Hindawi
Mathematical Problems in Engineering
Volume 2019, Article ID 3965845, 15 pages
https://doi.org/10.1155/2019/3965845

*Research Article*

# Holistic User Context-Aware Recommender Algorithm

**Tatenda D. Kavu**,[1] **Kudakwashe Dube**,[2] **and Peter G. Raeth**[3]

[1]*Computer Science Dept, University of Zimbabwe Harare, Harare, Zimbabwe*
[2]*School of Fundamental Sciences, Massey University Palmerston North, Palmerston North, New Zealand*
[3]*University of Gondar, Gondar, Ethiopia*

Correspondence should be addressed to Kudakwashe Dube; K.Dube@massey.ac.nz

Received 29 January 2019; Revised 5 June 2019; Accepted 14 July 2019; Published 29 September 2019

Academic Editor: Rosa M. Benito

Existing recommender algorithms lack dynamism, human focus, and serendipitous recommendations. The literature indicates that the context of a user influences user decisions, and when incorporated in recommender systems (RSs), novel and serendipitous recommendations can be realized. This article shows that social, cultural, psychological, and economic contexts of a user influence user traits or decisions. The article demonstrates a novel approach of incorporating holistic user context-aware knowledge in an algorithm to solve the highlighted problems. Web content mining and collaborative filtering approaches were used to develop a holistic user context-aware (HUC) algorithm. The algorithm was evaluated on a social network using online experimental evaluations. The algorithm demonstrated dynamism, novelty, and serendipity with an average of 84% novelty and 85% serendipity.

## 1. Introduction

Research in recommender systems research has recently diverted attention to context-aware recommender systems (CARS) since incorporating contextual information in recommender systems is an effective approach to create more accurate and relevant recommendations [1]. Recommender algorithms are generally limited to coping up with the dynamics of user preferences [2, 3]. In current recommender systems (RSs), adaptive learning is basically derived from computing neighbourhood of a user from ratings using distance functions such as Pearson correlation function and Euclidean distance function. Therefore, adaptation will be mainly the by-product of the neighbourhood computation. If a user rates something new, his/her neighbourhood changes, and as a result, the recommendations got by this user are likely to change. This is the type of adaptive recommendation which is experienced in current recommendation algorithms. However, preferences inferred from user actions only are inadequate to deduce sound user-centric predictions [4].

Recommender algorithms are found lagging behind the dynamics of user preferences, resulting in lack of sound dynamic, novel, and serendipitous recommendations [5]. Moreover, the incorporated contexts in CARS are not enough for tailoring personalized services to users. One or two contexts are only incorporated in recommender systems, and these are not enough to represent users' interests or preferences [6]. The literature indicates that recommender algorithms lack significant novel and serendipitous recommendations [7, 8].

To address the identified challenges, an investigation was carried out and it was found that recommender algorithms are biased significantly towards system-centric factors such as accuracy, diversity, and scalability [9]. There is a lack of incorporation of knowledge from these user-centric factors (user experience, user decision-making processes, and user interaction) [10]. Therefore, this article presents a unique method of incorporating users' decision-making knowledge into a recommender algorithm. The context-aware recommender algorithm uses the holistic contextual user profile in the form of social, cultural, psychological, and economic profile, together with items associated with the user to offer recommendations. When tested online, the algorithm proved to offer dynamic, novel, and serendipitous recommendations to any user as compared to any algorithm that uses ratings and rated items.

The main contribution of this work is the contextual modelling of the holistic user contextual profile in order to provide dynamic, novel, and serendipitous recommendations. The approach brings the benefit of satisfaction to the users of the recommender system. The rest of the paper is organized as follows: Section 2 shows related work, Section 3 shows the materials used and the theory behind the recommendation approach, Section 4 demonstrates the experimental setup, Section 5 shows the results, and Section 6 discusses the results. Finally, Section 7 summarizes and provides future directions.

## 2. Related Work

Contextualization is viewed as a paradigm for building intelligent systems that can better predict and anticipate the needs of users and acts more efficiently in response to their behavior. The incorporation of contextual information about the user in the recommendation process has attracted major interest [11]. Contextual information matters in RSs, and it is important to take this information into account when providing recommendations [12].

Contextual information/features of a user such as time and location are quite important in making novel and serendipitous recommendations [1, 7, 13, 14]. Modelling the contextual information about a user enables the recommendation engine to provide more reasonable recommendations for users [13, 15], and context adaptation is one major goal of session-aware recommendation engines [16].

It is a well-known fact from decision science that a decision made speaks a lot especially about the decision-maker's values [17]. If a user chooses an item or prefers an item, people with the same values with this particular user are likely to be interested in the same items [18]. Since users' preferences are likely to be similar to, or influenced by friends [19]. Users first identify the need of an item and then search for it and evaluate alternatives from the search before making a purchase [20]. Understanding users of RSs is the key issue for better recommendations, and users can be advised at the right place and at the right time with the right message [21]. The researchers of this article found from the literature that users are influenced by significant factors such as economic status, psychological status (e.g., mood), cultural values, and social status when choosing an item or a service [7, 21]. In summary, these are the factors that spell all conceptual issues that explain theoretical derivations of user preferences.

There are algorithms that incorporate time in the recommendation engines due to the fact that, in some circumstances, a recommendation that is relevant in the morning context may not be relevant in the evening and vice versa [7, 22]. In clothing industry, clothes needed during summer and winter may be different. Time can be aggregated such that it can be classified in daily periods for instance {morning, afternoon, evening} or seasonal periods such as {summer, winter, spring, autumn} or event periods such as {Christmas, new year, school reopening, etc.} [7, 21]. Recently RSs have been designed which incorporate time [2, 23]; however, an algorithm that focuses on the holistic approach of incorporating important decision-making factors in the context of {social, cultural, psychological, economic} factors has not been investigated or implemented. Generally, recommendation algorithms work with a general assumption that recent ratings are more important than older ratings [1, 22]. Users generally select content or items based on current desires or preferences at the time of selection. Such desires or preferences, however, often fluctuate over time [1, 19, 24].

Most of the work on CARS has been conceptual. Collection of big data for incorporation in CARS is still another major challenge that needs to be tackled [11]. Overall, the field of CARS is a relatively new and under explored area of research, and much more work is needed to investigate it comprehensively [12].

Vargas [8] did a phenomenal series of researches and proved that there was no significant groundwork that had been performed before on novelty. He came up with a unified framework for novelty and diversity coming from an integration of different mathematical models that describe novelty and diversity. He built the framework on the argument that novelty and diversity are inextricably interrelated. It is quite recognizable that computation of novel recommendations is done during the time of ranking the recommended list. The recommended list is shuffled to make sure that the final list is novel and diverse.

Lee and Lee [9] also proposed an algorithm to provide novel recommendations whereby a graph-based recommender algorithm was developed that uses positively rated items in users' profiles to construct a highly connected undirected graph. Items were represented as nodes and positive correlations as edges. Using the concept of entropy and the linked items in the graph, the proposed algorithm was able to find recommendations that were both novel and relevant.

The strategy of computing novel recommendations seems to be satisfactory from the status quo, but the argument still exists that serendipity is even more beneficial than novelty because it creates new preferences to the user thereby driving a user to make decisions in favor of the RS [7, 19, 25, 26] argues that new methods of extracting implicit information about users from their daily activities can be used to realize novel and serendipitous recommendations. Alternative methods can be explored to rank and recommend items to users by considering several criteria [13, 27, 28, 29], and that is the path taken by this article.

## 3. Materials and Methods

### 3.1. Relevant Theory

*3.1.1. Key Concepts, Terms, and Principles.* The user's contextual profile supplies substantial information about that particular user. Context in this case includes factors like time that a user interacts with a RS, social, and cultural background. Context can also involve the user's mood at a particular time (mood can be affected by events such as birthday, graduation day, or funeral time) and economic status (which include type of profession, social class, and possessions). Therefore, given that a recommender algorithm has access to user's contextual information, that algorithm should at least be able to give reasonable, unique novel recommendations to that user.

RS users are naturally found in different contexts [22, 30]. Users' tastes or behaviors can be affected by many contextual circumstances of the user, for instance, social life, mood, cultural background, economic status, location, and factors such as time (weekdays, weekends, and holidays) [22]. The geographical location of a user can have a significant influence on his/her preferences in clothing or food [31]. From an extensive review of the literature, the researchers found that the user contextual phenomenon can be classified into four classes. These classes are social, cultural, psychological, and economic status. If a user is in a certain social or cultural context, that user has particular needs. The same applies when a user is in a certain mood (psychological status), that user will like or dislike certain items [24]. If a user is in a certain economic status or context, that user is able or not able to afford certain items. It also entails that if the user's social status or economic status changes, he/she will have different preferences or tastes. Therefore, a recommender algorithm must take all the four contextual classes into consideration to offer dynamic, novel, and serendipitous recommendations to users.

