Mining and Predicting Users Clickstream Patterns from Noisy Interleaving Clicks

A. Alamoudi*, F. Fekri*, M. Mohandes**, B. Liu**
*School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, USA
{aalamoudi31, faramarz.fekri}@gatech.edu
** Electrical Engineering Department, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia
mohandes@kfupm.edu.sa, bo.liu.777@gmail.com

***Abstract*—With the recent advancement in technology and a vast amount of information available, research in pattern mining has started to attract more attention. Specifically, various techniques have been developed for clickstream mining, which is a specific type of sequential pattern mining, to discover the underlying patterns from the Internet user clickstream. Due to the complexity of clickstream patterns, many of the existing works applied sequential pattern algorithms to generate an exponential candidate space of patterns with respect to patterns letters. Further, those patterns were generated in a noiseless environment. To address this problem, we focus on a nonoverlapping clickstream pattern mining task with noisy interleaving clicks between the clickstream patterns letters. Additionally, we are interested in labeling the extracted patterns in the user browsing history. A modified suffix tree is proposed to extract those patterns with the exact occurrence in the user noisy database. Following this, we model the user browsing behavior via a Hidden Markov Model (HMM) to capture the dependencies between the extracted patterns and then predict the future clickstream patterns. Experimental results on both real-life and synthetic datasets show that our proposed algorithms outperform the state-of-the-art benchmarks in efficiency and prediction accuracy.**

***Index Terms*—Clickstream, Mining Patterns, HMM, Predicting Users Patterns**

I. Introduction

Pattern mining is considered an important task for discovering useful information or knowledge from data and it is one of the well-known research areas in various fields such as bioinformatics [1], online marketing [2], sightseeing recommendation [3], and task analysis [4]. In many domains, the sequential ordering of events cannot be ignored. For instance, a user web clickstream is a sequential record of events (clicks) when the user is browsing the web. Likewise, in network security, the order of events to detect the intrusion is needed. Given that, sequential pattern mining aims to generate frequent patterns by taking into account chronological orders of the events and the dependency between them.

Clickstream is a particular sequential data and it is a strict stream of events in which only one event occurs at any given time, although events can be repeated in the same sequence. Particularly, each action that the user performs on the web represents an event in the clickstream. Therefore, mining and analyzing clickstream patterns may provide an insight into the navigation paths and hence help improve the web design. Also, it can be used to prefetch the predicted pages to be visited next. As a result, it enhances the performance of the web application. Moreover, clickstream patterns of mobile users can be used for WiFi offloading where the traffic load is shifted from expensive cellular networks to much cheaper WiFi networks. Thus, it alleviates the burden on the transmission of the data as well as mitigates network operation costs and data costs on network operators and mobile users, respectively.

Many of the previous contributions on Clickstream pattern mining [5], [6] focus on extracting patterns based on a predefined measure of frequency. However, mining those patterns raises many challenges as not only the number of the discovered patterns is huge but also their run time is costly as they require multiple passes of the dataset. Moreover, [7], [8] focus on finding the patterns that satisfy various predefined constraints to reduce the time consumption of the mining and generate a compact set of patterns. However, those algorithms generate a very restrictive set of patterns and may not be applicable for noisy sequences. Besides mining the clickstream patterns, machine learning models were applied to clickstream logs for predicting user behavior over the internet. For instance, in [9], the prediction of user intention was modeled using a supervised learning problem. In addition, many works applied Hidden Markov models to determine the frequency of the most visited webpages during the session [10], [11]. However, all these models assume that the clickstreams are generated from a noiseless environment.

Departing from previous works, we focus on how to mine and predict the user clickstream patterns given that user clicks might be interleaved with other irrelevant clicks. It is not necessary for all dependent clicks in a pattern to occur successively in the browsing history. In reality, the user may interrupt his/her routine clickstream pattern to perform other tasks which do not relate to the current pattern. For example, consider a clickstream pattern of checking the news web application. For this pattern, the user has to click on the news type from the dashboard, select the particular section tab then click on the interesting news. However, in some cases, after clicking on the particular section tab, the user may check the email in-box, compose a message or even leave the news application and open the news application on a different tab. Further, the user clicks might be interleaved, the user clicks may be preceded by relevant clicks. These clicks include other preceding clicks in the same pattern and clicks that set up

