Search Engine Click Spam Detection Based on Bipartite Graph Propagation

ABSTRACT

Using search engines to retrieve information has become an important part of people's daily lives. For most search engines, click information is an important factor in document ranking. As a result, some websites cheat to obtain a higher rank by fraudulently increasing clicks to their pages, which is referred to as "Click Spam". Based on an analysis of the features of fraudulent clicks, a novel automatic click spam detection approach is proposed in this paper, which consists of 1. modeling user sessions with a triple sequence, which, for the first time to the best of our knowledge, takes into account not only the user action but also the action objective and the time interval between actions; 2. using the user-session bipartite graph propagation algorithm to take advantage of cheating users to find more cheating sessions; and 3. using the pattern-session bipartite graph propagation algorithm to obtain cheating session patterns to achieve higher precision and recall of click spam detection. Experimental results based on a Chinese commercial search engine using real-world log data containing approximately 80 million user clicks per day show that 2.6% of all clicks were detected as spam with a precision of up to 97%.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms

Experimentation, Algorithms, Performance

Keywords

Click Spam, Label Propagation, User Session Model, Frequent Sequential Patterns

1. INTRODUCTION

Search engines are very important in current network environment, which have become one of the major channels for people to access information on the Internet in recent years. After a query is submitted to a search engine, many relevant Web pages will be found. The search engine will rank the results based on a series of relevant ranking features, including textual information [18], link structure [20] and so on. According to the tasks of Yahoo! Labs Learning to Rank Challenge[1], hundreds of ranking features are used in current commercial search engines. Among the features, click information and relevant features have drawn much attention recently. Joachims [13] believes that click information can be used as a relevance feedback feature to rank the search results. Click model (such as cascade model [5], CCM [7], DBM [4] and so on) related works show that user click information plays an important role in ranking the search results.

According to the researches on search user behaviors [17], most users only pay attention to the top search results. So it is obvious that if a website can obtain a higher rank among all the related results, users will be more likely to click the site, and thus, the site traffic will increase. As a result, to raise the rankings of the results by forging user clicks has become the main channel for the illegal web page authors to improve user traffic and gain more economic benefits [6, 15, 24, 3]. Studies on cheating clicks show that various kinds of bots are used to issue queries and clicks automatically to search engines, which not only consume a great amount of capacity for the search engines but also affect the rankings of search results.

An increasing number of researchers focus on the task of anti-spam. Gyöngyi et al. [8] propose the algorithm of TrustRank, which is able to semi-automatically separate reputable, good pages from spam. They start from selecting a small set of reputable seed pages, and discover other pages that are likely to be good with the link structure of the web. Their results show that based on a good seed set of less than 200 sites, spam from a significant fraction of the web can be effectively filtered out. Krishnan et al. [16] introduce the Anti-Trust Rank algorithm. Similar to the Trust Rank algorithm, they select a manually labeled seed set of pages and their experiments on the WebGraph dataset show that their approach is effective at detecting spam pages from a small seed set.

At present, user behavior information has been introduced into search engines [1, 17], and thus, search engines will adjust the rankings of search results according to user reactions. As a result, new cheating methods, called "click spam", that aim to exploit user behavior analysis have arisen. A

[1]http://learningtorankchallenge.yahoo.com/index.php

typical example of click spam is “Google bombing”[2], which refers to raising the rank of a specific result in a search engine maliciously. Another two examples of click spam are the user sessions shown in Table 1, which are extracted from the real-world click logs of a popular Chinese commercial search engine. In the session shown in (a), after submitting a query, the user clicks on the same URL repeatedly. In the session shown in (b), the user submits different queries and clicks on the web pages with the same domain repeatedly. Both sessions may raise the rankings of the search results that the users click, which makes their websites easier to be clicked by other search engine users fraudulently. Obviously, click spam not only undermines the fairness among search results but also degrades user experience. For this reason, search engines are in urgent need of click spam detection techniques.

Table 1: Two examples of click spam

user IP	query	isclick	clicked URL
211.143.91.98	9999pp.com	0	
211.143.91.98	9999pp.com	1	http://369ii.com/
211.143.91.98	9999pp.com	1	http://369ii.com/
211.143.91.98	9999pp.com	1	http://369ii.com/
211.143.91.98	9999pp.com	1	http://369ii.com/
211.143.91.98	9999pp.com	1	http://369ii.com/

(a)

user IP	query	isclick	clicked URL
219.135.43.71	China	0	
219.135.43.71	China	1	http://www.zzyzzy .cn/html/322.html/
219.135.43.71	Shanghai	0	
219.135.43.71	Shanghai	1	http://www.zzyzzy .cn/html/151.html/
219.135.43.71	software	0	
219.135.43.71	software	1	http://www.zzyzzy .cn/html/188.html/
219.135.43.71	summit	0	
219.135.43.71	summit	1	http://www.zzyzzy .cn/html/56.html/
219.135.43.71	industry	0	
219.135.43.71	industry	1	http://www.zzyzzy .cn/html/220.html/

(b)

Click spam is a new method of cheating search engine rankings and involves user behavior, so it is difficult to distinguish normal and abnormal user reactions. In this paper, we propose a novel, automatic, click spam detection approach, which is, to the best of our knowledge, the first work to introduce the objective of user actions and time interval between actions into user session modeling for click spam detection. The proposed approach shows both effectiveness and efficiency at detecting click spam in a real commercial search engine. The approach uses the following procedure: 1. Model user session with a triple sequence; 2. Construct user-session bipartite graph and pattern-session bipartite graph to describe the relation between users and sessions as well as the relation between patterns and sessions; 3. Based on the seed cheating session modes, detect a large amount of click spam with a label propagation algorithm on the bipartite graph.

