MINING AND TRACKING EVOLVING WEB USER TRENDS FROM LARGE WEB SERVER LOGS

Basheer Hawwash and Olfa Nasraoui

Knowledge Discovery and Web Mining Lab

Dept. of Computer Engineering and Computer Science

University of Louisville

Recently, online organizations became interested in tracking users’ behavior on their websites to better understand and satisfy their needs. In response to this need, Web Usage Mining tools were developed to help them use web logs to discover usage patterns or profiles. However, since website usage logs are being continuously generated, in some cases, amounting to a dynamic data stream, most existing tools are still not able to handle their changing nature or growing size. This paper proposes a scalable framework that is capable of tracking the changing nature of user behavior on a website, and represent it in a set of evolving usage profiles. These profiles can offer the best usage representation of user activity at any given time, and they can be used as an input to higher-level applications such as a Web recommendation system. Our specific aim is to make the Hierarchical Unsupervised Niche Clustering (HUNC) algorithm more scalable, and to add integrated profile tracking and cluster-based validation to it. Our experiments on real web log data confirm the validity of our approach for large data sets that previously could not be handled in one shot.

1. Introduction

Organizations that provide online services, ranging from e-commerce transactions to document and multimedia browsing repositories, are in a continuous competition with each other to keep their existing customers and to lure new ones. Furthermore, this competition is
increasing because of the relative ease of starting an online business and the fact that a competitor’s website is often just one or two clicks away. The challenging nature of the online market has motivated online companies to start monitoring their online usage activity data to better understand and satisfy their website users. However, tremendous amounts of online clickstreams take place every day, making the data logged from them so large that trying to use conventional analysis methods is neither possible nor cost-effective. As a result, it has become imperative to use automated and effective data mining methods to turn this raw data into knowledge that can help online organizations to better understand their users.

Web Usage Mining is the process of applying data mining techniques on web log data to analyze the prevalent online activity patterns and extract from them user profiles (e.g. a set of usage clusters and the URLs that are relevant to each cluster). These profiles can then be used in higher-level applications such as recommendation or personalization services. There has been a considerable amount of research in web usage mining [1, 3, 4, 6, 14, 15, 18, 20]. However, there has been very few detailed studies that address the scalability and evolution concerns given the large streams of clickstream data and its evolving nature.

This paper proposes a new framework that can handle the changing behavior of the users of a website by continuously discovering, maintaining, and furthermore validating an unknown number of clusters and their corresponding usage patterns/profiles over time. Our specific aim is to make the Hierarchical Unsupervised Niche Clustering (HUNC) algorithm more scalable, and to add integrated profile tracking and cluster-based validation to it. In this framework, web usage mining is performed on a regular basis to discover new user profiles, and at each new time period, the recent user clickstreams or sessions are compared against the previous period’s profiles, then these profiles are incrementally updated taking into account their similarity to the new clickstreams. Each time that a new session matches an existing profile, it is used to update it, then the new session is discarded, thus greatly reducing the amount of new data to be mined in the new period. The remaining reduced data consisting of only new sessions, that do not match any existing profile, are then used
to discover new profiles to be added to the existing updated profiles. This approach results in a scalable profile discovery that furthermore adapts with the current usage trends.

The contributions of this paper can be summarized as follows: (i) enhancing the scalability of profile discovery, (ii) tracking the evolution of the discovered profiles over time by updating an archive of profiles on an on-going basis, (iii) generating user-friendly profiles that can be used in higher-level applications, (iv) evaluating the quality of the proposed profile discovery.

2. Related Work

2.1 Overview of Web Usage Mining

Web Usage Mining is the process of applying data mining techniques to extract useful knowledge such as typical usage patterns from web log data. The analysis of the discovered usage patterns can help online organizations gain a variety of business benefits, such as developing cross-product marketing strategies, enhancing promotional campaigns, and web personalization. Early research has been conducted in Web Usage Mining [1, 2, 4, 5, 14, 15, 25, 26] to address the challenges of pre-processing, usage data analysis and modeling, and data mining which ranged from clustering to association and sequential rule mining. Several tools have thus been developed [1, 3, 15, 18, 20] to infer usage patterns from web usage data (Pattern Discovery) and then to interpret these usage patterns (Pattern Analysis).

2.2 Web Usage Mining process

Typically, discovering the Web usage patterns, such as profiles or prediction models, consists of three steps: preprocessing the raw usage data, discovering patterns from the pre-processed data, and analyzing these discovered patterns.
There are two primary tasks in preprocessing: *data cleaning*, and *transaction identification* also known as *sessionization*. Data cleaning eliminates irrelevant items such as image requests and search engine requests from the server log. The transaction identification process groups the sequences of page requests into logical units, each of which is called a *session* which is the set of pages that are visited by a single user within a predefined period of time.

After pre-processing, the Web sessions are used as an input to pattern discovery methods that are typically rooted in areas such as data mining, artificial intelligence, or statistics. These discovery methods may include: *Statistical Analysis, Sequential Pattern mining* [21], *Path Analysis* [20, 23, 25, 26], *Association Rule Mining* [19, 24], *Classification* [22], and *Clustering* [2, 5, 6].

After discovery, the usage patterns are analyzed to better understand and interpret them, using a variety of analysis tools from the fields of statistics, graphics, visualization, or database querying. Examples of analysis tools can be found in [18, 19, 26].

