Journal of Network and Computer Applications 173 (2021) 102872

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Identifying click-requests for the network-side through traffic behavior

Xingrui Fei [a], Yi Xie [a,*], Shensheng Tang [b], Jiankun Hu [c]

[a] School of Data and Computer Science (GuangDong Province Key Laboratory of Information Security Technology), Sun Yat-sen University, Guangzhou, 510006, China
[b] Department of Electrical and Computer Engineering, St Cloud State University, St Cloud, MN 56301, USA
[c] School of Engineering and Information Technology, University of New South Wales at the Australian Defence Force Academy, Canberra, ACT 2600, Australia

ARTICLE INFO

Keywords:
Web traffic
Click
Traffic behavior
Hidden Markov model
Deep neural network

ABSTRACT

With the rapid development of web-based applications, clicking on hyperlinks has become a general means for accessing various network services. Understanding the visiting behavior of web users not only helps improve the personalized service quality and user experience, but also plays an important role in network management and early threat detection. Click-stream identification is a fundamental issue for user behavior analysis. However, most existing approaches are designed for non-encrypted HTTP requests and only focus on server-side scenarios, which makes them inapplicable to the increasingly popular HTTPS and network-side management. In this work, we propose an encryption-independent scheme from a network-side perspective that adopts the web traffic collected at the network boundary to identify the HTTP(S) requests generated by the click actions of web users. The proposed scheme employs hidden Markov models (HMMs) to describe the time-varying behavior of click and non-click web traffic. A deep neural network (DNN) is integrated into the HMMs to capture the context of web traffic, which eliminates the limitations caused by the independence hypothesis of the traditional HMMs. Finally, a DNN-based rear classifier is proposed to determine the type of HTTP(S) requests according to the fitting degree between the HTTP(S) requests and the HMM-based behavior models. We derive the algorithms for model learning and click identification. Experiments are conducted to validate the proposed approach. Performance-related issues and comparisons are discussed. Results show that both the average precision and recall rate of the proposed approach exceed 92%, which is better than most existing benchmark methods in terms of performance and stability.

1. Introduction

User behavior analysis (UBA) has been widely used to identify customers (Zaim et al., 2019), realize personalized service (Jiang et al., 2019), improve service quality (Wang et al., 2017), monitor and filter the access behavior (Peng et al., 2016; Cao et al., 2020), detect anomalies (Cao et al., 2017) and track the illegal collection of personal information in mobile applications (Liu et al., 2020). In these applications accurately extracting the click-streams is a fundamental issue to be addressed.

A click-stream comprises only those HTTP(S) requests that correspond to the web objects clicked by users. As a click-stream can directly indicate the concerns, intentions, and actions of web users, it has received much attention in UBA. Click-streams are currently extracted using three types of methods. The first type is designed for the server-side or service provider. One of the well-known methods is based on the type of accessed files (Cooley et al., 1999; Huiying and Wei, 2004; Chitraa et al., 2013) and the scripts embedded in the web pages (Silverstein et al., 1999; Spink et al., 2005; Rafter and Smyth, 2001). For example, htm/html files are usually considered to have a greater probability of being clicked by users than other elements of web pages (Anand and Aggarwal, 2012), while the scripts and controllers embedded in webpages can be used to record the mouse actions of users (Benevenuto et al., 2009). In addition, some methods based on machine learning are also used for click identification (Xie and Yu, 2008; Xu et al., 2013). However, the main issue of these type of methods is that they can only record the access paths on a specific web site rather than the complete click-trajectories of users. Considering that a user's web surfing usually involves multiple web sites and servers, client-side methods have been developed to obtain the complete click-actions of users. For example, plugins or resident codes are implanted into the terminals of web users to obtain their click-actions and targets (Atterer et al., 2006; Huang and White, 2010). However, this type of method is not widely accepted because it infringes on the privacy and security of users.

* Corresponding author.
E-mail address: xieyi5@mail.sysu.edu.cn (Y. Xie).

https://doi.org/10.1016/j.jnca.2020.102872
Received 11 January 2020; Received in revised form 20 May 2020; Accepted 13 October 2020
Available online 17 October 2020

To address the limitations of these methods, network-side methods have received a lot of attention (Schneider et al., 2009; Xie et al., 2013; Ben Houidi et al., 2014; Vassio et al., 2016; Rizothanasis et al., 2016). In these works, information located in the fields of HTTP header is measured from the web traffic and is adopted for click-request identification. Commonly used fields include "URL", "Referer", "Content-Type" and "User-Agent". Compared with the methods designed for the server-side and the client-side, the network-side methods based on the information of the HTTP header show better performance because they can reconstruct a user's complete click-trajectory when (s)he is surfing from one web site to another, particularly for those scenarios wherein management strategies are required to be adjusted based on the click-behavior of users, e.g., enterprise networks. However, the popularity of HTTPS has brought new challenges to these methods. As HTTPS encrypts the entire HTTP message, HTTP header-based methods cannot correctly access the required information. Therefore, the traditional network-side identification methods gradually become unworkable. Although the encryption of HTTPS does not involve most of the information used for UBA, such as the source/destination IP/port, the target host, the size of request/response, and the session duration, UBA on the network-side becomes extremely difficult because of the lack of effective methods in identifying click-requests from the HTTPS traffic.

To tackle this challenge, in this work we propose a new scheme to identify click-requests on the network-side, such as the access networks and the backbone networks of the Internet Service Providers (ISPs). As usually only encrypted HTTPS traffic can be observed on the network-side instead of plaintext HTTP, we achieve the click-request identification through traffic behavior without involving any information of the payload. Considering that there are only two possible sources for each HTTP(S) request, i.e., an HTTP(S) request can only be generated by a user's click-action or emitted automatically by a browser to request the embedded element of the web page, the essence of this scheme is to solve a binary classification problem for the HTTP(S) traffic. In contrast to existing works, the proposed scheme utilizes the inherent behavior characteristics of each type of the HTTP(S) request for identification rather than relying on the plain text information of the HTTP header. Consequently, the proposed scheme is encryption-independent and does not involve the privacy of users. To this end, we adopt two hidden Markov models (HMMs) to describe the time-varying behavior of the web traffic driven by the click activities of users and the automatic operations of browsers, respectively. Furthermore, a deep neural network (DNN) is integrated into the HMMs to capture the context of the web traffic and eliminate the limitations caused by the independence hypothesis of the traditional HMMs. Finally, a DNN-based rear classifier is proposed to determine the type of the HTTP(S) requests according to the fitting degree between the HTTP(S) request and the behavior models. The main contributions of this work are threefold.

- A new behavior-based classification scheme is introduced for click-request identification that is suitable for UBA on the network-side and is encryption-independent.
- A new HMM-DNN integrated model is proposed to capture the time-varying behavior patterns of click and non-click web traffic, which makes full use of the context-related information of the HTTP(S) request to be identified.
- New algorithms that do not rely on the Gaussian mixture models are derived for model learning and click-request detection. Experiments are conducted to validate the proposed approach.

The rest of the paper is organized as follows. In Section 2, we review the related works. In Section 3, we introduce the rationale of the proposed scheme. Experiments and performance evaluations are shown in Section 4. We discuss the performance related issues of the proposed approach in Section 5 and finally summarize our work in Section 6.

2. Related work

We summarize three types of methods for click-stream identification, including the server-side, the client-side, and the network-side.

The server-side methods have been well studied in the past two decades. The simplest way is using the suffix name of the accessed files to identify the click actions. For example, HTML files are considered to have a high probability of being accessed by click actions of users, whereas the JPGs and the GIFs are usually automatically requested by browsers (Cooley et al., 1999); the CGIs and the "robots.txt" are often accessed by the scripts (Huiying and Wei, 2004) and the crawlers (Anand and Aggarwal, 2012; Chitraa et al., 2013), respectively. According to these experiences, click-requests can be identified by filtering the files that are unrelated to the click-actions of users. Although this type of methods is simple and intuitive, it lacks versatility. For instance, on a web site that provides image download services, JPG files are likely to be clicked by users, which makes the file suffix unusable. In addition to the file name, another alternative is to use embedded scripts and controllers to capture the behavior of users. For example, scripts are embedded in search engines (Silverstein et al., 1999; Spink et al., 2005) to log users' search behavior. The basis of this method is that the HTTP(S) requests driven by the click actions of users usually include submission information, such as query terms and user-specified modifiers, but those issued automatically by the browsers usually do not carry additional parameters with the URLs. According to this assumption, researchers use scripts to record requests with parameters and label them as users' click-requests. Rafter and Smyth (2001) proposed a system that builds a user profile from analyzing the server log of a job search site. In this system, each line of the log file records an action of user-click, which is obtained by ignoring requests unrelated to the job service. Another similar example comes from social networking applications. A social network aggregator (Benevenuto et al., 2009) provides a common interface for users to access their accounts in various online social networks and directly obtain the click-stream data of users through the embedded scripts. In addition, machine learning is also used for click identification on the server-side. For example, hidden semi-Markov model is adopted to model the arrival process of the HTTP requests observed on the server-side (Xie and Yu, 2008; Xu et al., 2013). Based on the hyperlink relationship between the web pages provided by the web site, the model can be well trained and used for click identification. The main limitation of these server-side methods is that they can only extract the local click-paths for a specific web site instead of the full click-trajectories on the Internet. Moreover, some factors such as the proxy server and browser cache may affect the integrity of the server data, leading to incomplete click-paths.

In order to obtain a complete click-path of each web user, client-side methods have been developed. For example, a web proxy is deployed in an enterprise network and inserts scripts into each web page forwarded to the users to retrieve the mouse and keyboard events of users (Atterer et al., 2006). Another common method is to install plugins in the browser to record the interaction information between users and servers, e.g., "Timestamp", "URL" and "Referer" (Huang and White, 2010). Although the click-streams can be obtained directly by implanting codes in the terminals of users, the client-side methods are often criticized because they involve user privacy and system security.

To address the limitations of the above two methods, network-side methods have attracted widespread attention. In these methods, the fields of HTTP header are widely used for click-request identification, e.g., "Content-Type", "Referer" and "URL". In Schneider's work (Schneider et al., 2009), regular expressions were used to represent the URL patterns for click and non-click HTTP requests and were further employed to filter and identify the click-requests. Xie et al. (2013) assumed that each click-request should have at least K embedded objects. Based on this assumption, they first used the field of "Referer" to determine the hyperlink relationship between the accessed web objects and then identified the click-stream by using

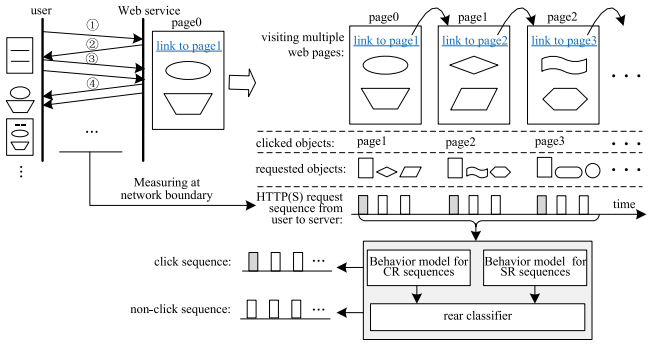


Fig. 1. Framework of the proposed scheme.