The rest of this paper is organized as follows: after a discussion of the related work in next section, we introduce the detection of the cheating on single click record in Section 3. The modeling of user sessions is described in Section 4. Section 5 introduces the recognition of session-level click spam. Section 6 presents the evaluation and discussion of our approach, and finally, Section 7 concludes the paper.

2. RELATED WORK

Over the last decade, there has been a growing interest in detecting click fraud. Jansen [11] introduces intentional clicks on sponsored links with the purpose of gaining undue monetary returns or harming a particular content provider, which is called “Click Fraud”. He also presents several countermeasures, such as strengthening the monitoring of click fraud, improving automated filters to identify and prevent click fraud, and changing the charging mode from pay-per-click to pay-per-action. Metwally et al. [19] present three forms of click fraud and methods to detect it. They find that several websites can cooperate with each other to create fraudulent clicks and thus advance their commercial interests. They develop an algorithm, called streaming-rules, to report association rules with tight guarantees on errors, using limited processing per element and minimal space, in order to detect fraud in advertising networks. The above previous work is related to the detection of click fraud. However, there are many differences between click spam for general search results and click fraud for sponsored advertisement links. Click fraud aims to increase the number of clicks on the advertisement links so as to gain more money from the websites that display the ads, while the goal of click spam is to make more clicks on the website to obtain a higher rank in the search results. And the detection of click fraud only needs to examine the IPs of the users that click on the ads, while the detection of click spam requires detailed analysis of click logs of the search engine. Consequently, the methods mentioned above cannot be directly applied to our work.

Radlinski [22] analyzes click spam from a utility standpoint. He finds that click spam plays an important role in the rankings of search results, which is harmful to search engines. He addresses malicious noise through partitioning the user population and suggests that personalizing search results can improve the robustness to click spam when using a simple ranking algorithm. Kang et al. [14] propose a semi-supervised learning algorithm for distinguishing program (bot) generated web search traffic from that of genuine human users. They first use the CAPTCHA technique to extract a large set of samples from the data logs as training data, and then develop a semi-supervised learning algorithm to take advantage of the unlabeled data to improve the classification performance. From their evaluation, the semi-supervised learning approach that they propose performs significantly better than the supervised learning algorithms.

All of these studies show that analysis of the features of single click records contribute to the detection of click spam. Other researchers focus on session-level click spam recognition. Sadagopan and Li [23] argue that it is important to identify typical and atypical user sessions in clickstreams. They model user session with a sequence of user actions. They classify user actions into 5 categories: page request, web click, sponsored click, next click and other click. They

[2]http://en.wikipedia.org/wiki/Google_bombing

then use a Markov chain model to represent user sessions and compute the transition probability as its likelihood score. A lower session score indicates a greater likelihood that the session is atypical. Their results show that their approach can identify typical and atypical sessions with a precision of about 89% and that filtering out the atypical sessions reduces the uncertainty of the mean CTR by about 40%.

To summarize, in the field of click spam detection, most research focuses on single click spam record detection, while few studies focus on session-level click spam recognition. However, click spam is user-level behavior. For example, if a user session is detected as a fraudulent session, all the actions of the user during the session should be regarded as fraudulent as well. Therefore, it is more important to detect click spam on a session level. Our approach focuses on session-level click spam recognition and is similar to the work of Sadagopan and Li; however, we also take into consideration the information about the action objective and the time interval between actions when modeling user sessions, and we adopt a bipartite graph propagation algorithm to further detect cheating sessions.

3. SINGLE CLICK SPAM RECORD DETECTION

The data used in our work consists of real click logs from a popular Chinese commercial search engine. There are about 80 million randomly sampled records in the click log per day with an anonymous ID and time information as well as submitted query, click type and clicked URL data. We selected the click logs from December 2011 for our experiment, which consist of 2.4 billion records in total.

In the click log, tag represents the type of click. For example, tag of "image" indicates that the user clicks on the "Image Search" tab. For normal users, a tag of the click should be consistent with the true action of the user. When confliction is found, click spam can be detected. As an example, if the tag is "video", but the clicked URL comes from a general search result but not video search result, the spam is indicated.

In total there are 40,130 different tags from one day sampled log data (2011.12.7), which can be divided into 53 categories according to the prefix of the tags. For 20 of these tag categories, the clicked URLs share the same characteristic. Based on the analysis above, for a click record, if its tag is missing, or the clicked URL does not match the tag, we believe that this click record is suspicious of cheating. By this method, experiments have been made and the results are shown in Table 2.

As shown in the table, in total 443,415 click spam records out of 82,113,889 click records are detected in this way, which take up 0.54% of all the click records (except for the above listed 11,716,766 records, there are more clicks that cannot be recognized by this method). We also run the experiment on two other days, and the ratio is 0.55% and 0.52% respectively, which keeps stable by our detection methods.

The detection of the single click spam is quite precise (with 100% precision), but only handles the simplest spam techniques. Hence it can be used as part of the seeds to find more complicated click spams in our model (details are shown in Section 5).