*3.1.2. Principles behind Computations.* The main principle behind the computation of the holistic user context-aware (HUC) algorithm is that users in the same context (social, cultural, psychological, and economic status) are likely to have the same taste or similar set of preferences. There are similarity functions such as Jaccard similarity function, Pearson correlation function, and Euclidean distance function which can be used to find the contextual similarity of users, using their profiles. Table 1 shows the comparison of these similarity measure functions.

Jaccard was used in this article because of the nature of the input data to the HUC algorithm. Jaccard is a simple statistic used for comparing the similarity and diversity of sample sets. It is a similarity distance measure similar to other methods of measuring similarity such as Euclidean distance, Pearson's correlation coefficient, and cosine similarity function. Its main advantage in this case is that it is best applicable in measuring the similarity of finite sets. Profiles can be converted into sets and are easily compared to find out similar profiles. The Jaccard index is a measure of similarity between sets and produces a value between 0 and 1.0 inclusive. It was introduced by Jaccard, the late professor of botany [32]. Many developers of recommender systems have used the Jaccard similarity to find the similarity between different types of sets especially in content-based recommender systems where it is used to measure the similarity of two documents [31]. Some of the published work where Jaccard was used include [32, 33] and [32, 34–37].

*3.1.3. Graphical Representation of the Computational Theory.* Similarity computation can be performed to find a link or similarity between user profiles. Using the graph theory, a graph is defined as an ordered triple V(G), E(G), and $\psi$G. Where V(G) is a nonempty set of vertices (where vertices are user profiles in this case), E(G) is a set of edges (edges are a representation of similarities), and $\psi$G which is associated with each edge of G as an unordered pair of (not necessarily distinct) vertices of G. $\psi$G is therefore the Jaccard similarity coefficient value that depicts the similarity between two vertices. If $e$ is an edge and $u$ and $v$ are vertices such that $\psi G(e) = uv = 0.9$, then $e$ is said to be joining $u$ and $v$; 0.9 shows the similarity value of the two vertices and depicts that $u$ and $v$ are very similar.

Therefore:

$$G = (V(G),\ E(G),\ \psi G)$$

V(G) = user1, user2, user3, user4, user5, user6, user7, user8
E(G) = $e1$, $e2$, $e3$, $e4$, $e5$, $e6$, $e7$

Suppose at some unique time $t$, $\psi$G is depicted as follows:

$$\psi G(e1) = \text{user1 user2} = 0.8,$$
$$\psi G(e2) = \text{user1 user3} = 0.7,$$
$$\psi G(e3) = \text{user4 user2} = 0.1,$$
$$\psi G(e4) = \text{user5 user6} = 0.5,$$
$$\psi G(e5) = \text{user3 user4} = 0.3,$$
$$\psi G(e6) = \text{user4 user7} = 0.5,$$
$$\psi G(e7) = \text{user4 user8} = 0.6$$

If user profiles have a similarity with a range of 0.5–1, it means that these users are similar. In other words, they are in related contexts and are likely to be interested in the same items. Therefore, from Figure 1, user 4 will be recommended items clicked/rated (that is actioned) by users 7 and 8. Looking at the graph, we can conclude that user 8 and user 7 are likely to be similar since they are both similar to user 4. Given that they have recent actions performed on the system according to the HUC algorithm, both are likely to be recommended items actioned by user 4. If the RS is deployed on a social network, then user 8 is likely to be recommended user 7 as a potential friend or a potential client.

*3.1.4. Computational Efficiency.* Novelty and serendipity are to a greater extent realized when using active profiles to compute neighbourhoods. Given that there are more than a million active users, it becomes computationally expensive to calculate the Jaccard similarity between the concerned users with the rest of the active users. This means that the efficiency and robustness of the algorithm are affected. Therefore, there is a need for a robust method that can be used to trim down the number of active users needed during similarity computation. We deliberated on classification and clustering algorithms such as $k$-means clustering, $k$-nn nearest neighbour, and decision trees. Decision trees were found to be the best method of classifying profiles efficiently because of the nature of the data used to compute neighbourhoods. Since the data come as sets, they will be the best for dynamic classification of

Table 1: Comparison of similarity measure metric functions.

| Jaccard similarity function | Euclidean distance function | Pearson correlation function |
|---|---|---|
| Measuring similarity of finite sets | Measure distance between data points | Measure linear correlation |
| Can work with strings | Work with vectors of real numbers | Corelational real numbers |
| Easy to work with social media profiles | Computationally expensive, when dealing with social media data | Weak on complex relationships |
| Best for content-based recommendations | Good for collaborative filtering algorithms | Good for collaborative filtering |

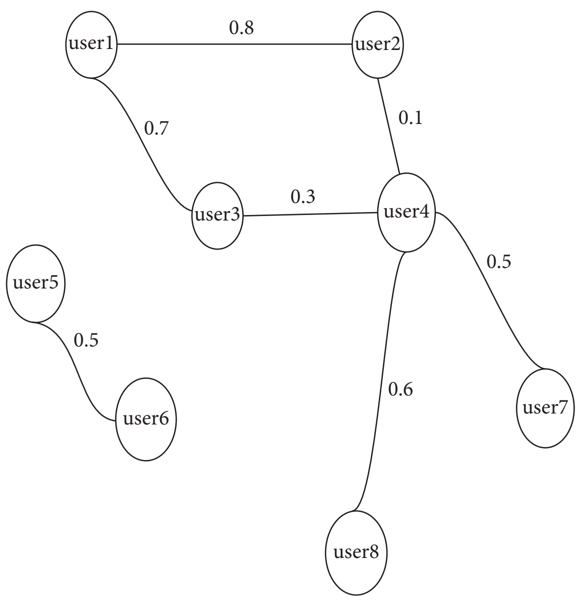

Figure 1: Similarity of user profiles.

profiles using decision trees. When a new user comes in, he/she will be placed in a specific class, and from that class, another computation will be done to compute his/her neighbourhood using the Jaccard similarity metric. This method reduces the computational time significantly. Moreover, classes will be changing systematically since the user profile will be updated by the user's activities. When a user profile changes, the user is migrated to another class of similar profiles. This approach gives rise to dynamic recommendations.

*(1) Classification using the ID$_3$ Decision Tree*. Using the ID$_3$ (Iterative Dichotomiser 3) version of a decision tree designed by Ross Quinlan, we start by calculating entropy, which is a statistical metric that measures the impurity of the data set. Given a set S of user profiles, which contains two classes: positive (meaning user did the anticipated action) and negative (meaning did not do the anticipated action), entropy with respect to this Boolean classification is

$$\begin{aligned}\text{entropy}(S) = &-p(\text{positive})\log 2\, p(\text{positive}) \\ &- p(\text{negative})\log 2\, p(\text{negative}),\end{aligned} \tag{1}$$

where $p$ positive is the probability of positive examples in S and $p$ negative is the probability of negative examples in S.

*Information gain* is the measure of the expected reduction in entropy. It decides which among the attributes of the concerned user's profile (the user that we want to classify) goes into a decision node (which attribute can be used to split the set). To minimize the decision tree depth, the attribute with the most entropy reduction is the best choice. The subset returned by a splitting decision must have a size of greater or equal to 50 profiles, and when the size is lower than 50, the splitting process stops and the user of concern is then classified:

$$\begin{aligned}
\text{Gain}(n) &= \text{Entropy}(n)^{-}\left([\text{weightedaverage}] * \text{entropy}\right),\\
\text{Gain}(S, A) &= \text{Entropy}(S)\text{Oentropy}\\
&\quad - \sum_{x\in\text{values}} (A)\frac{|S_v|}{|S|} * \text{Entropy}(S_v),\\
S &= \text{Each value } v \text{ of all possible values of attribute A},\\
S_v &= \text{Subset of S for which attribute A has value } v,\\
|S_v| &= \text{Number of elements in } S_v,\\
|S| &= \text{Number of elements in S}.
\end{aligned} \tag{2}$$

Given that on an e-commerce platform at a certain moment, a user was about to be classified into a certain class which shares the same attributes as him using the ID$_3$ decision tree. A set of user profiles together with their transactional history (whether they have bought or not on the e-commerce platform) was retrieved from the platform as shown in Table 2. These very short profiles will be used to demonstrate how the ID$_3$ decision tree can compute the class of user$_i$. In column action, 1 represents that the user bought something, 0 represents that the user did not buy anything.

