Multi-Stage Malicious Click Detection on Large Scale Web Advertising Data

Leyi Song, Xueqing Gong*, Xiaofeng He, Rong Zhang, Aoying Zhou
Center for Cloud Computing and Big Data
East China Normal University
3663 North Zhongshan Road, Shanghai, China
songleyi@ecnu.cn, {xqgong,xfhe,rzhang,ayzhou}@sei.ecnu.cn

ABSTRACT

The healthy development of the Internet largely depends on the online advertisement which provides the financial support to the Internet. Click fraud, however, poses serious threat to the Internet ecosystem. It not only brings harm to the advertisers, but also damages the mutual trust between advertiser and ad agency. Click fraud prediction is a typical big data application in that we normally need to identify the malicious clicks from massive click logs, therefore efficient detection methods in big data framework are much desired to combat this fraudulent behavior. In this paper, we propose a three-stage filtering system to attack click fraud. The serialized filters effectively detect the malicious clicks with decreasing confidence that can satisfy both advertisers and content providers.

1. INTRODUCTION

The fast development of the Internet depends not only on the increase of rich content, but also on online advertisement which provides the financial support to the Internet ecosystem. Online advertising tends to benefit all involved parties including content provider, advertiser, ad agency and ad network. It requires the mutual trust among all parties. This trust was at risk, however, by fraudulent clicks. Click fraud (also called click spam, malicious click) is an action of intentional clicking with the purpose of making undue profit, or doing harm to competitors. Click fraud is becoming a serious problem to the World Wide Web [5]. Failing to deter such behaviors will discourage advertisers from actively engaging in more online advertising activities, resulting in less revenue for content providers or ad agencies, and ultimately endangering the Internet ecosystem as a whole. Therefore it is of high importance to detect the malicious clicks.

Ad agencies or networks often deploy different filters to identify malicious clicks [10]. By setting a proper threshold or training a classifier, these methods can handle certain types of anomaly click behaviors generated by human or bots [7, 12, 13]. Despite such efforts, there still remains great challenge to address the problem of malicious clicks. In this paper, we focus on detecting malicious clicks from large scale web advertising data on ad agency side. Ad agency plays an intermediate role between advertisers and publishers in the online advertising system. In order to identify malicious clicks of different categories, for instance, suspicious users, stealthy click-bots and cheating publishers, we build a series of filters at different stages in big data computation framework. The filters are ordered by the decreasing confidence of predicting the malicious clicks. At first stage, a rule based filter identifies the malicious clicks with high confidence since these rules catch strong signals for abnormal clicks. At second stage, a supervised classification approach is used to detect malicious clicks, whose prediction results are of lower confidence than rule based ones. Finally we cluster the clicks into group at stage 3 with the hope that fraudulent clicks generated from one publisher will be grouped together. The clustering method is an unsupervised learning process, hence results in lowest prediction confidence. Sequentially organizing the filters in decreasing confidence order provides us with flexibility and scalability to add extra filters. For instance, we can replace one classifier with another classifier at stage 2, or add more classifiers to stage 2 to form an ensemble. Users have the freedom to use the result corresponding to different confidence.

We use the precision as the measure of confidence in this paper. The reason to use precision, instead of other metrics such as recall or F-measure is because 1) precision focuses on singling out bad clicks while trying to minimize the possibility of predicting valid clicks as fraudulent ones; 2) it is well-known that the judgement of whether or not a click is a fraudulent one is very subjective in many cases. Too large recall can potentially classify large number of valid clicks as bad ones unless we have high quality classifiers which is hard to obtain, especially when human judgement is difficult.

In this paper, we propose a click fraud detecting architecture which organizes filters sequentially by decreasing confidence order. This architecture was applied to large scale advertising log obtained from an ad agency. Specifically,

- we address the malicious click problem on the ad agency side. In addition to rule-based and supervised methods, we propose a clustering-based method to analyse the traffic quality;
- we design a click detecting mechanism in big data computation framework, i.e., Hadoop, to detect malicious

* Corresponding Author

clicks efficiently by sequentially organizing three types of filters of different confidence level;

- we carefully design and analyse important features, and verify our approach by evaluating the dissimilarity between predicted fraudulent clicks *vs.* valid clicks.