Table 2: Result of single click spam record detection

click type	#cheating clicks	#total clicks	spam ratio
–(missing tag)	285,449	285,449	100%
query recommendation	32,348	3,330,407	0.97%
next page	3,105	2,209,056	0.14%
page number	21,909	2,068,008	1.06%
sponsored search	12,092	2,200,346	0.55%
Snapshot	22,514	520,956	4.32%
Video	5,051	342,363	1.48%
Picture	4,844	246,900	1.96%
leftcolumn	9,957	122,421	8.13%
Music	4,841	113,434	4.27%
News	5,028	49,622	10.1%
Maps	9,737	33,917	28.7%
knowledge	7,128	33,440	21.3%
previous page	106	25,604	0.41%
searchForm	2,452	11,075	22.1%
web help	2,521	5,285	47.7%
Other	14,333	118,483	12.1%
Total	443,415	11,716,766	3.78%

4. MODELING USER SESSIONS

The general idea of our work is to model user sessions with action sequences so that we can detect spam by judging whether the user session matches abnormal sequence modes. In previous works, generally a symbolic definition of a single action is given (e.g. [23]) without taking into consideration the time interval between actions or the semantic meanings of the action (for example, the action is to submit the same query as a previous one or to submit a new query). The innovation of our work lies in these two points. Consequently, the key points of modeling the user session are 1. to define actions with semantic meanings and 2. to introduce time as a factor into the model.

4.1 Action definitions

We define a user session as the interaction between the user and the search engine in 30 minutes starting from the user submitting the first query to the search engine. In the click log of a single day, more than 50 million user sessions can be generated by this method. During a user session, after the user submits a query, he may browse the result page, click on the search results, click on the tabs and so on, or he may reform his query and continue his interaction with the search engine. In our work, we define 6 kinds of user actions:

- Qi: Submit a query, in which i is used to distinguish different queries (in a session, each time a new query is submitted, we assign a new ID)
- Wi: Click on web results, in which i is used to distinguish different web clicks
- Oi: Click on sponsored results, in which i is used to distinguish different sponsored clicks
- N: Load a new page, including clicking on the next page, the previous page and specific page numbers

- T: Scroll the page
- Ai: Perform other clicks, including clicking on tabs such as “Video” and “Music”

Thus the user actions in a user session can be divided into 6 categories. Unlike earlier studies, we assign different IDs for different queries and clicks on different links, which is able to distinguish action objectives. That is reasonable because it can be regarded as a normal behavior if a user clicks on 10 different web results after submitting a query. But if he clicks on the same web result 10 times after submitting a query, he is suspicious of cheating. In earlier studies, the 2 sessions are modeled as the same action sequence. While in our work, we can distinguish them according to the IDs of the actions.

4.2 Time interval

After defining the possible actions in a user session, we introduce the time interval information into our model. In a user session, there should be a reasonable time interval between actions. For example, after a user submits a query to the search engine, he takes time to read the search result page and then makes a click on one of the search results. He also takes time to read the landing page and then clicks on another search result. As a result, if the actions in a user session is excessive frequent, it is likely that the session is performed by a bot (a piece of program) and the user is suspicious of cheating. So it is important to take the time interval between actions into consideration when modeling user sessions. However, if we use the value of the time interval directly, the data will be too sparse because each different second of time interval means a different action. Instead, we need to segment the time interval. We analyse the time interval distributions of the 6 actions (the time interval of an action is the time interval between this action and the previous action, if this is the first action of the user session, the time interval is 0), which is shown in Figure 1.

As shown in Figure 1, the time interval distributions of the 6 actions are similar, so we apply the same segmentation strategy to the 6 actions. We segment the timeline into 4 parts with zero point and two inflection points according to the time interval distribution. To be specific, let t be the time interval value (in seconds) and T be the ID of the segment, so when:

- $t = 0, T = 0$
- $0 < t \leq 10, T = 1$
- $10 < t \leq 30, T = 2$
- $t > 30, T = 3$

Based on the analysis above, we are able to model user sessions involving action objective and time interval between actions. For each event in a session, we represent it with a triple (S, i, T), where $S \in \{Q, W, O, N, T, A\}$, which represents an action, in which i is used to distinguish different action objectives, and $T \in \{0, 1, 2, 3\}$, which represents the time interval. Thus each session can be represented by a triple sequence starting with $\{Q0, 0\}$, since each session starts from a user submitting a query to the search engine with no time interval. Below is an example of a modeled user session:

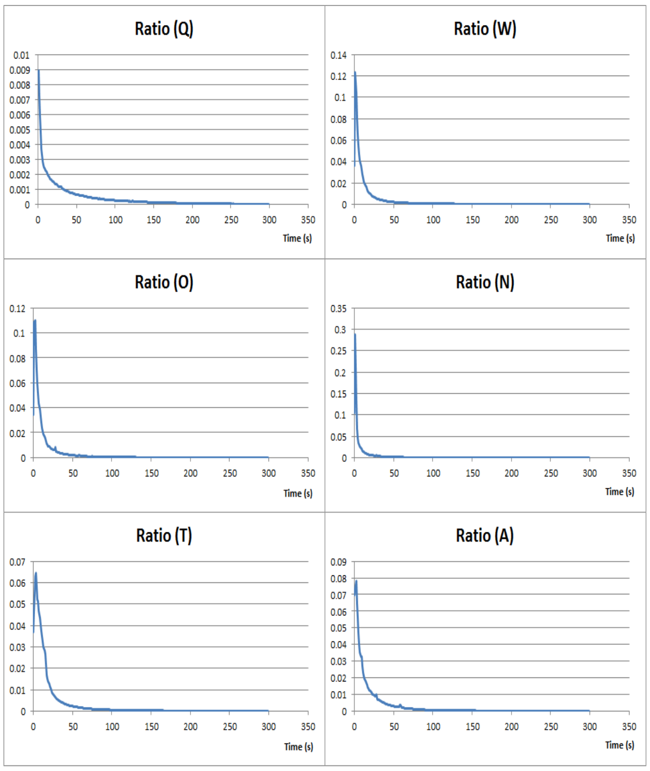

Figure 1: Time interval distribution of the 6 actions

$(Q0,0)$,$(T,1)$,$(Q0,1)$,$(Q0,0)$,$(W2,1)$,$(Q1,3)$,$(Q1,1)$,$(Q1,0)$, $(W2,1)$,$(T,2)$,$(Q1,1)$,$(Q1,0)$,$(W2,1)$,$(Q1,2)$,$(Q1,0)$,$(T,1)$

Thus, the user submits a query $(Q0, 0)$, scrolls the page a short time later $(T, 1)$, re-submits the same query after a short time $(Q0, 1)$, submits the same query again immediately $(Q0, 0)$, and clicks on the third web result after a short time interval $(W2, 1)$....

5. SESSION-LEVEL CLICK SPAM RECOGNITION

5.1 Detection of cheating sessions based on Markov transition probability

We first use the approach proposed in [23] as a baseline experiment. The main idea behind the approach is that within a session, the next event is largely affected by the previous one, which leads to a Markov Chain Model for user sessions. The state space of the Markov Chain Model consists every action that occurs in user sessions. The transition probability $Pr(i, j)$ from state i to j is estimated as follows:

$$Pr(i, j) = \frac{Q_{i,j}}{Q_i}$$

where $Q_{i,j}$ is the number of instances where state i is followed by state j, and $Q_i = \sum_j Q_{i,j}$.

Each user session can then be assigned a likelihood score by multiplying the probability of the individual state transitions in the session. Thus a session with a high likelihood score can be associated with normal behavior, while a session with a low likelihood score can be associated with rare behavior. Since the likelihood score is obtained by multiplying the probability of the individual state transitions, a user session with a longer sequence will get a smaller score. To avoid that case, they take a log of the likelihood score and normalize it by the length of the user session sequence to obtain the average Markovian LoglikeHood (MLH_{avg}). Consequently, a lower MLH_{avg} score indicates a greater likelihood that the session is atypical. Thus cheating sessions can be detected from the tail of the MLH_{avg} score distribution.

We use the approach of Sadagopan and Li to test our modeled user sessions from one day of click log data (2011.12.7), which consists of more than 50 million user sessions. We count the transition probability between states in user sessions and compute the MLH_{avg} score of each session. The distribution of MLH_{avg} scores is shown in Figure 2.

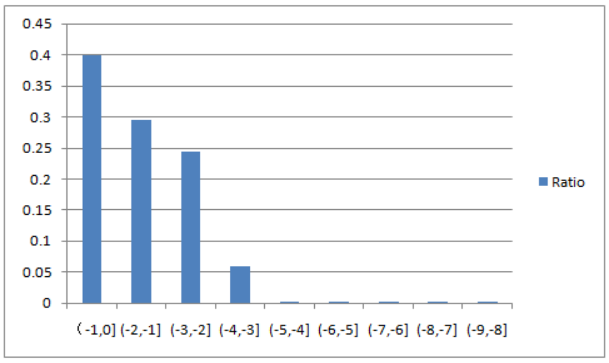

Figure 2: Distribution of MLH_{avg} scores

From the figure we can see that for 99.6% of the user sessions, the $\text{MLH}_{avg} \in (-4, 0]$, which can be regarded as normal sessions. We detect the user sessions with score lower than -4 as cheating sessions. The evaluation of precision of the approach and the comparison with our methods will be given in Section 6.

5.2 Discovering seed cheating session modes

The main idea of our work is to first find a small set of seed cheating sessions, and then detect cheating users and sequential patterns with a label propagation algorithm on the user-session bipartite graph and pattern-session bipartite graph, which will help detect more cheating sessions and improve the precision and recall of our work. To find the seed cheating sessions, we discover the modes used by these sessions with the detected single click spam records above. We analyze the sessions that contain these records because these sessions are more likely to be cheating sessions. We model these sessions with the method described in Section 4 and discover the modes that many of these sessions share. We discover 5 cheating session modes in total and represent them by regular expressions, their characteristics are described as below.

- $(QAi)*$: Submit different queries and click on the websites with the same domain repeatedly (as shown in Table 1 (b)), where i means the domains of the websites clicked by the user are the same.

- $(QiT)*$: Submit the same query and drag the page repeatedly, where i means the queries submitted by the user are the same.

- $(Qi)*$: Submit the same query repeatedly in order to increase the frequency of a certain keyword, where i means the queries submitted by the user are the same.

- $Q(Wi)*$: After submitting a query, click on a web page after a short time interval repeatedly, where i means the web pages clicked by the user are the same.

- $Q(Ai)*$: After submitting a query, click on a website with the same domain after a short time interval repeatedly, where i means the domains of the websites clicked by the user are the same.

