	1e-3	1250	1e-2	1250
	SV		2e-2 | 2e-2	1250	4e-5| 4e-4	1250	2e-2 | 2e-2	1250
	Adapter		5e-4	1000	5e-4	1000	5e-4	1000
Dynamic+AdapterPrompt	Latin	SP	1e-3	250	1e-3	1250	1e-3	250
		SV	4e-5 | 4e-4	1250	4e-5 | 4e-4	1250	4e-5 | 4e-4	250
	Sinhala	SP	1e-2	1500	1e-2	1250	1e-2	750
		SV	2e-2 | 2e-2	1250	2e-2 | 2e-2	1250	2e-2 | 2e-2	750
	Mixed	SP	1e-2	1250	1e-2	1250	1e-2	750
		SV	2e-2 | 2e-2	1250	2e-2 | 2e-2	1250	2e-2 | 2e-2	750
	Adapters		5e-4	500	5e-4	500	5e-4	500

Table 14 Hyperparameters: Kannada-English CMCS Experiments

Experiment			Sentiment Classification		Hate-Speech Detection	
			Learning Rate	Warmup Steps	Learning Rate	Warmup Steps
Full FT	PLM		5e-5	0	5e-5	0
Adapter-Based FT	Adapter		5e-4	0	5e-4	0
SP+SV	SP		1e-2	1250	1e-2	500
	SV		2e-2 | 2e-2	1250	2e-2 | 2e-2	500
DynamicPrompt	Latin	SP	1e-3	1000	1e-3	1000
		SV	4e-5 | 4e-4	1000	4e-5| 4e-4	1000
	Sinhala	SP	1e-2	1000	1e-2	1000
		SV	2e-2 | 2e-2	1000	2e-2 | 2e-2	1000
	Mixed	SP	1e-2	1000	1e-2	1000
		SV	2e-2 | 2e-2	1000	2e-2 | 2e-2	1000
AdapterPrompt	SP		1e-2	1250	1e-2	1500
	SV		2e-2 | 2e-2	1250	2e-2 | 2e-2	1500
	Adapter		5e-4	1000	5e-4	1000
Dynamic+AdapterPrompt	Latin	SP	1e-3	250	1e-3	250
		SV	4e-5 | 4e-4	1250	4e-5 | 4e-4	1250
	Sinhala	SP	1e-2	1500	1e-2	1500
		SV	2e-2 | 2e-2	1250	2e-2 | 2e-2	1250
	Mixed	SP	1e-2	1250	1e-2	1250
		SV	2e-2 | 2e-2	1250	2e-2 | 2e-2	1250
	Adapters		5e-4	500	5e-4	500

Table 15 Hyperparameters: Hindi-English CMCS Experiments

Experiment		Humour Detection	
		Learning Rate	Warmup Steps
Full FT	PLM	5e-5	0
Adapter-Based FT	Adapter	5e-4	0
SP+SV	SP	1e-2	1250
	SV	2e-2 | 2e-2	1250
AdapterPrompt	SP	1e-3	1250
	SV	2e-2 | 2e-2	1500
	Adapter	5e-4	1000

Appendix C: AdapterPrompt performance

Table 16 illustrates the results of the comprehensive study aimed at identifying the optimal adapter architecture for prompt-based learning in CMCS text classification, with a specific emphasis on the sentiment classification task in the Sinhala-English context.

Table 16 AdapterPrompt Results by Layers: Sentiment Classification for Sinhala-English