The rest of the paper is organized as follows. In Section 2, we introduce previous work related to click fraud detection. We present our detection architecture in Section 3. In Section 4, we comprehensively analyze the results of our approach by applying it to over one month real click log data from an ad agency, and conclude our paper in Section 5.

2. RELATED WORK

There are many participants play in online advertising ecosystem. Ad agency usually plays an match maker role between publishers and advertisers. Publishers own the website pages containing ad slots. Advertisers purpose their ad creativity to attract users to buy their products or to make other kinds of profit. Ad agency buys inventory from publishers and sells advertising traffic to advertisers. Thus, for a reputed ad agency, it is responsible to filter these malicious clicks before charging.

Undoubtedly there are great efforts in the area of fraud detection, including the click fraud problem, but most solutions are not easily available. The report by Tuzhilin [10] introduced some of the approaches that Google adapted to fight click fraud, in which both rule-based and anomaly-based filters were incorporated into search engine. Since search advertising is one of the major forms of online advertisement, this work inspired us to take similar approach in fraud detection system.

Another type of malicious click detection methods was based on click stream analysis techniques which identified patterns of fraudulent traffic. Metwally *et al.* proposed fraud detection solutions for data stream by combining association rules and duplicate detection methods [8]. Effective data structure such as modified Bloom Filter was used in this situation [13]. However, it is often difficult for most data mining-based detection methods to be implemented into stream analysis, hence limits the adaptation of such techniques to click fraud detection.

Supervised learning approach detects the malicious clicks by training a classifier. Data collection is one of the most important steps in this approach. Haddadi [3] proposed the idea of bluff ads, which is unrelated to the user search, user profile and recently activity. If a high ratio of such ads was clicked, the user could be flagged as suspicious. In some other work, CAPTCHA was used for training data generation and useful data collection [6]. For ad agencies, they have various advertiser and publisher sources, hence CAPTCHA approach is hardly applicable. The next key step is to extract features. The work [1] investigated query attributes between human and robot traffic. Different type of features could be extracted from the click attributes and user attributes, such as click count or geographic origin [1, 4].

Bot-generated click traffic is a big part of malicious clicks. The state-of-the-art bot detection work mostly aimed at clicks in search engine logs [6, 12]. Yu *et al.* proposed SBot-Miner, a system which automatically identified bot generated search traffic from query log, using history-based as well as matrix-based unsupervised methods [12].

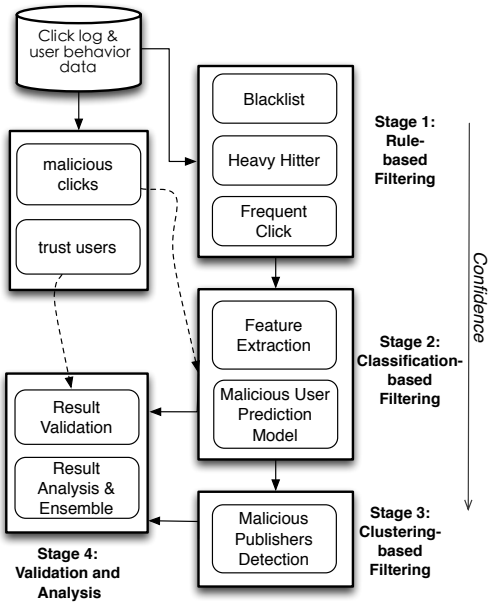


Figure 1: Architecture Overview.