the environment before the current pattern occurs. Dependent clicks always appear before the current click occurs while clicks for setting up the environment may occur only once. For instance, paying the rent bill from the bank account requires the user to type the bank website on the browser address bar, enter the user login credentials, select the account type, pay the required amount, and logout. Here, all clicks from login to logout are dependent on each other while typing the bank website, and any other necessary clicks before them are setting up the environment. In addition to the aforementioned reasons, user clicks may have been followed by other relevant clicks. For some web applications, the user may execute a sequence of clicks that constitutes the clean-up phase of the accomplished clickstream pattern. In the previous example of paying the rent bill, the logout click needs to be followed by some other clicks on the web application. Therefore, mining incomplete clickstream patterns, i.e. missing a few clicks, would not be useful for learning and predicting purpose. On the other hand, mining clickstream patterns with few extra clicks would be irrelevant and exert an additional overhead cost.

The contributions of our paper are summarized as follows:

- We propose a modified suffix tree for mining frequent clickstream patterns for web browsing users in a noisy environment where some irrelevant clicks are inserted into a pattern, and between different clickstream patterns in the clickstream. This new approach can eliminate those irrelevant clicks, extract the underlying patterns efficiently while maintaining the anti-monotonicity property, then identify them in the browsing history.
- Based on the extracted patterns, we model the user behavior over the Internet via HMM to learn and predict the future user navigational pattern.
- We evaluate the proposed algorithms through extensive experiments using both synthetic and real-world datasets.

II. User Clickstream Patterns Mining via Modified Suffix tree

Considering multiple web domains, we are given data from N Sessions collected from the user past activities. Each session $S^{(i)}$ consists of the order list of n_i clicks, i.e $S^{(i)} = \{a_1^{(i)}, \cdots, a_{n_i}^{(i)}\}$ where $a_j^{(i)} \in \{a_1, a_2, \cdots, a_L\}$ and L is the total number of possible clicks in the browsing history. Further, let's assume that there is a K higher order level of clickstream patterns (hereafter referred as patterns) $A_i \in \Gamma$ where $\Gamma = \{A_1, \cdots, A_K\}$ that each pattern A_i consists of an ordered list of corresponding clicks. Thus, given unlabeled sequences of clicks $a_j^{(i)}$ in each session S^i for $i = \{1, 2, \cdots, N\}$, we propose to find the correct set of K patterns $\Gamma = \{A_1, \cdots, A_K\}$ and group the clicks in each $S^{(i)}$ into sequences of some patterns from Γ. This problem is challenging since the click sequences may be interleaved with other irrelevant actions. For example, two imaginary click sequences $\{a_1, a_5, a_{10}, a_4\}$ and $\{a_1, a_{12}, a_4\}$ may both belong to the same pattern $A = \{a_1, a_4\}$. Further, the number of patterns is unknown in advance. Hence, mining and labeling the user patterns can be accomplished by resorting on a modified suffix tree based on shifted time window. Specifically, at each time window, different candidate patterns are generated by identifying the potential noisy clicks. Then, the extracted patterns are labeled as one of the K patterns in the user browsing history.

A. Modified Suffix Tree Structure

In this work, we use suffix tree as a model for generating a dictionary of subsequences from the user activities over the Internet. Suffix tree is a well-known model for many applications such as text searching and matching. It is also known for generating a linear set of candidates. In our proposed model, instead of using the standard suffix tree, we intend to use a modified version to incorporate the interleaving clicks of the user while browsing any domain over the Internet. Those clicks can be categorized as interrupting the user pattern clicks (hereafter referred to as noisy clicks) or setting /finishing up the session clicks (hereafter referred to as non-pattern clicks). As such we assume that there might exist some noisy clicks within the user pattern clicks as well as non-pattern clicks between the user patterns. In this framework, we consider two scenarios. In the first one, we assume at most a single noisy click per pattern. In the second scenario, multiple noisy clicks per pattern may exist.

1) Generating Non-overlapping Candidate Patterns Dictionary: In this setup, we generate candidate patterns to form a dictionary within a specified time window. This time window specifies the maximum length of the generated patterns. Particularly in the case of at most one noisy click per pattern, we generate a set of candidate patterns using suffix tree by considering the first click in the time window as the root of the tree and each click in the time window is potentially a noisy click, as depicted in Table I. Next, we shift the time window by one click and we repeat the process until we cover the entire browsing history of the user. For each generated candidate pattern, we consider the position index of the first and last click of the pattern in the user browsing session and the noisy click. As such, we remove any newly generated candidate patterns when their positions overlap with the existed ones in the dictionary to guarantee that all the candidate patterns in the dictionary are non-overlapping candidate patterns.