### 2.3 The Hierarchical Unsupervised Niche Clustering algorithm (HUNC)

HUNC is a hierarchical version of the Unsupervised Niche Clustering (UNC) algorithm. UNC is an evolutionary approach to clustering proposed by Nasraoui and Krishnapuram in [14], that uses a Genetic Algorithm (GA) [16] to evolve a population of cluster prototypes through generations of competition and reproduction. UNC has proven to be robust to noise, but was formulated based on an Euclidean metric space representation of the data. Later, the HUNC (Hierarchical Unsupervised Niche Clustering) algorithm [6] was proposed to generate a hierarchy of clusters which gives more insight into the Web Mining process, and makes it more efficient in terms of speed. HUNC does not assume the number of clusters in advance, can provide profiles to match any desired level of detail, and requires no analytical derivation of the prototypes. This allows HUNC to use specialized similarity measures based only on the user access patterns. HUNC algorithm is shown in Algorithm 1.
Algorithm 1 HUNC

Input: $x_j$: user sessions, $N_{\text{min}}$: minimum allowed cluster cardinality, $\sigma_{\text{split}}$: minimum allowed cluster scale

Output: $p_i$: User profiles (sets of URLs, later referred to as raw profiles) $\chi_i$: Clusters of user sessions closest to profile $i$

1: Encode binary session vectors $x_j$
2: Set current resolution Level $L = 1$
3: Apply UNC (one level clustering) to entire data set with small population size (this results in cluster representatives $p_i$ and corresponding scales $\sigma_i$)
4: repeat until: cluster cardinality $N_i < N_{\text{min}}$ or scale $\sigma_i < \sigma_{\text{split}}$
5: Increment resolution level: $L = L + 1$
6: For each parent cluster $p_i$ found at Level $(L - 1)$
7: if cluster cardinality $N_i > N_{\text{min}}$ or cluster scale $\sigma_i > \sigma_{\text{split}}$ then
8: Reapply UNC only on data records $x_j$ that are closest to this parent cluster $p_i$ based on distance measure (this results in cluster representatives $p_i$ and corresponding scales $\sigma_i$)

Algorithm: Unsupervised Niche Clustering (UNC)

Input: $x_j$: data records, in this case user sessions, $N_p$: population size, $G$: Number of generations

Output: Cluster representatives: a set of profiles $p_i$ and scales $\sigma_i$

1: Randomly select from $x_j$ an initial population of $N_p$ candidate representatives $p_i$
2: Set initial scales $\sigma_i = \frac{\max_{i,j} d_{ij}}{10}$
3: repeat for $G$ generations
4: Update the distance $d_{ij}$ of each data record $x_j$ relative to each candidate cluster representative using distance
5: Update the robust weight $w_{ij} = e^{-d_{ij}/(2\sigma_i)}$ of each data record $x_j$ relative to each candidate cluster representative $p_i$
6: Update the scale $\sigma_i = \frac{\sum_j w_{ij} d_{ij}}{\sum_j w_{ij}}$ for each candidate cluster representative (derived by setting fitness gradient: $\partial f_i / \partial \sigma_i = 0$ while $w_{ij}$ are fixed)
7: Update density fitness $f_i = \frac{\sum_j w_{ij}}{\sigma_i}$ of each cluster representative $p_i$
8: for $i = 1$ to $N_p/2$ do
9: Select randomly from population candidate parent $p_i$ without replacement
10: Select randomly from population another candidate parent $p_k$ without replacement
11: Obtain children $c_1$ and $c_2$ by performing crossover and mutation between the chromosome strings of $p_i$ and $p_k$
12: Update the scale $\sigma_i$ and the fitness $f_i$ of each child
13: Apply Deterministic Crowding to fill new population: Assign each child $c_i$ to closest parent if child’s fitness > closest parent’s fitness then child replaces closest parent in the new population else closest parent remains in the new population
2.4 Mining evolving profiles

Most research in web mining has focused primarily on data collected during a particular time frame. However, because of the dynamic nature of the Web, researchers have recently paid more attention to mining evolving Web user profiles that vary with time. For example, [7] proposed a stream clustering methodology that was inspired by the natural immune system’s ability to adapt to a dynamic environment, while [8] studied the evolving nature of the web, where the Web was represented as a directed graph. Earlier work in learning with concept drift is also related to mining evolving patterns. This includes the work in [13] which addressed the issue of learning evolving concepts, which are concepts that gradually change over time, as well as [11] which presented a method for gradual forgetting, that assigned a weight to each training example according to its appearance over time. More recently, [12] proposed a framework, based on the HUNC clustering algorithm, to discover profiles using HUNC, then track how the discovered profiles evolve over consecutive time periods, and further categorize this evolution based on several predefined categories of evolution such as emergence or birth, death, persistence, or atavism. The main distinction of this paper, compared to the work in [12] is that whereas [12] did not handle the scalability concerns, this paper directly addresses the computational cost by eliminating a large proportion of a weblog from further analysis during the pre-processing stage. Also unlike [12] which consisted of decoupled mining and evolution tracking processes, the method presented in this paper tightly integrates the evolution tracking with the actual cluster mining, and thus leverages the former to enhance the scalability of the latter. Another main distinction is in the validation procedure that we use in this paper, which is cluster-based, and thus directly assesses the validity of the discovered clusters in such familiar cluster validity terms as the cluster density, compactness, and separation between clusters. To take this validation further, we also tracked these validity criteria over time to visualize how cluster quality evolves over time. Consequently, whereas in [12], we only characterized the type of evolution qualitatively, here we track objective metrics that related to the clusters or profiles with time,
leading to a powerful quantitative temporal evaluation that can make it easier to support further decision making processes down the stream. For example, the quantitative metrics can form new features that can be used in further decision making.

3. Methodology

Most existing web usage mining algorithms have failed to address the notion of change and large data sets. Even the few approaches that tried to capture and handle the changing nature of the web, were concerned with understanding the changes in the users’ access patterns over a period of time [12], but failed to utilize this change directly in order to maintain and develop the evolving profiles. On the other hand, the approach proposed in this paper discovers the usage patterns over different periods of time, and furthermore captures the changes undergone by the patterns in order to update them accordingly.

3.1 Overview of the Proposed Methodology

The web usage mining process is traditionally done in several steps with only a few variations. It starts with preprocessing the log files, discovering the usage patterns using a web usage mining algorithm, and then interpreting the discovered patterns. These steps have been used to discover usage patterns, predominately, within one specific period of time, but they can arguably be reapplied periodically, over several periods, to capture the changes in navigation patterns. However, there are some concerns using this approach, as explained below.

- Reapplying the steps periodically can either be performed on the entire historic data, or on the new log files only. The former approach reduces the probability of discovering new trends because of their small weight compared to older trends, while the latter approach completely forgets all previous patterns which may not be reasonable or efficient, since some of these patterns might still be significant in the new period, and would have to be rediscovered again.
• Trying to always discover the new behaviors from all the accumulated log files up to
the current period will require significant computational resources, which defies the
scalability requirement.

• All the above approaches do not capture the changes in the usage behaviors in detail,
i.e. we do not know which URLs have changed or have become more interesting from
one period to another.