The attributes maybe {age range, gender, location, hometown, time period, profession}, and they can have the following values:

age range = {20–25, 25–30, 30–35, 35–40}

gender = F, M

location = {harare, gweru, bindura, mutoko, bulawayo}

hometown = {chinhoyi, gweru, masvingo, mutoko, bindura}

time period = {morning, afternoon, evening}

profession = {law, engineering, education, indigenous, health, student}

*We need to find which attribute will be the first decision node in the decision tree*:

Table 2: A short blueprint of user profiles.

| User | Age range | Gender | Time period | Action |
|---|---|---|---|---|
| 1 | 20–25 | F | Morning | 1 |
| 2 | 25–30 | F | Afternoon | 1 |
| 3 | 20–25 | M | Evening | 0 |
| 4 | 30–35 | M | Morning | 1 |
| 5 | 35–40 | F | Afternoon | 0 |
| 6 | 35–40 | M | Afternoon | 1 |
| 7 | 35–40 | F | Morning | 1 |
| 8 | 20–25 | F | Afternoon | 1 |
| user$_i$ | 20–25 | M | Morning | 1 |

$$\begin{aligned}
\text{Entropy}(S) &= -\left(\frac{7}{9}\right)\log 2\left(\frac{7}{9}\right) - \left(\frac{2}{9}\right)\log 2\left(\frac{2}{9}\right) = 0.764,\\
\text{Gain}(S,\ \text{gender}) &= \text{Entropy}(S) - \left(\frac{5}{9}\right) * \text{Entropy}(SF)\\
&\quad - \left(\frac{4}{9}\right) * \text{Entropy}(SM)\\
&= 0.764 - \left(\frac{5}{9}\right) * 0.7219 - \left(\frac{4}{9}\right) * 0.811\\
&= 0.723,\\
\text{Entropy}(SF) &= -\left(\frac{4}{5}\right) * \log 2\left(\frac{4}{5}\right) - \left(\frac{1}{5}\right) * \log 2\left(\frac{1}{5}\right)\\
&= 0.7219,\\
\text{Entropy}(SM) &= -\left(\frac{3}{4}\right) * \log 2\left(\frac{3}{4}\right) - \left(\frac{1}{4}\right) * \log 2\left(\frac{1}{4}\right)\\
&= 0.811.
\end{aligned} \tag{3}$$

Looking at the scenario above, information gain of other attributes is less than 0.723; therefore, gender will be used as the first decision node. User$_i$ end up in the class of males who made their transactions in the morning. Therefore, the minimum possible class size of user$_i$ will be returned to other modules of the HUC algorithm for further computation. As shown in Figure 2, user$_i$ will be in the same class as user$_4$. Therefore, a similarity function will be called to compute the similarity between user$_i$ and user$_4$.

When a user's context changes either in the form of social, cultural, psychological, or economic context, that user's friends (neighbourhood) change as well. Therefore, the user's predicted preferences will be derived from the user's recent context or profile. Thus, the HUC algorithm will not be stuck with the user's historical preferences. The change of neighbourhood will be an outcome of the new similarity computation. Novel and serendipitous recommendations will be a product of computing other recommendations from recent acted items within the neighbourhood of the concerned user.

### 3.2. The HUC Recommender Algorithm

Vectors:

$P_i$ = tuple ($s$, $c$, $p$, $e$) (a profile for user $i$)
$s$ = tuple ($a_1$, …, $a_7$), where $a_i$ is a social attribute (relationship status, age range, gender, education, likes, political affiliation, social status)
$c$ = tuple ($c_1$, …, $c_5$), where $c_i$ is a cultural attribute (religion, current location, hometown, timezone/time period, language)
$p$ = tuple ($p_1$, …, $p_4$), where $p_i$ is a psychological attribute (birthday, friends birthday, movies, upcoming events)
$e$ = tuple ($e_1$, . . ., $e_5$), where $e_i$ is an economic attribute (currency, work history, profession, residential category)

Sets:

I = {x: x is an item/product}

$I_r$ = {x: $x \in I$ AND recommended to $P_u$}
$U$ = {x: $x = P_u$, where $P_u$ is a profile for user $u$}
O = {x: $x \in U_{\text{o}}$, where o is an old user profile}
$R$ = {x: $x \in U$ AND $x$ is an active user}
N = {x: $x \in U_{\text{n}}$, where n is a new user profile}
$M$ = {x: $x \in R$ AND Similarity (xi, xj) $\geq n$, where n is 0.5, $\forall i, j$}

Check if user $u$ has previously expressed interest in product $i$: $x \in$ Boolean, isActive $(u, i) = x, u \in U$ AND $i \in I$

(i) Derivative expressions:

$R \subseteq U$, $U = O \cup N$, and $M \subseteq R$

Algorithm 1 computes recommendations to users, and it starts by passing existing users to Algorithm 2. Algorithm 2 starts by computing active users from the existing/old users and then comes up with similar users to the concerned user $u$ from active users, and it retains a list of users who are similar to the concerned user. Algorithm 1 then computes frequent items actioned by similar users by calling Algorithm 3. These frequent sets of items are the ones recommended to the user $u$. Time complexity for the ComputeRecommendations algorithm is mainly determined by the time complexity of Algorithms 2 and 3.

Algorithm 2 computes the similarity of profiles. Given a list of profiles, it computes the similarity of the given list in relation to a particular user. The main algorithm to be called is the Jaccard similarity algorithm, which does the underlying work of computing similarity. The time complexity of Algorithm 2 is shown below:

$n = |O|$, $m = |R|$, T(computeProfileClassSet(u, O)), T(JaccardSimilarity(m, u))

T(R) = $n * 1$, T(M) = $m * n^2$

T(ComputeSimilarProfile) = $n + m * n^2$

O($n$ + mn $*$ 2) ≈ $O(mn^2)$

Algorithm 3 uses association rule mining (apriori) to compute frequent sets. The main task of this algorithm is to come up with a set of items which are frequently actioned by similar users. The actions might be searching, clicking, rating, etc. Its time complexity is illustrated as follows.

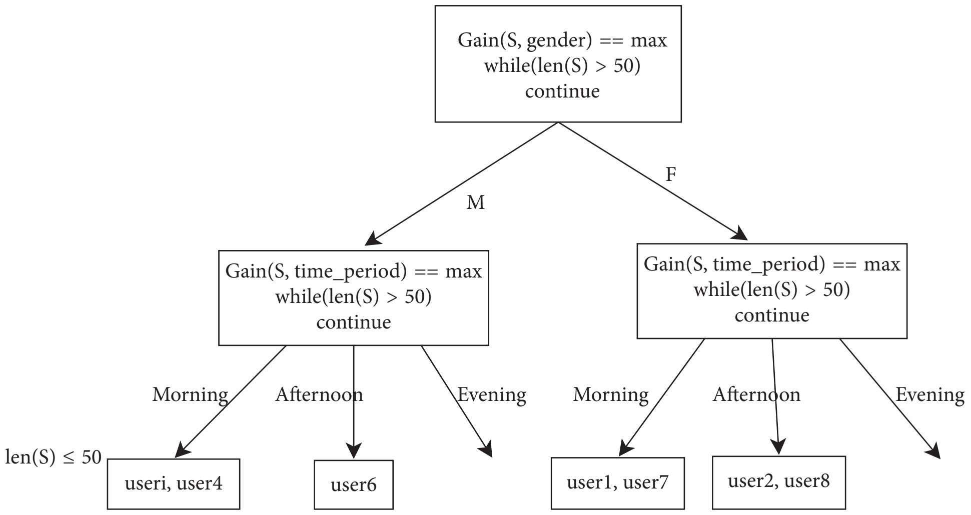

FIGURE 2: A snippet of the decision tree.

```
(1)  ComputeRecommendations
(2)  INPUT:
(3)  $I$ (set of items actioned by $M$)
(4)  $u$ (profile of user)
(5)  OUTPUT: $I_r$ (items recommended to $u$)
(6)  Begin
(7)    $M \longleftarrow$ (ComputeSimilarProfile ($u$, $O$))
(8)    Items_on_demand $\longleftarrow$ ComputeItemsActionedbySimilarProfile ($M$, $I$, $u$)
(9)    $I_r \longleftarrow$ I $\cup$ Items_on_demand
(10)   return $I_r$
(11) End
```

ALGORITHM 1: ComputeRecommendations.

```
(1)  ComputeSimilarProfile
(2)  INPUT: $u$ (profile of user)
(3)         O (set of old profiles)
(4)  OUTPUT: $M$ (set of profiles similar to profile $u$)
(5)  Begin
(6)         C $\longleftarrow$ computeProfileClassSet ($u$, $O$)
(7)         $R \longleftarrow \{r: r \in C$ and isActive $(r, I)\}$
(8)         $k \longleftarrow 0.5$; (Jaccard coefficient threshold for highly similar profiles)
(9)         $M \longleftarrow \{m: m \in R$ AND JaccardSimilarity $(m, u) \geq k, \forall m\}$
(10)        return $M$
(11) End
```

ALGORITHM 2: ComputeSimilarProfile.

```
(1)  ComputeItemsActionedbySimilarProfile
(2)  INPUT: $M$ (set of profiles similar to profile u)
(3)         $u$ (profile of user)
(4)         $I$ (set of items actioned by $M$)
(5)  OUTPUT: $I_t$ (set of items actioned by $T$)
(6)  Begin
(7)         freqSet $\longleftarrow$ Apriori ($M$, $I$, $u$)
(8)         $T \longleftarrow \{t: (t \in M$ AND (isActive ($t$, freqSet))$\}$
(9)         $I_t \longleftarrow \{i: i \in (I\ U$ freqSet) AND isActive($T$, $i$)$\}$
(10)        return $I_t$
(11) End
```

ALGORITHM 3: ComputeItemsActionedbySimilarProfile.