In a multiple noisy clicks scenario, again we generate the candidate patterns within the specified time window by resorting on the suffix tree. However, applying the same approach for a single noisy click would not be beneficial because the number of noisy clicks within the pattern is unknown. However, using the fact that the clicks of the user patterns occur in sequential order and many of the user patterns may have no noisy clicks, we can generate non-overlapping candidate patterns by considering their positions in the user browsing session as shown in Table II.

2) Labeling The Frequent Patterns: We further label the sequence of unlabeled clicks $a_j^{(i)}$ in each session $S^{(i)}$ for $i = \{1, 2, \cdots, N\}$ to one of the frequent generated patterns. For the case of only single noise insertion, we calculate the

frequency of the generated candidate patterns from the created dictionary as shown in Table I. Next, we sequentially assign the most frequent candidate pattern within the time window as a pattern for the user by maintaining the anti-monotonicity property, as explained in the following. let $P_1 = \{a_1, a_5, a_{10}\}$ and $P_2 = \{a_1, a_5\}$ are the two most frequent candidate patterns within a particular time window and let f_1 and f_2 indicate the frequencies of them, respectively. If $f_1 \geqslant f_2$ we select P_1 as a pattern otherwise we select P_2 as a pattern. In this work, we label the click sequence within the specified time window as either A_i or $\widehat{A_i}$ in order to understand and learn the user behavior. We use A_i to indicate the pattern is generated without inserting any noisy click from the user web activities and use $\widehat{A_i}$ to indicate that the pattern is generated with interleaving noisy clicks. Then, we label the remaining set of clicks in the session as NP if the number of the remaining clicks is less than the size of the time window, where NP indicates non-pattern clicks. Otherwise, we shift the time window by the length of the pattern A_i, and since we assume that there might be some non-pattern clicks, those clicks would have a low frequency in the user history. Hence we keep shifting the time window until we observe a spike in the frequency within the generated patterns in the particular time window. Again, we label the click sequence within the time window as A_j or $\widehat{A_j}$ by maintaining the anti-monotonicity property, and the clicks between the labeled patterns will be considered as NP as shown in Table I.

In a multiple noisy clicks scenario, we calculate the frequency of the generated candidate patterns from the created dictionary in Table II. Then the patterns that satisfy the threshold α will be elected as patterns for the user. Following this, we will trace the click positions of the patterns in the user browsing history in order to label the clicks. If the pattern symbols occurred sequentially without any noisy clicks insertions, it will be labeled as A_i otherwise it will be labeled as $\widehat{A_i}$. In addition, applying anti-monotonicity property will be considered while labeling the clicks. Then, the rest of the clicks that are not participating in the labeling process will be labeled as NP.

III. Modeling The User Web Browsing Behavior

Next, our goal is to model the user web browsing behavior from his/her clickstream using which we predict the next pattern of clicks that will be generated by the user in the future. Note that to develop a reliable prediction engine for the user, we need to learn the statistical properties of the patterns and whether or not the user patterns are interleaved with other irrelevant clicks. This challenging unsupervised learning problem can be accomplished by using models such as Hidden Markov Models (HMM) [12]. Particularly, we need to learn the joint distribution $\mathbb{P}(\mathbf{X^n}, \mathbf{Y^n}, \widehat{\mathbf{Y^n}})$ of unobserved states $\mathbf{X^n} = \{X_1, \cdots, X_n\}$ (where each $X_j \in \{1, 2, \cdots, K\}$), the latent true patterns $\mathbf{Y^n} = \{Y_1, \cdots, Y_n\}$ (where each Y_j in our case is in Γ), and the interleaved observed patterns $\widehat{\mathbf{Y^n}} = \{\widehat{Y_1}, \cdots, \widehat{Y_n}\}$ (where each $\widehat{Y_j} \in \{\widehat{A_j}, A_j\}$ for $j \in \{1, 2, \cdots, K\}$). We assume that the state transition has Markovian property and each observed pattern depends only on the current state as shown in Fig. 1. Training of the HMM can be performed by approximately estimating the maximum likelihood (MLE) parameters using the Expectation Maximization algorithm (EM).

