Received September 25, 2019, accepted October 16, 2019, date of publication October 23, 2019, date of current version November 5, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2949077

WebProfiler: User Interaction Prediction Framework for Web Applications

MINWOO JOO, (Student Member, IEEE), AND WONJUN LEE, (Senior Member, IEEE)
Network and Security Research Laboratory, School of Cybersecurity, Korea University, Seoul 02841, South Korea

Corresponding author: Wonjun Lee (wlee@korea.ac.kr)

This work was supported by the Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT under Grant 2017M3C4A7083676.

ABSTRACT User interaction prediction for Web applications is crucial to improve browsing experience. With the prediction, for instance, a browser can prefetch the content to be accessed next reducing long wait times in advance. However, predicting the user interaction is challenging in practice. Collecting detailed interaction data is difficult due to the constraints on target application and platform. Moreover, Web navigation prediction mechanisms for general applications have a low accuracy with conventional machine learning models. To this end, in this paper, we propose a Web interaction profiling framework, *WebProfiler*, which collects user interaction data in a generic way and accurately predicts the next navigation. Both navigation and click events are collected by using JavaScript event handlers and clicked objects are identified reliably through a document object model based approach. Furthermore, we adopt gated recurrent unit (GRU), a representative deep learning technique suitable for coping with time series Web interaction data, and present two advanced techniques for training the GRU-based model: uniform resource locator (URL) grouping to handle the variant URLs of a Web page and Web embedding to represent both events in a unified vector space. The experimental results based on the real user interaction data showed that click events within an application improved the overall prediction performance by 13.7% on average, which were overlooked by most of the previous research. In addition, WebProfiler achieved an average F-measure of 0.798 for top three candidates where URL grouping and Web embedding contributed to 52.4% of the performance improvement.

INDEX TERMS Deep learning, gated recurrent unit (GRU), navigation prediction, user interaction, web applications.

I. INTRODUCTION

Understanding user interaction for Web applications is key to improving users' Web browsing quality of experience (QoE). Web browsing consists of a series of clicks triggered by users and corresponding navigations to other applications. If such prior knowledge of user interaction is provided, a browser can prefetch the content to be accessed next in background, making it instantly available when a user requests that content. Prefetching enables Web applications accompanying multiple multimedia resources or subsidiary resources from external domains to reduce long wait times to download those resources. For example, Google Chrome provides a prediction service to load pages more quickly by initiating actions such as DNS prefetching, TCP and TLS preconnection, and preloading of Web pages before actual requests are made [1]. To generate those predictions, modern browsers leverage resource hints [2] provided by Web applications to hide latencies for networking, processing, and rendering. Resource hints are specified in source code by developers in advance, which are typically generated by using heuristic rules based on the static information of document markup and structure, navigation history, and user-dependent data (e.g., type of device, available compute and memory resources, network connectivity, user preferences). On the other hand, navigation prediction based on the relation between user interaction and Web applications produces more valuable hints beyond these simple heuristics. With conventional resource hints, a user can prefetch only the resources directly related to the current Web page and is unable to obtain any information of the application navigated next. With Web navigation prediction, a user can prefetch not only the relevant resources to the

The associate editor coordinating the review of this manuscript and approving it for publication was Cong Pu.

current application, but also the potential resources carried by the next navigation proactively. This prediction ability in turn shorten the response time of each navigation, enhancing the performance of Web prefetching even better.

However, user interaction prediction for Web applications is difficult in practice. First, collecting detailed data is challenging due to the limited scope of target application and platform. Most of the previous research efforts [3]–[5] tried to exploit the user interaction information collected at server side, which is limited to specific Web application services. Such application-specific data impedes a clear understanding of the behaviors of users who navigate to various Web applications. Meanwhile, other methods working at client side [6]–[8] tried to collect detailed user interaction data such as clicks, phone calls, and application executions. These client-side solutions, however, require modification of Web browser or application's source code, or they are only limited to be used on mobile platforms. Measurement overhead restricts the scalability of data collection tools, which leads to a narrow view of user interaction in real application usage.

Second, navigation prediction for general Web applications has a low accuracy with conventional machine learning models. As described in Section II, researchers have studied user interaction prediction for Web applications focusing on two typical application types: Web search [9]–[16] and online advertising [5], [17], [18]–[24]. Although they provided in-depth insights on these traditional Web applications, today's complex Web environment with diverse applications, including augmented or virtual reality, finance, healthcare, social network, video streaming, and Web of Things, has not been studied well. In addition, Web applications have a low predictability of user behavior inherently since a user can navigate anywhere through browsers whereas usage is restricted to installed applications in mobile environments. In [8], the authors presented methods to collect click data in Android mobile applications and predict click sequences of buttons using deep learning. As described later in this paper, unlike mobile applications, additional methods for performance improvement are essential for Web applications because of the low accuracy of conventional prediction models.

In this paper, we propose a Web interaction profiling framework, *WebProfiler*, which collects navigation and click events across Web applications, and predicts the next navigation accurately. We present an event tracing tool based on JavaScript event handling and identify individual click events using the document object model (DOM) architecture. By utilizing event handlers based on JavaScript, one of the fundamental technologies in Web environments, WebProfiler can collect navigation and click events for general applications without any modification of browser or application's source code. Clicked objects are represented by their positions and features in DOM trees, which leads to consistent object identification regardless of device or browser used for measurement. To design the navigation prediction model, we adopt gated recurrent unit (GRU), one of the variants of recurrent neural network (RNN) which is a representative deep learning technique suitable for handling time series Web interaction data. The performance of our GRU-based prediction model is optimized by analyzing various hyper-parameters and other machine learning models. In addition, we propose two advanced techniques to enhance the poor prediction performance of the baseline due to the variety of Web applications: uniform resource locator (URL) grouping to cope with the variant URLs for a single Web page and Web embedding to represent both navigation or click events in a unified vector space. To comprehend user interaction for Web applications and evaluate the proposed prediction model, we deploy the event tracing tool and collect real interaction data. Based on the collected data, we verify that WebProfiler can predict Web navigation accurately for general applications in practice. The major contributions of this work can be summarized as follows:

- WebProfiler not only collects detailed Web interaction data regardless of target application and platform, but also achieves reliable object identification for click events on any device or browser. Our JavaScript-based event tracing and object identification using the DOM tree contribute to the generic data collection method for Web interaction prediction. We validate the feasibility of our event tracing tool by developing a prototype operable on a Web browser with the highest usage share.
- We enhance the accuracy of our navigation prediction model by designing advanced techniques of URL grouping and Web embedding in addition to adopting the GRU-based prediction architecture. WebProfiler groups Web resources with relevant addresses and maps user interaction events to a low-dimensional space to maximize the accuracy of the GRU-based prediction model.
- We deploy our event tracing tool to real users to construct an extensive dataset for Web interaction and evaluate the prediction performance of WebProfiler based on the collected dataset. The experimental results show that click events occurring within a single application improved the navigation-only prediction performance by 13.7% on average whereas most of the previous research considered only the user interaction data directly related to navigations between Web pages. Furthermore, WebProfiler predicted next navigations with an F-measure of 0.798 for top three candidates where URL grouping and Web embedding contributed to 52.4% of the performance improvement.

The remainder of this paper is organized as follows. In Section II, we review the previous work related to user interaction data collection and prediction for Web applications. We outline the overall architecture of WebProfiler with its design principles in Section III. Section IV describes how to trace and identify interaction events for Web applications with the analysis on collected user interaction data. Section V presents the core components of our GRU-based navigation prediction model including URL grouping and Web embedding. In Section VI, we analyze the results of our experiments, demonstrating the effectiveness of our method compared to other schemes. Section VII discusses the limitations and opportunities for future work, and we conclude in Section VIII.