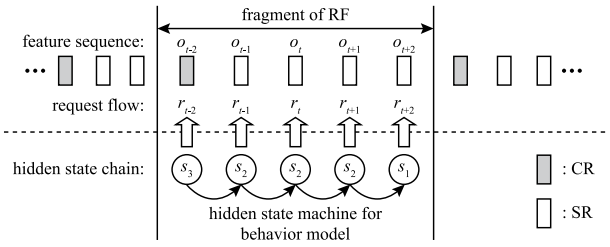


Fig. 2. Modeling the traffic behavior by FSM.

the rules with a predefined K. In Ben's work (Ben Houidi et al., 2014), the documents indicated by the "Referer" field of a request were regarded as a click-request. Then four filtering mechanisms were designed, including F-Referer, F-Children, F-Type, and F-Ad. F-Referer was used to filter all possible click-requests based on the "Referer" field. F-Children was responsible for eliminating those requests with a small number of embedded objects. F-Type and F-Ad were used to further exclude non-click requests by the predefined filename suffixes and advertising blacklist, respectively. In Vassio's work (Vassio et al., 2016), each request was described by 18 features selected from the HTTP header, including the "Content Type", the length of the "URL", and the number of children web pages derived from the main document. These features were divided into N decision trees that determine the click requests by voting. According to our survey, most existing works rely on the information from the HTTP header, which causes them to be used only for non-encrypted traffic instead of HTTPS scenarios. In addition, even under non-encrypted conditions, some HTTP header fields may be unavailable because they are hidden for security reasons (Ruiz-Martnez, 2012), e.g., the "Referer" field. In order to deal with the HTTPS traffic, Rizothanasis proposed a method based on time thresholds (Rizothanasis et al., 2016), where a request was determined as a click-request if the time interval between it and the two adjacent requests before and after is greater than the expected thresholds. The previous interval represents the duration between two consecutive click operations, whereas the next interval means that the related inline requests will appear at least one RTT after clicking. The limitation of this method is that it relies on the predefined thresholds, which affects its versatility and makes it unsuitable for the dynamically changing network scenarios.

In summary, server-side methods cannot obtain a complete click-trajectory of a user, while the client-side methods infringe on the privacy and security of users. Because most existing network-side methods rely on the plaintext information in the HTTP header, they do not apply to the increasingly popular HTTPS.

3. The proposed scheme

3.1. Rationale

Fig. 1 shows the basic framework of the proposed scheme. We start with a simple scenario in which a user clicks a hyperlink of a webpage. As shown in the upper left part of Fig. 1, the interaction process can be abstracted into four steps. First, the user's click-action triggers an HTTP(S) request to obtain the main page, i.e., the arrow labeled by ① in Fig. 1. Second, the server returns the main web document that contains the hyperlinks of all embedded objects to the user, i.e., the arrow labeled by ② in Fig. 1. Third, the user's browser automatically issues two HTTP(S) requests for all embedded objects according to the hyperlinks given in the returned document, i.e., the arrow labeled by ③ in Fig. 1. Finally, the browser obtains all inline objects from the server and renders them on the screen, i.e., the arrow labeled by ④ in Fig. 1. Based on the single-page access, we can further describe the process of continuously accessing multiple pages, and illustrate it in the upper right part of Fig. 1. In this process, there are three elements related to this work, including the clicked objects, requested objects, and the HTTP(S) request sequence from user to server. The clicked objects refer to the targets directly requested by the click actions of users, e.g., the HTML files. The requested objects represent a series of required web resources that are related to a clicked object, including the clicked object and its embedded objects. The HTTP(S) request sequence from user to server refers to the HTTP(S) traffic corresponding to the requested objects.

During the visit of page0 shown in Fig. 1, there are two types of HTTP(S) requests, including a click request (CR) and two spontaneous requests (SRs). The CR corresponds to the target web object that is requested via the user's active click-action, such as the main web document denoted by a rectangle shown in Fig. 1. The SRs are launched automatically by the user's browser to download the embedded objects of the clicked webpage, such as the requests for objects denoted by the ellipse and the trapezoid shown in Fig. 1. In contrast to existing works, we do not determine the category of each HTTP(S) request by analyzing its features in isolation, but from the perspective of time-varying behavior patterns of the request-flow (RF). Because there are only two types of requests (CRs/SRs), any fragment of an RF can only be one of two situations: starting with a CR or an SR. Therefore, the attribute inference of a single HTTP(S) request can be converted into a category analysis of an RF-fragment, i.e., using sequence analysis instead of individual detection. Theoretically, the sequence behavior can provide more available information for the CR identification than a single request because the former makes full use of the context-related information of the HTTP(S) request for its click attribute identification.

Based on this idea, at the lower right of Fig. 1 we adopt two models to describe the potential behavior mechanisms for the fragments of the CR-sequences and SR-sequences, respectively. Thus, the classification of an RF-fragment can be further solved by evaluating how well it fits the behavior models, which is a typical time-series classification problem. Traditionally, the likelihood of an RF-fragment fitting to a given model has been widely used as an indicator for classification, i.e., the label of the model with the maximum likelihood is considered the optimal category of the RF-fragment to be detected. However, this hard-decision-based method often leads to large deviations. In particular, when the likelihoods of the candidate models have similar statistical probabilities, it is not easy to make an effective distinction. To address this limitation, in this work we do not directly determine the category of a fragment by the values of the fitting degree between it and the behavior models, but achieve the final classification by adopting a DNN-based rear classifier to analyze the joint statistical distribution of the fitting degrees calculated by the behavior models.

To derive an analytical mathematical model, we use two finite state machines (FSMs) to describe the time-varying behavior mechanisms of CR-sequences and SR-sequences, respectively. The states of these FSMs come from a shared discrete state set in which each element represents a specific behavior pattern for launching an HTTP(S) request and controls the external features of the upstream requests. As shown

in Fig. 2, for a given RF-fragment, there are two related time-varying processes: an observable feature sequence and a hidden state chain. The observable feature sequence denotes the external attributes of the RF-fragment starting with the HTTP(S) request to be identified, whereas the hidden state chain describes the underlying time-varying behavior evolution mechanism of an observed RF-fragment and controls its external behavior features. The transition between two adjacent states represents a switch in behavior patterns of the RF-fragment. In most practical applications, the hidden state chain is usually unmeasurable and can only be estimated by the observable feature sequence.

Similar to most machine-learning-based solutions, the proposed scheme consists of two phases. The first phase is model learning, which estimates the parameters of both the FSM-based behavior models and the DNN-based rear classifier. The second phase is real-time detection based on the trained behavior models and classifier. For a given HTTP(S) request r, identification of its click attribute is equivalent to the category analysis of an RF-fragment that starts with r. The category analysis of an RF-fragment can be further solved by two steps: (i) evaluating the fitting degree of the RF-fragment to each behavior model and (ii) utilizing the rear classifier to evaluate the values of the fitting degree and determine the click attribute of the RF-fragment.

3.2. Formulation

Let r_w denote the wth HTTP(S) request of an RF-fragment with length W. Let O_w and S_w denote the random variables for the observable feature vector and the corresponding hidden state of r_w, respectively. Accordingly, we use the lowercase variables o_w and s_w to denote the instances of O_w and S_w, respectively. Thus, for an RF-fragment $r_{1:W} = (r_1, r_2, \dots, r_W)$, $o_{1:W} = (o_1, o_2, \dots, o_W)$ and $s_{1:W} = (s_1, s_2, \dots, s_W)$ form a complete sequence-pair to characterize the time-varying behavior of $r_{1:W}$, including the external observable traffic features and internal unmeasurable driving mechanisms. We further define an initial state function π_i to denote the probability that behavior pattern i is selected to generate r_1:

$$\pi_i = \Pr[S_1 = i], i \in Q, \tag{1}$$

where Q is the set of all possible hidden states.

Theoretically, any two elements in the hidden state sequence $s_{1:W}$ are related and interacting, i.e., the choice of any state $s_w, (w \in Q)$ requires a comprehensive consideration of the impact of the other states $s_{1:W} - \{s_w\}$. To highlight the modeling approach of the proposed scheme, we only take the first-order neighboring relationship as an example to introduce the modeling scheme, i.e., the hidden state of an HTTP(S) request is only affected by the state of its adjacent previous request.[1] To this end, the interaction model of behavior patterns can be described by Eq. (2):

$$\Pr[S_w | s_{1:W} - \{s_w\}] = \Pr[S_w | s_{w-1}]. \tag{2}$$

Let a_{ij} denote the conditional probability for the adjacent behavior patterns:

$$a_{ij} = \Pr[S_w = j | S_{w-1} = i], (i, j) \in Q. \tag{3}$$

We further use the output probability function $b_i(\vec{o}_w)$ to describe the statistical relationship between the underlying behavior pattern $i \in Q$ and the observed traffic features $\vec{o}_w$ of r_w:

$$b_i(\vec{o}_w) = \Pr[\vec{o}_w | S_w = i], i \in Q, \tag{4}$$

where $\vec{o}_w = (o_{1,w}, o_{2,w}, \dots, o_{D,w})^{\mathrm{T}}$ denotes the D-dimensional observation vector of r_w.

[1] The model based on higher-order state contexts can be derived by a similar method. Due to the limited space, we do not discuss it here.

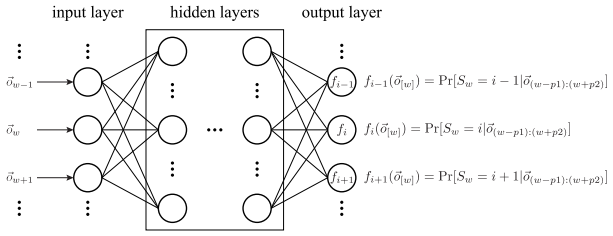


Fig. 3. DNN for the emission probability function of HMM.

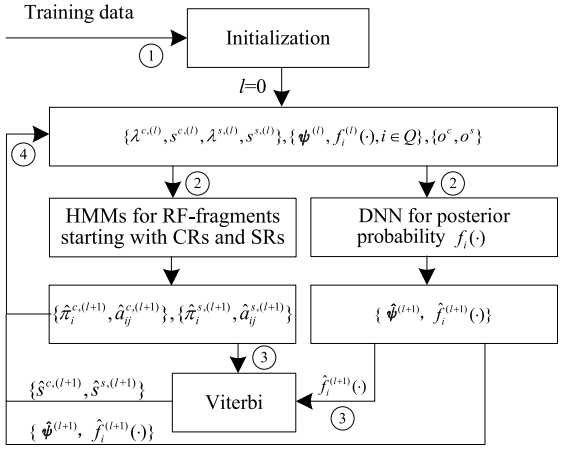


Fig. 4. Estimating the parameters of the behavior models.