Time complexity for the ComputeItemsActionedbySimilarProfile algorithm is as follows:

$n = |\text{freqset}|, m = |\text{T}|, z = |\text{M}|, p = |\text{I}|, \text{g} = |\text{It}|, k = \text{constant}$

$\text{T(freqSet)} = 2\text{p},\ \text{T(T)} = z + 1(2\text{p}),\ \text{T}(I_t) = p + 1 + n + p$

$\text{T(ComputeItemsActionedbySimilarProfile)} = p + 1 + n + p + z + 1(2\text{p}) + 2\text{p}$

$\text{O}(p + 1 + n + p + z + 1(2\text{p}) + 2\text{p}) \approx \text{O}(n)$

Algorithm 4 in particular is responsible for reducing the list of old users with recent events/actions. This algorithm implements the $\text{ID}_3$ decision tree. Given that on a platform there are a million active users, Algorithm 4 will trim down the number of active users to a very small list of maybe up to 50 users. Algorithm 2 will then compute similar users from this new list. This method reduces the time for extracting similar users significantly. Operations such as this can be run in the background so that they do not disturb user interaction.

The time complexity of the classification algorithm (computeProfileClassSet algorithm) is as follows:

$n = |O|,\quad m = |G_o|,\quad d = |O_d|,\quad k = \text{constant},\quad e = |\text{A}|$, $\text{T(Entropy(o)} = 1$,

$\text{T}(\sum_{x\in V_o}^{|V_o|} (|O_x|/|O_{a_i}| * \text{Entropy}\,(O_x))\})$

$T(\text{G}_\text{o}) = 1 + e * k,\ T(max) = n,\ T(O_d) = n$

$T(computeProfileClassSet) = 1 + e * k + n + n$

$\text{O}(1 + e * k + n + n) = e + n + \text{n} \approx \text{O}(n)$

isActive() is the fifth algorithm which checks whether a particular user is active on a platform or not. The time complexity for Active algorithm is only O(1) since there is one operation that checks if a condition is true or false.

The Jaccard similarity function takes two sets of items and returns a value from a range of 0–1, 1 meaning the sets are similar and 0 meaning they are not similar. Its known time complexity is as follows:

$\text{T(JaccardSimilarity)} = \text{O}(n^2)$

$\mathbf{O}(n^2)$ is the known time complexity for the Jaccard algorithm, where $n$ = number of sets.

The Apriori algorithm is the one that does the underlying work of implementing the association rule mining technique. It computes the frequent set using a threshold value. Its time complexity is $\mathbf{O}(\mathbf{n})$, where $n$ = number of unique items in set $I_m$ or in the set of all transactions.

*3.2.1. Definition of Terms*

$P_i = <s, c, k, e>$ is a profile for a user $i$ where the order of the elements in the tuple is important. The order of items in $s$, $c$, $k$, and $e$ is important as well.

$s = <s_1, \ldots, s_7>$ $s_i$ is a social attribute (relationship-status, age-range, gender, education, likes, political-affiliation, social status).

$c = <c_1, \ldots, c_7>$ $c_i$ is a cultural attribute (religion, current location, hometown, timezone/time period, language).

$k = <p_1, \ldots, p_4>$ $k_i$ is a psychological attribute (birthday, friends birthday, movies, upcoming events).

$e = <e_1, \ldots, e_5>$ $e_i$ is an economic attribute (currency, work history, profession, residential category).

$I = \{x : x$ is an item/product$\}$, $Ir = \{x : x \in I$ AND recommended to $P_u\}$, $U = \{x : x = P_u$, where $P_u$ is a profile for user $u\}$, $O = \{x : x \in U_o$, where $o$ is an old user profile$\}$, $R = \{x : x \in U$ AND $x$ is *an active* user$\}$, $N = \{x : x \in U_n$, where $n$ is a new user profile$\}$, $M = \{x : x \in R$ AND Similarity $(x_i, x_j) \geqq n$, where $n$ is 0.5, $\forall\ i, j\}$

If user $u$ has previously expressed interest in product $i$, this can be represented in the form of *isActive(u,i)* $= x$, where $x \in$ Boolean and $u \in U$ AND $i \in I$. Moreover, $R \subseteq U$, $U = O \cup N$ and $M \subseteq R$. $N$ and $M \subseteq R$.

Algorithm 1 computes recommendations to users, and it starts by passing existing users to Algorithm 2. Algorithm 2 starts by computing active users from the existing/old users and then comes up with similar users to the concerned user $u$ from active users, and it retains a list of users who are similar to the concerned user. Algorithm 1 then computes frequent items actioned by similar users by calling Algorithm 3. These frequent sets of items are the ones recommended to the user $u$. Time complexity for the ComputeRecommendations algorithm is mainly determined by the time complexity of Algorithms 2 and 3.

Algorithm 2 computes the similarity of profiles. Given a list of profiles, it computes the similarity of the given list in relation to a particular user. The main algorithm to be called is the Jaccard similarity algorithm, which does the underlying work of computing similarity. The time complexity of Algorithm 2 is shown below:

$n = |O|, m = |R|$, T(computeProfileClassSet(u,O)), T(JaccardSimilarity(m, u))

$\text{T(R)} = n * 1,\ \text{T(M)} = m * n^2$

$\text{T(ComputeSimilarProfile)} = n + m * n^2$

$\text{O}(n + \text{mn} * 2) \approx (Omn^2))$

Algorithm 3 uses association rule mining (apriori) to compute frequent sets. The main task of this algorithm is to come up with a set of items which are frequently actioned by similar users. The actions might be searching, clicking, rating, etc. Its time complexity is illustrated as follows.

Time complexity for the ComputeItemsActionedbySimilarProfile algorithm is as follows:

$n = |\text{freqset}|$——, $\quad m = |\text{T}|,\quad z = |\text{M}|,\quad p = |\text{I}|,\quad \text{g} = |\text{It}|$, $k = \text{constant}$

$\text{T(freqSet)} = 2\text{p},\ \text{T(T)} = z{+}1(2\text{p}),\ \text{T}(I_t) = p{+}1 + n + p$

$\text{T(ComputeItemsActionedbySimilarProfile)} = p{+}1 + n + p + z{+}1(2\text{p}){+}2\text{p}$

$\text{O}(p{+}1 + n + p + z{+}1(2\text{p}){+}2\text{p}) \approx \text{O}(n)$

Algorithm 4 in particular is responsible for reducing the list of old users with recent events/actions. This algorithm

```
(1)  computeProfileClassSet
(2)  INPUT: $u$ (profile of user)
(3)         $O$ (set of old profiles)
(4)  OUTPUT: $O_d$ (set of profiles in same class as $u$)
(5)  Begin
(6)      $T \longleftarrow \{\mathrm{x} : x\ is\ a\ user\ attribute\}$
(7)      $tuple \longleftarrow < v_1, \ldots, v_n >$
(8)      $O \longleftarrow \left\{\mathrm{x} : x = tuple, v_i \in Dom_{a_i}, a_i \in T, i = 1, 2, \ldots, n, n = |T|\right\}$
(9)      $V_O \longleftarrow \left\{\mathrm{x} : x \in Dom_{O_{a_i}}, a_i \in T, \exists_{v_{a_i}} : x = v_{a_i} \in tuple\ for\ some\ tuple\ O\right\}$
(10)     $g_{a_i} \longleftarrow \left\{Entropy(O) - \sum_{x \in V_o}^{|V_o|} (|O_x| / |O_{a_i}| * Entropy(O_x))\right\}$
(11)     $O_d \longleftarrow \left\{\mathrm{p} : p \in O \forall p_{a_i} \in p \exists_{a_i} : g_{a_i} = max_y(G_o)\right\}$
(12)     **return** $O_d$
(13) End
```