II. RELATED WORK

In spite of the importance of user interaction profiling and prediction for general Web applications, only few research efforts have focused on these topics thoroughly and most of the existing studies have their own limitations to be addressed. In this section, we discuss the state-of-the-art in a brief manner.

A. USER INTERACTION AND BETTER WEB QOE

With the evolution of Web architectures and protocols for dynamic and complex Web applications, many attempts have been made to improve Web QoE. Especially, since one of the most commonly used metrics to perceive Web QoE is latency (e.g., page load time) [25], there were several efforts to prefetch Web content utilizing user interaction information for better Web QoE. Google Chrome included user action predictors from resource requests and response traffics to reduce page load time with speculative optimization [26]. [27] suggested a mobile content prefetcher of social network feeds for a popular social networking service (i.e., Twitter) by analyzing content consumption behaviors of users. [28] showed that prefetching Web resources based on gaze activity with external tracking equipment improved Web navigation latency. While these efforts showed that user interaction data could be helpful to improve Web QoE, their browser-specific, application-specific, or high measurement overhead solutions are hard to be applied to other Web environments. On the other hand, WebProfiler is applicable to both user interaction data collection in modern Web browsers and navigation prediction for general Web applications, which leads to better Web QoE regardless of target environment.

B. DATA COLLECTION FOR USER INTERACTION

There have been various approaches to collect user interaction data within various contexts. Amongst them, we discuss Web-based and mobile-based approaches.

1) WEB-BASED APPROACHES

[3] described a general architecture to collect users' Web browsing behaviors at server side in order to realize automatic Web personalization. [4] collected a sequence of each user's HTTP requests for Sybil user detection from a Chinese social networking service (i.e., Renren). Reference [5] utilized various features such as advertising information, user information, advertising-user interaction, and advertising position bias collected at the Twitter servers. These server-side collection approaches were typically suitable for particular applications and the collected data had a low level of granularity on user interaction, most of which occurs at client side. On the other hand, [6] presented a client-side method to collect users' click events to analyze Web page revisitation by modifying the source code of the Firefox browser. Since modern Web browsers have fast update tempos [29], it is desirable to decouple data collection tools from Web browsers or operating systems. WebProfiler achieves this separation by implementing its event tracing tool as a standard-compliant browser extension.

2) MOBILE-BASED APPROACHES

Collecting user interaction data in mobile environments is challenging due to constrained user inputs (e.g., no mouse movement) and limited infrastructure support. To this end, [7] proposed an Android service application programming interface (API) for collection and prediction of user interaction so that application developers can manually inject the API calls into their source code. However, this approach would not suitable for Web applications since modification of source code for every existing application is impossible in practice and calling platform-dependent APIs in mobile Web browsers is rarely supported as well. PathFinder [8] proposed a method to identify dynamically-located buttons appeared at mobile display, which could collect user interaction data in an application-independent manner. However, this approach could not be extended and applied directly to Web applications in two aspects. First, PathFinder relied on the Android accessibility service (i.e., operating-specific), which reduced its portability filtering out all the events occurring in Web-based applications (i.e., WebView). Moreover, the clicked button identification method based on the size and position of each candidate button caused unreliable object identification over different types of devices or client software. On the contrary, WebProfiler is relatively portable and efficient thanks to its sole dependency on the modern Web architecture.

C. WEB NAVIGATION PREDICTION FOR MODERN WEB APPLICATIONS

Web navigation prediction has been studied for a decade using server logs on user requests [30] or click-driven search queries [9]–[12] in diverse contexts. [30] proposed a combined predictor of Markov model and support vector machine (SVM) with Dempster's rule which can predict the next page given a user's session log. Although [30] worked towards similar goals, we employ a novel GRU-based prediction model for WebProfiler with advanced techniques for performance improvement using fine-grained user interaction data. Meanwhile, there have been a lot of research efforts on click models which tried to simulate users' click behaviors with Bayesian networks [9], [10] or neural networks [11], [12]. Since these efforts mainly focused on Web search scenarios, it is difficult to apply their approaches to general Web applications. Another study for Web search [13] proposed a decision tree ensemble predictor which utilized mouse cursor movements using a JavaScript-based logger for better prefetching on search engine result pages. However, this prediction method has high measurement overhead for logging mouse cursor positions frequently (every 250 milliseconds) and is hard to be utilized in mobile environments where no mouse cursor exists, and, above all, is limited to Web search as well. Recently, [14] proposed a regression-based expectation-maximization algorithm to enhance traditional position bias click models for personal search and [15] explored the unique biases compared to desktop environments for mobile search, respectively. In addition, [16] introduced a non-parametric calibration method called isotonic regression to improve click models trained with suboptimal hyperparameters. Despite these persistent research efforts on Web search, however, they have only tried to understand and predict user interaction based on inherent properties such as relevance between user queries and resulting documents, and relative positions on search engine result pages.

In the context of online advertising, on the other hand, many research efforts have been made for predicting click-through rates (CTR). Reference [17] explored the importance of bounce rate, the fraction of users who click on advertisements but almost immediately move on to other tasks, and utilized it to predict the effectiveness and quality of advertisements. [18] introduced an RNN-based framework which directly models the dependency on users' sequential behaviors into the click prediction process through the recurrent structure. [5] presented a learning-to-rank approach for advertising in the Twitter timeline to address the training signal sparsity and update its model online. Reference [19] investigated the effects of feature sequence on the performance of a convolutional neural network based CTR prediction model. Reference [20] proposed a deep interest network model for embedding and multi-layer perceptron based approaches to design a local activation unit to adaptively learn the representation of user interests from historical behaviors with respect to a certain advertisement, while [21] extended the model to explore the change of users' interests by distinguishing global and recent click-through behaviors. Reference [22] introduced both character-level and work-level approaches to predict the CTR of a query-ad pair using deep convolutional neural networks where the textual content appearing in a query-ad pair and the page position on a search engine result page are only given as input. Ouyang *et al.* examined various types of relationships including user-to-ad, ad-to-ad, and feature-to-CTR in [23], while, in [24], they tried to improve the CTR prediction of a target ad considering auxiliary ads from both the spatial domain and temporal domain. As in Web search, these studies are dedicated solutions tailored to the characteristics of sponsored search or contextual advertising and thus additional methods to relax the assumptions should be devised to utilize them for general Web applications.

Other than the previous research for typical applications mentioned above, [8] predicted click sequence of buttons in mobile environments for general applications by adopting a deep learning technique. Although the authors achieved an acceptable level of prediction performance, applying their solution to Web environments is not straightforward because of the low accuracy of conventional prediction models when applied to Web applications as well as the diverse

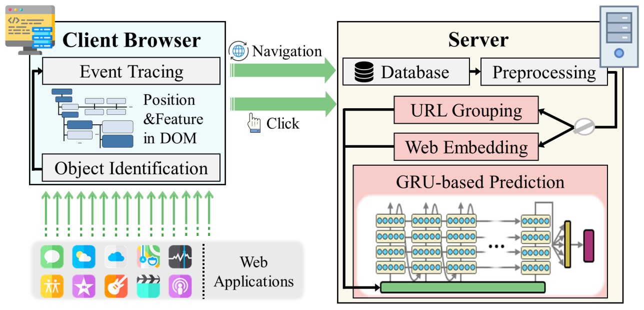

FIGURE 1. Overall architecture of WebProfiler.

prediction targets in Web applications. To address the challenges found in the literature, WebProfiler provides an accurate prediction method robust to navigations across various Web applications.

In summary, the novelty of WebProfiler lies in its scalability of user interaction data collection and accurate navigation prediction for general Web applications. WebProfiler collects fine-grained Web interaction data regardless of target application and platform, and identifies clicked objects reliably on any device or browser, whereas the existing data collection methods for user interaction are limited to particular applications or rely on support from underlying platforms. Furthermore, we designed advanced techniques of URL grouping and Web embedding to enhance the navigation prediction accuracy of our GRU-based model, which can be applied extensively to general Web applications as well, other than the typical application types explored by the previous research on Web interaction prediction.