The proposed framework, depicted in Figure 1, overcomes the above issues and can be
summarized as follows, assuming that we start with a set of initial (previous or seed) profiles
mined from an initial period:

1. Preprocess the new web log data to extract the current user sessions,

2. Update the previous profiles using the extracted user sessions,

3. Re-apply clustering to the distinct user sessions only (i.e. the ones not used in step 2
to update the previous profiles),

4. Post-process the distinct (new) profiles mined in step 3,

5. Combine the updated profiles with the distinct profiles to create the new seed profiles
for future periods,

6. Interpret and evaluate the discovered profiles

7. Go to step 1 as data from a new period becomes ready
3.2 Preprocessing the logs

The first step in pre-processing is data cleaning where all irrelevant elements such as images, requests from crawling agents, and unsuccessful requests are removed. The next step in preprocessing is sessionization, where consecutive page requests with the same originating IP address, session, or cookie ID, are grouped into units called sessions. Each session represents all the pages visited by a particular user within a predefined period of time. The $i^{th}$ user session is encoded as an $N_u$-dimensional binary vector $s_j$ with the following property:
\[ s^{(i)}_j = \begin{cases} 
1 & \text{if user accessed URL } j \text{ during session } i \\
0 & \text{otherwise} 
\end{cases} \]

where \( j \) is a unique number in \( 1,..,N_u \) and \( N_u \) is the total number of valid URLs.

### 3.3 Updating profiles

After pre-processing, the sessions are categorized into *matching sessions* and *distinct sessions*. The new matching sessions, that match any of the old profiles, will be used to update these profiles, the *old profiles* are the accumulation of the evolving profiles discovered and updated with arriving data, up to the previous period. The *distinct sessions* are the remaining sessions extracted during pre-processing, and that were not found to be “similar” enough to any old profile. Only these distinct sessions, that do not match any profile, will undergo the next step of discovering the new profiles/trends.

A profile \( P_i \) is a vector representation of the cluster \( X_i \), which is a set of user sessions that are more similar to each other than to the sessions in other clusters. The profile is represented as \( P_i = (P_{i1},..,P_{iN_u}) \) where \( P_{ij} \) is the relevance weight of \( URL_j \) in cluster \( i \), and is estimated by the conditional probability of accessing \( URL_j \) during the sessions assigned to \( i^{th} \) cluster \( X_i \), i.e.,

\[
P_{ij} = p(s_j^{(k)} = 1|s_j^{(k)} \in X_i) = \frac{|X_{ij}|}{|X_i|} \quad (1)
\]

\[
X_{ij} = \{s^{(k)} \in X_i|s_j^{(k)} > 0\} \quad (2)
\]

\( X_{ij} \) represents the set of sessions in cluster \( X_i \), that accessed the \( j^{th} \) URL, \( URL_j \), and \( |X_i| \) is the number of sessions in cluster \( X_i \). \( |X_{ij}| \) will be denoted as \( N_{ij} \) in the following discussion.

A profile \( P_i \) has a cardinality, \( N_i \), which is the number of sessions that are closest to the cluster prototype of \( X_i \), i.e. \( N_i = |X_i| \). Moreover, the profile \( P_i \) has a scale measure \( \sigma_i^2 \) that determines to what extent the sessions in cluster \( X_i \) are dispersed around its cluster
representative. The scale measure is a cluster parameter in HUNC which is derived to optimize HUNC’s density criterion as follows

\[ \sigma_i^2 = \frac{\sum_{j=1}^{N} w_{ij} d_{ij}^2}{\sum_{j=1}^{N} w_{ij}}, \]  

(3)

where \( w_{ij} \) is a robust typicality weight that measures how typical a session \( x_j \) is in cluster \( X_i \), and serves to distinguish outliers. It is given by

\[ w_{ij} = e^{-\frac{d_{ij}^2}{2\sigma_i^2}}. \]  

(4)

\( d_{ij} \) is the distance between session \( x_j \) and the prototype of cluster \( X_i \), as given by (6). The scale measure \( \sigma_i^2 \) can be considered as a radius of the cluster or profile, and the more similar the input sessions are to the cluster prototype, the smaller \( \sigma_i^2 \). Moreover, based on the robust weight \( w_{ij} \), it is possible to detect outliers which will have smaller weights.

In order to match new sessions with old profiles, we compute the similarity between a session \( S \) and a profile \( P \) using the Cosine Similarity between two sets as follows:

\[ Sim_{\text{cos}}(P, S) = \frac{|P \cap S|}{\sqrt{|P| \cdot |S|}} \]  

(5)

Where \( |P| \) is the number of URLs in profile \( P \), and \( |S| \) is the number of URLs in session \( S \). The similarity is then mapped to the dissimilarity:

\[ d^2 = (1 - Sim_{\text{cos}}(P, S))^2. \]  

(6)

Another way to match a new session to an old profile is by computing the session’s robust weight relative to the profile, defined in (4), and comparing it to a threshold value; this matching will be referred to as Robust Weight Similarity. The advantage of using the robust weight is that the weights are normalized by the scale of each cluster; hence,
they are less sensitive to the threshold, and they also depend on the profile’s dispersion or compactness. Thus, profiles with smaller scales will be more strict in matching, and vice versa for large-scale profiles.

One important issue when assessing the similarity between a profile and a session is to consider only significant common URLs, i.e., URLs whose weight exceeds a threshold value, when computing the cosine similarity. Thresholding URLs is an additional precaution to filter out the effect of potential outliers that might contaminate a profile. However, if the threshold is too strict, then this might cause discarding important updates to profiles, while still in their infancy stages (thus their URL weights are still low). Hence, choosing the optimal threshold value is vital, and is an important parameter to study. In addition to the above parameters, we record the starting and ending times for each profile. The former is the time stamp of the first session that gave rise to the profile, while the latter is the time stamp of the most recent session that was used to update the profile. Finally, after defining all the needed quantities, we can summarize the profile updating in Algorithm 2.
Algorithm 2 Update Profiles

Input: The set of all new sessions \( S_a \), The (old) seed profiles and their parameters \( P(N_{ij}, N_i, \sigma^2_i) \)

Output: The set of distinct sessions \( S_d \), The updated profiles \( P_u \)