Algorithm 4: ComputeProfileClassSet.

```
(1)  **isActive**
(2)  INPUT: $u$ (profile of user)
(3)         $i$ (item or set of items)
(4)  OUTPUT: bool (True or False)
(5)  Begin
(6)      isActive $\longleftarrow$ False
(7)      ACTIONS $\longleftarrow$ {click, search, rate, bought}
(8)      $A \longleftarrow \{x$: $x$ is recent action on $i$ by u AND $x \in$ ACTIONS$\}$
(9)      if ($A \neq \varnothing$)):
(10)     isActive $\longleftarrow$ True
(11)     return isActive
(12) End
```

Algorithm 5: isActive.

```
(1)  JaccardSimilarity
(2)  INPUT: $u_i$ (profile of user)
(3)         $u_j$ (profile of user)
(4)  OUTPUT: $k$ (similarity value)
(5)  Begin
(6)      $k \longleftarrow [(u_i \cap u_j) / (u_i \cup u_j)$
(7)      return $k$
(8)  End
```

Algorithm 6: JaccardSimilarity.

```
(1)  Apriori
(2)  INPUT: $M$ (set of profiles similar to profile $u$)
(3)         $I_m$ (set of sets of items actioned by $M$)
(4)         $u$ (profile of user)
(5)  OUTPUT: $F$ (set of items frequently actioned by $M$)
(6)  Begin
(7)      $k \longleftarrow 0.25$;(support threshold)
(8)      $I_m \longleftarrow$ {X: X $\in I$ AND isActive($M$, $X$) $\forall X$}
(9)      FrequentSets $\longleftarrow$ {$X$: Support ($X$, |$I$m|)>=$k$, $XI$m}
(10)     $F \longleftarrow \{i$: $i \in X$, $X \in$ FrequentSets,$\forall X\}$
(11)     return $F$
(12) End
```

Algorithm 7: Apriori.

```
(1) Entropy
(2) INPUT: $O$ (set of old profiles)
(3) OUTPUT: Entropy Value
(4) Begin
(5)         Entropy$\longleftarrow^{-}(|O_+|/|O|)\text{Log2}(|O_+|/|O|) - (|O_-|/|O|)\text{Log2}(|O_-|/|O|)$
(6)         $O_+$ number of profiles in $O$ with positive actions
(7)         return Entropy
(8) End
```

Algorithm 8: Entropy.

```
(1) Support
(2) INPUT: $X$ (set of items)
(3)         $|I_m|$ (number of items transacted)
(4) OUTPUT: $S_{\text{ratio}}$ (support ratio of $X$)
(5) Begin
(6)     count $\longleftarrow$ 0
(7)     for all $i$ in $I_m$
(8)     if ($i == X$)
(9)         count $\longleftarrow$ count + 1
(10)        $S_{\text{ratio}} \longleftarrow (\text{count}/|I_m|) * 100$
(11)        return $S_{\text{ratio}}$
(12) End
```

Algorithm 9: Support.

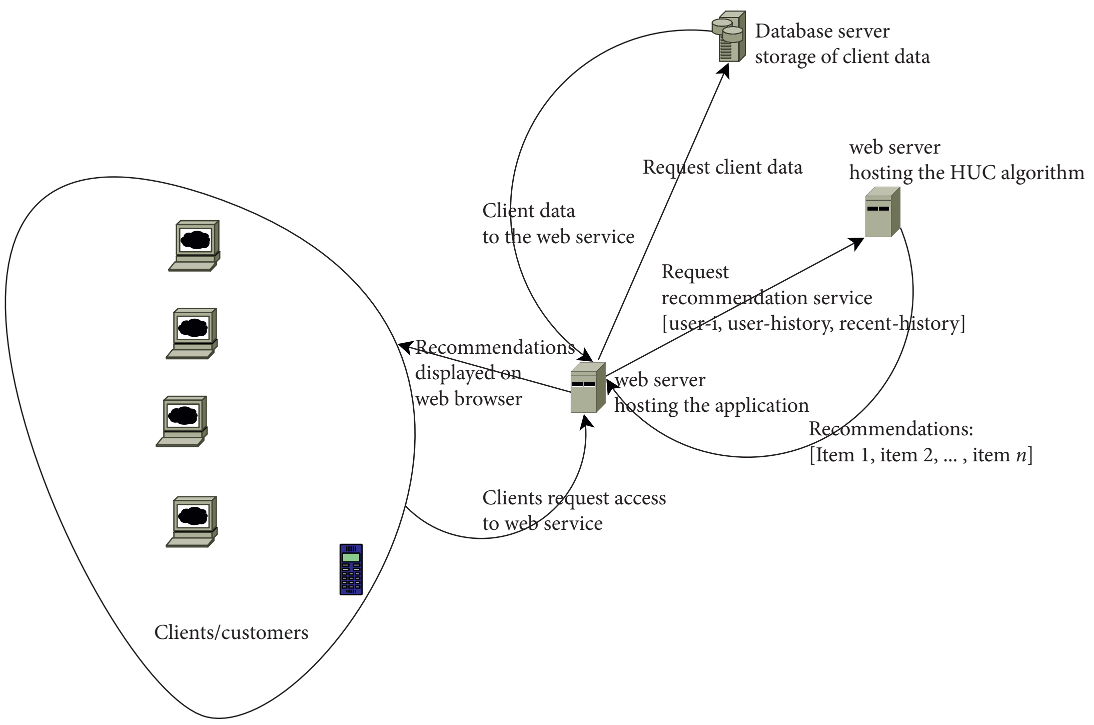

Figure 3: Experimental setup: clients interacting with the HUC algorithm.

implements the $ID_3$ decision tree. Given that on a platform there are a million active users, Algorithm 4 will trim down the number of active users to a very small list of maybe up to 50 users. Algorithm 2 will then compute similar users from this new list. This method reduces the time for extracting similar users significantly. Operations such as this can be run in the background so that they do not disturb user interaction.

Table 3: Transactional data.

| Data captured | Data type | Description |
|---|---|---|
| Recommendation ID | Int | Identify recommendations |
| Subject user | Object | Users, receiving recommendations |
| Recommended time | Time stamp | Time recommendation received |
| Clicked user | Object | User who clicked recommendation |
| Clicked time | Time stamp | Time a recommendation was clicked |

The time complexity of the classification algorithm (computeProfileClassSet algorithm) is as follows:

$n = |O|, \quad m = |G_o|, \quad d = |O_d|, \quad k = \text{constant}, \quad e = \text{—A—},$
$\mathrm{T(Entropy(o)} = 1,$

$\mathrm{T}($

$\sum_{x \in V_o}^{|V_o|} (|O_x| / |O_{a_i}|| * \text{Entropy}\,(O_x))$

$\})$

$T(\mathrm{Go}) = 1 + e * k, T(max) = n, T(Od) = n$

$T(computeProfileClassSet) = 1 + e * k + n + n$

$\mathrm{O}(1 + e * k + n + n) = e + n + \mathrm{n} \approx \mathrm{O}(n)$

isActive() is the fifth algorithm which checks whether a particular user is active on a platform or not. The time complexity for Active algorithm is only O(1) since there is one operation that checks if a condition is true or false.

The Jaccard similarity function takes two sets of items and returns a value from a range of 0–1, 1 meaning the sets are similar and 0 meaning they are not similar. Its known time complexity is as follows:

$\mathrm{T(JaccardSimilarity)} = \mathrm{O}(n^2)$

$\mathbf{O}(n^2)$ is the known time complexity for the Jaccard algorithm, where $n =$ number of sets.

The Apriori algorithm is the one that does the underlying work of implementing the association rule mining technique. It computes the frequent set using a threshold value. Its time complexity is $\mathbf{O(n)}$, where $n =$ number of unique items in set $I_m$ or in the set of all transactions.

## 4. Experimental Setup

The HUC algorithm was evaluated using online experimental evaluation methods since it was integrated with Unipals, a growing social site/network for Zimbabwean universities. This social site is used by university employees and students. However, any web user can open an account on the platform just like Facebook and make friends. By the time it was tested in December 2018, the platform had 1486 users and the number of users is still increasing. 1486 users were a good initial step for a meaningful online evaluation.

*4.1. Aim and Objectives.* The aim is to find out if the HUC algorithm is coping up with the dynamics of user preferences and does it offer novel and serendipitous recommendations.

*4.2. Background.* The HUC algorithm can be integrated with a web/mobile application that provides recommendation services to its clientele. When a client is interacting with an application, the client's information and segmented history are sent to the algorithm so that it will compute recommendations. After computation, the algorithm returns items to be recommended to the client which would then be presented on the client's interface. This procedure is shown in Figure 3.