III. DESIGN OF WEBPROFILER

WebProfiler is a comprehensive framework to improve Web QoE through accurate navigation predictions by profiling user interaction data. There are two fundamental design principles that the prediction framework should follow. First, it should be a generic method to collect user interaction data without any restriction on target application and platform. Measurement overhead must be minimized to understand users' behaviors properly extending the coverage of collection tools in practice. Second, accurate navigation prediction should be made for various Web applications. In Web environments where users can navigate to diverse applications according to their preferences or contexts, additional methods to enhance the low accuracy of existing prediction models must be provided.

Fig. 1 depicts the overall architecture of WebProfiler including the primary components at both client side and server side. On client browsers, the event tracing tool collects navigation and click events occurring as users navigate to general Web applications and sends them to the server. The clicked objects are identified consistently by their positions and features in DOM trees regardless of device or browser used for measurement. The server stores the profile logs forwarded by client browsers in its database and utilizes it for training the Web navigation prediction model. Unnecessary data is filtered out and transformed into a proper form for training through preprocessing. To improve prediction accuracy, URL variants for a single Web page are combined as a group and both navigation and click events are embedded in a unified vector space to reduce data sparsity. Finally, the GRU-based model is trained for Web navigation prediction. The trained model is stored in the server and used to predict the application likely to be navigated next, given a new user interaction data sequence.

IV. USER INTERACTION DATA COLLECTION

To collect navigation and click events in a generic way, we design an event tracing tool which can be applied to any Web application on any platform. This section presents the detailed process of event tracing in Web browsers and how clicked objects are identified reliably. Then, we analyze the real user interaction data collected by our event tracing tool, highlighting the necessity of advanced prediction techniques for Web navigation prediction.

A. EVENT TRACING IN WEB BROWSERS

To capture users' behaviors in Web browsers without any restriction, we designed an event tracing tool based on JavaScript event handlers. Alongside hypertext markup language (HTML) and cascading style sheets (CSS), JavaScript is one of the core technologies for the Web, which enables Web pages to operate interactively and is the key part of Web applications. JavaScript is used to handle the events which occur when a browser accesses Web pages, where these events are generated from Web page rendering, interaction with Web pages' contents, device-related processing issues, or other programming instances such as media stream playback and animation timing. Most of today's Web applications (over 95%) use JavaScript as a client-side programming language [31] and major Web browsers have dedicated engines to execute it (e.g., V8 for Google Chrome, SpiderMonkey for Firefox, JavaScriptCore for Safari, Chakra for Microsoft Edge and Internet Explorer). Server-side JavaScript implementations such as Node.js are developed and used extensively as well. In this context, designing an event tracing method based on JavaScript is the best way to maximize the scalability of user interaction data collection regardless of target application or platform thanks to its flexibility.

We embraced two kinds of JavaScript events for user interaction prediction: one is fired when the document for a Web application is loaded and parsed, and the DOM is fully constructed (i.e., navigation event), and the other is fired when a user clicks an HTML element within the document (i.e., click event). When a navigation event is triggered by a user, the event tracing tool records the user's identifier, event type, URL navigated, and timestamp, and stores it as a profile log. For a click event, the detailed information of the HTML element clicked by a user is collected in addition

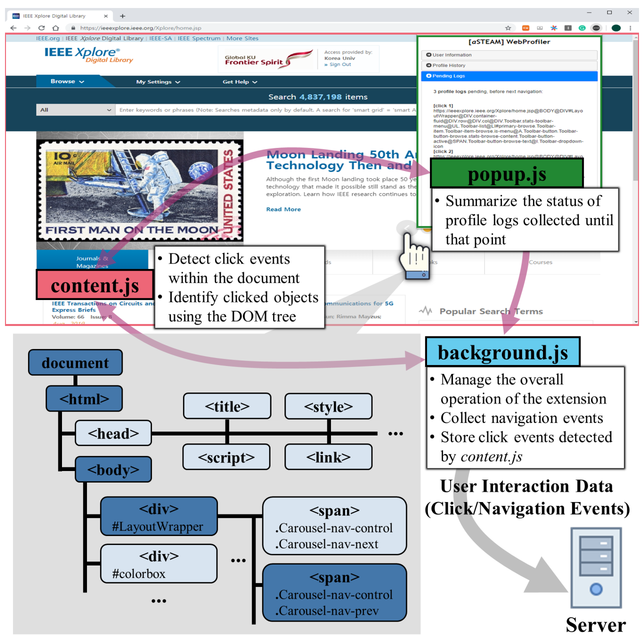
popup.js
• Summarize the status of profile logs collected until that point
content.js
• Detect click events within the document
• Identify clicked objects using the DOM tree
background.js
• Manage the overall operation of the extension
• Collect navigation events
• Store click events detected by content.js
User Interaction Data (Click/Navigation Events)
Server
document
<html>
<head>
<body>
<title>
<style>
<script>
<link>
<div> #LayoutWrapper
<div> #colorbox
<span> .Carousel-nav-control .Carousel-nav-next
<span> .Carousel-nav-control .Carousel-nav-prev
[aSTEAM] WebProfiler
IEEE Xplore Digital Library
FIRST MAN ON THE MOON
Moon Landing 50th Anniversary Technology Then and Now

FIGURE 2. Operating principle of event tracing and object identification for click events in WebProfiler. The example shows the case when a user navigates to *ieeexplore.ieee.org* and clicks a user interface object to display previous items on a carousel.

to the items recorded for a navigation event. Whenever a new Web page is loaded, an event handler for click events is attached to the <*body*> element, which covers the entire document area. This event handler catches every click event occurring in the document with the dedicated path to the clicked HTML element in the DOM tree. Web pages of an application always have more than one <*body*> elements for each HTML document and thus the JavaScript-based event tracing tool of WebProfiler can record every user interaction event for Web applications in their original forms, whether it is planted at either client side or server side.

To assess the feasibility, we implemented our event tracing tool as an extension on the Chrome browser [32], which is also built on Web technologies such as HTML, JavaScript, and CSS. Fig. 2 presents how the extension operates with a working example. A user navigates to *ieeexplore.ieee.org* and the extension inserts the content script (i.e., *content.js*) into the document of the Web application as its DOM contents loaded. The background script (i.e., *background.js*) which manages the overall operation of the extension collects observed navigation events while collecting click events detected by the content script as well. If the user accesses the extension user interface page (i.e., *popup.js*), the status of profile logs collected until that point is summarized and displayed. The collected interaction events are stored in the local storage in browsers temporarily, forwarded to the server when an application transition occurs, and accumulated in the database. Although we implemented the prototype of our event tracing tool as a form of extension in this work, the same software can be utilized with a slight interface modification since it is designed on the basis of JavaScript, one of the essential components of Web applications. It can be implemented on the JavaScript engine inside a browser or imported at any part of source code of a Web application. Furthermore, such client-side approach provides descriptive statistics on the Web interaction of individual users over multiple applications. It is expected that our event tracing tool can be widely used for the research on user interaction data collection for Web applications by relieving the constraints on target application and platform.

B. OBJECT IDENTIFICATION FOR CLICK EVENTS

Identifying clicked objects is challenging because the sizes and positions of objects can vary as the state of a browser window displays them. In [8], for example, the authors presumed that a button is equal to the button which has the most similar size and position among candidate buttons considering the display characteristics of the device used for measurement. However, such method does not guarantee reliable object identification over different types of devices or client software. For Web applications where users interact with various objects (e.g., text links, boxes, images) as well as explicitly specified buttons, a more generic and reliable object identification method should be devised.