We use one day of click log data (2011.12.7) to evaluate the precision of the session modes above. We model the user sessions and obtain the sessions that match the 5 modes like this: if the length of the subsequence of a session that matches the mode exceeds 50% of the length of the session (only the actions of which the time interval is 0 or 1 are considered), we determine that the session matches the cheating mode. Here we do not consider the actions of which the time interval is 2 or 3 because we believe the actions in a cheating session are usually with a shorter time interval so that the user can perform more cheating sessions during the same period of time. For each of the 5 modes, we extract 100 sessions that match the mode randomly for manual annotation and evaluation the precision. The number and precision of the sessions are shown in Table 3.

Table 3: Click ratio and precision of the 5 seed cheating modes

mode	#session	#click	%click	precision
$(QAi)_{0-1}*$	11,981	143,214	0.226%	100.0%
$(QiT)_{0-1}*$	3,964	36,964	0.058%	98.0%
$(Qi)_{0-1}*$	17,258	14,956	0.024%	95.0%
$Q(Wi)_{0-1}*$	2,069	36,097	0.057%	100.0%
$Q(Ai)_{0-1}*$	4,079	82,843	0.130%	100.0%
total	39,351	314,074	0.495%	98.6%

In Table 3, the first column is the cheating mode, the second column is the number of the sessions that match the corresponding mode, the third column shows the number of actions (queries and clicks) in the sessions and the fourth column is the ratio between the actions in the sessions and all the actions in the one day click log, the last column is the precision of the corresponding mode. From the table we can see that the precision of the 5 cheating modes are rather high (98.6%) so that the detected 39,351 sessions can be used as seed cheating sessions.

5.3 User-session bipartite graph propagation algorithm

The user-session bipartite graph propagation algorithm is based on the assumption that if a certain number of a user's sessions are cheating sessions, the other sessions of this user are likely to be cheating sessions as well. According to that assumption, we start from the seed cheating sessions and find cheating users based on the cheating sessions, and then find more cheating sessions based on the cheating users that we have found. We repeat this process several times to detect a large number of cheating sessions to improve the precision and recall of our work. In order to complete that process, we construct the relationship between users and session sequences. In a search engine, a user may initiate sessions with different sequences, and the same session sequence may be made by different users. Their relationship forms the user-session bipartite graph, as shown in Figure 3. In the click log data of one day, there are nearly 40 million users and 1 million session sequences.

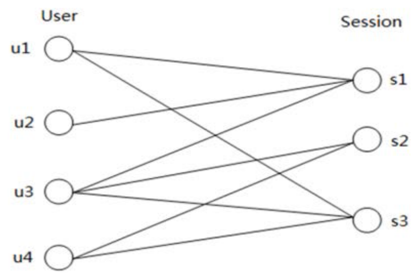

Figure 3: User-session bipartite graph

Based on the assumption and analysis, we propose the user-session bipartite graph propagation algorithm. First, we assign a score for each session. Specifically, for the sessions that match the cheating modes identified above, we assign an initial score of 1, and all other sessions are assigned a score of 0. Then, for each user, we update the score with the weighted average of the scores of the sessions that the user makes. Then, for each session, we update the score with the weighted average of the scores of users who make the session to finish the first iteration. This process is repeated until the scores of the sessions between two iterations change little. A description of the algorithm is shown in Algorithm 1, where w_{ij} is the frequency of user i making session j.

Algorithm 1 User-session bipartite graph propagation algorithm

```
Require:
    The set of cheating session sequences, $C$;
    The set of users, $U$;
    The set of session sequences, $S$;
    The user-session weight matrix, $W$;
    The threshold to end the iteration, $\epsilon$;
 1: for each $s_j$ in $S$ do
 2:     $sscore(s_j) = 0$
 3: end for
 4: for each $s_j$ in $C$ do
 5:     $sscore(s_j) = 1$
 6: end for
 7: $n = 1$
 8: while $|sscore_n - sscore_{n-1}| > \epsilon$ do
 9:     for each $u_i$ in $U$ do
10:         $uscore(u_i) = \sum_i w_{ij} \times sscore(s_j)$
11:     end for
12:     for each $s_j$ in $S$ do
13:         if $s_j$ in $C$ then
14:             $sscore(s_j) = 1$
15:         else
16:             $sscore(s_j) = \sum_j w_{ij} \times uscore(u_i)$
17:         end if
18:     end for
19:     $n = n + 1$
20: end while
```

5.4 Pattern-session bipartite graph propagation algorithm

From the user-session bipartite graph propagation algorithm, we can obtain a list of cheating users and cheating session sequences. However, our goal is to discover all the cheating sequential patterns so that we can detect the cheating sessions that match these patterns. The pattern-session bipartite graph propagation algorithm is based on the assumption that if a certain number of sessions that match a sequential pattern are cheating sessions, other sessions that match the sequential pattern are likely to be cheating sessions as well. The main idea of the pattern-session bipartite graph propagation algorithm is to find frequent sequential patterns from the modeled user sessions and to diffuse the cheating score of the seed sessions on the pattern-session bipartite graph in order to obtain a list of cheating sequential patterns.