Adapter Architecture	Layers	Ac.	Pr.	Re.	F1
Houlsby	0	77.93	54.67	55.87	55.19
	0,1	77.81	53.60	55.20	54.21
	0,1,2	78.06	54.02	56.19	54.83
	0,1,2,3	78.23	57.50	52.54	53.64
	0,1,2,3,4	80.30	59.93	56.73	57.86
	0,1,2,3,4.5	78.50	57.29	57.27	56.75
	0,1,2,3,4.5,6	78.99	60.52	58.31	57.97
	0,1,2,3,4.5,6.7	78.40	57.14	55.91	55.94
	0,1,2,3,4.5,6.7,8	79.56	65.46	56.49	59.10
	0,1,2,3,4.5,6.7,8,9	79.39	66.95	60.44	59.88
	0,1,2,3,4.5,6.7,8,9,10	78.85	59.59	62.31	**60.67**
	All	79.56	59.54	56.84	57.78
Pfeiffer	0	79.36	54.94	55.67	55.29
	0,1	78.21	59.00	54.41	55.34
	0,1,2	77.74	53.77	57.30	55.19
	0,1,2,3	78.95	58.58	56.08	**56.49**
	0,1,2,3,4	75.86	54.85	53.08	52.36
	0,1,2,3,4,5	78.38	55.68	53.38	54.26
	0,1,2,3,4.5,6	79.31	55.93	56.73	56.13
	0,1,2,3,4.5,6.7	78.25	54.47	56.51	55.22
	0,1,2,3,4.5,6.7,8	78.03	53.50	55.87	54.30
	0,1,2,3,4.5,6.7,8,9	78.35	55.27	53.27	54.10
	0,1,2,3,4.5,6.7,8,9,10	78.92	60.07	55.00	56.42
	All	78.57	65.80	55.46	56.44

Note: Ac. denotes Accuracy, while Precision (Pr.), Recall (Re.), and F1 correspond to macro averages

Author Contributions P.U., I.U., and C.G. contributed to the literature review, dataset analysis, and visualization, conceptualization and methodology, system implementation and evaluation, manuscript preparation, and editing. R.S. was involved in conceptualization, supervision, and manuscript reviewing. S.R. was involved in conceptualization, research administration, supervision, manuscript reviewing, and editing.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions. No external funding was received for this research.

Data Availability The data, which comprises existing open datasets, can be accessed through the corresponding URLs and references provided in the paper.

Code Availability The code is available on the GitHub repository.

Declarations

Competing interests The authors declare no competing interests.

Consent for Publication We confirm that this work is original and has not been published elsewhere, nor is it currently under consideration for publication elsewhere. All authors consent to the publication of this manuscript.

Ethical Approval and Consent to Participate Ethical approval and consent to participate are not applicable.

References

1. Bali, K., Sharma, J., Choudhury, M., Vyas, Y.: I am borrowing ya mixing? an analysis of English-Hindi code mixing in Facebook. In: Diab, M., Hirschberg, J., Fung, P., Solorio, T. (eds.) Proceedings of the First Workshop on Computational Approaches to Code Switching, pp. 116–126. Association for Computational Linguistics, Doha, Qatar (2014). https://doi.org/10.3115/v1/W14-3914 . https://aclanthology.org/W14-3914
2. Gundapu, S., Mamidi, R.: Word level language identification in English Telugu code mixed data. In: Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation. Association for Computational Linguistics, Hong Kong (2018). https://aclanthology.org/Y18-1021
3. Zirker, K.A.H.: Intrasentential vs. intersentential code switching in early and late bilinguals (2007). https://api.semanticscholar.org/CorpusID:60154198
4. Hande, A., Puranik, K., Yasaswini, K., Priyadharshini, R., Thavareesan, S., Sampath, A., Shanmugavadivel, K., Thenmozhi, D., Chakravarthi, B.R.: Offensive Language Identification in Low-resourced Code-mixed Dravidian languages using Pseudo-labeling (2021). https://doi.org/10.48550/arXiv.2108.12177
5. Srivastava, V., Singh, M.: Code-mixed nlg: Resources, metrics, and challenges. In: 5th Joint international conference on data science & management of data (9th ACM IKDD CODS and 27th COMAD). CODS-COMAD 2022, pp. 328–332. Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3493700.3493766
6. Rathnayake, H., Sumanapala, J., Rukshani, R., Ranathunga, S.: Adapter-based fine-tuning of pre-trained multilingual language models for code-mixed and code-switched text classification. Knowl. Inf. Syst. **64**, 1937–1966 (2022). https://doi.org/10.1007/s10115-022-01698-1