To address this issue, we identified a clicked object as a specific node located in its DOM tree. First, a unified string for an HTML element corresponding to each click event is generated by extracting all the elements located alongside the path from the <*body*> node in the document to that node in the DOM tree and separating them by a special character (@ in this work). To distinguish between the elements with the same tag name, the class (denoted as *.class*) and ID (denoted as *#ID*) for each element are marked as well. For example, in Fig. 2, the user clicks the arrow-shaped user interface to browse previous items on a carousel and this object is identified as a <*span*> element with the classes of *Carousel-nav-control* and *Carousel-nav-prev* among the descendant nodes of the <*div*> element with the ID of *LayoutWrapper*, which is one of the child nodes of the <*body*> element. That is, it is represented as a unified string of *@BODY@DIV#LayoutWrapper@... @SPAN.Carousel-nav-control.Carousel-nav-prev*. In addition, the URL of each navigated Web page precedes such unified string. If a clicked object is a descendant node of a *frame* or *iframe* tag, the URL of the corresponding *(i)frame* is attached instead of the navigated URL in the parent window to specify its DOM tree. The DOM-based identification method enables our event tracing tool to minimize incorrect identifications over different types of devices or client software, since it is designed based on the intrinsic architecture of HTML documents without relying on any estimation-based or platform-dependent methods.

C. REAL USER INTERACTION DATA

We collected Web interaction data in real application usage of 15 users for 8 weeks. Indiscriminate collection of user interaction data might raise privacy concerns of participants.

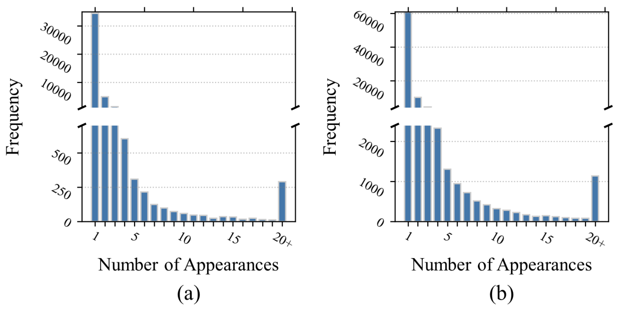

FIGURE 3. Histograms of the number of appearances of Web applications and objects for both event types. (a) Frequency of unique applications for navigation events. (b) Frequency of unique objects for click events.

To ensure the privacy of participants, we used a randomly generated 32-digit hexadecimal string as each user's identifier and processed collected interaction data in an anonymous way. The user identifier (UID) remains the same over different machines if the browser with our event tracing tool is signed in with an identical account. This allows us for consistent observations of user interaction regardless of environment. We deployed our event tracing tool to volunteer users from May 7 to July 4, 2019, and collected 305,064 interaction events in total: 90,267 for navigation events and 214,797 for click events, respectively.

Fig. 3 illustrates the distributions of the number of appearances of Web applications and objects for both navigation and click events. Fig. 3 (a) shows how many times each unique URL is observed for the navigation events in the real user interaction dataset. For the entire set of navigation events, 42,731 unique URLs appear indicating that each Web application is observed only 2.112 times on average. As shown in the histogram, the URLs observed less than five times account for the majority, which implies the characteristics of Web users who navigate to diverse applications without any restriction whereas their usage patterns are only limited to the finite set of installed applications in mobile environments. Fig. 3 (b) shows the frequencies of clicked objects as the number of appearances. Note that the scales of y-axes are different from the other subplot. For click events, 84,417 unique objects are observed where the average is 2.544 times for each object. Compared to the result of navigation events, the average observation number is slightly high, but the distribution is still skewed toward the objects observed less than five times. It means that the opposite cases are observed simultaneously where users click several popular objects a lot or they interact with various objects rarely. Such numerous prediction targets emphasize the necessity of a precise design of the prediction model for Web applications.

V. WEB NAVIGATION PREDICTION

This section describes the core components of WebProfiler to predict the application likely to be navigated next. Individual steps for preprocessing collected data to be suitable for training are presented first and then the robustness of a representative deep learning model for navigation prediction is examined. To remedy the shortcomings on prediction performance, two additional techniques are devised: URL grouping and Web embedding.

A. PREPROCESSING

To prevent from corrupting data patterns and misleading results, we developed sequence processing and filtering techniques in our Python preprocessing implementation. First, the collected user interaction data is loaded from the database and the events with non Web-specific URLs (e.g., browser-specific custom pages) are eliminated. The data is then clustered by UID and rearranged by timestamp. Adjacent events with timestamps apart more than the pre-defined maximum time gap (10 minutes in this work) are separated as distinct sequences. If the time gap between adjacent events becomes too large, the context is not maintained anymore and they should be distinguished as different samples for training. After slicing the sequences to fit in the input size of our prediction model, we filtered out minor data based on UID, navigated URL, and clicked object string to reduce noise on the dataset. Five UIDs with too small data to train are excluded, navigated URLs are restricted to the items ranked in the top 200, and clicked objects observed less than two times are represented as a dedicated identifier. All the criteria used for filtering are decided empirically. Other advanced techniques are applied as well, which will be explained later in this section.

After generating sequences, we transformed the user interaction data into a suitable form for training. All the events observed in the dataset (i.e., URLs for navigation and object strings for click) are used to construct the overall event dictionary. Since the prediction model only deals with numeric values as input, we assigned a unique integer identifier for each event. Also, these integer values are divided by the maximum identifier before training to make them fall within a range between 0 and 1 because the performance of a machine learning algorithm is highly dependent on input scales. We generated a training set, a validation set, and a test set by randomly permutating the samples in the entire dataset, where each set accounts for 70%, 15%, and 15%, respectively.

B. GRU-BASED PREDICTION MODEL

We adopted GRU as a baseline of our navigation prediction model, one variant of the popular deep neural network architectures used for analyzing time series data, called RNN. Unlike a feedforward neural network where the activations flow only in one direction, an RNN has connections pointing both forward and backward. Such backward flow enables a recurrent neuron to receive not only an input vector, but also the output vector from the previous time step. It works as a form of memory because the output of a recurrent neuron at a certain time step t is a function of all the inputs at previous time steps and that is why RNN is suitable for time series data. To resolve issues faced by the legacy RNN, the long short-term memory (LSTM) architecture was proposed in 1997 [33] and has been gradually improved by other researchers. An LSTM unit maintains both the short-term state and long-term state by recognizing an important input with an input gate, preserving it as long as needed with a forget gate, and extracting it whenever needed with an output gate. It is known that LSTM makes training converge faster and detects well long-term dependencies in data than RNN. Recently, a simplified version of LSTM, GRU, was proposed [34] and widely used by many applications, particularly by those in natural language processing. Compared to LSTM, it has a single vector for both short-term and long-term states, a single gate controller controls both the input gate and the forget gate, and the full state vector is output at every time step. In WebProfiler, the GRU-based prediction model was used as it performs roughly the same with other variants of LSTM, in spite of its simple architecture. To build our navigation prediction model, we first set the input layer to accept input sequences with variable lengths since the number of navigation or click events occurred before a navigation prediction differs in context. We then layered GRU cells to construct the body of the prediction model and added a fully connected layer and a softmax layer before the output layer to transform activated values from the GRU cells into prediction targets. By refining the baseline GRU model architecture and tuning the hyperparameters extensively, we achieved the elaborate design of our GRU-based prediction model tailored to navigation prediction tasks for Web applications.

TABLE 1. Test set precision at top *k* with the GRU-based prediction model as a baseline.

Model	Precision						
	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$	$k=7$
GRU[a]	0.230	0.340	0.423	0.492	0.543	0.582	0.619

[a]The basic GRU-based prediction model is tested as a baseline with neither URL grouping nor Web embedding.