These formulas form a typical first-order HMM (Rabiner, 1989). Although classic HMMs have shown excellent performance in time-series modeling, they suffer from three innate limitations. First, the independence assumption widely adopted by HMMs causes the model to be unable to describe the context of observations. Second, in most HMMs the emission probability function $b_i(\cdot)$ is usually limited to common statistical distributions such as the Gaussian mixture model (GMM), which makes it difficult to describe the complex and nonlinear relationships between the hidden states and the observations. Moreover, it is difficult for the classic HMMs to process the high-dimensional observation vectors.

Thus, we develop a new context-dependent HMM based on DNN. Different from the definition of $b_i(\cdot)$, we define a context-dependent posterior probability function $f_i(\vec{o}_{[w]})$ by Eq. (5):

$$f_i(\vec{o}_{[w]}) = \Pr[S_w = i | \vec{o}_{(w-p1):(w+p2)}], \tag{5}$$

where $p1$ and $p2$ represent the lower and upper limits of the RF-fragment's truncation window, respectively. We further adopt a DNN to calculate the $f_i(\cdot), (i \in Q)$, which is shown in Fig. 3. The input layer of the DNN consists of $p1+p2+1$ neuron nodes. Because the observation vector is D-dimensional, each neuron of the input layer shown in Fig. 3 is actually composed of D sub-nodes. The output layer consists of $|Q|$ neuron nodes, each of which outputs a posterior probability of the corresponding hidden state when $\vec{o}_{(w-p1):(w+p2)}$ is given.

Consequently, the parameter of the proposed approach consists of three parts: (i) the initial state function and the state transition function of HMMs $\lambda^u = \{\pi_i^u, a_{ij}^u\}$, where $u \in \{c, s\}$ denotes the model instance for CR- or SR-sequences, (ii) the parameters ψ of the DNN for calculating the context-dependent posterior probability $f_i(\cdot)$, and (iii) the parameters χ of the DNN-based rear classifier.

3.3. Parameters and states estimation

Different from the existing GMM-based DNN-HMM designed for speech recognition (Yan et al., 2013) and content type recognition (Tan

et al., 2019), in this work, we introduce a new GMM-free training algorithm that makes the proposed integrated HMM-DNN scheme independent from the GMM. We estimate the parameters of the HMM-DNN models simultaneously using a new iteration approach. Fig. 4 shows our parameter estimation method. There are three model instances in this scheme, including two independent HMMs for RF-fragments starting with CRs and SRs, respectively, and a shared DNN for the posterior probability function $f_i(\cdot), (i \in Q)$. Considering that both HMMs for CRs and SRs use the same parameter estimation algorithm, except for the training data in the following algorithm derivation, we do not make a distinction between the two HMM instances unless a special note is made.

As shown in Fig. 4, the iterative training of the integrated HMM-DNN model consists of four steps:

- *Step* 1: using the training data to initialize the parameters of the models and estimate the hidden states,
- *Step* 2: updating the parameters of the HMMs and DNN simultaneously based on the results of the previous iteration,
- *Step* 3: inferring the new hidden states for the observations by the updated parameters of the models based on the Viterbi algorithm (Forney, 1973).
- *Step* 4: looping until the stop condition of the iteration is met.

To achieve the above method, we let $\alpha_i^k(w)$ and $\beta_i^k(w)$ denote the forward and backward variables of the kth $\in K$ sequence sample, respectively, where K is the size of the training CR-fragments or SR-fragments. The definitions and recursive formulas of $\alpha_i^k(w)$ and $\beta_i^k(w)$ are given by Eqs. (6) and (7), respectively.

$$\begin{aligned}\alpha_i^k(w) &= \Pr[\vec{o}_{1:w}^k, S_w^k = i|\lambda] \\ &= \begin{cases}\pi_i \rho_w(i) f_i(\vec{o}_{[w]}^k), & w = 1 \\ \sum_{j=1}^{|Q|} \alpha_j^k(w-1) a_{ji} \rho_w(i) f_i(\vec{o}_{[w]}^k), & w > 1\end{cases},\end{aligned} \tag{6}$$

$$\begin{aligned}\beta_i^k(w) &= \Pr[\vec{o}_{(w+1):W^k}^k | S_w^k = i, \lambda] \\ &= \begin{cases}\sum_{j=1}^{|Q|} a_{ij} \rho_{w+1}(j) f_j(\vec{o}_{[w+1]}^k) \beta_j^k(w+1), w < W^k \\ 1, w = W^k\end{cases},\end{aligned} \tag{7}$$

where W^k denotes the length of the kth sample and $\rho_w(i) = \Pr[\vec{o}_w^k]/\Pr[S_w = i]$. $\Pr[\vec{o}_w^k]$ and $\Pr[S_w = i]$ are the priori probabilities of the observed feature $\vec{o}_w^k$ and the hidden state i, respectively.

Let $\xi_{ij}^k(w)$ denote the posterior joint probability of two adjacent hidden states S_w^k and S_{w+1}^k given the feature sequence $\vec{o}_{1:W^k}^k$:

$$\begin{aligned}\xi_{ij}^k(w) &= \Pr[S_w^k = i, S_{w+1}^k = j | \vec{o}_{1:W^k}^k, \lambda] \\ &= \alpha_i^k(w) a_{ij} \rho_{w+1}(j) f_j(\vec{o}_{[w+1]}^k) \beta_j^k(w+1)/P^k.\end{aligned} \tag{8}$$

P^k is the likelihood of the kth sequence sample relative to the model λ, which acts as the normalization factor and is calculated using Eq. (9):

$$P^k = \Pr[\vec{o}_{1:W^k}^k|\lambda] = \sum_{i=1}^{|Q|} \alpha_i^k(W^k). \tag{9}$$

Let $\gamma_i^k(w)$ denote the posterior probability of S_w^k given the feature sequence $\vec{o}_{1:W^k}^k$:

$$\gamma_i^k(w) = \Pr[S_w^k = i|\vec{o}_{1:W^k}^k, \lambda] = \sum_{j=1}^{|Q|} \xi_{ij}^k(w)/P^k. \tag{10}$$

Based on the forward–backward algorithm (Rabiner, 1989), the initial state probability function π_i and the state transition probability function a_{ij} of the HMM-DNN can be estimated by Eqs. (11) and (12), respectively.

$$\hat{\pi}_i = \frac{1}{K} \sum_{k=1}^{K} \gamma_i^k(1). \tag{11}$$

$$\hat{a}_{ij} = \frac{\sum_{k=1}^{K} \sum_{w=1}^{W^k-1} \xi_{ij}^k(w)}{\sum_{k=1}^{K} \sum_{w=1}^{W^k-1} \gamma_i^k(w)}. \tag{12}$$

Given the parameters of $\{\hat{\pi}_i, \hat{a}_{ij}\}$ and the posterior probability $\hat{f}_i(\cdot)$, we can infer the hidden state sequence for each RF-fragment via the Viterbi algorithm. For a feature sequence $\vec{o}_{1:W^k}$, let $\delta_i^k(w)$ denote the maximum probability of the underlying state path that accounts for the first w requests and ends in state i. The definition and the recursive formulas of $\delta_i^k(w)$ are given by Eq. (13):

$$\begin{aligned}\delta_i^k(w) &= \max_{S_1,\ldots,S_{w-1}} \Pr[S_1, \ldots, S_w = i, \vec{o}_{1:w}^k|\lambda] \\ &= \begin{cases}\pi_i \rho_w(i) f_i(\vec{o}_{[w]}^k), & w = 1 \\ \max_{j \in Q}[\delta_j^k(w-1) a_{ji}] \rho_w(i) f_i(\vec{o}_{[w]}^k), & w > 1\end{cases}.\end{aligned} \tag{13}$$

Let $\phi_w^k(i)$ record the state of S_{w-1} of the path with the maximum probability:

$$\phi_w^k(i) = \begin{cases}0, & w = 1 \\ \arg\max_{j \in Q}[\delta_j^k(w-1) a_{ji}], & w > 1\end{cases}. \tag{14}$$

Then, we can obtain $\hat{s}_{1:W^k}^k$ by backtracking the state path:

$$\begin{cases}\hat{s}_{W^k}^k = \arg\max_{i \in Q}[\delta_i^k(W^k)], \\ \hat{s}_w^k = \phi_{w+1}^k(\hat{s}_{w+1}^k)\end{cases}. \tag{15}$$

To evaluate the fitting degree between the model and the data, we define an average logarithmic probability $\mathcal{L}$ by

$$\mathcal{L} = \sum_{k=1}^{K} \frac{1}{KW^k} \log \Pr[o_{1:W^k}^k|\lambda] = \sum_{k=1}^{K} \frac{\log P^k}{KW^k}. \tag{16}$$

In the above iterative operations, we require two functions: the coefficient $\rho_w(i)$ and the posterior state probability $f_i(\cdot)$. $\rho_w(i)$ is calculated using the priori probabilities of $\Pr[\vec{o}_w^k]$ and $\Pr[S_w = i]$. For a given training data set, $\Pr[\vec{o}_w^k]$ is a constant that can be obtained by the statistical distribution of the observations, and $\Pr[S_w = i]$ can be estimated by the frequency of each hidden state based on the state sequences inferred by the Viterbi algorithm after each iteration.

Different from the traditional HMMs that use regular statistical distributions (e.g., GMM) or a discrete state output matrix to describe the conditional probability $b_i(\vec{o}_w)$, here we adopt the context-related posterior probability function $f_i(\cdot)$ and calculate it using a DNN. The DNN consists of a full-connection BP neural network with $L + 1$ layers, including one input layer, $L - 1$ hidden layers, and one output layer (Jain et al., 1996). The numbers of neurons in the input layer and output layer are equal to $(p1 + p2 + 1)D$ and $|Q|$, respectively. Let x_n^l $(n \in Z_l)$ denote the output of the nth neuron in the lth layer, V_{mn}^l $(m \in Z_l, n \in Z_{l-1})$ denote the connection weight between the nth neuron of the $(l-1)$th layer and the mth neuron of the lth layer, and θ_m^l $(m \in Z_l)$ denote the offset of the mth neuron in layer l. Then, the output of the mth neuron in layer l is calculated by Eq. (17):

$$x_m^l = \sigma\left(\sum_{n=1}^{Z_{l-1}} V_{m,n}^l x_n^{l-1} + \theta_m^l\right), \tag{17}$$

where $\sigma(z) = (e^z - e^{-z})/(e^z + e^{-z})$ is hyperbolic tangent.