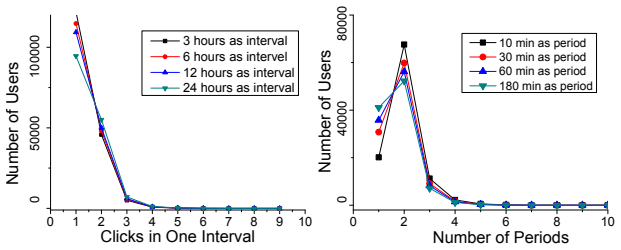
Previous approaches to identify and understand malicious clicks focused on one specific method, or one specific problem. Researchers adopted click-through rate related methods for web spam in search engine [11]. However, it is difficult for an ad agency to access the click-though rate, therefore some fraudulent patterns can not be easily identified by traditional methods. In our framework, we design a more general way of data collection and filtering strategy for malicious click detection, implemented in a stage-wise architecture.

3. STAGE-WISE CLICK FRAUD FILTERING ARCHITECTURE

3.1 Architecture Overview

In order to accommodate different filtering methods with different confidence in predicting the fraudulent clicks, we argue that it is advantageous to take the approach of stage-wise filtering architecture. The filters are sequentially connected such that the filter generating results with higher predicting confidence is put in the chain before the one with lower confidence. The intuition is that we need to identify most confident malicious clicks, and reduce the data size which need further processing, hence reduce the complexity of the problem. Furthermore, stage-wise structure offers prediction results of different confidence, which enables the users to utilize the result with more flexibility.

The stage-wise filtering system architecture is illustrated in Figure 1. The major components are following 3 stages. 1) Rule-based filters at stage 1 to detect obvious invalid clicks with high confidence. They can identify two types of malicious clicks: heavy hitter and frequent clicker. 2) Classification-based filters at stage 2 to determine more complicated clicks with human judged training set. 3) The clustering-based filter at stage 3 to identify cheating groups from similar publisher websites. We use intra-cluster distance and query diversity to separate malicious groups.



(a) Click density (b) Click frequency

Figure 2: Click statistics of a trusted user click log dataset in one month

3.2 Rule-based Filtering

To fight click fraud, generating blacklist in the filtering system is the most reliable method. It is easy to compile the blacklist for violating entities such as user agent(UA) or IP address, but the coverage of a blacklist is limited. Setting certain rules is efficient to exclude more malicious clicks from entering accounting system. In this paper we focus on setting rules for the heavy hitter and frequent click problems [8, 13, 7].

Heavy hitter in click logs means, at a specific time interval, the click rate of a user is relatively higher than a threshold λ_1, while frequent click problem refers to the situation that user's click appears in relatively more periods[1] than a predefined threshold λ_2.

To obtain a reasonable filtering threshold, we note from Figure 2 that the number of clicks and periods follows the *Zipfian distribution*. We set the maximum number for normal user behavior as the lower value of the two: number of clicks in one interval and number of periods the user clicks. For better accuracy, we can also determine the threshold based on the p-quantile value on the entire log dataset.

3.3 Classification-based Filtering

Classification-based methods are widely used in fraud detection or spam detection field [9]. Classification is an effective way for addressing malicious clicks, especially for stealthy clicks that are hard to be captured by rule-based approach. The classification has been applied to real data with success [4, 10]. One of the biggest advantages with classification approach is that once a model has been built, the prediction of new instance is usually quite fast. There is also research on the attributes which can distinguish human and bot traffic [1]. In our work, we use the traditional features used in previous work, and also engineer new features useful for training classifiers. We will discuss the features in more details later. The result of this stage is a set of malicious users from the click log.

3.4 Clustering-based Filtering

It is obvious that results of supervised methods highly depend on the accuracy, coverage and labeling scheme of the labeled corpus. The classified malicious clicks are limited by the fraudulent types in training set. Hence, we develop our clustering-based method, which is based on our observation that, for agency's customers(advertisers), some of the suspicious ad traffic from the same website shows high similarity. We try to group similar traffic and analyze them as a group in order to detect abusive clicks from the publisher-side. This approach can also be applied to search engine traffic, since the query diversity can be used to separate the suspicious groups from the search ad log. Considering the result confidence and filtering cost, this stage can be an optional choice for each advertiser. For valuable customers, this is an attractive feature, which differs from previous work.