We implemented the GRU-based prediction model using TensorFlow [35] including other deep learning models and verified the performance of the baseline model as an early test. Table 1 shows the results of test set precision at top k with the GRU-based navigation prediction model after training. Note that the dataset and experimental parameters used for these preliminary results are identical to those of the main evaluation part (see Section VI for further details). We measured the accuracy based on top-k candidates because determining a set of candidate applications is a pragmatic solution in terms of accuracy rather than predicting the exact application which will be navigated next. Specifically, the top-k precision means that if the actual next application was found in the top-k candidate set where only k predicted targets are included and sorted by their probabilities of occurrences in descending order, we regarded it as correct. As shown in the table, the basic GRU-based model predicted the next application exactly only in 23% and the prediction accuracy just achieved 42.3% with a practical candidate set size of $k = 3$. The result shown here was evaluated even after all the hyperparameters were optimized, which implies additional techniques are essential for a high degree of prediction accuracy and motivates us to devise URL grouping and Web embedding to improve the prediction performance of WebProfiler for practical use.

C. URL GROUPING

To improve navigation prediction accuracy, we should consider the various nature of Web browsing. The part of a Web application's URL called query string assigns values to specified parameters. Query string is typically used to process the contents which vary according to the context of Web pages with a same format. A search engine result page is a typical example of utilizing query string, where different results of search queries are displayed on the same page architecture. Unifying Web pages with similar URLs which only differ for the query string part as a single prediction target leads to meaningful navigation prediction because they share most of their resources such as image, CSS, and JavaScript.

WebProfiler applies URL grouping for the GRU-based prediction model, where the variants of a single URL are processed as a unified group. For a navigation event, we eliminated the query string part of its target URL and used it to identify that event. That is, the substring of a URL after the question mark, if exists, is truncated. Here is a trivial example of navigations over search engine result pages (i.e., *result.search.com*). Suppose that URL_1, URL_2, and URL_3 are *https://result.search.com?where=blogpost&query=machine learning*, *https://result.search.com?where=searchbar&query=deeplearning&oquery=machinelearning*, and *https://result.search.com?where=searchbar&query=gru&oquery=deeplearning*, respectively. A query string in this example consists of type of previous page (i.e., *where*), search query (i.e., *query*), and old search query (i.e., *oquery*). The instances specified by the query string here are common features analyzed by search engines to examine the relevance between Web pages. For these three pages with different URLs, the dominant factor for page load time is the size of multimedia resources which are requested to the server and downloaded identically. With URL grouping, these URLs are handled as a unified group and a browser can prefetch the multimedia resources in advance, if the predicted target belongs to this group. Similarly, for a click event, the URL of its parent window or *(i)frame* is truncated as well and then the string for the clicked HTML element follows intactly. This simple processing technique can increase the performance of Web navigation prediction drastically by reducing the prediction target space of navigated URLs (see Section VI for detailed description).

D. WEB EMBEDDING

Word embedding is a machine learning technique widely used in natural language processing where words or phrases from a vocabulary are mapped to a lower dimension. Unlike image and audio processing systems where high-dimensional datasets exist sufficiently, in the systems treating discrete atomic symbols (e.g., words in natural language processing), a prediction model learns very little from individual symbols in the dataset, which leads to data sparsity. Vector space models such as word embedding can alleviate this problem by representing symbols in a continuous vector space. With word embedding, semantically similar words are embedded nearby each other.

As an extension of word embedding, we designed Web embedding where both types of events (i.e., navigation and click) are mapped to a unified low-dimensional vector space. Web embedding takes the entire sequences of events in the dataset as input and assigns a dense vector in 64 dimensions to each event identifier. Different from the conventional word embedding where the same types of words or phrases are processed, in Web embedding, two distinct types of navigation and click events are serialized first in an identical target space together and then the vectorization to a low-dimensional space proceeds. For implementation, we adopted the skip-gram model of Word2vec [36], which consists of two-layer neural networks trained to reconstruct the contexts of input symbols. In this work, we set the number of skips as 2 and the size of skip window as 1 for the skip-gram model. For example, suppose that the events are given in the context of {N to URL_a, C to OBJ_x, C to OBJ_y, C to OBJ_z, N to URL_b, N to URL_c, ...}, where N and C stands for navigation and click, respectively, and OBJ denotes the clicked object identified in the DOM tree. That is, the user in this example first navigated to the Web application with URL_a, clicked the objects of OBJ_x, OBJ_y, and OBJ_z in order, and then navigated to the Web application with URL_b and URL_c in a row. First, we can generate (context, target) pairs as {([N to URL_a, C to OBJ_y], C to OBJ_x), ([C to OBJ_x, C to OBJ_z], C to OBJ_y), ([C to OBJ_y, N to URL_b], C to OBJ_z), ...}. To generate these pairs, we can consider syntactic contexts and the context is defined as the left and the right event of a target event, since the size of skip window is set as 1. Since the skip-gram model predicts each context symbol from its target symbol, the dataset becomes {(C to OBJ_x, N to URL_a), (C to OBJ_x, C to OBJ_y), (C to OBJ_y, C to OBJ_x), (C to OBJ_y, C to OBJ_z), ...} of (input, output) pairs. The objective function is optimized with stochastic gradient descent (SGD) thereafter and the final embeddings are stored in the form of dictionary. In the learning phase, each event is converted into a corresponding vector representation by looking up the dictionary and then fed into the prediction model as input. As shown in Section VI, Web embedding enhances the performance of our GRU-based prediction model even better in addition to URL grouping by locating interaction events that share common contexts in close proximity with each other in the dense vector space.

VI. EVALUATION

In this section, we evaluated the performance of WebProfiler based on the real user interaction data collected by our event

TABLE 2. Experimental parameters for evaluation.

Parameter	Value
Number of users	15
Data collection period	From May 7 to July 4, 2019 (8 weeks)
Number of total events	305,064
Number of navigations	90,267
Number of clicks	214,797
Training : validation : test sets	0.7 : 0.15 : 0.15
Initialization	He initialization
Number of unrolled cells	10
Learning rate	0.01
Number of units	64, 128, 256, 512
Number of layers	1, 2, 4, 8
Batch size	25, 50, 100
Activation function	Sigmoid, hyperbolic tangent, ReLU, ELU
Optimizer	Gradient descent, Momentum, NAG, RMSProp, Adam

tracing tool. First, we optimized the hyperparameters of our GRU-based prediction model extensively in terms of number of units and layers, batch size, activation function, and optimizer. We then examined the effectiveness of click events other than navigation events and investigated the performance gains from adopting URL grouping and Web embedding. Finally, we compared the GRU-based model with other typical models in machine learning: RNN, LSTM, SVM, logistic regression (LR), and naive Bayes (NB).

We determined several training options empirically and utilized averaged F-measure to measure prediction performance. Throughout the evaluation, we used the same dataset containing the real user interaction data collected by our event tracing tool as described in Section IV. For deep learning models, we adopted the He initialization strategy [37] and set the number of unrolled cells as 10 and the learning rate as 0.01, respectively. Other hyperparameters for deep learning models are explored intensively in the following subsection. All the experimental parameters are summarized in Table 2. The performance of Web navigation prediction is evaluated using F-measure, the harmonic mean of precision and recall. To deal with multi-label data, we used the micro-averaging operation [38], which is usually considered for averaging accuracy measures in information retrieval tasks.

A. HYPERPARAMETER TUNING

Prior to the evaluation on the proposed methods, we derived optimal values for the hyperparameters to maximize the performance of the GRU-based prediction model through extensive experiments. During hyperparameter tuning, validation set is used for evaluation to avoid overfitting, unless otherwise mentioned. Fig. 4 shows the prediction performance according to the combination of the number of units and

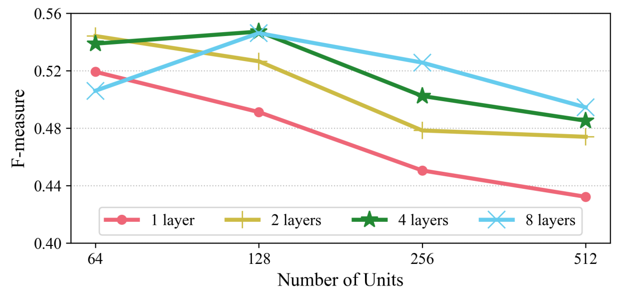

FIGURE 4. Prediction performance as the number of units and layers.