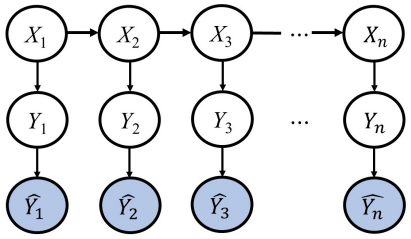

Fig. 1: HMM structure for learning the user behavior

A. User Behavior Prediction Structure

Based on the theory of Hidden Markov Models, HMM is widely used for learning and predicting the future state over time series by exploiting the initial probabilities of all the hidden states, the transition probability matrix between the hidden states and the emission probability matrix between the hidden state and the observed states. In our proposed model, instead of using the standard HMM, we use a modified version of HMM in order to adapt it to the interleaved patterns. Thus, for a sequence of length n, the joint distribution $\mathbb{P}(\mathbf{X^n}, \mathbf{Y^n}, \widehat{\mathbf{Y^n}})$ can be parametrized as $\mathbb{P}(\mathbf{X^n}, \mathbf{Y^n}, \widehat{\mathbf{Y^n}}|\theta) = \mathbb{P}(X_1; \pi) \prod_{i=2}^{n} \mathbb{P}(X_i|X_{i-1}; M) \prod_{i=1}^{n} \mathbb{P}(Y_i|X_i; \phi) \prod_{i=1}^{n} \mathbb{P}(\widehat{Y_i}|Y_i; \mu)$, where π, M, ϕ, μ are the learning parameters of the initial state distribution, the state transition probability matrix, the states-latent patterns emission probability matrix and the latent patterns-observed patterns emission probability matrix, respectively. Formally, given an observed sequence, we would like to compute the likelihood that the observed sequence is generated by the model. Hence, we can learn the model parameters by resorting on the forward-backward approach where the observed sequence can be divided into two sub-sequences, past sub-sequence $\{\widehat{Y_1}, \cdots, \widehat{Y_t}\}$, and future sub-sequence $\{\widehat{Y_{t+1}}, \cdots, \widehat{Y_n}\}$. The forward algorithm calculates the joint probabilities between pattern label is being in a state i at time t, $X_t(i)$, and the emitted patterns until time t, $\{\widehat{Y_1}, \cdots, \widehat{Y_t}\}$. Whereas, the backward algorithm calculates the posterior probabilities of the emitted patterns from time $t+1$ to n, $\{\widehat{Y_{t+1}}, \cdots, \widehat{Y_n}\}$, given the pattern label is being in a state i at time t, $X_t(i)$. More formally, the forward-backward formulations are shown in (1) and (2)

$$\begin{aligned} \alpha_t(i) &= \mathbb{P}(\widehat{Y_1}, \cdots, \widehat{Y_t}, X_t = i) \\ &= \sum_{j=1}^{K} \alpha_{t-1}(j) \mathbb{P}(X_t = i|X_{t-1} = j) \mathbb{P}(\widehat{Y_t}|Y_t = A_i) \\ &\quad \mathbb{1}[Y_t = A_i]. \end{aligned} \tag{1}$$

TABLE I: Generating the candidate patterns and labeling the user browsing activities in single interleaving noisy click setup

User activities=[abcfabdc], time window= 4				
Timestamp (t)	noisy click	Generated candidate pattern	Location	Frequency
1	f	abc	f,1,3	1
	c	abf	c,1,3	1
	b	acf	b,1,4	1
2	a	bcf	a,2,4	1
⋮	⋮	⋮	⋮	⋮
5	b	adc	b,5,8	1
Labeled user activities=[$A_i np\widehat{A_i}$]				

TABLE II: Generating the candidate patterns and labeling the user browsing activities in multiple interleaving noisy clicks setup