7. Krishnan, J., Anastasopoulos, A., Purohit, H., Rangwala, H.: Cross-lingual text classification of transliterated hindi and malayalam. In: 2022 IEEE International Conference on Big Data (Big Data), pp. 1850–1857. IEEE Computer Society, Los Alamitos, CA, USA (2022). https://doi.org/10.1109/BigData55660.2022.10021079
8. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (2019). https://doi.org/10.18653/v1/N19-1423. https://aclanthology.org/N19-1423
9. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (2019). https://doi.org/10.18653/v1/N19-1423. https://aclanthology.org/N19-1423
10. Han, X., Zhao, W., Ding, N., Liu, Z., Sun, M.: Ptr: Prompt tuning with rules for text classification. AI Open **3**, 182–192 (2022). https://doi.org/10.1016/j.aiopen.2022.11.003
11. Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., Neubig, G.: Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. ACM Comput. Surv. **55**(9) (2023). https://doi.org/10.1145/3560815
12. Hu, S., Ding, N., Wang, H., Liu, Z., Wang, J., Li, J., Wu, W., Sun, M.: Knowledgeable prompt-tuning: Incorporating knowledge into prompt verbalizer for text classification. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 2225–2240. Association for Computational Linguistics, Dublin, Ireland (2022). https://doi.org/10.18653/v1/2022.acl-long.158. https://aclanthology.org/2022.acl-long.158
13. Tu, L., Xiong, C., Zhou, Y.: Prompt-tuning can be much better than fine-tuning on cross-lingual understanding with multilingual language models. In: Findings of the Association for Computational Linguistics: EMNLP 2022, pp. 5478–5485. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates (2022). https://aclanthology.org/2022.findings-emnlp.401
14. Zhao, M., Schütze, H.: Discrete and soft prompting for multilingual models. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 8547–8555. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic (2021). https://doi.org/10.18653/v1/2021.emnlp-main.672. https://aclanthology.org/2021.emnlp-main.672
15. Karimi Mahabadi, R., Zettlemoyer, L., Henderson, J., Mathias, L., Saeidi, M., Stoyanov, V., Yazdani, M.: Prompt-free and efficient few-shot learning with language models. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 3638–3652. Association for Computational Linguistics, Dublin, Ireland (2022). https://doi.org/10.18653/v1/2022.acl-long.254. https://aclanthology.org/2022.acl-long.254
16. Huang, L., Ma, S., Zhang, D., Wei, F., Wang, H.: Zero-shot cross-lingual transfer of prompt-based tuning with a unified multilingual prompt. In: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pp. 11488–11497. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates (2022). https://aclanthology.org/2022.emnlp-main.790
17. Fu, J., Ng, S.-K., Liu, P.: Polyglot prompt: Multilingual multitask prompt training. In: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pp. 9919–9935. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates (2022). https://aclanthology.org/2022.emnlp-main.674
18. Winata, G.I., Madotto, A., Lin, Z., Liu, R., Yosinski, J., Fung, P.: Language models are few-shot multilingual learners. In: Proceedings of the 1st Workshop on Multilingual Representation Learning, pp. 1–15. Association for Computational Linguistics, Punta Cana, Dominican Republic (2021). https://doi.org/10.18653/v1/2021.mrl-1.1. https://aclanthology.org/2021.mrl-1.1
19. Chakravarthi, B.R., Priyadharshini, R., Muralidaran, V., Jose, N., Suryawanshi, S., Sherly, E., McCrae, J.P.: DravidianCodeMix: sentiment analysis and offensive language identification dataset for dravidian languages in code-mixed text **56**(3), 765–806 (2022). https://doi.org/10.1007/s10579-022-09583-7
20. Joshi, P., Santy, S., Budhiraja, A., Bali, K., Choudhury, M.: The state and fate of linguistic diversity and inclusion in the NLP world, pp. 6282–6293. Association for Computational Linguistics, Online (2020). https://doi.org/10.18653/v1/2020.acl-main.560. https://aclanthology.org/2020.acl-main.560
21. Howard, J., Ruder, S.: Universal language model fine-tuning for text classification. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 328–339. Association for Computational Linguistics, Melbourne, Australia (2018). https://doi.org/10.18653/v1/P18-1031. https://aclanthology.org/P18-1031