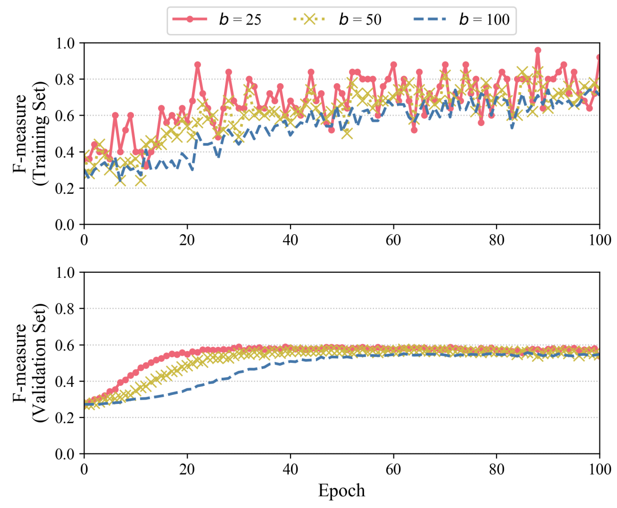

FIGURE 5. Learning curve as the batch size *b* for both training set prediction performance (top) and validation set prediction performance (bottom).

number of layers. As shown in the figure, the F-measure decreased as the number of units increased above a certain level. The complexity of a prediction model increases as the number of units and layers becomes too large, which results in overfitting where the performance for training set becomes better while that for validation set decreases failing to learn general patterns of data. With too simple model, on the other hand, the F-measure appears to be low, because it cannot learn hidden characteristics of data properly. We adopted the 4-layer architecture with 128 units, which showed the best prediction performance among the combinations.

Fig. 5 shows the learning curves for different batch sizes of $b = 25, 50, 100$. Since the majority of training methods using deep learning are performed based on mini-batch iterative optimization, batch size is one of the important hyperparameters. The result for training set showed that a prediction model with a smaller batch size made more noisy estimation in the short term because less samples were considered per iteration. In the result for validation set, the F-measure converged to a higher value quickly with a smaller batch size as the number of updates per epoch became large. However, the computation time for each epoch became longer and the entire training time increased with a smaller batch size, causing low availability of parallelism. On the other hand,

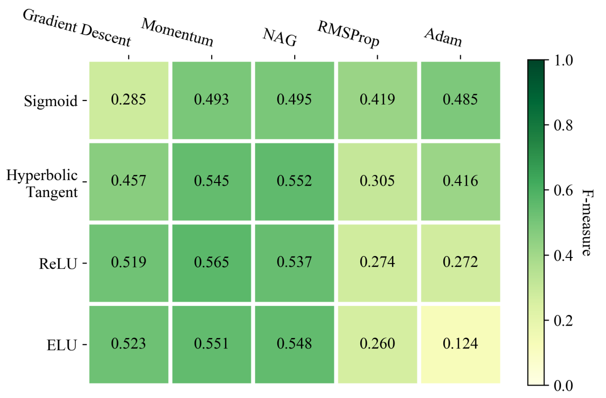

FIGURE 6. Prediction performance as the combination of activation functions and optimizers.

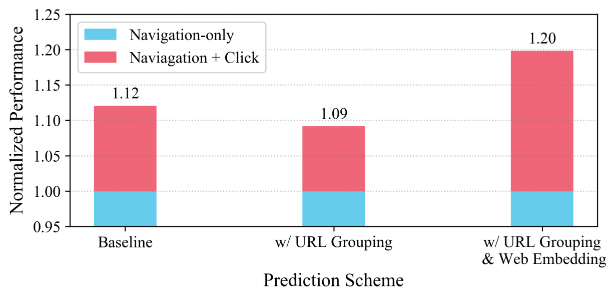

FIGURE 7. Performance gain by adopting click events for prediction.

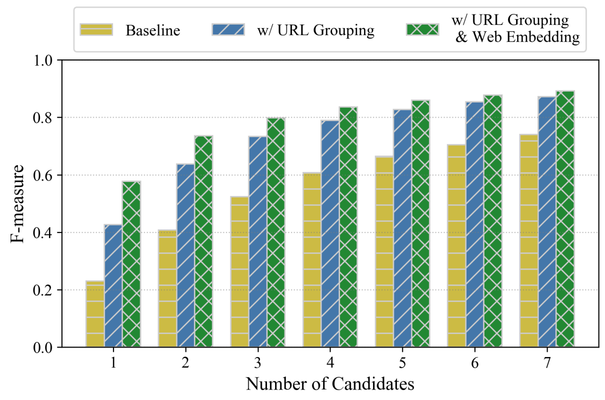

FIGURE 8. Performance gain at top-*k* with URL grouping and web embedding.

there is a possibility of inaccurate weight updates with a large batch size since old gradients are used for training and gradients fluctuate as the state of parameters for non-convex functions like loss functions used in deep learning. Therefore, in this work, we used a batch size of $b = 25$, which is small enough to obtain positive effects on both reliable training and generalized performance even if training is a bit slow.

We examined the prediction performance according to the combination of activation functions and optimizers as shown in Fig. 6. In deep learning, a poor choice of activation function leads to the vanishing/exploding gradients problems where training fails to converge due to unstable gradients. We tested rectified linear unit (ReLU), known to work well for deep neural networks, and its recent variant, exponential linear unit (ELU) as well as traditional activation functions of sigmoid and hyperbolic tangent. Optimizer is another important factor affecting training speed and reliability. In this work, we covered popular optimizers used for deep learning: gradient descent, Momentum, Nesterov accelerated gradient (NAG), root mean square propagation (RMSProp), and adaptive moment estimation (Adam). For activation functions, ReLU and its variant performed well, while hyperbolic tangent showed comparable performance as well. For optimizers, Momentum and NAG provided consistent performance over activation functions. We selected the combination of ReLU and Momentum, which showed the best F-measure. For subsequent experiments, we used these hyperparameter values for evaluation.

B. IMPACT OF CLICK EVENTS ON PREDICTION

Before evaluating the proposed methods, we analyzed how much click events occurring within an application are helpful for Web navigation prediction. Fig. 7 shows the performance gains by adopting click events for prediction over three types of prediction schemes: the first one is the baseline model based on GRU, the second one is the model where URL grouping is added to the baseline, and the last one is our complete prediction model with both URL grouping and Web embedding. The performance of each prediction scheme trained by using only navigation data was measured first and then the performance of each scheme was measured again after re-training based on both navigation and click data, which was represented as normalized values to the navigation-only performance. That is, the normalized performance for each of the navigation plus click results is represented as a relative value when the corresponding navigation-only performance is regarded as 1. As shown in the figure, using click data for training as well as navigation data improved the prediction performance by 12.0% for the baseline GRU-based model, 9.2% for the model only with URL grouping, and 19.8% for our complete prediction model with both URL grouping and Web embedding, respectively (13.7% on average). This is because that click data provides more fine-grained information (i.e., hidden states between adjacent navigations) to grasp users' intentions compared to navigation data and its impact on prediction performance turns out to be more than expected. Different from the previous research focusing only on navigation data between Web pages, it showed that user interaction data occurring within an application is another important factor to enhance prediction performance.