*4.3. Procedure.* In this experiment, we integrated the HUC algorithm with a growing social network for universities in Zimbabwe Unipals: Unipals. The source code of the algorithm implementation is found on GitHub. The algorithm recommends friends or pals on the platform. The users' profiles {social, cultural, psychological, economic} are implicitly built as the user interacts with the platform. The user profile or context which was used by the algorithm to compute recommendations consists of {location, countryOfOrigin, time, friends, age, gender, picture-tags, language, likes, hobbies, interests}. This holistic user context increases the knowledge about the user so as to predict novel recommendations. The algorithm computes and then recommends pals or friends to the subject user. The summarized procedure of the whole process is depicted in Figure 3. Since this was an online experiment, the users were not aware that the algorithm is under test. The algorithm captured how the subject user responded to recommendations made.

*4.4. Data Set Design and Preparation.* The interaction of users with the social network and the interaction of users with the recommender algorithm were logged and saved in a csv file which can be accessed on dataset or on GitHub. For security reasons, the user profile data are not given for analysis. The information captured during transactions is shown in Table 3.

The recommendation ID in Table 3 represents the recommendations that have been offered to users. The subject user is the user that is receiving recommendations, and the clicked user is the user that is clicked by the subject user. When a subject user logs in, he/she is recommended a friend/pal, and if the user clicks the recommended user, the recommended user becomes the clicked user. The recommendation time is the time when the recommendation is made, and the clicked time is the time when the clicked user is clicked.

*4.5. Online Evaluation Experiment.* RS research is often based on comparisons of predictive accuracy: the better the evaluation scores, the better the RS. However, it is difficult to compare results from different RSs due to many options in design and implementation of evaluation strategies [38]. Therefore, a holistic approach of evaluating an RS will be a fair method of evaluation. While offline analysis is useful, user satisfaction (which is the main focus of this research) can only be measured in an online context [39].

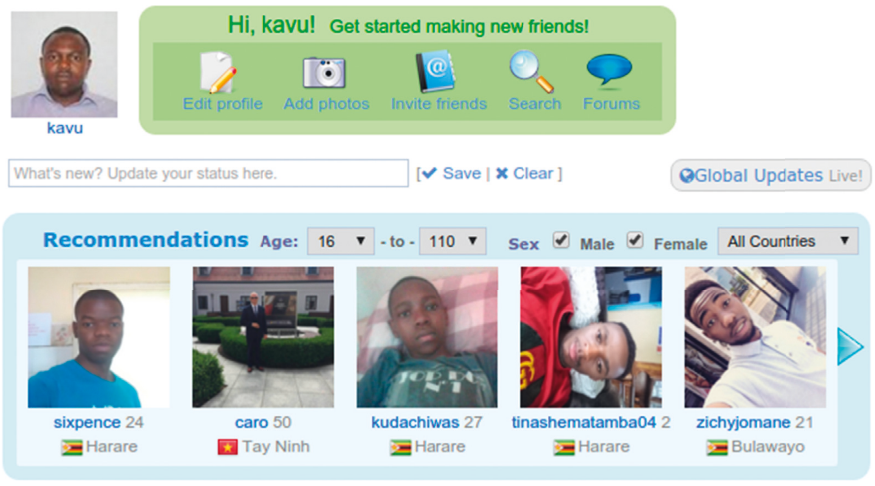
kavu
Hi, kavu! Get started making new friends!
Edit profile
Add photos
Invite friends
Search
Forums
What's new? Update your status here.
[✔ Save | ✖ Clear ]
Global Updates Live!
Recommendations Age: 16 - to - 110 Sex ✓ Male ✓ Female All Countries
sixpence 24 Harare
caro 50 Tay Ninh
kudachiwas 27 Harare
tinashematamba04 2 Harare
zichyjomane 21 Bulawayo

FIGURE 4: A snippet of recommendations made when a user logs in.

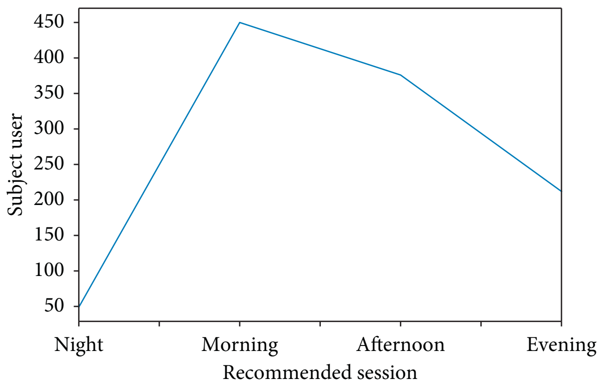

FIGURE 5: The relationship between subject users and recommendation time.

During online evaluation, the users are real users in a fully deployed system. This approach is less susceptible to bias from the recruitment process because the users are often directly using the system in the natural course of affairs [22]. From this background, we applied the online evaluation approach as a way to assess the performance of the algorithm. The interaction of the users with the algorithm is recorded. The live process can be viewed at dataset. This dataset in csv has the latest and recorded old transactions. In the analysis section, the algorithm was assessed using the following metrics: conversion rate, coverage, robustness, accuracy, novelty, and serendipity.

## 5. Results

Figure 4 shows one of the author's home pages just after logging in. A user received recommendations even before having friends on the platform since recommendations were derived from the user's profile. This approach eliminates the cold start problem which is also a serious problem in many RSs. When the logged in user clicks one of the recommended friends, the algorithm again computes new novel recommendations. User interactions were captured and exported as csv dataset for analysis, and

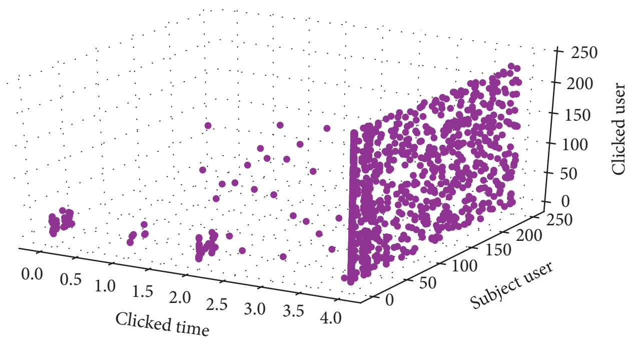

FIGURE 6: The relationship of subject user, clicked user, and time.

python libraries were used for analysis and for visualization. The Jupyter notebook file where result analysis was performed is found on GitHub. From the data gathered for analysis, (TheUser) is the one that received recommendations and clicked (the user which was clicked by the TheUser). From the dataset, more than a thousand recommendations where generated and the first 700 recommendations generated a lot of responses from the users. One thousand (1000) recommendations were offered to 250 users, which entailed that there were 250 active users on the platform and those 250 active users received recommendations at least once from the algorithm.

Figure 5 shows the record of the number of active users and the time periods which they responded to recommendations. Redundant records were found in the form of one active user responding to more than one recommendation. The graph demonstrates that there was much activity in the morning as compared to other time sessions.

Considering the fact that, in Figure 6, the time sessions were encoded as follows: 0.0–1.0 (midnight), 1.0–2.0 (afternoon), 2.0–3.0 (evening), and 3.0–4.0 (morning), we can see that a lot of activity happens in the morning, probably, that is, when the users which were mostly students got free to chat on the platform. Figure 6 also shows that the first 100 subject users responded a lot to their recommendations. This demonstrates that the algorithm was not affected by the cold

Table 4: A snippet of recorded transactions.

| Date recommended | Date clicked | Time recommended | Session | TheUser | Clicked |
|---|---|---|---|---|---|
| 2018-04-20 07 : 57 : 00 | 2018-04-20 07 : 57 : 00 | 7 : 57 : 0 | Morning | 0 | 0.0 |
| 2018-04-20 09 : 05 : 00 | 2018-04-20 09 : 05 : 00 | 9 : 5 : 0 | Morning | 0 | 0.0 |
| 2018-04-20 09 : 14 : 00 | 2018-04-20 09 : 15 : 00 | 9 : 14 : 0 | Morning | 0 | 1.0 |
| 2018-04-22 11 : 42 : 00 | 2018-04-22 11 : 42 : 00 | 11 : 42 : 0 | Morning | 1 | 5.0 |
| 2018-04-22 11 : 42 : 00 | 2018-04-22 11 : 42 : 00 | 11 : 42 : 0 | Morning | 1 | 6.0 |
| 2018-04-23 08 : 17 : 00 | 2018-04-23 08 : 17 : 00 | 8 : 17 : 0 | Morning | 2 | 7.0 |
| 2018-04-23 11 : 16 : 00 | 2018-04-23 11 : 17 : 00 | 11 : 16 : 0 | Morning | 3 | 8.0 |
| 2018-04-23 14 : 29 : 00 | 2018-04-23 14 : 30 : 00 | 14 : 29 : 0 | Afternoon | 5 | 9.0 |
| 2018-04-24 13 : 20 : 00 | 2018-04-24 13 : 20 : 00 | 13 : 20 : 0 | Afternoon | 6 | 11.0 |
| 2018-04-23 17 : 13 : 00 | 2018-04-23 17 : 13 : 00 | 17 : 13 : 0 | Afternoon | 1 | 11.0 |
| 2018-04-24 14 : 07 : 00 | 2018-04-24 14 : 08 : 00 | 14 : 7 : 0 | Afternoon | 1 | 3 |

start problem. Cold start is failure to draw any inferences for new users due to insufficient information of those users. The user profiles which participated in this experiment were not availed for analysis due to privacy issues from the service providers of the platform.