User activities=[abcfabdcabc], time window= 3			
Timestamp (t)	Generated candidate pattern	Location	Frequency
1	abc	1,3	1
2	bcf	2,4	1
⋮	⋮	⋮	⋮
9	abc	9,11	2
$\alpha = 1$, Elected patterns=[abc] and Labeled user activities=[$A_i np\widehat{A_i} A_i$]			

$$\begin{aligned}\beta_{t-1}(i) &= \mathbb{P}(\widehat{Y_t}, \cdots, \widehat{Y_n} | X_t = i) \\ &= \sum_{j=1}^{K} \beta_t(j) \mathbb{P}(X_t = j | X_{t-1} = i) \mathbb{P}(\widehat{Y_t} | Y_t = A_i) \\ &\quad \mathbb{1}[Y_t = A_i],\end{aligned} \tag{2}$$

where $\alpha_t(i)$ and $\beta_t(i)$ denote the forward and backward probabilities at time t, respectively, for $t = \{1, 2, \cdots, n\}$ and $i \in \{1, 2, \cdots, K\}$. Next, to learn the parameters of the model, we resort on EM algorithm to iteratively find the maximum likelihood estimation. Using the definitions of forward and backward probabilities, we can calculate auxiliary variables $\gamma_t(i)$, $\xi_t(i,j)$, $\eta_t(i, A_j)$ and $\lambda_t(A_i)$. These variables represent the posterior probabilities of the pattern label at state i , the posterior probabilities between adjacent pattern label at state i and state j, the posterior probabilities between the pattern label at state i and the latent true pattern at state A_j, and the posterior probabilities of the latent true pattern at state A_i, respectively. More specifically, we have

$$\gamma_t(i) = \mathbb{P}(X_t = i | \widehat{Y_1}, \cdots, \widehat{Y_n}) = \frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{K} \alpha_t(j)\beta_t(j)} \tag{3}$$

$$\begin{aligned}\xi_t(i,j) &= \mathbb{P}(X_t = i, X_{t+1} = j | \widehat{Y_1}, \cdots, \widehat{Y_n}) \\ &= \alpha_t(i) \mathbb{P}(X_{t+1} = j | X_t = i) \\ &\quad \mathbb{P}(\widehat{Y_{t+1}} | Y_{t+1} = A_j) \mathbb{1}[Y_{t+1} = A_j] \beta_{t+1}(j),\end{aligned} \tag{4}$$

$$\begin{aligned}\eta_t(i,j) &= \mathbb{P}(X_t = i, Y_t = A_j | \widehat{Y_1}, \cdots, \widehat{Y_n}) \\ &= \begin{cases} \sum_{m=1}^{K} \alpha_{t-1}(m) \\ \mathbb{P}(X_t = i | X_{t-1} = m) \mathbb{1}[Y_t = A_i] \\ \mathbb{P}(\widehat{Y_{t+1}} | Y_t = A_i) \beta_t(i) & i = j \\ \\ 0 & i \neq j, \end{cases}\end{aligned} \tag{5}$$

$$\begin{aligned}\lambda_t(A_i) &= \mathbb{P}(Y_t = A_i, \widehat{Y_1}, \cdots, \widehat{Y_n}) \\ &= \begin{cases} \sum_{m=1}^{K} \alpha_{t-1}(m) \\ \mathbb{P}(X_t = i | X_{t-1} = m) \mathbb{1}[Y_t = A_i] \\ \mathbb{P}(\widehat{Y_t} | Y_t = A_i) \beta_t(i) & i = j \\ \\ 0 & i \neq j. \end{cases}\end{aligned} \tag{6}$$

Then, using these variables we can calculate the HMM parameters using equations (7) through (10)

$$\pi_i = \mathbb{P}(X_0 = i) = \frac{1}{L} \sum_{l=1}^{L} \gamma_0^l(i) \tag{7}$$

$$\begin{aligned}M_{i,j} &= \mathbb{P}(X_{t+1} = j | X_t = i) \\ &= \frac{\sum_{l=1}^{L} \sum_{t=1}^{n} \xi_t^l(i,j)}{\sum_{j=1}^{K} \sum_{l=1}^{L} \sum_{t=1}^{n} \xi_t^l(i,j)}\end{aligned} \tag{8}$$

$$\begin{aligned}\phi(i,j) &= \mathbb{P}(Y_t = A_j | X_t = i) \\ &= \frac{\sum_{l=1}^{L} \sum_{t=1}^{n} \eta_t^l(i,j)}{\sum_{j=1}^{K} \sum_{l=1}^{L} \sum_{t=1}^{n} \eta_t^l(i,j)}\end{aligned} \tag{9}$$