For clustering, We define the **dissimilarity** between two log entries x, y as:

$$D(x,y) = \sum_{f \in Fields} w_f d_f(x,y) \tag{1}$$

where each log entry is represented by a vector of click attributes as $< user, IP, referrer, UA, area, query >$. $Fields$ is the feature set used in click attributes and $d_f(x,y)$ is the distance measure defined on each attribute f, normalized between [0,1]. w_f is the weight of $d_f(x,y)$.

For attributes like refer URL and user agent, we care about the longest matching prefix, and the distance measure is defined as:

$$d_{url}(x,y) = 1 - \frac{LCP(x,y)}{\max(|x|,|y|)} \tag{2}$$

where LCP means the longest common prefix of two strings.

Even though network address translation (NAT) might allow many users behind a single IP address, the cheating groups always show strong similarity in IP address. To amplify the importance of the tail part in IP address, we take 32 bits IPv4 address as an example to define the distance. If $LCB(longest\ common\ bits) \geq 16$, then

$$d_{IP}(x,y) = 1 - \frac{LCB(x,y)}{total\ bits}, \tag{3}$$

otherwise, the distance between two IP addresses is 1.

For other attributes such as area, we simply treat their dissimilarity as binary value (0 or 1). Furthermore, we can easily prove that $D(x,y)$ is a metric, since it follows the properties: (1) $D(x,y) = 0 iff x = y$, (2) $D(x,y) \geq 0$, (3) $D(x,y) = D(y,x)$, and (4) $D(x,y) \leq D(x,z) + D(z,y)$(triangle inequality). Due to space limit, we skip the proof here.

After obtaining the dissimilarity matrix of a log set, we use *k-medoids* algorithm to produce the clustering in this stage. K-medoids is an adaptation of k-means algorithm. Rather than calculating the mean of the items for each cluster, which is not applicable in our situation, a representative item, or medoid, is chosen for each cluster. Medoids for each cluster are calculated to finding object i by minimizing

$$\tilde{J} = \sum_{j \in C_i} D(i,j) \tag{4}$$

where C_i is the cluster containing object i and $D(i,j)$ is the dissimilarity function defined in equation 1. Since the algorithm simply looks up the dissimilarity matrix, it only needs to be calculated once in the beginning.

The next step is how to distinguish cheating groups from all clusters. Obviously, if one group agrees on most fields, it indicates this group of clicks come from a botnet using similar terminals and browsers or a real interested user with high probability. From the click statistics we can figure that,

[1]We use *period* to represent the window for counting in frequent clicks problem, in order to distinguish the *interval* of heavy hitter. Clicks of each user occur within the same period will be ignored.

Table 1: Examples of Labeled Malicious Clicks

User	Advertiser	Area	IP	Referer	Query	User agent
1	c1	1	x.x.25.177	none	none	Mozilla/5.0
2	c2	1	x.x.25.178	none	none	Mozilla/5.0
3	c3	2	y.y.51.137	http://r1.com/s?wd=wvihv	wvihv	Mozilla/4.0(compatible; MSIE 7.0;)
3	c3	2	y.y.51.142	http://r1.com/s?wd=jmfitxm	jmfitxm	Mozilla/4.0(compatible; MSIE 7.0;)
4	c3	2	y.y.51.145	http://r2.com/s?wd=qyfsoc	qyfsoc	Mozilla/4.0(compatible; MSIE 7.0;)