Given a training data $\vec{o}_{[w]}^k$ and its hidden state $\hat{s}_w^k$ obtained by the Viterbi algorithm, the purpose of DNN learning is to realize the following equation by adjusting its weight parameters:

$$\begin{aligned}f_i(\vec{o}_{[w]}^k) &= \Pr[S_w^k = i|\vec{o}_{[w]}^k] \\ &= \exp(x_i^L)/\sum_{j=1}^{|Q|} \exp(x_j^L) = \begin{cases}0, i \neq \hat{s}_w^k \\ 1, i = \hat{s}_w^k\end{cases}.\end{aligned} \tag{18}$$

Algorithm 1 Parameter learning for HMM-DNN

Input: $o^u, \iota, \varepsilon, u \in \{c, s\}$;
Output: $\hat{\lambda}^u, \hat{\psi}, u \in \{c, s\}$;
1: $l \Leftarrow 0$, $\Delta \Leftarrow +\infty$;
2: Initialize $\hat{\lambda}^{u,(l)}$ and $\hat{s}^{u,(l)}$ for $\forall u \in \{c, s\}$; $\hat{\psi}^{(l)}$ and $\hat{f}_i^{(l)}(\cdot)$ for $\forall i \in Q$;
3: **while** $(l < \iota)$ and $(\Delta > \varepsilon)$ **do**
4: **for** $\forall u \in \{c, s\}$ **do**
5: Calculate $\alpha_i^{u,k}(w)$, $\beta_i^{u,k}(w)$, $\xi_i^{u,k}(w)$, $\gamma_i^{u,k}(w)$, $\delta_i^{u,k}(w)$, $\phi_i^{u,k}(w)$, for $\forall i \in Q$, $\forall w \in W^k$ and $\forall k \in K$;
6: Estimate $\hat{\lambda}^{u,(l+1)}$ and $\mathcal{L}^{u,(l+1)}$;
7: **end for**
8: Calculate $E^{(l+1)}$;
9: Update $\hat{\psi}^{(l+1)}, \hat{f}_i^{(l+1)}(\cdot)$ for $\forall i \in Q$;
10: Estimate $\hat{s}^{u,(l+1)}$ for $u \in \{c, s\}$;
11: $l \Leftarrow l + 1$; $\Delta \Leftarrow \sum_u |\mathcal{L}^{u,(l+1)} - \mathcal{L}^{u,(l)}|$;
12: **end while**

To evaluate the error between the theoretical output and the DNN's actual output, we define the cross-entropy E by Eq. (19):

$$E = -\frac{1}{A}\sum_{k=1}^{K}\sum_{w=1}^{W^k}\sum_{i=1}^{|Q|} f_i(\vec{o}_{[w]}^k)\log \hat{f}_i(\vec{o}_{[w]}^k), \tag{19}$$

where $A = \sum_{k=1}^{K} W^k$ is the total number of observation vectors for DNN training. $\hat{f}_i(\cdot)$ and $f_i(\cdot)$ are the actual output and the theoretical value of the ith neuron node in the DNN's output layer, respectively.

The weights and offsets are updated using the BP algorithm (Rumelhart et al., 1986) at every iteration as follows:

$$\begin{cases} V_{ij}^l \leftarrow V_{ij}^l - \eta \dfrac{1}{A}\displaystyle\sum_{a=1}^{A} \frac{\partial E(a)}{\partial V_{ij}^{l,(a)}} \\ \theta_i^l \leftarrow \theta_i^l - \eta \dfrac{1}{A}\displaystyle\sum_{a=1}^{A} \frac{\partial E(a)}{\partial \theta_i^{l,(a)}} \end{cases}, \tag{20}$$

where η and $E(a)$ denote the learning rate and the cross-entropy, respectively. The updated items are calculated by

$$\begin{cases} \dfrac{\partial E(a)}{\partial V_{ij}^{l,(a)}} = \boldsymbol{\Phi}_i^l x_j^{l-1} \\ \dfrac{\partial E(a)}{\partial \theta_i^{l,(a)}} = \boldsymbol{\Phi}_i^l \end{cases}, \tag{21}$$

and the recursive formula of $\boldsymbol{\Phi}_i^l$ is given by Eq. (22).

$$\boldsymbol{\Phi}_i^l = \begin{cases} [\hat{f}_i(\vec{o}_{[w]}^k) - f_i(\vec{o}_{[w]}^k)]\sigma'(z_i^L), & l = L \\ \displaystyle\sum_{m=1}^{Z_{l+1}} V_{mi}^{l+1}\boldsymbol{\Phi}_m^{l+1}\sigma'(z_i^l), & l < L \end{cases}, \tag{22}$$

where $z_m^l = \sum_{n=1}^{Z_{l-1}} V_{mn}^l x_n^{l-1} + \theta_m^l$ denotes the input of the mth neuron in the lth layer. The training for the DNN stops when the cross-entropy reaches the expected threshold.

We show the complete pseudocode for the parameter training of the integrated HMM-DNN model in Algorithm 1, where the right superscript $u \in \{c, s\}$ is used to label the behavior models of CR-sequences and SR-sequences. The inputs of the algorithm are the observation sequences, the maximum number of iterations ι, and a minimum positive number ε. ι and ε are used to evaluate the convergence of the model and are assigned before implementing the algorithm. The output is the parameters of HMMs and DNN. In the initialization process (2nd line), $\hat{\lambda}^{u,(0)}$ and $\hat{s}^{u,(0)}$ can be initialized by prior knowledge or clustering approaches, e.g., K-means. After $\hat{\psi}^{(0)}$ is initialized by pre-training, it can be used to calculate the $\hat{f}_i^{(0)}(\cdot)$. Before the stop condition is reached, there are three parts to be trained in each iteration, including two HMMs (lines 4-7) and one DNN (lines 8 and 9). All of them can be executed in parallel, as they are independent of each other. $\hat{s}^{u,(l+1)}$ shown in the 10th line is estimated by the Viterbi algorithm, i.e., Eqs. (13)–(15).

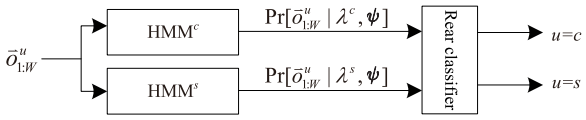


Fig. 5. Identifying the CR by the rear classifier.

Algorithm 2 Click attribute identification

Input: $\vec{o}_{1:W}$;
Output: $u, u \in \{c, s\}$;
1: **for** $w = 1 \rightarrow W$ **do**
2: Calculate $f_i(\vec{o}_{[w]})$;
3: **end for**
4: Calculate P_u for $u \in \{c, s\}$;
5: Calculate the click attribute of r_1;

Once the parameter estimation of the behavior models is completed, the rear classifier is trained. Let r_1 denote the HTTP(S) request to be detected. As shown in Fig. 5, the classifier uses the likelihoods outputted by the two HMM-DNN models to classify the RF-fragment $r_{1:W}$ and further determines the click attribute of r_1. The proposed rear classifier consists of a fully connected DNN that includes two inputs and two outputs. The inputs are the likelihoods of labeled training sequences calculated through the two behavior models, whereas the outputs are the labels of the RF-fragments to be identified. Similar to the above DNN for the context-related posterior probability function $f_i(\cdot)$, the weight coefficients of the rear classifier can be estimated by general DNN algorithms, such as the BP algorithm.

3.4. Click identification and deployment

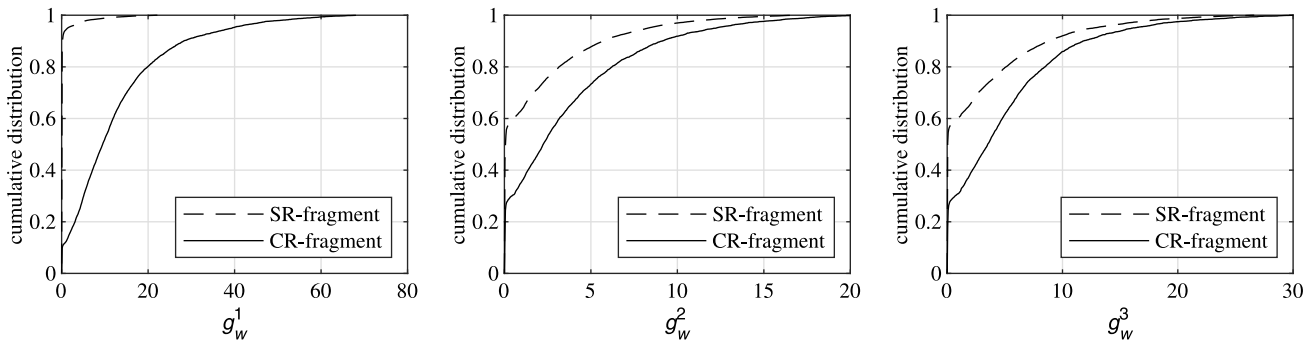
Given an RF-fragment $r_{1:W}$, the mission of the proposed scheme is to determine whether r_1 is emitted by a user's click-action or the automatic operation of the user's browser. The pseudocode of click identification is shown in Algorithm 2. The algorithm proceeds in three steps:

- *Step* 1: Calculate the posterior probability $f_i(\vec{o}_{[w]})$ by the trained DNN (lines 1-3) and the feature sequence $\vec{o}_{1:W}$ of $r_{1:W}$,
- *Step* 2: Calculate the likelihoods (P_c, P_s) of $\vec{o}_{1:W}$ fitting to the CR and SR behavior models, respectively,
- *Step* 3: Using the rear classifier, label the click attribute of r_1 based on the likelihoods (P_c, P_s).

As the proposed scheme is designed for the network-side, it can be easily deployed in actual networks. The deployment involves the following two steps: (i) collecting traffic data from the mirror port of the aggregation switch located at the network and (ii) inputting the collected traffic data into the proposed scheme. The complexity of the entire deployment is mainly contributed by the computation of the models (two HMMs and two DNNs).

4. Experiments

This section evaluates the performance of the proposed scheme on real traffic data provided by a group of volunteers. We develop a simple browser plugin to record the click information (e.g., URL and timestamp) for labeling and install it in each volunteer's browser that is specially configured for this experiment. When the volunteers



(a) cumulative distribution of g_w^1 (b) cumulative distribution of g_w^2 (c) cumulative distribution of g_w^3

Fig. 6. The distributions of features.

are willing to share their click data, they can access the websites through the customized browser. The traffic collected on the mirror port of the aggregation switch is then labeled using the log files of the browser plugins. The collected data were approximately 4 GB in size, and included 17,863 RF-fragments, 1,046 of which are CR-fragments. We further divide these data into two sets for training and testing.[2]

4.1. Evaluation metrics

The evaluation metrics used in this work include accuracy, precision (P), recall (R), $F1$ (Singh et al., 2018), and the kappa coefficient ($Kappa$) (Peng et al., 2016). The accuracy is used to evaluate the classification performance of the shared DNN, and it is the percentage of the observation vectors that are correctly classified. Other metrics are used to evaluate the overall performance of the models, they are defined by Eqs. (23)–(26), respectively.

$$P = \frac{TP}{TP + FP}, \tag{23}$$

$$R = \frac{TP}{TP + FN}, \tag{24}$$

$$F1 = \frac{2 \times P \times R}{P + R}, \tag{25}$$

$$Kappa = \frac{\kappa_0 - \kappa_e}{1 - \kappa_e}, \tag{26}$$

where TP, FP, and FN denote the numbers of true positives, false positives, and false negatives, respectively. $F1$ is a comprehensive metric that can be seen as the harmonic mean of P and R. $Kappa$ is used to measure the agreement between the classification result and the true label. $Kappa \geq 0.75$ signifies excellent agreement, while $Kappa \leq 0.40$ means poor agreement (Fleiss et al., 2003). κ_0 and κ_e in Eq. (26) are calculated by Eqs. (27) and (28):

$$\kappa_0 = \frac{TP + TN}{TP + FN + FP + TN}, \tag{27}$$

$$\kappa_e = \frac{(TN + FP)(TN + FN) + (TP + FP)(TP + FN)}{(TP + FN + FP + TN)^2}, \tag{28}$$

where TN denotes the number of true negatives. In this work, the CR-fragments are treated as the positive samples.