$$\begin{aligned}\mu(A_i, u) &= \mathbb{P}(\widehat{Y_t} = u | Y_t = A_i) \\ &= \begin{cases} \frac{\sum_{l=1}^{L} \sum_{t=1}^{n} \lambda_t^l(A_i) \mathbb{1}[\widehat{Y_t^l} = u]}{\sum_{l=1}^{L} \sum_{t=1}^{n} \lambda_t^l(A_i) \mathbb{1}[\widehat{Y_t^l} = A_i] + \lambda_t^l(A_i) \mathbb{1}[\widehat{Y_t^l} = \widehat{A_i}]} \\ \qquad\qquad u \in \{A_i, \widehat{A_i}\} \\ \\ 0 \qquad\qquad \text{otherwise,} \end{cases}\end{aligned} \tag{10}$$

where t and L are the timesteps, and the number of iterations, respectively. The model is built based on the training dataset and validated using the testing dataset. Specifically, we validated our model based on the accuracy performance

of predicting the probability distribution of the user pattern at $t+1$ as

$$\mathbb{P}(\widehat{Y_{t+1}}|\widehat{Y_1},\cdots,\widehat{Y_t}) \propto \mathbb{P}(\widehat{Y_1},\cdots,\widehat{Y_t},\widehat{Y_{t+1}}) \quad (11)$$
$$= \sum_{Y_{t+1}}\sum_{X_{t+1}}\sum_{X_t}\mathbb{P}(\widehat{Y_{t+1}}|Y_{t+1})\mathbb{P}(Y_{t+1}|X_{t+1})$$
$$\mathbb{P}(X_{t+1}|X_t)\mathbb{P}(X_t|\widehat{Y_1},\cdots,\widehat{Y_t})$$

IV. Evaluation

In this section, we conduct several experiments on the real world and synthetic datasets to validate and demonstrate the effectiveness of our mining and predictive models. We compare our model with a set of benchmark references. In mining the user patterns setup, We compare our model with CM-WSPADE [13] which is used for mining weighted sequential patterns as well as NOSEP [8] that is used for mining non-overlapping sequential patterns with predefined gaps. Pertaining the predictive setup, we evaluate the accuracy of our model with the Long short-term memory (LSTM) model in [14] and SlidingWindow-MC algorithm [15] as benchmarks for top-n predicted patterns.

A. Generating Ground Truth Patterns from Synthetic and Real-World Datasets

The web content today is highly dynamic and hence it is challenging to mimic the user browsing behavior. However, consistent user actions can be learned from the past and applied to consistent content layouts. By the usability principle of consistency [16], website layouts should follow the same design templates, despite changes in the content, to ensure high usability of the website. The consistency of a web-page layout is maintained through an HTML Document Object Model (DOM) [17] tree. The user may perform a sequence of DOM actions, separated by non-DOM actions such as opening the browser, opening a tab, typing a URL in the address bar, clicking on a bookmark, etc. While DOM actions depend on the result of a previous DOM action, they do not have any dependency on non-DOM actions. Following this, we extract the DOM tree of multiple web domains to generate a set of patterns for a particular user where the clicks in the patterns are obeying the path rules in the DOM tree. In this study, we generated patterns with varying sizes per user. Using the generated patterns, we created the browsing history of the user by establishing the transition matrix between the patterns, assuming that the transition between the patterns has a Markovian property. Further, each pattern in the browsing history is followed by arbitrary non-pattern clicks. Then, we considered the case that some patterns are probabilistically interleaved by noisy clicks. For this work, we first consider the case of only a single inserted noisy click per pattern followed by multiple inserted noisy clicks per pattern.

Next, we used a real-world dataset that is collected by wireless provider from over 1000 cell tower locations for more than 5000 mobile users. This dataset contains the web access log of the mobile users for the period of one month. Each access log in the dataset is accompanied by the encrypted subscriber ID, session time, traffic volume, cell tower ID, and website address. In this setup, we first assumed each web access log is a click initiated by the user and extracted the web access logs of the top ten users who have rich browsing history. We divided the browsing history for each user into multiple sessions based on the time elapsed between the web access logs and we selected the time threshold for each user seperatly. After that, we passed the generated sessions into Prefixspan [18] to extract the underlying patterns based on the provided relative support. According to that, any unlabeled clicks in the user browsing history were assumed to be non-pattern clicks. Again, we assumed that the transition between the patterns has a Markovian property and each click depends only on the current pattern. Lastly, we probabilistically inserted some patterns with noisy clicks for the case of single insertion per pattern and multiple insertions per pattern.