## 6. Discussion

Table 4 shows the first 20 records of user transactions. TheUser column is a unique identification of active users which received recommendations. The column Clicked represents the users which were clicked by TheUser, and ID numbers for TheUser is different from those for Clicked. If we look at The User1, we can see that the user is quite active, The User1 clicked ClickedID (5, 6, 11, and 3) at different times, that is, 11 : 42, 17 : 13, and 14 : 08, respectively. These users were clicked at different dates and time, showing that the algorithm was a bit dynamic and changes its recommendations with time and the user preferred those recommendations.

True positive (tp): refers to the recommended users which TheUser clicks.

False positive (fp): refers to the recommended users which TheUser did not click.

*6.1. Precision.* Since users where logged in sessions, it is important to figure out that unique recommendations per session can give us valuable information. Suppose that each session would take 10 minutes before a user is logged out, the total number of recommendations generated by the algorithms per the recorded transactions was 646. From these 646, those which the active users responded to were 77 per each unique session.

Therefore, the accuracy rate or precision is given below: $(\text{tp}/\text{tp} + \text{fp}) * 100$.

*6.2. Diversity.* Diversity refers to the uniqueness of the recommended items, i.e., out of the recommended users how many were unique? Getting back to the recorded transactions, from the generated unique sessional recommendations which were 646, 251 were unique recommendations (users which were never been recommended before). This meant that the diversity rate of the algorithm was $(251/646) * 100 = 38.9\%$ and shows that many of the registered users where not active on the platform. Therefore, the algorithm was left with no option but to recommend the remaining active users which were not that unique.

*6.3. Coverage.* Coverage looks at the total number of items on the platform, and out of the total, it identifies how many of these were ever recommended during recommendations. From the time when the transactions were recorded, the social network had 1486 users, and out of these, 1115 were picked for recommendations. Therefore, coverage rate becomes users involved/total users $= (1115/1486) * 100 = 75.03\%$.

*6.4. Novelty.* Novelty determines how unknown recommended items are to a user. The novelty of a retrieval set has been defined with respect to the end user as the proportion of known and unknown relevant items in the recommended list [40].

That is, given $L \subseteq I_R$, where $I_R$ is the recommended list, $L$ is the set of items in $I_R$ that the user(u) likes, $L$ can be partitioned as $L = L_k \cup L_u$ into those items, $L_k$ is already known items to the user, and $L_u$ is unknown items to the user. Then, the novelty per user is Novelty(R) $= |L_u|/|L|$, where $R$ is the set of all active users who received recommendations. The average novelty of the algorithm is

$$\sum_{u \in R} \frac{|L_u|/|L|}{|R|} = 85.561\%. \tag{4}$$

This shows that the novelty of the algorithm was quite high, from the transactions which were recorded.

*6.5. Serendipity.* An item is serendipitous if it is novel and relevant [8]. Serendipity is the measure of how surprising the relevant recommendations are.

(1) Average number of recommendations: $R$ at time $t_1$ and time $t_2$

(2) Average number of obvious items on time $t_1$ and $t_2$: $q$

(3) Number of nonobvious items on time $t_1$ and $t_2 = R - q$

```
$S_{\mathrm{user}_i}$:
count = 0.0
$\forall i$ in $[R - q]$
if $[R - q]$ are useful at $t_1$ and $t_2$
count = count + 1
$S_{\mathrm{user}_i}$ = count/|total$_{\mathrm{Recommendations}}$|
```

$$\text{AverageSerendipity} = \sum^{n_i} \left( \frac{S_{\mathrm{user}_i}}{\left|\mathrm{total}_{\mathrm{recommendations}}\right|} \right) = 85.47\%. \tag{5}$$

The best way to measure novelty and serendipity is asking users whether they were already familiar to a recommended item. However, this is impracticable in offline experiments [40]. It is for this reason that we took an online experimental approach to dealing with real users. The user's familiarity with an item was tracked from the user's history. If a user has been recommended an item before, that item is labeled familiar and is done implicitly. However, we did not consider other ways which the user might be familiar with the item.

Taking into consideration the level of novelty, serendipity, and dynamism of this algorithm, we can tell that the algorithm was really novel and serendipitous given that the rate was 84% at average. The level of novelty was better as compared to other novel algorithms which are found in the literature such as [40, 41] which experience a maximum of 77%. The algorithm's recommendations were also changing with time sessions especially the four time sessions (morning, afternoon, evening, and midnight), which means it was considering time context very well. We found out that its accuracy rate was very low at 36%. It was discovered that there was a trade-off between serendipity and accuracy rate [42]. The hypothesis that, if an algorithm takes a holistic stance of incorporating contextual user profile in the form of social, cultural, psychological, and economic profile, together with user-actioned items, that algorithm can offer dynamic, novel, and serendipitous recommendations was proven.

*6.6. Conclusion.* From the algorithm evaluation and the discussion made, we found out that the algorithm was quite novel and serendipitous, given that the average novelty and serendipity to each user was an average of 85%. The diversity of the algorithm was a bit low with an average of 38% due to the number of active users on the platform since the algorithm only considers active users to generate neighbourhood. This diversity is comparable to one of the best novel algorithms which were implemented by Hurley and Zhang [41] which had a novelty range of 38–44% and Zanitti et al. [43] which ranges from 25–52%. The platform did not perform well on poor mobile networks, and this resulted in users visiting the platform only when using public networks or WiFi. This reduced the number of active users and affected the performance of the algorithm. However, the coverage rate of the algorithm was at an average rate of 75%, and this entails that the number of available active users were mostly considered when constructing recommendations. Finally, the article demonstrated that if a recommender algorithm is given timely and detailed profiles of users, dynamic, novel, and serendipitous recommendations can be realized, and this will increase (1) traffic on a site, (2) click streams, (3) desired decisions, and (4) retention rates.

## 7. Summary

This article proposed a novel mechanism of generating recommendations based on a holistic understanding of user context to tackle three issues: (1) adequate adaptation to dynamic user preferences, (2) user-centric recommendations (user satisfaction), (3) adequate novel and serendipitous recommendations. The mechanism was further demonstrated as a process of deriving a user-centric recommender algorithm. The algorithm takes a hybrid approach of collecting holistic user contextual information which can be accessed from social media and then uses a user-user collaborative approach to generate and recognize novel, serendipitous, and dynamic recommendations. The holistic users' contextual knowledge mainly involves the social, cultural, economic status, and psychological profile of a user. The algorithm had an average time complexity of $O(n^2)$.

The results were quite promising for a social platform and we could not verify the performance of the algorithm over many platforms due to the lack of availability of other platforms to integrate with such as e-commerce and e-learning. This can be performed as future work. Another open issue is finding out which profile category among the four (social, cultural, psychological, and economic status) is more significant on platforms such as social networks, e-commerce, e-learning, movie sites, and online news. The social network did not allow users to tag most of their activities, and this also provided a limitation to this algorithm since the algorithm was supposed to be fed with updated user activities. Updated user activities would provide the algorithm with current interests and mood of users. In future, user activities could be tapped outside the service provider so that the algorithm is given a wider experience of a user to provide diverse and novel recommendations to the user. Our future work also involves developing an API that implements this algorithm so that different services can use the API to generate recommendations on different platforms.

## Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## References

[1] Z. D. Champiri, S. R. Shahamiri, and S. S. B. Salim, "A systematic review of scholar context-aware recommender

systems," *Expert Systems with Applications*, vol. 42, no. 3, pp. 1743–1758, 2015.

[2] J. Beel, C. Breitinger, S. Langer, A. Lommatzsch, and B. Gipp, "Towards reproducibility in recommender-systems research," *User Modeling and User-Adapted Interaction*, vol. 26, no. 1, pp. 69–101, 2016.

[3] F. O. Isinkaye, Y. O. Folajimi, and B. A. Ojokoh, "Recommendation systems: principles, methods and evaluation," *Egyptian Informatics Journal*, vol. 16, no. 3, pp. 261–273, 2015.