1 Update Profiles \((S_a, P)\)
2 {
3 For each session \( S \) in \( S_a \)
4 {
5 Compute \( \text{Sim}(S, P_i) \) for all current profiles \( P_i \)
6 Find profile \( P_k \) that is closest to \( S \)
7 If \( \text{Sim}(S, P_k) > \text{Sim}_{\min} \) then
8     UpdateProfile\((P_k, S)\)
9  Else
10     Add \( S \) to \( S_d \)
11  }
12 }

13 UpdateProfile \((P_i, S)\)
14 {
15 For each URL\(_j\) in \( P_i \)
16 {
17     If URL\(_j\) in present in Session \( S \) then
18         \( P_{ij} = \frac{N_{ij}+1}{N_i+1} \)
19     Else
20         \( P_{ij} = \frac{N_{ij}}{N_i+1} \)
21  }
22 For each URL\(_k\) in \( S \) but not in \( P_i \)
23 {
24     Add URL\(_j\) to \( P_i \)
25     \( P_{ij} = \frac{1}{N_i+1} \)
26  }
27 Compute \( d_{\text{new}}^2 = (1 - \text{Sim}(P_i, S))^2 \)
28 Update profile scale: \( \sigma_{\text{new}}^2 = \frac{\sigma_{\text{old}}^2 N_i + d_{\text{new}}^2}{N_i+1} \)
29 \( N_i = N_i + 1 \)
30 Update the End-Date of \( P_i \) to the last date of last access in \( S_a \)
31 }

The algorithm starts by finding the closest matching profile to each session, and then uses the matching session to update that profile. If the session does not match any profile, i.e., its similarity with all profiles is not greater than the threshold \( \text{Sim}_{\min} \), then it is classified as a distinct session. In this algorithm, a crisp membership is used, i.e., only the closest profile
is updated, as opposed to updating all profiles close to the session (which would be a fuzzy membership). It can be argued, intuitively, that a crisp or hard membership should yield higher precision at the cost of lower coverage, which, from a business perspective, means that each user is mapped to exactly one detailed profile, instead of many.

Updating the closest matching profile is done by incrementing its cardinality by one, increasing the weight of common URLs, adding new URLs (that are in the session but not in the profile), updating the weight for the remaining URLs and updating the profile’s starting and ending dates. Finally, the scale of the profile is updated by adding the effect of the new session’s distance ($d^2_{\text{new}}$). The weight of each new session is considered to be 1 (unlike the variance definition in (3)), since we are already restricting only very similar sessions to update a matching (and furthermore, the closest) profile.

The complexity of Algorithm 2 can be shown to be

$$O(N_s \times |P| \times \text{Max}(|URL|_P, |URL|_S))$$

where $N_s$ is the number of new sessions, $|P|$ is the number of profiles, $|URL|_P$ is the maximum number of URLs in a profile, and $|URL|_S$ is the maximum number of URLs per sessions. Furthermore, we exploit the fact that Web sessions are extremely sparse (typically fewer than 10 URLs per session), when comparing sessions and profiles as sets. This is reduced even further when applying a threshold on the URL significance weight. Moreover, the number of profiles tends to be small because only strong profiles are mined at any period. Thus, the number of new sessions is what really affects the complexity of the updating algorithm the most, albeit still linearly.

3.4 Discovering distinct profiles

After the old profiles have been updated, only distinct sessions will undergo the pattern discovery process. The HUNC algorithm described in Section 2.3 is used to discover the new patterns, and the output will be denoted as the new profiles. These new trends may
represent a potential new interesting market niche, or a radical change in the current market needs and behavior. The output of the pattern discovery process is a set of new clusters that are represented as a set of URLs. Each cluster has a variance measure $\sigma_i^2$, cardinality measure $N_i$, sum of weights, and a density or fitness. The variance is defined in (3), the cardinality is the number of sessions assigned/closest to this cluster up to period $(t)$ and is donated as $N_i$. The Density $f_i$ of Profile $i$ is defined as:

$$f_i = \frac{\sum_{j=1}^{N_i} w_{ij}}{\sigma_i^2}.$$  

Since the distinct sessions are only part of the original sessions, HUNC’s run time and resource usage are reduced, which increases the scalability of this approach.

### 3.5 Post-processing the distinct profiles

The purpose of post-processing is to formulate the discovered patterns in a way that is more understandable by Humans, and usable by higher-level applications. The post-processing phase is applied only on the newly discovered clusters. During the post-processing phase, each URL is mapped to the closest cluster. The set of all URLs in the same cluster constitutes a profile. A relevance weight is calculated for each URL as in (4), and the profile variance and cardinality are calculated as well - which are the same as in the cluster. A sample profile is shown in Figure 2. This profile has 11 significant URLs with their weight shown on the left next to each URL. It represents 58 sessions in the period 1/30/98 until 2/4/98, and its variance is low which means that the URLs are very close to each other.
3.6 Combining profiles

In this phase, the updated profiles and the newly discovered profiles are combined into one set of profiles that will serve as the new seed of profiles for the next pattern discovery period. Moreover, statistics about the pattern discovery process are collected, such as the number of updated profiles, the number of newly discovered profiles, and the number of distinct sessions. These statistics help in monitoring the performance of the discovery process, and in analyzing the output. For example, they can answer questions such as:

“During which period did the browsing behavior change the most?”

“Did changing the content of the website in period T change the users’ behavior?”

3.7 Interpreting the profiles

Once the browsing behavior of users is summarized in the form of profiles, these profiles could serve as an input to higher-level tools that can analyze the usage patterns and make conclusions that help in a further decision making process. The analysis tools can draw on statistics, graphics, visualization or database querying. The information presented in the profiles could also be used directly in higher level applications (e.g. the list of URLs to implement collaborative filtering recommendations), or to evaluate the profiles quality (e.g. the profile scale or variance). There are also other metrics that are gathered for each pattern.
discovery cycle, and used for a higher level analysis of the profile evolution and interaction, e.g., number of distinct sessions per period. These will be presented in Section 4.