B. Simulation Results for Mining User Pattern

We evaluated all the models on each user and we reported the average results among all the users. In this framework, we examined all the models on 5% and 10% noise rates. Based on [19], we considered the pattern sizes between 3 and 10 clicks for mining the user patterns. We utilized the precision, recall, and F1 score as measures of performance. Also, we conducted extensive experiments on a synthetic dataset and due to page limitations, we highlighted the results of the real dataset. Table III shows the performance of mining users patterns for our proposed model and the benchmarks for varying noise rate insertion in the setup of a single interleaving noisy click and multiple interleaving noisy clicks. Our proposed model based on suffix tree achieves the best performance among the benchmark methods in terms of precision and F1 score on both setups. We also noted that our model is robust against the variation of noise rate. However, the performance of the benchmark models is degraded when the rate of noise insertion is increased particularly in the multiple noisy clicks setup.

C. Simulation Results for Predicting User Patterns

We first trained our proposed model on 70% of the training data and evaluated on the test data for each user. Then, we reported the average performance among all users. Since all the extracted pattern are relevant, we utilized the hit rate as a measure of performance accuracy for all models. Fig. 2 and Fig. 3 show the performance of our model in comparison with the benchmarks for the hit rate accuracy on top n most probable candidates. Further, we showed the superiority of our model when we compare it with the SlidingWindow-MC algorithm. Particularly, our proposed model achieved an accuracy of 60% in comparison to 37% in the SlidingWindow-MC model. Additionally, the results show that our proposed model could achieve an accuracy of $\sim 70\%$ on the real dataset for multiple noisy clicks setup. It also shows a slightly better performance of our model than the LSTM model. Although LSTM is widely used for learning and capturing the dependency over time series, these simulation results suggest that our proposed model was a better fit for the datasets.

TABLE III: Performance results for mining user patterns in single and multiple interleaving noisy click setups using real dataset

Noise rate	Models	Noise interleaving setups					
		(Single)			(Multiple)		
		Precision	Recall	F1 score	Precision	Recall	F1 score
5%	Modified Suffix Tree	0.9	0.75	0.82	0.8	0.53	0.64
	NOSEP	0.7	0.35	0.47	0.7	0.33	0.45
	CM-WSPADE	0.7	0.32	0.44	0.7	0.28	0.4
10%	Modified Suffix Tree	0.9	0.75	0.82	0.7	0.39	0.5
	NOSEP	0.7	0.28	0.4	0.6	0.23	0.33
	CM-WSPADE	0.7	0.23	0.35	0.6	0.22	0.33

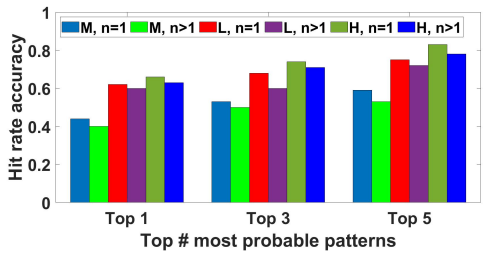

Fig. 2: Performance comparison of various prediction models on synthetic dataset where M, L, H, n=1 and n>1 denote SlidingWindow-MC, LSTM, Proposed HMM, single interleaving noisy click setup and multiple interleaving noisy clicks setup, respectively.

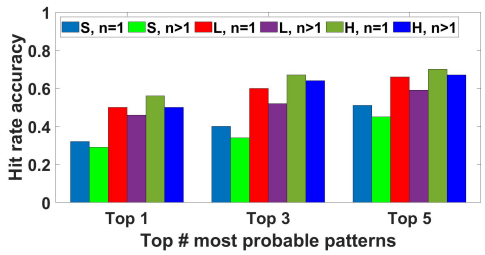

Fig. 3: Performance comparison of various prediction models on real dataset where M, L, H, n=1 and n>1 denote SlidingWindow-MC, LSTM, Proposed HMM, single interleaving noisy click setup and multiple interleaving noisy clicks setup, respectively.

V. Conclusion

Understanding and utilizing the consistency of the user browsing behavior through the webpage layout is the key element for extracting the underlying patterns of the user that tackles the dynamic changes of the content in the websites. Also, predicting the user patterns accurately in advance is another major solution for WiFi offloading and hence alleviates the cost and traffic on the user and the network providers, respectively. In this paper, we developed clickstream mining for the user browsing activities with interleaving noisy clicks. Those interleaving noisy clicks were randomly inserted per pattern and considered in two different setups, a single noisy click, and multiple noisy clicks. Our simulation results suggested that we can efficiently and accurately extract the underlying patterns of the user click activity when compared to the state of art models. Moreover, it showed that our model is more robust to the noisy interleaving clicks. Finally, the employed pattern predictive model based on HMM showed an impressive performance with an accuracy of 81% for outputting the top 5 most probable user patterns.