First, we introduce the definition of *frequent sequential patterns*, which is given by [2]. An item set $I = \{i_1, i_2, ..., i_k\}$ is the set of all the items. A sequence $\alpha = \langle t_1, t_2, ..., t_m \rangle (t_i \subseteq I)$ is an ordered list of item subsets. A sequence $\langle a_1, a_2, ..., a_n \rangle$ *is contained in* another sequence $\langle b_1, b_2, ..., b_m \rangle$ if there exist integers $i_1 < i_2 < ... < i_n$ such that $a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, ..., a_n \subseteq b_{i_n}$. Given a sequence database $D = \{s_1, s_2, ..., s_n\}$, the *support for a sequence* α is defined as the number of sequences in D that contain α. If the support for a sequence α is larger than $\theta|D|$ (θ is the user-specified threshold), α is called the *frequent sequential pattern.*

A variety of algorithms for finding frequent sequential patterns in very large sequential databases have been developed over the years, such as FreeSpan [9], PrefixSpan [10, 21], CloSpan [25] and so on. We use the PrefixSpan approach proposed in [10, 21] to mine frequent sequential patterns from the modeled user sessions. PrefixSpan works through divide and conquer. The complete set of sequential patterns is partitioned into different subsets according to different prefixes, and corresponding projected databases are constructed and frequent sequential patterns are mined recursively. Based on the mined frequent sequential patterns, we construct the pattern-session bipartite graph. Since each sequential pattern may be matched by different session sequences and each session sequence may match different sequential patterns, their relationship forms the pattern-session bipartite graph. We use the same algorithm with the user-session bipartite graph propagation algorithm to diffuse the cheating score of the seed sessions and obtain the cheating sequential patterns, as shown in Algorithm 2.

The frequent sequential patterns do not require the actions in a session to be continuous, which effectively reduce the influence of the noise. The high robustness of the algorithm increases the recall of click spam detection.

6. EVALUATIONS AND DISCUSSIONS

6.1 Performance of the user-session bipartite graph propagation algorithm

We use the user-session bipartite graph propagation algorithm to experiment on the sampled click logs of one week, which contain about 80 million click records per day. First we evaluate the performance of the algorithm with the click log of a single day (2011.12.7). Each session sequence receives a score after the iteration completes. Obviously the

Algorithm 2 Pattern-session bipartite graph propagation algorithm

```
Require:
    The set of cheating session sequences, $C$;
    The set of frequent sequential patterns, $P$;
    The set of session sequences, $S$;
    The pattern-session weight matrix, $W$;
    The threshold to end the iteration, $\epsilon$;
 1: for each $s_j$ in $S$ do
 2:     $sscore(s_j) = 0$
 3: end for
 4: for each $s_j$ in $C$ do
 5:     $sscore(s_j) = 1$
 6: end for
 7: $n = 1$
 8: while $|sscore_n - sscore_{n-1}| > \epsilon$ do
 9:     for each $p_i$ in $P$ do
10:         $pscore(p_i) = \sum_i w_{ij} \times sscore(s_j)$
11:     end for
12:     for each $s_j$ in $S$ do
13:         if $s_j$ in $C$ then
14:             $sscore(s_j) = 1$
15:         else
16:             $sscore(s_j) = \sum_j w_{ij} \times pscore(p_i)$
17:         end if
18:     end for
19:     $n = n + 1$
20: end while
```

score of each session sequence is between 0 and 1. We calculate the number and *click ratio* of the sessions for different score ranges. Here, the *click ratio* of the sessions is defined as the ratio between the number of actions (queries and clicks) of the sessions in a specific score range and the number of actions of all the sessions. The click ratios for different score ranges are shown in Figure 4.

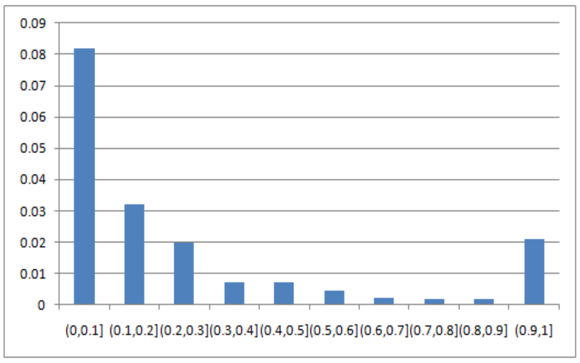

Figure 4: Click ratios for different score ranges on one day with the user-session bipartite graph propagation algorithm

From the algorithm we know that a higher score indicates a larger likelihood that the session is cheating. So for the sessions with a score larger than 0.5, we randomly sample 200 sessions for manual annotation to evaluate the precision, and the results are shown in Table 4.

From the table we can see that the sessions in a higher score range get a higher precision to be cheating sessions,

Table 4: Precision in the different score ranges

range	#session	#correct	precision
(0.9,1]	59	57	0.966102
(0.8,0.9]	15	13	0.866667
(0.7,0.8]	22	19	0.863636
(0.6,0.7]	28	22	0.785714
(0.5,0.6]	76	55	0.723684
total	200	166	0.83

which meets our expectation. Since the detection of click spam requires high precision, we take the sessions within the score range of (0.9,1] as cheating sessions, which make up 2.1% of the number of queries and clicks in all the sessions. We define the click ratio of the sessions within the score range of (0,9,1] as the *click spam ratio*.

We also run the same experiment on one week's sampled data (2011.12.1 - 2011.12.7), containing 550 million click records. The click ratios for different score ranges on separate days are shown in Figure 5.