C. BENEFITS OF URL GROUPING AND WEB EMBEDDING

We investigated the benefits of URL grouping and Web embedding, which are devised to cope with the diversity on Web navigation prediction. Fig. 8 shows the performance gains at top k over three types of prediction schemes for test set. With URL grouping, the F-measure increased 39.3%

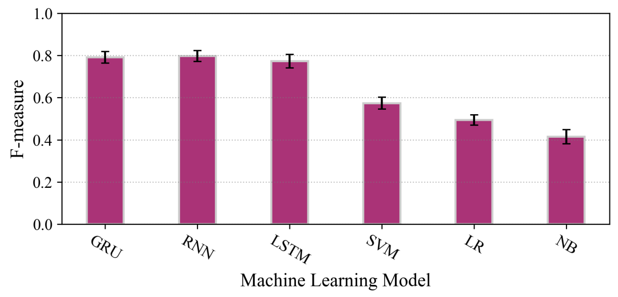

FIGURE 9. Prediction performance comparison to other machine learning models.

on average compared to that of the baseline where the performance gain was maximized to 85.6% with a small number of candidates. URL grouping enables WebProfiler to identify the variants of a Web application as a unified one and this reduction of target space leads to significant performance improvements for Web navigation prediction. Moreover, the prediction performance improved another 10.6% on average when Web embedding was applied in addition to URL grouping. Because of numerous navigation targets and clickable objects associated with them, the problem of data sparsity is severe in Web navigation prediction. The substantial performance gains of Web embedding stem from our design of embedding, which maps both type of events to a unified low-dimensional vector space together. In summary, adopting both techniques proposed in this work achieved an average performance gain of 56.6%. As shown in the figure, using more candidates brought higher prediction performance, but too many candidates made the prediction results useless. We determined $k = 3$ to be the optimal number of candidates since it showed an acceptable F-measure of 0.798 (with performance gain of 52.4%) and the slope of performance improvement became gentle around this point. The prediction performance observed in this work is sufficient for practical use considering the diversity on Web navigation prediction, surpassing the performance in mobile environments. It was also confirmed through the hyperparameter tuning process that these two techniques reduced the complexity of our GRU-based model other than prediction accuracy.

D. COMPARISON TO OTHER MACHINE LEARNING MODELS

Finally, we compared the prediction performance of GRU with those of other machine learning models. Fig. 9 illustrates the comparison results of different models including GRU, RNN, LSTM, SVM, LR, and NB. For RNN and LSTM, we trained the models using TensorFlow as the GRU-based model with both URL grouping and Web embedding enabled. The radial basis function was used as a kernel function for SVM, SGD training was adopted for LR, the Gaussian event model was used for NB, which outperformed other event models (e.g., Bernoulli, complement, multinomial). For non deep learning models, previous ten events were utilized as input with URL grouping enabled. All the F-measure values were measured as top-3 prediction performance for the test sets generated by varying the random seeds to permutate samples in the entire dataset. In the presented results, the variants of RNN (i.e., GRU, RNN, LSTM) performed well since they predict the targets based on all the inputs at previous time steps, while the performance of LSTM was slightly low compared to other two models. Although RNN showed almost the same accuracy with GRU, it was observed that the RNN-based model showed unreliable performance according to the hyperparameter settings. On the other hand, the performance levels of the prediction models other than deep learning were poor since they cannot capture the hidden patterns in Web interaction data properly.

VII. DISCUSSION

The event tracing tool of WebProfiler can be improved by collecting more detailed user interaction data. In addition to synchronous navigation events, the navigations triggered by JavaScript-based asynchronous requests such as Ajax or those in single page applications using JavaScript libraries such as React can be included for data collection. As many Web applications utilize asynchronous designs to accomplish interactive Web, collecting such asynchronous will enable us to understand dynamic user interaction in depth. Distinguishing events from multiple windows or tabs can be beneficial to capture context precisely as well. The current implementation of our event tracing tool collects all the events from any window or tab in a unified manner. If such multi-source data is identified by individual flows, we can figure out a user's intention more clearly. Also, collected user interaction data occurring within an application can be extended beyond single click. Various types of interaction events other than simple click will be helpful for better navigation prediction including keyboard typing, mouse scroll, double click, drag, and zoom-in/zoom-out.

Client-side data collection solutions including the event tracing tool of WebProfiler might cause extra network usage to send collected data to a dedicated server. Such additional traffic would consume a certain portion of network bandwidth and it might have negative effect on users' Web browsing experience. Moreover, in mobile environments where users pay for their network usage according to the amounts of cellular traffic they use, it would result in poor user satisfaction at a different aspect. For further improvement, the potential impact of additional network usage should be analyzed carefully and a way of incentivizing users to participate in data collection should be devised as well.

We can enhance our object identification method and prediction model design further. With our object identification method for click events, there is a possibility of identifying different objects as a same one, if they have identical IDs and classes over all their elements and only the contents of leaf nodes differ. It might be helpful for learning common properties by clustering them as an identical object on one hand and imply the potential of improvement for more precise event identification on the other. The current prediction model ignores the time intervals between input events considering only the order of those events for training. If the prediction model is designed to utilize inter-event time or dwell time in Web applications, various intentions of users can be inferred from event sequences. In addition, we can provide personalized prediction results if the model is trained further with the data generated by each user.

The freshness of the navigation prediction model should be maintained when utilizing WebProfiler in real application usage. Usage pattern for Web applications changes over time even for a same user. Therefore, to utilize the propose prediction method in practice, new trends should be reflected consistently by re-training the prediction model with fresh data. In addition to the prediction model, the corpus of events used for Web embedding should be updated regularly to catch the common context between events properly. For more general usage, reinforcement learning can be a promising solution where a model is trained by interacting with environments continuously.

VIII. CONCLUSION

In this paper, we proposed a profiling framework for Web applications called WebProfiler, which collects user interaction data without any restriction and predicts Web navigation accurately for general applications. We developed and deployed an event tracing tool which collects real user interaction data with low measurement overhead using JavaScript event handlers and identifies clicked objects reliably through a DOM tree based approach. To improve the GRU-based navigation prediction performance, we designed two advanced techniques for training, URL grouping and Web embedding. The experimental results based on the real user interaction data demonstrated that click events occurring within an application played a significant role in navigation prediction, which improved the overall performance by 13.7% on average. This insight will encourage succeeding research on in-application user interaction data. In addition, WebProfiler achieved an average F-measure of 0.798 for top three candidates where the performance gain from URL grouping and Web embedding accounted for 52.4%. The prediction performance of WebProfiler is sufficient for practical use considering its accuracy and feasibility. Our future work aims to develop an online prefetching mechanism based on the navigation prediction model with other types of interaction events.

REFERENCES

[1] Google Chrome. *Google Chrome Privacy Whitepaper*. Accessed: Sep. 18, 2019. [Online]. Available: https://www.google.com/chrome/privacy/whitepaper.html

[2] I. Grigorik. Resource hints. W3C Working Draft. Accessed: Jul. 2, 2019. [Online]. Available: https://www.w3.org/TR/resource-hints/

[3] B. Mobasher, J. Srivastava, and R. Cooley, "Automatic personalization based on Web usage mining," *Commun. ACM*, vol. 43, no. 8, pp. 142–151, Aug. 2000.

[4] G. Wang, T. Konolige, C. Wilson, X. Wang, H. Zheng, and B. Y. Zhao, "You are how you click: Clickstream analysis for sybil detection," in *Proc. SEC*, Washington, DC, USA, 2013, pp. 241–255.

[5] C. Li, Y. Lu, Q. Mei, D. Wang, and S. Pandey, "Click-through prediction for advertising in Twitter timeline," in *Proc. KDD*, Sydney, NSW, Australia, 2015, pp. 1959–1968.

[6] H. Obendorf, H. Weinreich, E. Herder, and M. Mayer, "Web page revisitation revisited: Implications of a long-term click-stream study of browser usage," in *Proc. CHI*, San Jose, CA, USA, 2007, pp. 597–606.