References

[1] X. Wu, X. Zhu, Y. He, and A. N. Arslan, "Pmbc: Pattern mining from biological sequences with wildcard constraints," *Comput. Biol. Med.*, vol. 43, no. 5, p. 481–492, jun 2013. [Online]. Available: https://doi.org/10.1016/j.compbiomed.2013.02.006

[2] Y. Thushara and V. Ramesh, "A study of web mining application on e-commerce using google analytics tool," *International Journal Of Computer Applications*, vol. 149, no. 11, pp. 21–26, 2016.

[3] L. Zhang, P. Luo, L. Tang, E. Chen, Q. Liu, M. Wang, and H. Xiong, "Occupancy-based frequent pattern mining," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 10, no. 2, pp. 1–33, 2015.

[4] Y. J. M. Pokou, P. Fournier-Viger, and C. Moghrabi, "Authorship attribution using small sets of frequent part-of-speech skip-grams," in *The Twenty-Ninth International Flairs Conference*, 2016.

[5] A. Rajimol and G. Raju, "Web access pattern mining–a survey," in *International Conference on Data Engineering and Management*. Springer, 2010, pp. 24–31.

[6] S. Vijayalakshmi, V. Mohan, and S. S. Raja, "Mining of users' access behaviour for frequent sequential pattern from web logs," *International Journal of Database Management System (IJDM)*, vol. 2, 2010.

[7] T. Van, A. Yoshitaka, and B. Le, "Mining web access patterns with super-pattern constraint," *Applied Intelligence*, vol. 48, no. 11, pp. 3902–3914, 2018.

[8] Y. Wu, Y. Tong, X. Zhu, and X. Wu, "Nosep: Nonoverlapping sequence pattern mining with gap constraints," *IEEE transactions on cybernetics*, vol. 48, no. 10, pp. 2809–2822, 2017.

[9] G. Suchacka, M. Skolimowska-Kulig, and A. Potempa, "Classification of e-customer sessions based on support vector machine." *ECMS*, vol. 15, pp. 594–600, 2015.

[10] M. A. Awad and I. Khalil, "Prediction of user's web-browsing behavior: Application of markov model," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 4, pp. 1131–1142, 2012.

[11] Y. Shi, Y. Wen, Z. Fan, and Y. Miao, "Predicting the next scenic spot a user will browse on a tourism website based on markov prediction model," in *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*. IEEE, 2013, pp. 195–200.

[12] L. Rabiner and B. Juang, "An introduction to hidden markov models," *ieee assp magazine*, vol. 3, no. 1, pp. 4–16, 1986.

[13] H. M. Huynh, L. T. Nguyen, B. Vo, A. Nguyen, and V. S. Tseng, "Efficient methods for mining weighted clickstream patterns," *Expert Systems with Applications*, vol. 142, p. 112993, 2020.

[14] A. Alamoudi, M. Liu, A. Payani, F. Fekri, and D. Li, "Predicting mobile users traffic and access-time behavior using recurrent neural networks," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2021, pp. 1–6.

[15] S. D. Bernhard, C. K. Leung, V. J. Reimer, and J. Westlake, "Clickstream prediction using sequential stream mining techniques with markov chains," in *Proceedings of the 20th International Database Engineering & Applications Symposium*, 2016, pp. 24–33.

[16] J. Butler, W. Lidwell, and K. Holden, *Universal principles of design*. Rockport publishers Gloucester, MA, USA, 2010, 112–113., 2003.

[17] J. Stenback, P. Le Hégaret, and A. Le Hors, "Document object model (dom) level 2 html specification," *W3C Recommendation*, vol. 9, 2003.

[18] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu, "Mining sequential patterns by pattern-growth: The prefixspan approach," *IEEE Transactions on knowledge and data engineering*, vol. 16, no. 11, pp. 1424–1440, 2004.

[19] L. Test. (2021) 2021 digital experience benchmarks by industry. [Online]. Available: https://contentsquare.com/blog/2021-digital-experience-benchmarks-by-industry/