4. Experiments

The proposed framework was applied on two real web site logs: a smaller log from the University of Missouri’s CECS department’s web site, and a much larger log from University of Louisville’s library website (library.louisville.edu). For each web site, a set of evolving usage profiles were extracted, and their quality was evaluated using a set of metrics. This section starts by defining these evaluation metrics in Section 4.1, then discusses the different parameters that will be used and their configurations in Section 4.2. Experiments on both datasets showed similar results but due to paucity of space, only the results for the bigger data set, University of Louisville Library, will be discussed in Section 4.3. Finally, a summary of the experiments and the optimal configuration is presented in Section 4.4.

4.1 Evaluation Metrics

The profiles that result from the pattern discovery process can be evaluated manually, by visiting the URLs and trying to determine whether these links are related. However this evaluation method can be subjective, time consuming, or even outdated since URLs are dynamically changing. Moreover, it is not enough that the profile “descriptions” are plausible; the profiles should form good “clusters”, i.e. they should be compact and separated. Thus, a more reliable and accurate method is to use a set of objective evaluation metrics to assess the quality of profiles. Table 1 lists the evaluation metrics, and their desired and expected ranges, where $N_s$ is the total size of the data. These metrics will be described in the next sections in detail. Table 2 lists the evaluation metrics and a short description that reflects their behavior in the context of web usage mining.
Table 1: Evaluation Metrics Summary

<table>
<thead>
<tr>
<th>Evaluation Metric</th>
<th>Notation</th>
<th>Desired Value</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile Variance</td>
<td>$\sigma_i^2$</td>
<td>Low</td>
<td>[0-1]</td>
</tr>
<tr>
<td>Profile Cardinality</td>
<td>$N_i$</td>
<td>High</td>
<td>[1-$N_s$]</td>
</tr>
<tr>
<td>Matching Sessions</td>
<td>-</td>
<td>High</td>
<td>[1-$N_s$]</td>
</tr>
<tr>
<td>Profiles Count</td>
<td>-</td>
<td>High</td>
<td>[1-$N_s$]</td>
</tr>
<tr>
<td>Profile Pair-wise Similarity</td>
<td>$Sim_{cos}(P_i, P_j)$</td>
<td>Low</td>
<td>[0-1]</td>
</tr>
<tr>
<td>Profile Density</td>
<td>$D_i$</td>
<td>High</td>
<td>[1-$\infty$]</td>
</tr>
</tbody>
</table>

Table 2: Metrics Quality Description

<table>
<thead>
<tr>
<th>Evaluation Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Profile Variance</td>
<td>Low variance reflects high similarity in users’ behavior</td>
</tr>
<tr>
<td>Profile Cardinality</td>
<td>High cardinality means that this behavior is shared by many users</td>
</tr>
<tr>
<td>Matching Sessions</td>
<td>High value reflects similar users’ behavior at consecutive time intervals</td>
</tr>
<tr>
<td>Profiles Count</td>
<td>High value reflects diversity in users’ behavior represented by different patterns</td>
</tr>
<tr>
<td>Pair-wise Similarity</td>
<td>Low similarity emphasizes differences in the users’ behaviors</td>
</tr>
<tr>
<td>Profile Density</td>
<td>High density means high similarity in behaviors of many users</td>
</tr>
</tbody>
</table>

4.1.1 Profile Variance

The profile variance or scale ($\sigma$) was defined in (3), and reflects how much the sessions in that cluster are dispersed. The profile variance is normalized in the range [0-1], and its value approaches 1 as the sessions in the corresponding cluster get further from the cluster prototype, and thus further from each other (i.e. are less similar to each other). Hence, a lower profile variance is desirable. The profile variance is proportionally related to the dissimilarity (difference) between the session and a profile. As the dissimilarity increases, the variance will increase. The dissimilarity was defined in (6). In fact, the cosine similarity can be reformulated using the profile variance as

$$Sim_{cos} = 1 - \sqrt{\sigma}$$ (9)

4.1.2 Profile Cardinality

The profile cardinality is the number of sessions assigned to the corresponding cluster. A
higher cardinality means that this profile is more popular and interesting. However, a higher cardinality profile might also be a result of the “default” cluster that captures all the “noise” in the data, which are sessions that are too different to form any strong and consistent usage patterns. When the profile gets updated, its cardinality is increased. Hence, the cardinality of a profile at any given time is the sum of the number of sessions that the profile has acquired up to and including that time period. The cardinality $N_t$ of a profile $P_i$ at time period $t$ can be written as follows

$$N_t^i = \sum_{j=1}^{t} n_{ij}^j,$$

(10)

where $n_{ij}^j$ is the number of sessions identified to be closest to the profile $i$ at time $j$. Furthermore, for evaluation purposes, the accumulated cardinality is normalized by the sum of all profile cardinalities up to time period $t$, as follows

$$(N_t^i)_{\text{norm}} = \frac{N_t^i}{\sum_{j=1}^{\mid P \mid} N_j^j},$$

(11)

where $|P|$ is the total number of profiles.

4.1.3 Matching vs. distinct sessions

The matching sessions refer to the sessions that matched an existing profile and are thus used to update the properties of that profile, while the distinct sessions are the ones that are not close enough to any of the existing profiles, and therefore they are used to extract new profiles in a new period. A high matching session percentage indicates high quality profiles (that have been discovered up to the previous period), and furthermore reduces the cost of pattern discovery, since fewer (distinct) sessions would have to undergo the mining process.

4.1.4 Number of profiles

Tracking the numbers of profiles discovered and updated at each cycle of the mining
process can give an insight about which time periods have witnessed changes in usage behavior, and what were the trends. Each profile at each cycle can be categorized as Discovered (created from the distinct sessions), Static (was not updated by any matching sessions), or Updated (by matching sessions).

4.1.5 Profile Pair-wise Similarity

The Binary Cosine Dissimilarity defined in (6) will be used for creating a profile-to-profile dissimilarity matrix. The value of 0.01 will be used as the threshold value, i.e., if the difference between two profiles is less than 0.01 then the profiles will be considered similar. The value of 0.01 in the difference means a similarity of 0.9 between profiles as given in (5). A large number of similar profiles indicates a poor clustering result.