The data analysis process includes the following steps:

- Splitting the RFs into equal-length fragments, and extract the feature sequences,
- Training the HMM-DNN models and the rear classifier,
- Calculating the likelihoods of the test sequences fitting to the HMM-DNN models,
- Inferring the click attribute of each test sequence by the rear classifier,
- Evaluating the performance based on the given metrics.

[2] The data will be openly available via the Internet for research purposes.

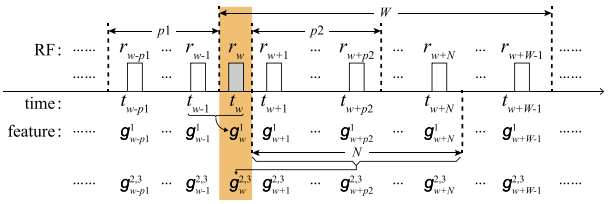


Fig. 7. Schematic diagram of the RF.

4.2. Features

Similar to most of the applications based on machine learning, the choice of observation characteristics will affect the final decision. Here, we only adopt three time-related features to verify the performance of the proposed approach. These features are defined by Eqs. (29)–(31).

$$g_w^1 = t_w - t_{w-1}, \tag{29}$$

$$g_w^2 = \sum_{n=1}^{N} g_{w+n}^1 / N, \tag{30}$$

$$g_w^3 = \left[\sum_{n=1}^{N} (g_{w+n}^1 - g_w^2)^2 / N \right]^{\frac{1}{2}}, \tag{31}$$

where t_w ($w \in \{1, \ldots, W\}$) denotes the time of the wth observed HTTP(S) request r_w. N is the length of the backward truncation window used to calculate g_w^2 and g_w^3 for r_w. Thus, g_w^1 is the time interval between r_w and r_{w-1}, and g_w^2 and g_w^3 are the means and the standard deviation of $g_{(w+1):(w+N)}^1$, respectively. Fig. 6 shows the cumulative distributions of these temporal characteristics for the CR-fragments and the SR-fragments. From a statistical point of view, these features show significant differences, which indicates their usability in click attribute identification. Moreover, these results are consistent with the basic principles of web access shown in Fig. 1. For example, the interval between a CR and its previous request in the same flow mainly depends on the frequency of the click actions, while the interval between two adjacent SRs is usually very small because they are issued concurrently by the browser. Therefore, these time-related features (g_w^1, g_w^2, and g_w^3) are suitable for forming the observation vector $\vec{o}_w = (g_w^1, g_w^2, g_w^3)^{\mathrm{T}}$ for click attribute identification.

To form an observation sequence $\vec{o}_{1:W}$ based on these features, there are four parameters that need to be considered, including the size of the backward truncation window N shown in Eqs. (30) and (31), the length of context ($p1, p2$), and the size of RF-fragment (W). Fig. 7 shows the relationships between these variables and the given request (r_w) to be identified.

Because N cannot be obtained by calculation, here we employ an alternative method to determine the optimal value of N by investigating the joint entropies (H_N) (Adi et al., 2017) of $(g_w^{2,k}, g_w^{3,k})$ and the

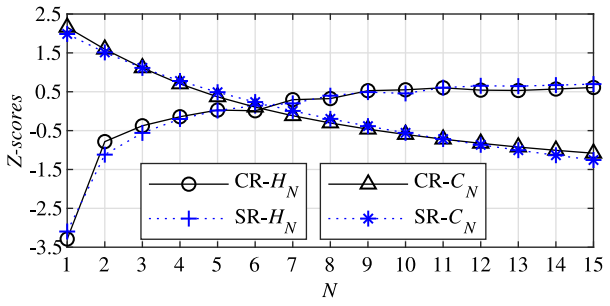
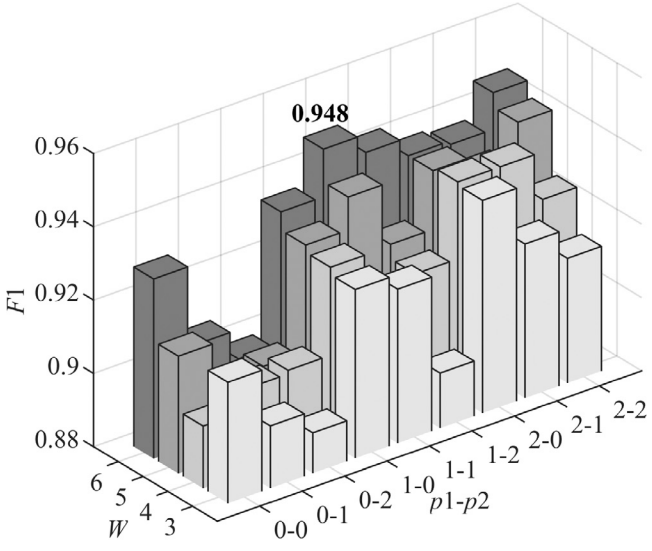
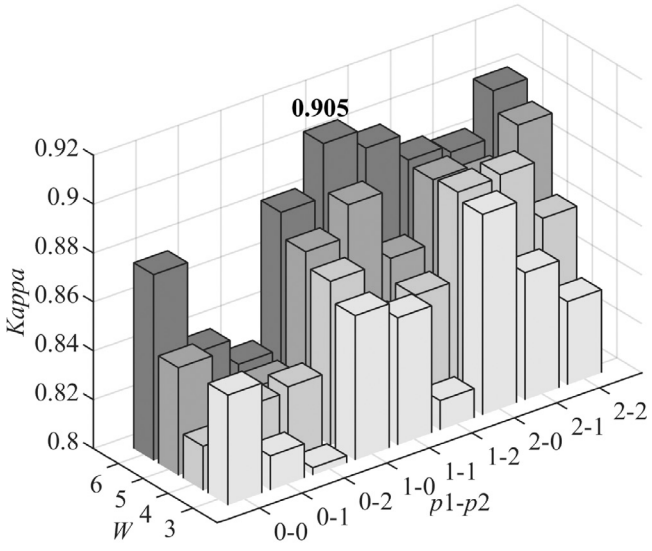


Fig. 8. The influence of N.



(a) $F1$



(b) $Kappa$

Fig. 9. The performance of models with different parameters.

Table 1
The performance of DNN using different activation functions.

Activation functions	Accuracy (%)	Time (s)
sigmoid(x)	95.54	582.61
tanh(x)	96.37	501.05

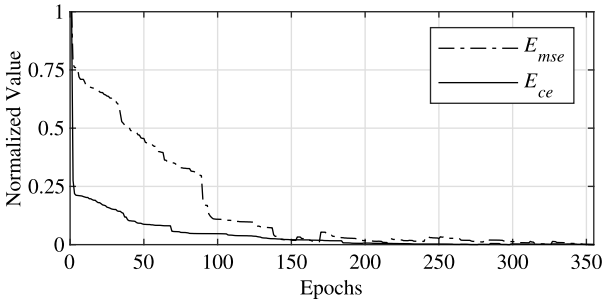


Fig. 10. The curves of cost.

correlation (C_N) between requests. Given K samples of RF-fragments and N, H_N is calculated by

$$H_N = -\sum_{\forall(k,w)} \Pr[g_w^{2,k}, g_w^{3,k}] \log \Pr[g_w^{2,k}, g_w^{3,k}], \tag{32}$$

where $k = \{1, \ldots, K\}$ and $w = \{1, \ldots, W^k\}$. k and W^k denote the index and the length of the kth sample, respectively. C_N is calculated by

$$C_N = \frac{1}{A} \sum_{k=1}^{K} \sum_{w=1}^{W^k} CorrNum(r_w)/N, \tag{33}$$

where $A = \sum_{k=1}^{K} W^k$ is the total number of requests. $CorrNum(r_w)$ represents the number of requests with the same "Host" as r_w in the subsequent N requests. Fig. 8 shows the curves of H_N and C_N vs. N for CRs and SRs. H_N and C_N are normalized by Z-scores (Kreyszig, 2011), respectively. The result shows that both the entropies of the CRs and SRs gradually increase and tend to stabilize as the value of N increases, but both the correlations gradually decrease. To balance the two types of indicators, in the following experiment, we let $N = 6$.

Fig. 9 shows the impact of W, $p1$ and $p2$ on the identification performance. Each value in the figures is the average of 20 experiments. The results indicate the following relationships:

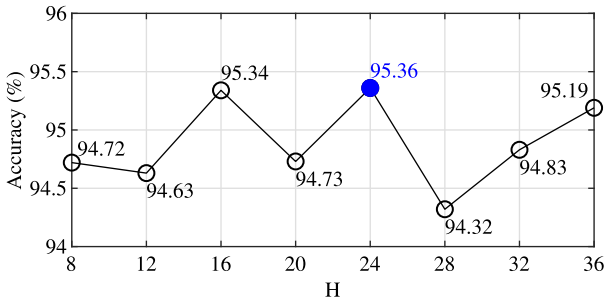
- There is a weak positive correlation between W and the detection performance, including $F1$ and $Kappa$;
- $F1$ is positively correlated with $p1$ but is negatively correlated with $p2$ in some cases, e.g., $W = 3$;
- $Kappa$ shows that all combinations of parameters achieved good results.

Based on these results, in the following experiments, we let W, $p1$, and $p2$ take the values of 6, 1, and 1, respectively.

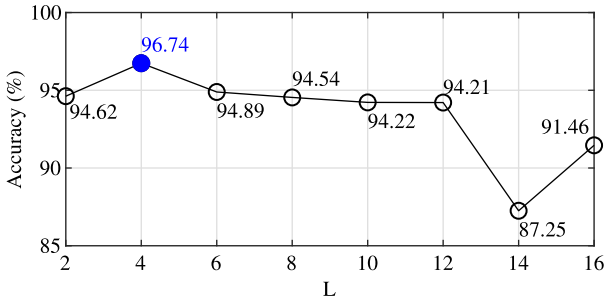
4.3. Configuration of the DNN-based models

We consider three pending factors for the DNN-based models, including the activation function, loss function and structure.

The sigmoid function $\text{sigmoid}(x) = 1/(1 + e^{-x})$ and the hyperbolic tangent $\tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$ are the most frequently used activation functions. $\tanh(x)$ usually converges faster than $\text{sigmoid}(x)$ because its outputs are close to zero (LeCun et al., 2012). In Table 1, we compare the accuracy and time cost of the $\text{sigmoid}(x)$ and the $\tanh(x)$ for the DNN. The results indicate that in the case of similar accuracy, the running time of DNN with $\tanh(x)$ is less than that with $\text{sigmoid}(x)$. Therefore, we adopt $\tanh(x)$ to be the activation function of the DNN in this work.