22. Li, X.L., Liang, P.: Prefix-tuning: Optimizing continuous prompts for generation. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 4582–4597. Association for Computational Linguistics, Online (2021). https://doi.org/10.18653/v1/2021.acl-long.353 . https://aclanthology.org/2021.acl-long.353
23. Liu, X., Zheng, Y., Du, Z., Ding, M., Qian, Y., Yang, Z., Tang, J.: Gpt understands, too. AI Open (2023). https://doi.org/10.1016/j.aiopen.2023.08.012
24. Liu, X., Ji, K., Fu, Y., Tam, W., Du, Z., Yang, Z., Tang, J.: P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp. 61–68. Association for Computational Linguistics, Dublin, Ireland (2022). https://doi.org/10.18653/v1/2022.acl-short.8. https://aclanthology.org/2022.acl-short.8
25. Qin, G., Eisner, J.: Learning how to ask: Querying LMs with mixtures of soft prompts. In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 5203–5212. Association for Computational Linguistics, Online (2021). https://doi.org/10.18653/v1/2021.naacl-main.410. https://aclanthology.org/2021.naacl-main.410
26. Ding, N., Hu, S., Zhao, W., Chen, Y., Liu, Z., Zheng, H., Sun, M.: OpenPrompt: An open-source framework for prompt-learning. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 105–113. Association for Computational Linguistics, Dublin, Ireland (2022). https://doi.org/10.18653/v1/2022.acl-demo.10. https://aclanthology.org/2022.acl-demo.10
27. Hambardzumyan, K., Khachatrian, H., May, J.: WARP: Word-level Adversarial ReProgramming. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 4921–4933. Association for Computational Linguistics, Online (2021). https://doi.org/10.18653/v1/2021.acl-long.381. https://aclanthology.org/2021.acl-long.381
28. Zhao, T., Wallace, E., Feng, S., Klein, D., Singh, S.: Calibrate before use: Improving few-shot performance of language models. In: International Conference on Machine Learning (2021). https://api.semanticscholar.org/CorpusID:231979430
29. Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., Gelly, S.: Parameter-efficient transfer learning for NLP. In: Proceedings of the 36th International Conference on Machine Learning. PMLR, vol. 97, pp. 2790–2799 (2019). https://proceedings.mlr.press/v97/houlsby19a.html
30. Pfeiffer, J., Vulić, I., Gurevych, I., Ruder, S.: Mad-x: An adapter-based framework for multi-task cross-lingual transfer. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP) (2020). https://doi.org/10.18653/v1/2020.emnlp-main.617
31. Rathnayake, H., Sumanapala, J., Rukshani, R., Ranathunga, S.: Adapterfusion-based multi-task learning for code-mixed and code-switched text classification. Eng. Appl. Artif. Intell. **127**, 107239 (2024). https://doi.org/10.1016/j.engappai.2023.107239
32. Rücklé, A., Geigle, G., Glockner, M., Beck, T., Pfeiffer, J., Reimers, N., Gurevych, I.: AdapterDrop: On the efficiency of adapters in transformers. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pp. 7930–7946. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic (2021). https://doi.org/10.18653/v1/2021.emnlp-main.626. https://aclanthology.org/2021.emnlp-main.626
33. Shah, A., Thapa, S., Jain, A., Huang, L.: ADEPT: Adapter-based efficient prompt tuning approach for language models. In: Proceedings of The Fourth Workshop on Simple and Efficient Natural Language Processing (SustaiNLP), pp. 121–128. Association for Computational Linguistics, Toronto, Canada (Hybrid) (2023). https://aclanthology.org/2023.sustainlp-1.8
34. Reynolds, L., McDonell, K.: Prompt programming for large language models: Beyond the few-shot paradigm. In: Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems. CHI EA '21. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3411763.3451760. https://doi.org/10.1145/3411763.3451760
35. Bohra, A., Vijay, D., Singh, V., Akhtar, S.S., Shrivastava, M.: A dataset of Hindi-English code-mixed social media text for hate speech detection. In: Proceedings of the Second Workshop on Computational Modeling of People's Opinions, Personality, and Emotions in Social Media, pp. 36–41. Association for Computational Linguistics, New Orleans, Louisiana, USA (2018). https://doi.org/10.18653/v1/W18-1105. https://aclanthology.org/W18-1105
36. Vilares, D., Alonso, M.A., Gómez-Rodríguez, C.: EN-ES-CS: An English-Spanish code-switching Twitter corpus for multilingual sentiment analysis. In: Proceedings of the Tenth International Conference on Lan-