4.1.6 Profile Density

Profile density was defined in Section 3.4 as the sum of session weights divided by the profile variance. The weight of each session is now considered to be 1, since we are already restricting only very similar sessions to update the profile. Hence, the profile density \(D^t_i\) at time period \(t\) can be defined as follows

\[
D^t_i = \frac{N^t_i}{\sigma^2_i}, \tag{12}
\]

where \(N^t_i\) is the profile cardinality up to time period \(t\). The profile quality generally increases as its variance decreases as discussed in Section 4.1.1, and a higher cardinality is also desirable as discussed in Section 4.1.2. Hence, a high density is desirable and indicates a high quality profile with high compactness (low variance) and more importance (high cardinality).
4.2 Experimental Configuration

The pattern discovery process depends on a number of options and parameters that affect the usage profiles and their quality. Varying the number and values of these parameters would help in determining the best configuration to be used, and help point out the weaknesses and strengths of the proposed framework. The configuration options and parameters include:

1. Method of discovery: *evolution* (the method used in this paper) or *traditional* (Full) where profiles are discovered from all historic logs at once.

2. Method of matching profiles: *Binary Cosine Similarity* Eq.(5) or *Robust Weight Similarity* Eq.(4).

3. Similarity threshold: used as cut off value to determine how similar a session is to a profile (i.e. if $Sim(S_j, P_i) > Sim_{min}$ then update $P_i$).

4. URL weight threshold: used as cut off value in two phases of the discovery process: the post-processing phase in HUNC discussed in Section 3.5 to make sure that only URLs that are significant enough in the cluster are selected in the final profiles, and in the profile updating algorithm (Algorithm 2) where only significant URLs in the profile are compared to the current sessions.

These parameters and their different values are listed in Table 3. For the URL significance weight threshold, the term `URL_TH` will be used, for the Binary Cosine Similarity threshold, the term `CosSim_TH` is used, and for the Robust Weight Similarity threshold, the term `RobWT_TH` is used.

The HUNC algorithm is used to discover the profiles from distinct sessions. Table 4 shows all the HUNC parameters and their values. Changing the values of these parameters might affect the resulting profiles quality. But they were chosen based on previous experiments on many website logs. Since this paper aims to study the changing usage behavior over the
Table 3: Experimental Configuration

<table>
<thead>
<tr>
<th>Method</th>
<th>Matching Criteria</th>
<th>URL_TH</th>
<th>CosSim_TH</th>
<th>RobWT_TH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evolution</td>
<td>Binary</td>
<td>0.04</td>
<td>0.3</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.04</td>
<td>0.5</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1</td>
<td>0.3</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1</td>
<td>0.5</td>
<td>-</td>
</tr>
<tr>
<td>Robust</td>
<td></td>
<td>0.04</td>
<td>-</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.04</td>
<td>-</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1</td>
<td>-</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.1</td>
<td>-</td>
<td>0.6</td>
</tr>
<tr>
<td>FULL</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

web, and not the performance of HUNC, these parameter values will be the default values for all experiments.

Table 4: HUNC Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min Cardinality</td>
<td>The smallest allowable size of an individual</td>
<td>20</td>
</tr>
<tr>
<td>Min Cardinality Split</td>
<td>If cardinality of individual is less than this, it will not be split</td>
<td>30</td>
</tr>
<tr>
<td>Variance Threshold Split</td>
<td>If individual threshold is less than this, it will not be split</td>
<td>0.1</td>
</tr>
<tr>
<td>Variance Factor</td>
<td>Factor of variance used as threshold for within niche distance</td>
<td>0.5</td>
</tr>
<tr>
<td>Min Fitness Cross</td>
<td>Threshold for identifying valid individuals when restricting mating of good individuals from different niches</td>
<td>1000</td>
</tr>
<tr>
<td>Max Levels</td>
<td>Maximum number of levels of hierarchical clustering</td>
<td>8</td>
</tr>
<tr>
<td>Population Size</td>
<td>The initial population size used in genetic algorithm</td>
<td>20</td>
</tr>
<tr>
<td>CrossOver Probability</td>
<td>The probability of performing crossover</td>
<td>0.0</td>
</tr>
<tr>
<td>Mutation Probability</td>
<td>The probability of parts of chromosome of being mutated</td>
<td>0.0000005</td>
</tr>
</tbody>
</table>

4.3 University of Louisville Library

4.3.1 The Dataset

The logs were collected for requests done to the University of Louisville’s Library main website (library.Louisville.edu) for five consecutive days: from Wednesday February 27th, 2008 until Sunday March 2nd, 2008. There were a total of 364,409 requests, grouped into 14,888 sessions. The profile evolution tracking was done on a daily basis, so there were five time periods, with the data properties shown in Table 5.
Table 5: U of L Library: Dataset Properties

<table>
<thead>
<tr>
<th>Period Date</th>
<th>Access Requests Count</th>
<th>Sessions Count</th>
<th>Percentage of Total Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wed, Feb 27th, 2008</td>
<td>104,794</td>
<td>4,196</td>
<td>28.76%</td>
</tr>
<tr>
<td>Thu, Feb 28th, 2008</td>
<td>92,446</td>
<td>3,657</td>
<td>24.65%</td>
</tr>
<tr>
<td>Fri, Feb 29th, 2008</td>
<td>70,722</td>
<td>2,919</td>
<td>19.41%</td>
</tr>
<tr>
<td>Sat, Mar 1st, 2008</td>
<td>40,834</td>
<td>1,791</td>
<td>11.2%</td>
</tr>
<tr>
<td>Sun, Mar 2nd, 2008</td>
<td>55,613</td>
<td>2,325</td>
<td>15.26%</td>
</tr>
</tbody>
</table>

In the five time periods, around 85% of the requests were “bad access" requests, i.e. irrelevant and noise requests. This observations shows the importance of the preprocessing phase, because all these bad requests would have adversely affected the resulting profiles discovery.

4.3.2 Profile Variances

The profiles evolution over time is shown in Figures 3, 4, 5, and 6. Since the number of profiles is large for some configurations (about 50), only the highest cardinality profiles are shown because their behavior is more reliable. Fig. 3 (b) shows that the majority of profiles maintain a stable low variance, whereas Fig. 3 (a) shows a slow increase in profile variances. Hence, a stricter similarity threshold resulted in better quality profiles when using the Binary Cosine Similarity. Fig. 4 shows that all profiles are of high quality with variance less than 0.3, which is more desirable than the results in Fig. 3. Hence, a higher URL threshold caused more stability and better quality of profiles.
Figs. 5 and 6 show the variance evolution when using the Robust Weight Similarity. The profiles show similar metrics as in Fig. 3. The last configuration in Fig. 6 (b) shows the best quality of profiles, because most profiles began with a small variance, and they kept improving, in contrast with the other configurations, where the variances either remained the same or increased.