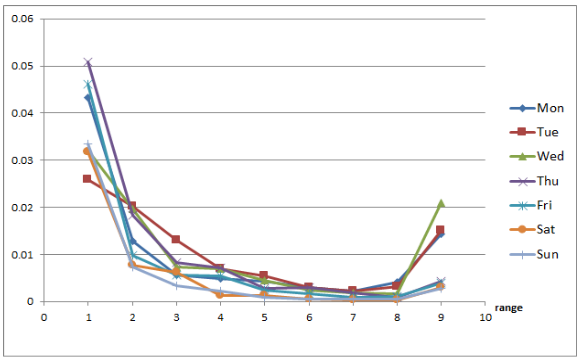

Figure 5: Click ratios for different score ranges on separate days in a week with the user-session bipartite graph propagation algorithm

As shown in Figure 5, the number of cheating sessions detected is larger from Monday to Wednesday than from Thursday through Sunday. Perhaps it is because the cheating users or company are more active at the beginning of the week. So we suggest that the detection of click spam should be more concentrated at the beginning of each week.

6.2 Performance of the pattern-session bipartite graph propagation algorithm

To select the proper support threshold θ for mining frequent sequential patterns, we use a range of 0.005 to 0.01 for θ and test the sampled one-day log data using the pattern-session bipartite graph propagation algorithm and compute the click spam ratio. From the results for each θ, we randomly sample 100 sessions within the score range of (0.9, 1] for manual annotation to evaluate the precision. The click spam ratio and precision for each θ are shown in Table 5.

These results indicate that with the decrease of θ, the click spam ratio increases while the precision decreases. To ensure the precision of the detection of click spam, we set θ to be 0.01. The click ratio in different score ranges is shown in Figure 6.

The figure shows that the click spam ratio reaches 2.6%, which is higher than the 2.1% for the user-session bipar-

Table 5: Click spam ratio and precision for each θ

support threshold θ	click spam ratio	precision
0.01	0.026172068	97%
0.009	0.026976257	92%
0.008	0.027539188	93%
0.007	0.028182539	92%
0.006	0.028665052	91%
0.005	0.028986728	91%

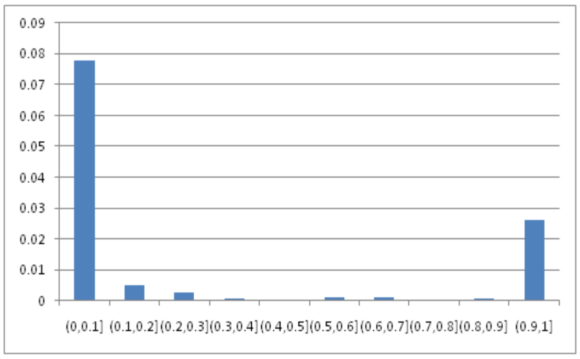

Figure 6: Click ratio in different score ranges with the pattern-session bipartite graph propagation algorithm

tite graph propagation algorithm, thus proving the high robustness of the pattern-session bipartite graph propagation algorithm.

6.3 Comparison of the three approaches

We compare the performance of our algorithm with the traditional click spam detection approach [23]. For the baseline approach, we consider sessions with scores lower than -4 to be cheating sessions and randomly sample 100 sessions to evaluate the precision. The click spam ratio and precision of the three approaches are shown in Table 6.

Table 6: Comparison of the three approaches

	click spam ratio	precision
Baseline[23]	1.7%	90%
User-session bipartite graph propagation algorithm	2.1%	97%
Pattern-session bipartite graph propagation algorithm	2.6%	97%

The table shows that our approaches outperform the baseline for both precision and recall. While maintaining high precision, the pattern-session bipartite graph propagation algorithm is able to detect more cheating sessions than the user-session bipartite graph propagation algorithm.

6.4 Effects of click spam on search results

The Normalized Discounted Cumulative Gain (NDCG, [12]) is an important metric to evaluate the relevance of the search results. The NDCG at position $p(NDCG@p)$ is computed as:

$$DCG@p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{log_2(1+i)}$$

$$NDCG@p = \frac{DCG@p}{IDCG@p}$$

where rel_i is the manual label of the i-th document, and $IDCG@p$ represents an ideal DCG value obtained when sorting the documents by relevance. The formula indicates that the relevance of higher-ranking results is more important than that of lower-ranking results. For example, if a low relevance result is ranked high, the NDCG value will become relatively small and the ranking results will be less satisfactory. We filter out the detected cheating sessions and use NDCG metric to evaluate the changes.

We extract the queries in which the number of clicks is larger than 1000 and the click spam ratio is higher than 10% from the sampled one-day log data. These queries involve large number of click spam records, and we would like to evaluate whether our approach of click spam detection helps the search engine perform better. For each query, we rank the URLs by CTR before and after filtering out the cheating sessions. Here CTR is computed as:

$$CTR = \frac{\#clicks\ on\ the\ URL\ when\ searching\ the\ query}{\#the\ query\ is\ searched}$$

Since the detected cheating sessions involves large number of clicks on some of the results, these results will rank lower after we filter out the cheating sessions. We label the top 20 URLs for each query according to their relevance to the query and compute the NDCG value. We compare the average nDCG@5, nDCG@10 and nDCG@20 of the queries, and the results are shown in Figure 7.