(a) Accuracy vs. number of hidden layer nodes ($L = 1$)



(b) Accuracy vs. number of hidden layers ($H = 24$)

Fig. 11. Accuracy of the different DNN structures.

The loss function is used to estimate the fitting degree of the model. In addition to the cross-entropy (CE) E_{ce}, the mean square error (MSE) E_{mse} is also often used. Similar to Eq. (19), E_{mse} is defined by

$$E_{mse} = \frac{1}{2A}\sum_{k=1}^{K}\sum_{w=1}^{W^k}\sum_{i=1}^{|Q|}[f_i(\vec{o}^k_{[w]}) - \hat{f}_i(\vec{o}^k_{[w]})]^2. \tag{34}$$

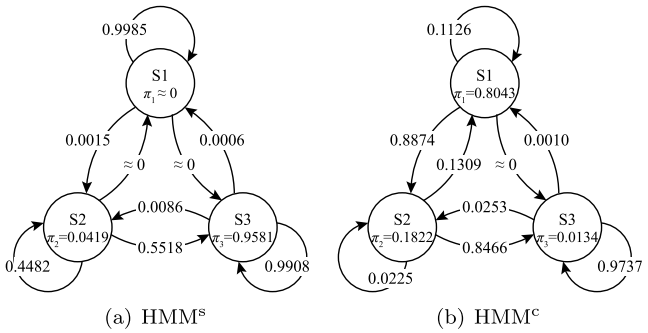
Fig. 10 shows the normalized values of MSE and CE versus the number of epochs in the DNN training phase. The results indicate that the value of CE decreases faster than that of MSE, which means that CE can reduce the number of iterations. Therefore, CE is used as the loss function in this experiment.

The optimal structure can be determined by two steps (Yu and Deng, 2015): (i) determining the optimal number of hidden layer nodes (H) in a single-layer neural network and (ii) adding more hidden layers (L) with the same number of nodes. Fig. 11(a) shows the influence of H on the accuracy of DNN with $L = 1$. The results indicate that the accuracy of $H = 24$ is better than others. Considering the trade-off between accuracy and computational complexity, H is set to 24 in the following experiments. Fig. 11(b) shows the influence of L on the accuracy of DNN. Results indicate that the accuracy (96.74%) of $L = 4$ is better than others. Moreover, the results show that the accuracy of the DNN is quite stable as the values of H and L increase. Thus, the shared DNN is set to four hidden layers, with 24 nodes per layer.

4.4. The behavior characteristics of the models

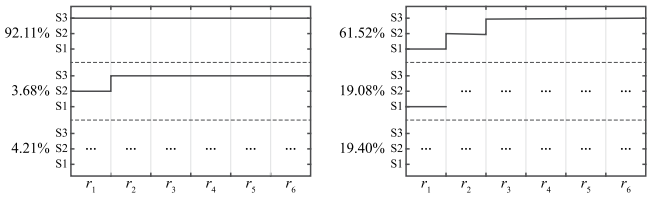
Fig. 12 shows the state transition diagrams of the trained models. The circles denote the hidden states, each of which represents a specific behavior pattern. $\pi_i, i \in Q$ denotes the initial state probability of state i. The arrows and their values indicate the state transition relationship and the state transition probabilities, respectively. According to the probabilities, we can obtain the following inferences:

- The SR-sequences usually start from state 3 and continue to maintain self-transfer;
- The CR-sequences usually start from state 1 and then transfer to state 2 and 3 in turn and finally stay in state 3.



(a) HMM^s (b) HMM^c

Fig. 12. The state transition diagrams of the models.



(a) SR-sequences (b) CR-sequences

Fig. 13. The state transition sequences.

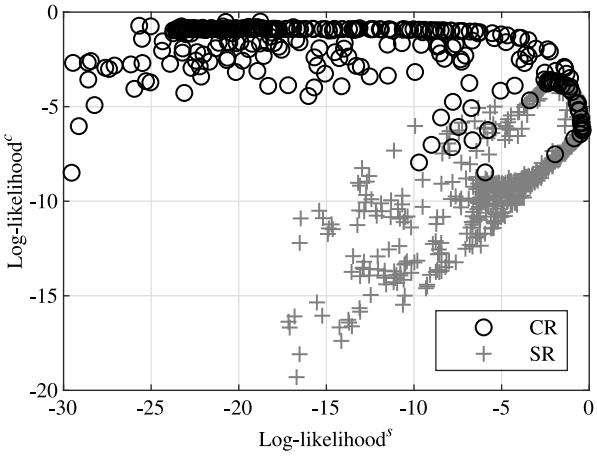


Fig. 14. The distribution of logarithmic likelihoods.

Furthermore, Figs. 13(a) and 13(b) show the typical state transition sequences of the SR- and CR-sequences, respectively. We sort them according to their occurrence probabilities in the training data. The typical time-varying state sequences $(S3, S3, S3, \ldots, S3)$ and $(S1, S2, S3, \ldots, S3)$ account for 92.11% and 61.52% of all SR- and CR-sequences, whereas the remaining 19.08% of CR-sequences begin in state 1. These results indicate that the proposed model can effectively capture the time-varying behavior characteristics of both the CR- and SR-sequences.

4.5. Comparison of the decision means

Fig. 14 shows the distribution of all RF-fragments in the two-dimensional space composed of the logarithmic likelihoods of both HMM^s and HMM^c. "+ " and "○" represent the true labels of RF-fragments as the SR and CR, respectively. Log-likelihood^s and Log-likelihood^c denote the logarithmic likelihoods calculated using HMM^s and HMM^c, respectively. The essence of click attribute identification is to separate CRs from the observed HTTP(S) traffic. As this step is independent of the HMM-DNN model, we compare different decision

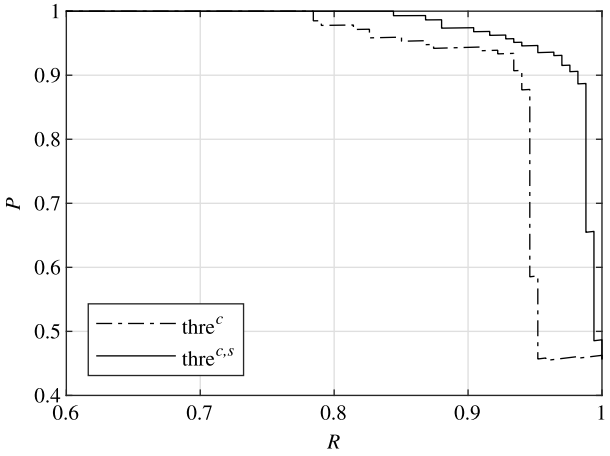


Fig. 15. The PRCs of the threshold-based methods.

Table 2
The performance of $thre^{c,s}$ and the DNN rear classifier.

Methods	P	R	$F1$	$Kappa$
$thre^{c,s}$	1.000	0.759	0.862	0.773
BP	0.938	0.975	0.956	0.918

means for the optimal solution. There are three options, including (i) the single model threshold ($thre^c$), (ii) the dual model threshold ($thre^{c,s}$), and (iii) the DNN rear classifier with the dual model. The decision formulas of (i) and (ii) are given by (35) and (36):

$$u = \begin{cases} c, & P_c \geq thre_1 \\ s, & P_c < thre_1 \end{cases}, \tag{35}$$

$$u = \begin{cases} c, & P_c - P_s \geq thre_2 \\ s, & P_c - P_s < thre_2 \end{cases}, \tag{36}$$

where P_c and P_s denote the likelihood outputted by the HMM^c and the HMM^s, respectively. $thre_1$ and $thre_2$ are the thresholds calculated by the historical data. Fig. 15 shows the precision–recall curves (PRCs) for both threshold-based methods. This result indicates that the recognition performance of the dual model is better than that of the single model.

To evaluate the performance of the threshold-based methods and the DNN rear classifier, we adopt Zou's algorithm (Zou et al., 2016) to determine the best threshold for $thre^{c,s}$ and compare it with the DNN rear classifier using the same test data. The average results are shown in Table 2. Although the P of $thre^{c,s}$ is slightly higher than that of the DNN rear classifier, the other indicators are not as good as it, including the R, the $F1$, and the $Kappa$. Thus, thresholds calculated based on limited historical data are not suitable for complex application scenarios.

4.6. Performance comparison

To validate the proposed HMM-DNN, we compare its recognition performance with other peer methods. The selected methods are divided into two categories: (i) the first is based on the features of single-request, including support vector machine with sigmoid (S-SVM) and radial basis function (R-SVM) as the kernel function (Kong et al., 2017), Gaussian Naive Bayes (GNB) (Fadlil et al., 2017), k-means (Kumari et al., 2016), GMM (Alizadeh et al., 2015), J48 decision trees (DT) (Sahu and Mehtre, 2015), multilayer perceptron (MLP) (Ramchoun et al., 2016), and "$P + N$" (Rizothanasis et al., 2016); (ii) the second is based on the sequence analysis, including recurrent neural networks (RNN) (Yin et al., 2017), long short-term memory networks (LSTM) (Vu et al., 2018), and HMM-GMM. For each method, the same data and features are used for model training and determining the optimal model configuration.

Table 3
Statistical results of P+N, HMM-GMM, and HMM-DNN (m, μ, and σ denote the median, mean, and standard deviation, respectively).

Methods	$F1$			$Kappa$		
	m	μ	σ	m	μ	σ
8	0.893	0.891	0.016	0.798	0.796	0.029
11	0.882	0.881	0.032	0.777	0.779	0.058
12	0.943	0.933	0.025	0.892	0.881	0.042

In Fig. 16, we show the results of the performance comparison from four aspects, including the precision in Fig. 16(a), the recall in Fig. 16(b), the $F1$ in Fig. 16(c), and the $Kappa$ in Fig. 16(d). These results are based on the data that come from 200 independent random experiments. An intuitive conclusion is that sequence-based methods are generally better than those based on single-request. Moreover, compared to other solutions, the proposed HMM-DNN shows two advantages: (i) the average of each indicator is quite stable and better than most other methods; (ii) both the box and the tail of the composite indicator, $F1$ and $Kappa$, are more compact than other methods. We notice that the performance of both RNN and LSTM in this work is not so good, although they perform well in other fields. The reason mainly comes from two aspects. First, in RNN the number of derivative terms increases with the number of hidden layers, which causes the gradient to decay quickly and terminates the update of the parameters early. A similar issue has been identified in Le's work (Le et al., 2015). Therefore, it is difficult for RNN to effectively capture the underlying laws of the data. Second, CR may not be suitable for LSTM, because in each sequence the number of non-click requests is much larger than that of the CRs, which results in the signal characteristics of the CRs being easily overwhelmed by non-click requests and not being detected by the LSTM. This limitation has been mentioned in Tax's work when they apply the LSTM to log analysis (Tax et al., 2017). Moreover, both the RNN and LSTM are based on a neural network, which makes them inevitably have the inherent limitations of neural networks, such as the over-fitting issue, inexplicable model, and parameter instability. All of these issues affect their final performance.