An overall conclusion can be drawn that more restrictive threshold values combined with the use of the profile and scale-sensitive Robust Weight Similarity tend to result in discovering very high quality profiles over time. The changes in the existing profile variance and the emergence of new profiles show that the users change their behavior over time, and that the
proposed framework was able to capture these changes.

Figure 5: U of L Library: Profile Variances (Robust Weight Similarity , URL TH(0.04))

![Figure 5: U of L Library: Profile Variances (Robust Weight Similarity , URL TH(0.04))](a) ![Figure 5: U of L Library: Profile Variances (Robust Weight Similarity , URL TH(0.04))](b)

Figure 6: U of L Library: Profile Variances (Robust Weight Similarity , URL TH(0.1))

![Figure 6: U of L Library: Profile Variances (Robust Weight Similarity , URL TH(0.1))](a) ![Figure 6: U of L Library: Profile Variances (Robust Weight Similarity , URL TH(0.1))](b)

To study the overall profile quality, Fig. 7 shows the minimum, maximum, median, and average variance of the final profiles at the end of evolution. The minimum, median, and average variance remained stable and within the same range for all different configurations which indicates that the overall quality of profiles is good. Fig. 7 (a) shows that moving the URL weight threshold value from 0.04 to a more restrictive value of 0.1 caused the maximum variance to drop drastically. On the other hand, the value of the cosine similarity threshold did not seem to have any effect. A low URL threshold value will cause more URLs to be added to the profile. Hence the profile would be matched with more sessions, which risks
reducing its quality. Fig. 7 (b) shows an opposite behavior, where the similarity threshold has more effect on the value of the maximum variance. In contrast to the Binary Cosine Similarity, the Robust Weight Similarity takes into account the profile variance, not just the number of common URLs in the profile and session. Hence, if the profile has low variance, it becomes harder for new sessions to match it. For this reason, changing the number of URLs in the profile by changing the URL threshold value will not have the same affect as it had when using the Binary Cosine Similarity.

Figure 7: U of L Library: Profiles Sigma Aggregates

4.3.3 Profile Evolution

The percentage of each type of profile (discovered vs. updated) for each time period is shown in Fig. 8 for the Binary Cosine Similarity, and in Fig. 9 for the Robust Weight Similarity. The percentages of profiles seem to be similar using the different configurations, with the majority of profiles being updated. However, a higher similarity threshold value caused more restricted profile updates, with more new profiles being discovered or equivalently, more old profiles remaining the same, as shown in Figs. 9 (b) and (d). From the users’ behavior perspective, having the majority of profiles being updated means that their activity is changing slowly over time, with small part of their activity to be completely new (which is reflected by discovering new profiles). Hence, this further proves that profiles were
able to capture the changes over time while maintaining high quality.

Figure 8: U of L Library: Profile Counts (Binary Cosine Similarity)

(a) ![CosSim(0.3) URL TH(0.04)](image)
(b) ![CosSim(0.5) URL TH(0.04)](image)
(c) ![CosSim(0.3) URL TH(0.1)](image)
(d) ![CosSim(0.5) URL TH(0.1)](image)

Figure 9: U of L Library: Profile Counts (Robust Weight Similarity)

(a) ![RobWT(0.3) URL TH(0.04)](image)
(b) ![RobWT(0.6) URL TH(0.04)](image)
(c) ![RobWT(0.3) URL TH(0.1)](image)
(d) ![RobWT(0.6) URL TH(0.1)](image)

4.3.4 Matching vs. Distinct sessions

Fig. 10 shows the percentages of matching and distinct sessions for each time period when using the Binary Cosine Similarity. All configurations show that at each time period, the percentage of matching sessions was between 15% and 25%. Fig. 11 shows the number of matching and distinct sessions when using the Robust Weight Similarity. Figs. 11 (a) and (c) show a similar behavior where the vast majority of sessions got matched, which is compatible with the findings in Fig. 9 (a) and (c). Figs. 11 (b) and (d) show a completely opposite behavior to Figs. 11 (a) and (c) where the majority of sessions are treated as distinct. This is reflected in Figs. 9 (b) and (d) where some profiles were distinct. However having high percentage of distinct sessions doesn’t mean high number of distinct profiles being discovered, because these distinct sessions might be similar and account to only few profiles. And this can be explained as changes in user behavior over time, but these changes
are similar to each other, i.e. users are interested in the same new links at the same time period.

Figure 10: U of L Library: Matching vs. Distinct Sessions (Binary Cosine Similarity)

![Figure 10: U of L Library: Matching vs. Distinct Sessions (Binary Cosine Similarity)](image)

Figure 11: U of L Library: Matching vs. Distinct Sessions (Robust Weight Similarity)

![Figure 11: U of L Library: Matching vs. Distinct Sessions (Robust Weight Similarity)](image)

The overall trend of matching sessions is shown in Fig. 12, which shows that for the Binary Cosine Similarity, the percentage of matching sessions is similar for all configurations. However, for the Robust Weight Similarity, the percentage gets much higher when the matching threshold is lower.

Figure 12: U of L Library: Matching Sessions Percentage

![Figure 12: U of L Library: Matching Sessions Percentage](image)
4.3.5 Profile Cardinality

In Figure 13, the cardinality of the profiles, with maximum variance (thus lowest quality) and minimum variance (thus highest quality), is plotted for each configuration. Fig. 13 (a) presents the results when using the Binary Cosine Similarity, showing that the URL weight threshold has most effect on the high quality profile (a high URL threshold reduces cardinality), whereas the similarity threshold value was the main factor affecting the lower quality profile. Fig. 13 (b) shows the results when using the Robust Weight Similarity. The cardinality of the high quality profile (with low variance) remains the same for all different configurations. However, the cardinality of the profile with maximum variance is sensitive only to the similarity threshold value, where a lower threshold value causes higher cardinality.