[4] P. Brusilovsky, A. Felfernig, P. Lops et al., "RecSys'16 Joint Workshop on Interfaces and Human Decision Making for Recommender Systems," in *Proceedings of the 10th ACM Conference on Recommender Systems*, pp. 413-414, Boston, MA, USA, September 2016, https://dl.acm.org/citation.cfm?id=2959199.

[5] M. H. Aghdam, "Context-aware recommender systems using hierarchical hidden Markov model," *Physica A: Statistical Mechanics and Its Applications*, vol. 518, pp. 89–98, 2019.

[6] K. Haruna, M. A. Ismail, D. Damiasih, H. Chiroma, and T. Herawan, "A comprehensive survey on comparisons across contextual pre-filtering, contextual post-filtering and contextual modelling approaches," *Telkomnika (Telecommunication Computing Electronics and Control)*, vol. 15, no. 4, pp. 1865–1874, 2017.

[7] D. Kotkov, S. Wang, and J. Veijalainen, "A survey of serendipity in recommender systems," *Knowledge-Based Systems*, vol. 111, pp. 180–192, 2016.

[8] S. Vargas, "Novelty and diversity evaluation and enhancement in recommender systems," Ph.D. Dissertation, Universidad Autónoma de Madrid, Madrid, Spain, 2015.

[9] K. Lee and K. Lee, "Escaping your comfort zone: a graph-based recommender system for finding novel recommendations among relevant items," *Expert Systems With Applications*, vol. 42, no. 10, pp. 4851–4858, 2015.

[10] T. D. Kavu, K. Dube, P. G. Raeth, and G. T. Hapanyengwi, "A characterisation and framework for user-centric factors in evaluation methods for recommender systems," *International Journal of ICT Research in Africa and the Middle East*, vol. 6, no. 1, pp. 1–16, 2017.

[11] K. Verbert, N. Manouselis, X. Ochoa et al., "Context-aware recommender systems for learning: a survey and future challenges," *IEEE Transactions on Learning Technologies*, vol. 5, no. 4, pp. 318–335, 2012.

[12] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, *Recommender Systems Handbook*, Springer, Berlin, Germany, 2011.

[13] M. G. Campana and F. Delmastro, "Recommender systems for online and mobile social networks: a survey," *Online Social Networks and Media*, vol. 3-4, pp. 75–97, 2017.

[14] F. Keikha, M. Fathian, and M. R. Gholamian, "TB-CA: a hybrid method based on trust and context-aware for recommender system in social networks," *Management Science Letters*, vol. 5, no. 5, pp. 471–480, 2015.

[15] Li. Lei, "Next generation of recommender systems: algorithms and applications," Dissertation, Digital Commons, Berkeley, CA, USA, 2014.

[16] M. Quadrana, P. Cremonesi, and D. Jannach, "Sequence-aware recommender systems," *ACM Computing Surveys*, vol. 51, no. 4, pp. 1–36, 2018.

[17] L. Buchanan, *Leigh Buchanan (2006), A Brief History of Decision Making*, Harvard Business Publishing, Brighton, MA, USA, 2006.

[18] R. Chen, Q. Hua, Y.-S. Chang, B. Wang, L. Zhang, and X. Kong, "A survey of collaborative filtering-based recommender systems: from traditional methods to hybrid methods based on social networks," *IEEE Access*, vol. 6, pp. 64301–64320, 2018.

[19] M. Eirinaki, J. Gao, I. Varlamis, and K. Tserpes, "Recommender systems for large-scale social networks: a review of challenges and solutions," *Future Generation Computer Systems*, vol. 78, pp. 413–418, 2018.

[20] S. Karimi, *A purchase decision-making process model of online consumers and its influential factor a cross sector analysis*, Ph.D. thesis, pp. 1–326, 2013, https://www.escholar.manchester.ac.uk/api/datastream?publicationPid=uk-ac-man-scw:189583{&}datastreamId=FULL-TEXT.PDF.

[21] E. Frolov and I. Oseledets, "Tensor methods and recommender systems," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 7, no. 3, article e1201, 2017.

[22] C. C. Aggarwal, *Recommender Systems*, Springer, Berlin, Germany, 2016.

[23] Y. Koren, "Collaborative filtering with temporal dynamics," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Paris, France, June 2009.

[24] K. Kapoor, V. Kumar, L. Terveen, J. A. Konstan, and S. Paul, "I like to explore sometimes: adapting to dynamic user novelty preferences," in *Proceedings of the 9th ACM Conference on Recommender Systems - RecSys'15*, pp. 19–26, Vienna, Austria, September 2015.

[25] R. Mu, X. Zeng, and L. Han, "A survey of recommender systems based on deep learning," *IEEE Access*, vol. 6, pp. 69009–69022, 2018.

[26] M. Taghavi, J. Bentahar, K. Bakhtiyari, and C. Hanachi, "New insights towards developing recommender systems," *The Computer Journal*, vol. 61, no. 3, pp. 319–348, 2018.

[27] C. V. Sundermann, M. A. Domingues, M. D. Silva Conrado, and S. Oliveira Rezende, "Privileged contextual information for context-aware recommender systems," *Expert Systems with Applications*, vol. 57, pp. 139–158, 2016.

[28] X. Yang, Y. Guo, Y. Liu, and H. Steck, "A survey of collaborative filtering based social recommender systems," *Computer Communications*, vol. 41, pp. 1–10, 2014.

[29] Z. Yang, B. Wu, K. Zheng, X. Wang, and L. Lei, "A survey of collaborative filtering-based recommender systems for mobile internet applications," *IEEE Access*, vol. 4, pp. 3273–3287, 2016.

[30] B. P. Knijnenburg, M. C. Willemsen, Z. Gantner, H. Soncu, and C. Newell, "Explaining the user experience of recommender systems," *User Modeling and User-Adapted Interaction*, vol. 22, no. 4-5, pp. 441–504, 2012.

[31] J. J. Levandoski, M. Sarwat, E. Ahmed, and M. F. Mokbel, "LARS: a location-aware recommender system," *IEEE 28th International Conference on Data Engineering*, vol. 1, pp. 450–461, 2012.

[32] S. Mahapatra and A. Tareen, "A cold start recommendation system using item correlation and user similarity," *ACM Transactions on Information Systems*, 2015, https://www.acsu.buffalo.edu/{~}suchismi/iRec.pdf.

[33] M. Millan, M. Trujillo, and E. Ortiz, "A collaborative recommender system based on asymmetric user similarity," in *Proceedings of the 8th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL'07)*, pp. 663–672, Springer-Verlag, Birmingham, UK, December 2007.

[34] P. Kumar and S. Chalotra, "An efficient recommender system using hierarchical clustering algorithm," *Internationa Journal of Computer Science Trends and Technology (IJCST)*, vol. 2,

no. 4, pp. 1–6, 2014, http://www.ijcstjournal.org/volume-2/issue-4/IJCST-V2I4P1.pdf.

[35] J. Leskovec, *Mining of Massive Data Sets - Ullman*, Cambridge University Press, Cambridge, UK, 2014.

[36] R. Singh, B. Kr. Patra, and B. Adhikari, “A complex network approach for collaborative recommendation,” CoRR, 2015, http://arxiv.org/abs/1510.00585.

[37] S. D. Sondur, S. Nayak, and A. P. Chigadani, “Similarity measures for recommender systems: a comparative study,” *International Journal for Scientific Research and Development*, vol. 2, no. 3, pp. 76–80, 2016, http://www.journal4research.org/Article.php?manuscript=J4RV2I3036.

[38] A. Said, A. Bellogín, A. De Vries, and B. Kille, “Information retrieval and user-centric recommender system evaluation,” in *Proceedings of the 21st Conference on User Modeling, Adaptation and Personalization (UMAP 2013)*, pp. 5–8, Rome, Italy, June 2013.

[39] C. Hayes, P. Massa, P. Avesani, and P. Cunningham, “An online evaluation framework for recommender systems,” in *Proceedings of the Recommendation and Personalization in eCommerce (2002)*, p. 50, Malaga, Spain, January 2002.

[40] L. Zhang, “The definition of novelty in recommendation system,” *Journal of Engineering Science and Technology Review*, vol. 6, no. 3, pp. 141–145, 2013.

[41] N. Hurley and M. Zhang, “Novelty and diversity in top-N recommendation—analysis and evaluation,” *ACM Transactions on Internet Technology*, vol. 10, no. 4, pp. 1–30, 2011.

[42] M. Kaminskas and D. Bridge, “Measuring surprise in recommender systems,” *Ir.Ii.Uam.Es*, vol. 69, pp. 107–144, 2011.

[43] M. Zanitti, S. Kosta, and J. Kirk Sørensen, “A user-centric diversity by design recommender system for the movie application domain,” in *Proceedings of the Companion of the “The Web Conference 2018”*, pp. 1381–1389, Lyon, France, April 2018.