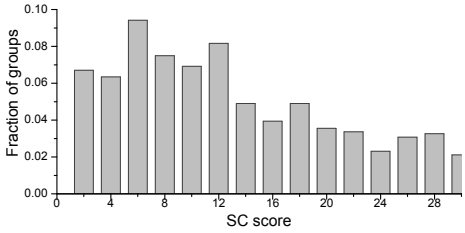


Figure 3: Fraction of groups vs. Scatter scores

if the group size is reasonable, it is less possible to be an interested user. In other words, if the intra-cluster similarity of a group is lower, then the probability of the traffics in the group being malicious is higher. Moreover, the high similarity of the referrer in a suspicious group means the low traffic quality of the publisher except search engine. In particular, we add the query diversity factor for search engine traffic. We define the *Scatter(SC)* score of each group as:

$$SC = intra\_distance \times query\_diversity \quad (5)$$

where *query_diversity* is defined as the ratio of distinct search phrases to total search phrases. The *query_diversity* is set to 1 for non-search engine traffic. By adding query diversity, we want to give more importance to the website traffic groups, since it is more difficult to locate the root cause of problem in search engine groups. Thus, we use the intra-cluster distance times query diversity to measure the ads click *Scatter*-ness of groups. Finally, the groups with low *SC* score will be regarded as suspicious groups. Figure 3 shows the distribution of *Scatter* score across our test groups. A brief description is shown in Algorithm 1.

input : clicks on each advertiser
output: groups of malicious clicks
initialize dissimilarity matrix;
while *not at end of advertisers' traffic set* **do**
  step 1: apply K-medoids using $D(x, y)$ distance to get traffic groups with similar features
  step 2: select suspicious groups with lower SC score
end

Algorithm 1: Major steps of getting cheating groups

3.5 Feature Extraction

The features extracted from data that work in web page spam domains may not work in ad log analysis. We inspect the click data and introduce several features specifically designed for classifiers to predict malicious clicks. Features are created by studying the labeled users' activity patterns.

In order to define useful features, we need to analyze the difference between normal and malicious click behaviors. Below, we discuss some of the features we identified to represent the user.

Number of clicked advertisers. This feature counts the number of distinct advertisers each user clicked. Malicious users show extreme patterns, for instance most have empty cookie and others have dense clicks on one advertiser.

Click ratio on advertisers. This feature takes into account both the total clicks and the distinct clicked advertisers, defined as total clicks/total clicked advertisers. For each user, it represents the average clicks per advertiser. We observe that the trusted users show higher diversity by comparing their histograms.

We also define features that can be used to characterize the attribute of a user. For instance, a fraudulent user might carry out the malicious click behaviors from one device, but with many dynamically allocated IP addresses. Thus, we derive features from user agent, IP and cookie. Short cookies are more suspicious than normal cookies, same goes for user agent. Some details about these features are shown below.

Click/IP ratio. This feature is defined as the total clicks for a user/total unique IP addresses the clicks come from.

Variance of IP clicks. After counting the number of clicks from each IP address, we can calculate the variance of these clicks. It will be suspicious if one user launches clicks with consistent frequency from some IP addresses. Besides, we also use features such as the number of user agents, number of referrers, length of agent, length of cookie.

There are some other features derived from geographic or temporal attributes. For click timestamp, we divide a day into four six-hour periods: night(0:00 to 5:59), morning(6:00 to 11:59), afternoon(12:00 to 17:59) and evening(18:00-23:59). With the information available, the features below are also extracted: *most frequent area*, *most frequent period*, *number of clicks in each period*, *mean/std deviation of clicks in periods* and so on. We are not going to list out all the 17 features used in classification considering the space limit.

4. EXPERIMENTS

In this section we present the experimental results of the stage-wise filtering framework.

4.1 Dataset

The data set used in the experiments is over one month click log we get from an ad agency company. It contains about 35 million records of ad clicks. The advertising log data normally contains attributes as user ID, click timestamp, user's IP, cookie, query phrase (*if the ad traffic is from a search engine*), user agent and referrer (*url of the page where user clicked the ad link*).

To protect privacy, users who click the ads are assumed to be only temporarily identified by cookie. We assume that

Table 2: Brief description of training set.