guage Resources and Evaluation (LREC'16), pp. 4149–4153. European Language Resources Association (ELRA), Portorož, Slovenia (2016). https://aclanthology.org/L16-1655
37. Chathuranga, S., Ranathunga, S.: Classification of code-mixed text using capsule networks. In: Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021), pp. 256–263. INCOMA Ltd., Held Online (2021). https://aclanthology.org/2021.ranlp-1.30
38. Kamble, S., Joshi, A.: Hate speech detection from code-mixed Hindi-English tweets using deep learning models. In: Proceedings of the 15th International Conference on Natural Language Processing, pp. 150–155. NLP Association of India, International Institute of Information Technology, Hyderabad, India (2018). https://aclanthology.org/2018.icon-1.22
39. Tatariya, K., Lent, H., De Lhoneux, M.: Transfer learning for code-mixed data: Do pretraining languages matter? In: Proceedings of the 13th Workshop on Computational Approaches to Subjectivity, Sentiment, & Social Media Analysis, pp. 365–378. Association for Computational Linguistics, Toronto, Canada (2023). https://doi.org/10.18653/v1/2023.wassa-1.32. https://aclanthology.org/2023.wassa-1.32
40. Takawane, G., Phaltankar, A., Patwardhan, V., Patil, A., Joshi, R., Takalikar, M.: Language augmentation approach for code-mixed text classification. Natural Language Processing Journal **5**, 100042 (2023). https://doi.org/10.1016/j.nlp.2023.100042
41. Laureano De Leon, F.A., Tayyar Madabushi, H., Lee, M.: Code-mixed probes show how pre-trained models generalise on code-switched text. In: Calzolari, N., Kan, M.-Y., Hoste, V., Lenci, A., Sakti, S., Xue, N. (eds.) Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024), pp. 3457–3468. ELRA and ICCL, Torino, Italia (2024). https://aclanthology.org/2024.lrec-main.307
42. Winata, G., Cahyawijaya, S., Liu, Z., Lin, Z., Madotto, A., Fung, P.: Are multilingual models effective in code-switching?, pp. 142–153 (2021). https://doi.org/10.18653/v1/2021.calcs-1.20
43. Thara, S., Poornachandran, P.: Transformer based language identification for malayalam-english code-mixed text. IEEE Access **9**, 118837–118850 (2021). https://doi.org/10.1109/ACCESS.2021.3104106
44. Zhang, R., Cahyawijaya, S., Cruz, J.C.B., Winata, G., Aji, A.: Multilingual large language models are not (yet) code-switchers. In: Bouamor, H., Pino, J., Bali, K. (eds.) Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp. 12567–12582. Association for Computational Linguistics, Singapore (2023). https://doi.org/10.18653/v1/2023.emnlp-main.774. https://aclanthology.org/2023.emnlp-main.774
45. Qin, L., Ni, M., Zhang, Y., Che, W.: Cosda-ml: Multi-lingual code-switching data augmentation for zero-shot cross-lingual nlp. In: Bessiere, C. (ed.) Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20, pp. 3853–3860. International Joint Conferences on Artificial Intelligence Organization, ??? (2020). https://doi.org/10.24963/ijcai.2020/533 . Main track. https://doi.org/10.24963/ijcai.2020/533
46. Hande, A., Hegde, S.U., Priyadharshini, R., Ponnusamy, R., Kumaresan, P.K., Thavareesan, S., Chakravarthi, B.R.: Benchmarking multi-task learning for sentiment analysis and offensive language identification in under-resourced dravidian languages. CoRR **abs/2108.03867** (2021) arXiv:2108.03867

Authors and Affiliations

Pasindu Udawatta[1] · Indunil Udayangana[1] · Chathulanka Gamage[1] · Ravi Shekhar[2] · Surangika Ranathunga[3]

✉ Surangika Ranathunga
s.ranathunga@massey.ac.nz

Pasindu Udawatta
pasinduu.18@cse.mrt.ac.lk

Indunil Udayangana
induniludayangana.18@cse.mrt.ac.lk

Chathulanka Gamage
chathulanka.18@cse.mrt.ac.lk

Ravi Shekhar
r.shekhar@essex.ac.uk

1 Department of Computer Science and Engineering, University of Moratuwa, Moratuwa, Sri Lanka

2 School of Computer Science and Electronic Engineering, University of Essex, Colchester, United Kingdom

3 School of Mathematical and Computational Sciences, Massey University, Palmerston, New Zealand