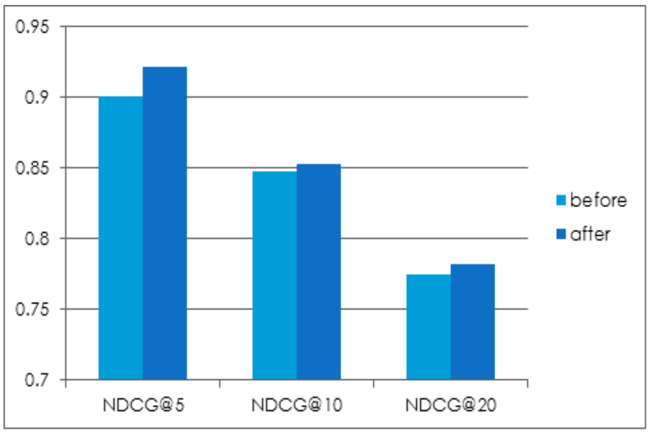

Figure 7: Comparison of nDCG before and after filtering out the cheating sessions

The figure shows that after filtering out the cheating sessions, there is a larger increase in nDCG@5 than in nDCG@10 and nDCG@20. This result indicates that the detected click spammers tended to succeed in placing their websites at a higher position ($\leq$5) in search results.

7. CONCLUSIONS

Previous studies show that click spam has a large influence on the performance of search engines. In this paper, we propose a novel automatic session-level click spam detection approach. We first model user sessions with a triple sequence that accounts for the action objective and the time interval between actions. Then, we construct the user-session bipartite graph and the pattern-session bipartite graph to describe their relations. Finally, based on the detected seed cheating sessions, we diffuse the cheating score on the bipartite graph and improve the precision and recall of our work. The approaches that we propose perform better than the traditional session-level click spam detection method, as our approach can detect 2.6% of all the queries and clicks as spam with a precision of 97%. After filtering out the cheating sessions that we have detected, the nDCG of the search results increases, which indicates that the search engine performs better.

8. REFERENCES

[1] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–26. ACM, 2006.

[2] R. Agrawal and R. Srikant. Mining sequential patterns. In *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*, pages 3–14. IEEE, 1995.

[3] L. Becchetti, C. Castillo, D. Donato, S. Leonardi, and R. A. Baeza-Yates. Link-based characterization and detection of web spam. In *AIRWeb*, pages 1–8, 2006.

[4] O. Chapelle and Y. Zhang. A dynamic bayesian network click model for web search ranking. In *Proceedings of the 18th international conference on World wide web*, pages 1–10. ACM, 2009.

[5] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 87–94. ACM, 2008.

[6] G. Gu, R. Perdisci, J. Zhang, W. Lee, et al. Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection. In *USENIX Security Symposium*, pages 139–154, 2008.

[7] F. Guo, C. Liu, A. Kannan, T. Minka, M. Taylor, Y.-M. Wang, and C. Faloutsos. Click chain model in web search. In *Proceedings of the 18th international conference on World wide web*, pages 11–20. ACM, 2009.

[8] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with trustrank. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 576–587. VLDB Endowment, 2004.

[9] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu. Freespan: frequent pattern-projected sequential pattern mining. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 355–359. ACM, 2000.

[10] J. Han, J. Pei, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *proceedings of the 17th international conference on data engineering*, pages 215–224, 2001.

[11] B. J. Jansen. Click fraud. *Computer*, 40(7):85–86, 2007.

[12] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[13] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM, 2002.

[14] H. Kang, K. Wang, D. Soukal, F. Behr, and Z. Zheng. Large-scale bot detection for search engines. In *Proceedings of the 19th international conference on World wide web*, pages 501–510. ACM, 2010.

[15] A. Karasaridis, B. Rexroad, and D. Hoeflin. Wide-scale botnet detection and characterization. In *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, volume 7. Cambridge, MA, 2007.

[16] V. Krishnan and R. Raj. Web spam detection with anti-trust rank. In *AIRWeb*, volume 6, pages 37–40, 2006.

[17] Y. Liu, R. Cen, M. Zhang, S. Ma, and L. Ru. Identifying web spam with user behavior analysis. In *Proceedings of the 4th international workshop on Adversarial information retrieval on the web*, pages 9–16. ACM, 2008.

[18] M. Marchiori. The quest for correct information on the web: Hyper search engines. *Computer Networks and ISDN Systems*, 29(8):1225–1235, 1997.

[19] A. Metwally, D. Agrawal, and A. E. Abbadi. Using association rules for fraud detection in web advertising networks. In *Proceedings of the 31st international conference on Very large data bases*, pages 169–180. VLDB Endowment, 2005.

[20] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[21] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. Mining sequential patterns by pattern-growth: The prefixspan approach. *Knowledge and Data Engineering, IEEE Transactions on*, 16(11):1424–1440, 2004.

[22] F. Radlinski. Addressing malicious noise in clickthrough data. In *Learning to Rank for Information Retrieval Workshop at SIGIR*, volume 2007, 2007.

[23] N. Sadagopan and J. Li. Characterizing typical and atypical user sessions in clickstreams. In *Proceedings of the 17th international conference on World Wide Web*, pages 885–894. ACM, 2008.

[24] T. Schluessler, S. Goglin, and E. Johnson. Is a bot at the controls?: Detecting input data attacks. In *Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games*, pages 1–6. ACM, 2007.

[25] X. Yan, J. Han, and R. Afshar. Clospan: Mining closed sequential patterns in large datasets. In *Proc. 2003 SIAM Intąŕl Conf. Data Mining (SDMąŕ03)*, pages 166–177, 2003.