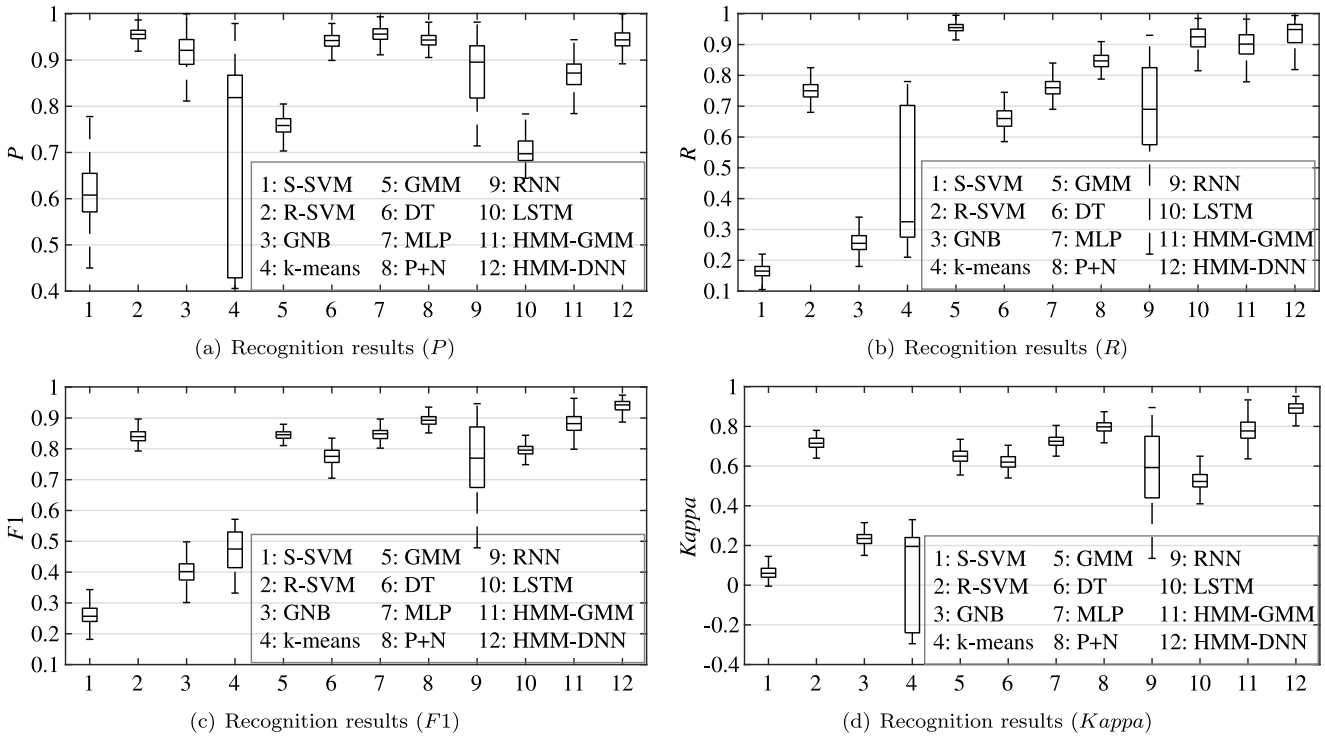
Furthermore, the detailed statistical results of the three best methods ("$P + N$", HMM-GMM, and HMM-DNN) are listed in Table 3. Although "$P + N$" provides the best performance in the methods of single-request, the predefined thresholds affect its versatility. The statistical results of HMM-GMM and HMM-DNN show that the performance of DNN is better than that of GMM. For example, HMM-DNN increases the mean of $F1$ by 5.2 percentage points, whereas each of its indicators has a smaller standard deviation. The reason is that GMMs are statistically inefficient for modeling the data that lie on or near a nonlinear manifold in the data space (Hinton et al., 2012); however, the relationship between the hidden states and the observations is usually nonlinear.

5. Discussion

5.1. Number of hidden states

The number of hidden states of HMM has always been the main factor affecting its application, because currently there is no suitable mathematical method to solve it. In this work, we adopt a simple method to determine the optimal number of hidden states from the perspective of clustering. For the proposed model, each hidden state represents a specific behavior pattern for launching the HTTP(S) requests, and can also be treated as a class of observations. Therefore, the number of hidden states is equivalent to the number of clusters of the observations. Based on this idea, we use Eq. (37) to evaluate the clustering performance and employ it to determine the number of hidden states.

$$J = \sum_{k=1}^{K} \sum_{i \in C_k} (g_k^i - m_k)^2 / |C_k|, \tag{37}$$



(a) Recognition results (P) (b) Recognition results (R)

(c) Recognition results ($F1$) (d) Recognition results ($Kappa$)

Fig. 16. Comparison with other methods.

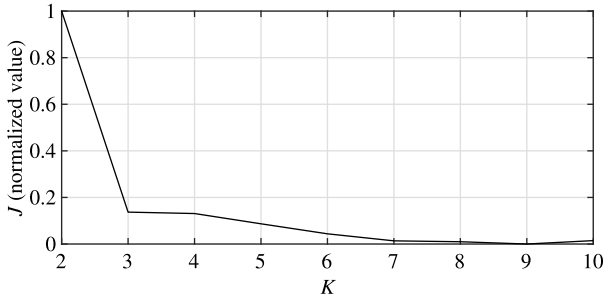


Fig. 17. Mean square error vs. number of clusters.

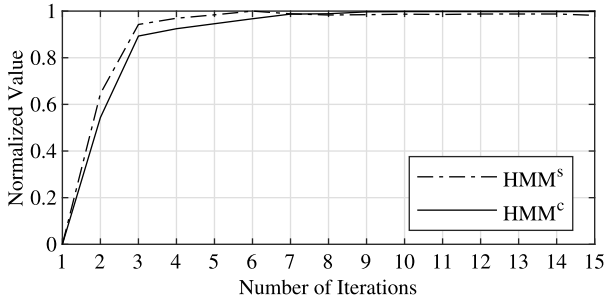


Fig. 18. The variation in the likelihood of the model.

where K denotes the number of different clusters. $|C_k|$ is the size of the kth cluster. g_k^i denotes the distance between the ith observation vector in the kth cluster and the centroid of the kth cluster. m_k is the average distance of the kth cluster. J represents the mean square error of the total distance of the clusters. Then, we use k-means to test the performance of different number of states and plot the results in Fig. 17. It indicates that J tends to be stable when the number of clusters is greater than 3. Considering the performance and computational complexity, we choose 3 to be the number of hidden states.

5.2. Convergence and complexity analysis

The convergence of the model directly affects its execution efficiency. For the HMM framework, the average logarithmic probability $\mathcal{L}$ calculated by Eq. (16) is a good indicator for evaluating the convergence. Fig. 18 shows the curves of normalized $\mathcal{L}$ vs. the number of iterations. The results indicate that both the click and non-click models (HMM^c and HMM^s) tend to converge after seven iterations.

The computational complexity is related to the number of parameters that need to be estimated. Let Q be the number of an HMM's hidden states and $p1$ and $p2$ be the length of the context. D is the dimension of the observation vector. W is the size of RF-fragments. Let H_d and L_d be the number of nodes per layer and the hidden layers for a neural network $d \in \{\text{shared DNN, rear classifier}\}$, respectively. The training phase must train two HMMs, each with $Q^2 + Q$ parameters. The shared DNN has $[(p1 + p2 + 1)D + Q + L_1]H_1 + (L_1 - 1)H_1^2 + Q$ parameters; meanwhile, the classifier that evaluates the likelihoods must train $(L_2+4)H_2+(L_2-1)H_2^2+2$ parameters. In the identification phase, the computational complexity of calculating the likelihood of an RF-fragment through the HMM-DNN is $O(WQ^2 + H_1^2)$, which includes the computation of the forward algorithm of the HMMs and the DNN for generating the posterior probabilities. Moreover, the computational complexity of identifying the CR using the rear classifier is $O(H_2^2)$. The total computational complexity of identifying whether an RF-fragment starts with a CR then becomes $O(WQ^2+H_1^2+H_2^2)$. In addition, the storage space of the HMM-DNN parameters must accommodate four parts: $2Q$ for the initial state probability, $2Q^2$ for the transition probability, $(L_1-1)H_1^2+[(p1+p2+1)D+Q+L_1]H_1+Q$ for the weight and offset of the shared DNN, and $(L_2-1)H_2^2+(L_2+4)H_2+2$ for the weight and offset of the rear classifier.

6. Conclusion and future work

In this work, a new scheme was proposed to identify CRs based on the traffic behavior. We adopted two HMMs to describe the time-varying behavior of click and non-click web traffic. DNN was integrated into the HMMs to capture the context of the observed traffic features and eliminate the limitations caused by the independence hypothesis of the traditional HMMs. In contrast to the existing works, we designed new GMM-independent algorithms for this integrated model. Finally, we introduced a DNN-based rear classifier to determine the click attribute of an HTTP(S) request according to the fitting degree between the requests and the behavior models. We evaluated the performance of the proposed scheme with some existing benchmark methods. The experiment results showed that the proposed behavior-based scheme can effectively identify the click property of HTTP(S) requests. Since the proposed scheme does not rely on the payload information of the packets/requests, it is encryption independent and suitable for both the network-side and server-side.

Some interesting issues arising from this work will be further explored in our future research. For example, in this work we only adopt the features related to the time of HTTP(S) requests. However, in the experiments we have found that the information of the TCP connections is also very useful. Combining the information of the network layer and the TCP layer might improve the effectiveness of our method. Moreover, the type of website affects the access patterns of users, which leads to a significant difference in the traffic generated by clicks and makes it difficult to use a unified method for all types of recognition. A fine-grained identification scheme is a challenge that warrants subsequent investigation.

CRediT authorship contribution statement

Xingrui Fei: Software, Validation, Formal analysis, Investigation, Data curation. **Yi Xie:** Conceptualization, Methodology, Resources, Writing - original draft, Supervision. **Shensheng Tang:** Writing - review & editing. **Jiankun Hu:** Methodology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank all anonymous reviewers and editors for their valuable comments to improve this work. This work was supported by the Natural Science Foundation of China (No. 61972431), the Natural Science Foundation of Guangdong Province, China (No. 2018A030313303, 2017B030306015, and 2017B010125003), and foundation of the Science and Technology Development Center of Ministry of Education of China (No. 2018A06002).

References

Adi, E., Baig, Z., Hingston, P., 2017. Stealthy Denial of Service (DoS) attack modelling and detection for HTTP/2 services. J. Netw. Comput. Appl. 91, 1–13.

Alizadeh, H., Khoshrou, A., Zquete, A., 2015. Traffic classification and verification using unsupervised learning of Gaussian mixture models. In: 2015 IEEE International Workshop on Measurements Networking, M N, pp. 1–6.