[7] Y. Li, "Reflection: Enabling event prediction as an on-device service for mobile interaction," in *Proc. UIST*, Honolulu, HI, USA, 2014, pp. 689–698.

[8] S. Lee, R. Ha, and H. Cha, "Click sequence prediction in Android mobile applications," *IEEE Trans. Human-Mach. Syst.*, vol. 49, no. 3, pp. 278–289, Jun. 2019.

[9] F. Guo, C. Liu, A. Kannan, T. Minka, M. Taylor, Y.-M. Wang, and C. Faloutsos, "Click chain model in Web search," in *Proc. WWW*, Madrid, Spain, 2009, pp. 11–20

[10] A. Chuklin, I. Markov, and M. D. Rijke, "Basic click models," in *Click Models for Web Search*. San Rafael, CA, USA: Morgan & Claypool, 2015, pp. 9–22.

[11] A. Borisov, I. Markov, M. de Rijke, and P. Serdyukov, "A neural click model for Web search," in *Proc. WWW*, Montreal, QC, Canada, 2016, pp. 531–541

[12] A. Borisov, M. Wardenaar, I. Markov, and M. de Rijke, "A click sequence model for Web search," in *Proc. SIGIR*, Ann Arbor, MI, USA, 2018, pp. 45–54.

[13] F. Diaz, Q. Guo, and R. W. White, "Search result prefetching using cursor movement," in *Proc. SIGIR*, Pisa, Italy, 2016, pp. 609–618.

[14] X. Wang, N. Golbandi, M. Bendersky, D. Metzler, and M. Najork, "Position bias estimation for unbiased learning to rank in personal search," in *Proc. WSDM*, Marina del Rey, CA, USA, 2018, pp. 610–618.

[15] J. Mao, C. Luo, M. Zhang, and S. Ma, "Constructing click models for mobile search," in *Proc. SIGIR*, Ann Arbor, MI, USA, 2018, pp. 775–784.

[16] A. Borisov, J. Kiseleva, I. Markov, and M. de Rijke, "Calibration: A simple way to improve click models," in *Proc. CIKM*, Torino, Italy, 2018, pp. 1503–1506.

[17] D. Sculley, R. G. Malkin, S. Basu, and R. J. Bayardo, "Predicting bounce rates in sponsored search advertisements," in *Proc. KDD*, Paris, France, 2009, pp. 1325–1334.

[18] Y. Zhang, H. Dai, C. Xu, J. Feng, T. Wang, J. Bian, B. Wang, and T.-Y. Liu, "Sequential click prediction for sponsored search with recurrent neural networks," in *Proc. AAAI*, Quebec City, QC, Canada, 2014, pp. 1369–1375.

[19] P. P. K. Chan, X. Hu, L. Zhao, D. S. Yeung, D. Liu, and L. Xiao, "Convolutional neural networks based click-through rate prediction with multiple feature sequences," in *Proc. IJCAI*, Stockholm, Sweden, 2018, pp. 2007–2013.

[20] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, "Deep interest network for click-through rate prediction," in *Proc. KDD*, London, U.K., 2018, pp. 1059–1068.

[21] M. Gan and K. Xiao, "R-RNN: Extracting user recent behavior sequence for click-through rate prediction," *IEEE Access*, vol. 7, pp. 111767–111777, Jul. 2019.

[22] X. Bai, R. Abasi, B. Edizel, and A. Mantrach, "Position-aware deep character-level CTR prediction for sponsored search," *IEEE Trans. Knowl. Data Eng.*, to be published.

[23] W. Ouyang, X. Zhang, S. Ren, C. Qi, Z. Liu, and Y. Du, "Representation learning-assisted click-through rate prediction," in *Proc. IJCAI*, Macao, China, 2019, pp. 4561–4567.

[24] W. Ouyang, X. Zhang, L. Li, H. Zou, X. Xing, Z. Liu, and Y. Du, "Deep spatio-temporal neural networks for click-through rate prediction," in *Proc. KDD*, Anchorage, AK, USA, 2019, pp. 2078–2086.

[25] D. N. da Hora, A. S. Asrese, V. Christophides, R. Teixeira, and D. Rossi, "Narrowing the gap between QoS metrics and Web QoE using above-the-fold metrics," in *Proc. PAM*, Berlin, Germany, 2018, pp. 31–43.

[26] I. Grigorik, "High performance networking in Chrome," in *The Performance of Open Source Applications*, T. Armstrong, Ed. Morrisville, NC, USA: Lulu.com, 2013, pp. 1–19.

[27] Y. Wang, X. Liu, D. Chu, and Y. Liu, "EarlyBird: Mobile prefetching of social network feeds via content preference mining and usage pattern analysis," in *Proc. MobiHoc*, Hangzhou, China, 2015, pp. 67–76.

[28] D. Rozado, A. El Shoghri, and R. Jurdak, "Gaze dependant prefetching of Web content to increase speed and comfort of Web browsing," *Int. J. Hum. Comput. Stud.*, vol. 78, pp. 31–42, Jun. 2015.

[29] G. Keizer. Browser updates: Here's how often Chrome, Firefox, Edge and Safari get refreshed. Computerworld. Accessed: Jun. 22, 2018. [Online]. Available: https://www.computerworld.com/article/3284365/browser-updates-heres-often-chrome-firefox-edge-and-safari-get-refreshed.html
[30] M. Awad, L. Khan, and B. Thuraisingham, "Predicting WWW surfing using multiple evidence combination," *VLDB J.*, vol. 17, no. 3, pp. 401–417, May 2008.
[31] W3Techs. *Usage Statistics of JavaScript as Client-Side Programming Language on Websites*. Accessed: Aug. 4, 2019. [Online]. Available: https://w3techs.com/technologies/details/cp-javascript/all/all
[32] Google Chrome. *What are Extensions?* Accessed: Aug. 4, 2019. [Online]. Available: https://developer.chrome.com/extensions
[33] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[34] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. EMNLP*, Doha, Qatar, 2014, pp. 1724–1734.
[35] M. Abadi *et al.* (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. [Online]. Available: https://www.tensorflow.org/
[36] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. NIPS*, Lake Tahoe, NV, USA, 2013, pp. 3111–3119.
[37] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. ICCV*, Santiago, Chile, 2015, pp. 1026–1034.
[38] G. Tsoumakas, L. Katakis, and I. Vlahavas, "Mining multi-label data," in *Data Mining and Knowledge Discovery Handbook*, O. Maimon and L. Rokach, Eds., 2nd ed. New York, NY, USA: Springer, 2010, pp. 667–685.

MINWOO JOO (S'13) received the B.S. degree in computer and communication engineering from Korea University, Seoul, South Korea, in 2013, where he is currently pursuing the Ph.D. degree in computer science and engineering.

His research interests include autonomous networking, next-generation transport protocols, and deep learning.

WONJUN LEE (M'00–SM'06) received the B.S. and M.S. degrees in computer engineering from Seoul National University, Seoul, South Korea, in 1989 and 1991, respectively, and the M.S. degree in computer science from the University of Maryland, College Park, MD, USA, in 1996, and the Ph.D. degree in computer science and engineering from the University of Minnesota, Minneapolis, MN, USA, in 1999.

In 2002, he joined the Faculty of Korea University, Seoul, where he is currently a Professor with the School of Cybersecurity. He has authored or coauthored over 200 articles in refereed international journals and conferences. His research interests include communication and network protocols, optimization techniques in wireless communication and networking, security, and privacy in mobile computing, and radio frequency-powered computing and networking. He has been served as a Technical Program Committee Member for the IEEE International Conference on Computer Communications (INFOCOM), since 2008. He has also served as a Technical Program Committee Member for the Association for Computing Machinery (ACM) International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), from 2008 to 2009, and the IEEE International Conference on Computer Communications and Networks (ICCCN), from 2000 to 2008, and over 120 international conferences.