Figure 13: U of L Library: Cardinality of Max and Min Sigma

![U of L Library: CosSim(Cardinality)](a)

![U of L Library: RobWT(Cardinality)](b)

The cardinality percentage of each profile at each time period are plotted when using the Binary Cosine Similarity and Robust Weight Similarity in Figs. 14 and 15, respectively. For all configurations, the majority of profiles witnessed a slight decrease in their cardinality over time, which means that there is a high competition between profiles over acquiring new sessions. Only two profiles are of poor quality: one in Fig. 15 (a) and one in Fig. 15 (c), where they have a high cardinality and variance, which means that they are most likely the “default” profiles that end up acquiring all outlying sessions that did not match any of the
good clusters. The default profile is an expected artifact of the session to cluster assignment process, and is easy to recognize thanks to the evaluation metrics. Despite the fact that most profiles has less than 10% cardinality, a lot of profiles were generated which means diversity of users’ behavior. Section 4.3.7 will elaborate more on the similarities among profiles.

Figure 14: U of L Library: Profile Cardinalities (Binary Cosine Similarity)
4.3.6 Profile Counts

The number of profiles generated from the usage behavior is shown in Fig. 16. Since HUNC is a hierarchical algorithm, a high number of profiles indicates more detailed profiles, which is desirable. However, a larger number of profiles typically causes a lower cardinality in the profiles, as expected. The Robust Weight Similarity Threshold value seems to be the only factor affecting the number of profiles, where a stricter value (0.6) generated more detailed profiles.
4.3.7 Profile Pairwise Similarity

The maximum, minimum, average, and median profile pairwise similarities are plotted in Fig. 17, as well as the percentage of duplicate profiles. Duplicate profiles are those whose dissimilarity from each other is less than 0.01. Both Figs. 17 (a) and (b) show similar trends, where the maximum and median dissimilarity is 1 (which means that the majority of profiles are completely different), while the average difference is around 0.7 which can be translated to profiles which are only 16% similar based on (5) and (6). The percentage of duplicate profiles is almost zero in all cases, which supports our desire of developing high quality, distinct, and evolving profiles through different time periods, which translates to the dynamic nature of users’ behavior.

Figure 17: U of L Library: Profile Pair-wise Similarity Aggregates
### 4.3.8 Profile Density

The profile Density evolution for the Binary Cosine Similarity and Robust Weight Similarity are shown in Figs. 18 and 19 respectively. The majority of profiles seemed to improve their density over time, with the exception of a few profiles as shown in Fig. 18 (a), which means that their variance kept increasing (since the cardinality cannot decrease over time).

Figure 18: U of L Library: Profile Densities (Binary Cosine Similarity)
4.3.9 Evolution vs. traditional or full discovery

All the profiles generated when using the traditional discovery approach had variance less than 0.2. And their density was high as shown in Fig. 20. Some of the profiles generated using the evolution-based approach showed a comparable density as was seen in Fig. 19.
Fig. 21 compares the aggregate metrics for both discovery approaches (traditional vs. evolution). The median, maximum, and average variances are lower when using the traditional approach than when using the evolution-based approach. However, Fig. 21 (b) shows that using the Robust Weight Similarity in tandem with evolution-based mining has discovered profiles (at least one) whose variance is the lowest (hence highest quality) compared to all other approaches.
4.3.10 Analysis of the Evolution of the profiles metrics

Monitoring the profiles metrics over time described in previous sections gives some insight on the evolution process of profiles. Evaluating the quality of profiles at each time period independently might result in misjudging the profile quality, whereas tracking the profile’s metrics over different time periods would result in a more accurate description of the profiles’ behavior, hence adding a model of profiles’ metrics evolution over time would improve the quality of the evaluation process (which will be conducted as a future research). For example, Profile 7 in Figure 19 (a) had a low density in the first two time periods compared to Profile 0, which would falsely indicate a lower quality, however, if compared along all time periods it shows that the density of Profile 7 is increasing whereas the density of Profile 0 is stable, hence Profile 7 is attracting more sessions (higher cardinality) while these sessions are closer to the profile prototype (lower variance) over time (i.e. Profile 7 became more specialized in a certain users’ behavior over time).

4.4 Experiments Summary

The results of the experiments showed that using the Robust Weight Similarity with a high similarity threshold value of 0.6, and a strict URL weight threshold of 0.1, gives the best results. The compactness of the profiles for this configuration kept improving over time, as seen in Fig. 6 (b). A lower profile variance means that the sessions are more similar to the profile. The high quality of the profiles has made these profiles able to match more incoming new sessions, as seen in Fig. 9 (d), where the majority of the profiles got updated. Moreover, this configuration resulted in more detailed profiles, where the average number of profile was around 50% as seen in Fig. 16. These detailed profiles were also very different from each other (based on the profile pair-wise similarity measure), as seen in Fig. 17 (b), where the percentage of duplicate profiles was almost 0. The profile quality using this configuration was further confirmed when the density of profiles was found to be increasing over time, as seen in Fig. 19 (d).
Conclusion

Online organizations and businesses have recently started using Web Usage Mining techniques to help track the behavior of their website users, and hence, better understand and satisfy their users’ needs and maintain their loyalty. However, only very few web usage mining algorithms were able to deal with the changing nature of the web user activities, and a very limited body of literature has even addressed this evolution at all. We have presented an innovative and scalable framework that is capable of capturing the changing group behavior of the users of a website by mining its Web access log data over several periods, and by evaluating the results of this web usage mining using an arsenal of objective evaluation metrics that study the quality and characteristics of the profiles over time. Unlike our previous work in [12], the work in this paper naturally threads together the profiles over several periods, instead of having to do so post-mining. Also, unlike [12], we perform the threading pre-mining, which means that we match and thus discard large portions of the web access data during the pre-processing phase itself, leaving only a smaller portion of data to undergo rediscovery, thus improving scalability. The proposed framework develops a set of evolving profiles that represent the usage patterns as profiles, with each profile in turn consisting of a set of URLs, and each URL having its own degree of significance in that profile. These profiles offer a simple way to model the users’ behavior at any given time. In the future, we plan to tie our web usage mining directly into a model-based collaborative filtering recommendation module. We will also investigate alternative and faster optimization techniques for the profile discovery (clustering), as well as methods to model the evolution of the evaluation metrics over time.
Bibliography