Anand, S., Aggarwal, R.R., 2012. An efficient algorithm for data cleaning of log file using file extensions. Int. J. Comput. Appl. 48 (8), 13–18.

Atterer, R., Wnuk, M., Schmidt, A., 2006. Knowing the user's every move: User activity tracking for website usability evaluation and implicit interaction. In: Proceedings of the 15th International Conference on World Wide Web. WWW '06, ACM, New York, NY, USA, pp. 203–212.

Ben Houidi, Z., Scavo, G., Ghamri-Doudane, S., Finamore, A., Traverso, S., Mellia, M., 2014. Gold mining in a river of internet content traffic. In: Dainotti, A., Mahanti, A., Uhlig, S. (Eds.), Traffic Monitoring and Analysis. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 91–103.

Benevenuto, F., Rodrigues, T., Cha, M., Almeida, V., 2009. Characterizing user behavior in online social networks. In: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement. IMC '09, ACM, New York, NY, USA, pp. 49–62.

Cao, Y., Huang, Z., Yu, Y., Ke, C., Wang, Z., 2020. A topology and risk-aware access control framework for cyber-physical space. Front. Comput. Sci. 14 (4), 144805.

Cao, Q., Qiao, Y., Lyu, Z., 2017. Machine learning to detect anomalies in web log analysis. In: 2017 3rd IEEE International Conference on Computer and Communications, ICCC, pp. 519–523.

Chitraa, V., Antony, S., Thanamani, 2013. A novel technique for sessions identification in web usage mining preprocessing. Int. J. Comput. Appl. 34 (9), 24–28.

Cooley, R., Mobasher, B., Srivastava, J., 1999. Data preparation for mining world wide web browsing patterns. Knowl. Inf. Syst. 1 (1), 5–32.

Fadlil, A., Riadi, I., Aji, S., 2017. Ddos attacks classification using numeric attribute-based Gaussian naive Bayes. Int. J. Adv. Comput. Sci. Appl. 8, 42–50.

Fleiss, J.L., Levin, B., Paik, M.C., 2003. Statistical Methods for Rates and Proportions, third ed. John Wiley & Sons, Hoboken, New Jersey.

Forney, G.D., 1973. The viterbi algorithm. Proc. IEEE 61 (3), 268–278.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Kingsbury, B., Sainath, T., 2012. Deep neural networks for acoustic modeling in speech recognition. IEEE Signal Processing Magazine IEEE Signal Process. Mag. 29, 82–97.

Huang, J., White, R.W., 2010. Parallel browsing behavior on the web. In: Proceedings of the 21st ACM Conference on Hypertext and Hypermedia. HT '10, ACM, New York, NY, USA, pp. 13–18.

Huiying, Z., Wei, L., 2004. An intelligent algorithm of data pre-processing in Web usage mining. In: Fifth World Congress on Intelligent Control and Automation, IEEE Cat. No.04EX788, Vol. 4, pp. 3119–3123.

Jain, A.K., Mao, J., Mohiuddin, K.M., 1996. Artificial neural networks: A tutorial. Computer 29 (3), 31–44.

Jiang, M., Fang, Y., Xie, H., Chong, J., Meng, M., 2019. User click prediction for personalized job recommendation. World Wide Web 22, 325–345.

Kong, L., Huang, G., Wu, K., 2017. Identification of abnormal network traffic using support vector machine. In: 2017 18th International Conference on Parallel and Distributed Computing, Applications and Technologies, PDCAT, pp. 288–292.

Kreyszig, E., 2011. Advanced Engineering Mathematics, tenth ed. John Wiley & Sons, Hoboken, New Jersey, p. 1014.

Kumari, R., Sheetanshu, Singh, M.K., Jha, R., Singh, N.K., 2016. Anomaly detection in network traffic using K-mean clustering. In: 2016 3rd International Conference on Recent Advances in Information Technology, RAIT, pp. 387–393.

Le, Q.V., et al., 2015. A tutorial on deep learning part 2: Autoencoders, convolutional neural networks and recurrent neural networks. Google Brain 1–20.

LeCun, Y.A., Bottou, L., Orr, G.B., Müller, K.-R., 2012. Efficient BackProp. In: Montavon, G., Orr, G.B., Müller, K.-R. (Eds.), Neural Networks: Tricks of the Trade, second ed. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 9–48.

Liu, Y., Song, T., Liao, L., 2020. TPII: Tracking personally identifiable information via user behaviors in HTTP traffic. Front. Comput. Sci. 14 (3), 143801.

Peng, J., Choo, K.-K.R., Ashman, H., 2016. User profiling in intrusion detection: A review. J. Netw. Comput. Appl. 72, 14–27.

Peng, L., Zhang, H., Yang, B., Chen, Y., Zhou, X., 2016. Early stage internet traffic identification using data gravitation based classification. In: 2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, pp. 504–511.

Rabiner, L.R., 1989. A tutorial on hidden Markov models and selected applications in speech recognition. Proc. IEEE 77 (2), 257–286.

Rafter, R., Smyth, B., 2001. Passive profiling from server logs in an online recruitment environment. In: IJCAI 2001.

Ramchoun, H., Amine, M., Janati Idrissi, M.A., Ghanou, Y., Ettaouil, M., 2016. Multilayer perceptron: Architecture optimization and training. Int. J. Interact. Multimed. Artif. Intel. 4, 26–30.

Rizothanasis, G., Carlsson, N., Mahanti, A., 2016. Identifying user actions from HTTP(S) traffic. In: 2016 IEEE 41st Conference on Local Computer Networks, LCN, pp. 555–558.

Ruiz-Martnez, A., 2012. A survey on solutions and main free tools for privacy enhancing Web communications. J. Netw. Comput. Appl. 35 (5), 1473–1492, Service Delivery Management in Broadband Networks.

Rumelhart, D., Hinton, G., Williams, R., 1986. Learning representations by back propagating errors. Nature 323, 533–536.

Sahu, S., Mehtre, B.M., 2015. Network intrusion detection system using J48 Decision Tree. In: 2015 International Conference on Advances in Computing, Communications and Informatics, ICACCI, pp. 2023–2026.

Schneider, F., Feldmann, A., Krishnamurthy, B., Willinger, W., 2009. Understanding online social network usage from a network perspective. In: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement. IMC '09, ACM, New York, NY, USA, pp. 35–48.

Silverstein, C., Marais, H., Henzinger, M., Moricz, M., 1999. Analysis of a very large web search engine query log. SIGIR Forum 33 (1), 6–12.

Singh, K., Singh, P., Kumar, K., 2018. User behavior analytics-based classification of application layer HTTP-GET flood attacks. J. Netw. Comput. Appl. 112, 97–114.
Spink, A., Koshman, S., Park, M., Field, C., Jansen, B.J., 2005. Multitasking Web search on Vivisimo.com. In: International Conference on Information Technology: Coding and Computing, ITCC'05 - Vol. II, Vol. 2, pp. 486–490.
Tan, X., Xie, Y., Ma, H., Yu, S., Hu, J., 2019. Recognizing the content types of network traffic based on a hybrid DNN-HMM model. J. Netw. Comput. Appl. 142, 51–62.
Tax, N., Verenich, I., La Rosa, M., Dumas, M., 2017. Predictive business process monitoring with LSTM neural networks. In: Dubois, E., Pohl, K. (Eds.), Advanced Information Systems Engineering. Springer International Publishing, Cham, pp. 477–492.
Vassio, L., Drago, I., Mellia, M., 2016. Detecting user actions from HTTP traces: Toward an automatic approach. In: 2016 International Wireless Communications and Mobile Computing Conference, IWCMC, pp. 50–55.
Vu, L., Thuy, H., Nguyen, U., Ngoc, T., Nguyen, D., Dinh Thai, H., Dutkiewicz, E., 2018. Time Series Analysis for Encrypted Traffic Classification: A Deep Learning Approach. pp. 121–126.
Wang, G., Zhang, X., Tang, S., Wilson, C., Zheng, H., Zhao, B.Y., 2017. Clickstream user behavior models. ACM Trans. Web 11 (4), 21:1–21:37.
Xie, G., Iliofotou, M., Karagiannis, T., Faloutsos, M., Jin, Y., 2013. ReSurf: Reconstructing web-surfing activity from network traffic. In: IFIP Networking Conference.
Xie, Y., Yu, S.-Z., 2008. A large-scale hidden semi-Markov model for anomaly detection on user browsing behaviors. IEEE/ACM Trans. Netw. 17 (1), 54–65.
Xu, C., Du, C., Zhao, G., Yu, S., 2013. A novel model for user clicks identification based on hidden semi-Markov. J. Netw. Comput. Appl. 36 (2), 791–798.
Yan, Z., Huo, Q., Xu, J., 2013. A scalable approach to using DNN-derived features in GMM-HMM based acoustic modeling for LVCSR. In: 14th Annual Conference of the International Speech Communication Association, InterSpeech 2013, 14th Annual Conference of the International Speech Communication Association, InterSpeech 2013.
Yin, C., Zhu, Y., Fei, J., He, X., 2017. A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access 5, 21954–21961.
Yu, D., Deng, L., 2015. Automatic Speech Recognition. Springer, London.
Zaim, H., Haddi, A., Ramdani, M., 2019. A novel approach to dynamic profiling of e-customers considering click stream data and online reviews. Int. J. Electr. Comput. Eng. 9 (1), 602–612.
Zou, Q., Xie, S., Lin, Z., Wu, M., Ju, Y., 2016. Finding the best classification threshold in imbalanced classification. Big Data Res. 5, 2–8, cited By 30.

Xingrui Fei received the B.S. degree in Soochow University. He is currently a post-graduate student at Sun Yat-sen University, Guangzhou, China. His research interests focus on cyber security.

Yi Xie received the B.Sc., M.Sc. and Ph.D. degrees from Sun Yat-Sen University, Guangzhou, China. He was a visiting scholar at George Mason University and Deakin University during 2007 to 2008, and 2014 to 2015, respectively. He is currently an Associate Professor at the School of Information Science and Technology, Sun Yat-Sen University. His recent research interests include networking, network security, behavior modeling and algorithms.

Shensheng Tang is currently with the Department of Electrical and Computer Engineering at St Cloud State University, USA. He received his Ph.D. from The University of Toledo, USA. He has eight years of product design and development experience in wireless industry as hardware engineer, system engineer, and manager respectively. His current research interests include embedded systems, networking (wireless, wired), Internet of things (IoT), and modeling and performance evaluation. He has served or is serving as an editor or Guest Editor for International Journals and a TPC member of international conferences. He is a senior member of IEEE.

Jiankun Hu is a full professor of Cyber Security at the School of Engineering and Information Technology, the University of New South Wales at the Australian Defence Force Academy (UNSW@ADFA), Australia. Prof. Hu received his Bachelor's degree in Industrial Automation in 1983 from Hunan University, PR China; a Ph.D. degree in engineering in 1993 from the Harbin Institute of Technology, PR China; and a Master's degree by research in the School of Computer Science and Software Engineering from Monash University, Australia, in 2000. Hu's major research interest is in computer security, especially biometric security.