Description	# of Clicks/Users
Fraudulent clicks	174,642
Non-fraudulent clicks	779,980
Malicious users	83,061
Benign users	649,204

some user actions such as buying stuff or registering are benign. In this way, we extract non-fraudulent clicks from benign users for training. On the other hand, some malicious clicks were addressed using domain knowledge. Examples of malicious clicks are shown in Table 1. We notice that attack may come from similar IP addresses with fake user agent, fake referrer, or meaningless query phrase. Besides, the training data includes complete clicks of three advertisers that can be used to evaluate clustering performance. A brief description of dataset for training is shown in Table 2.

4.2 Experimental Setup

We run our stage-wise method on an 8-node Hadoop cluster using Pig Latin [2]. We split each stage into Map/Reduce modules in pipeline. Each module can be converted into one or several rounds of Map/Reduce tasks. For example the filtering phase in rule-based stage, users are partitioned through mappers and the counting filter UDFs(User Define Functions) on each user are performed in reducer. How to minimize the I/O cost is a major research for analysis on large scale dataset. Taking this problem into consideration, briefly, we design our UDFs to get the most results through one-pass over the data.

In the first stage of our framework, the bound rules for click density and click frequency are determined using p-quantile of the entire distribution. From our training data, we find there contains over 0.5% heavy hitters. There exists a certain percentage of noisy clicks in the log, 0.995 or similar (consider the distributions in Figure 1) can be chosen as value of p in our experiment for rule setting. The large click logs are filtered by passing through click counting, period counting, quantile calculating and filtering phases.

Next, the reduced dataset is passed to feature extraction modules as well as classifiers in the second stage. In our experiment, 17 features which have been discussed above are created. Then, we use training set in Table 2 to train classification models and compare the performance of the following classifiers: Naïve Bayes, Decision Table, Bayes Net, REP Tree and Random Forest(10 trees, 5 random features each). The comparison results of 5-fold cross validation over training set are presented in Table 3. Since the labelled malicious are just partial of real malicious clicks, the relatively recall performance is less important than precision. The performance could be very good result of the simple labeling scheme, thus we need unsupervised methods.

Finally, three advertisers with different click size(100K, 10K, 1K) are chosen as the evaluation dataset in the clustering stage. We set equal weight for different features in experiment. According to Figure 3, we use 2.0 as the lower bound for the SC score of groups, which means the group is quite dense. The number of K in applying K-medoids are determined experimentally.

4.3 Results Analysis and Validation

Table 3: Precision performance for classifiers.

Classifier	% Precision	Run Time (s)
Random Forest	**97.6**	322
REP Tree	**96.8**	98
Bayes Net	96.7	115
Decision Table	96.1	199.5
Naïve Bayes	52.9	27

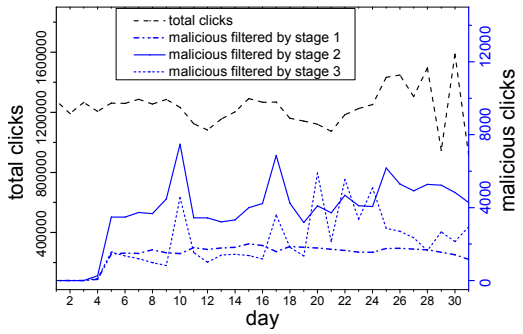


Figure 4: Analysis of the click log dataset.

Figure 4 illustrates the result of our implementation of stage-wise approach applied on click log of one month.

From the daily result, we find that a high percentage of clicks follows the format *http://search-engine.com/s?word* with the same short UA. Further study suggests that these clicks are coming from bot-net which directly inject noise into our logs using fake referer, user agent and IP address. We also find four suspicious publisher groups, which generate high density clicks with similar IP addresses and user agents from their website. Moreover, part of the suspicious clicks were generated by download manager showing in UA.

Figure 4 shows that the percentage of malicious in first stage distributed evenly. We could reasonably assume that all the heavy hitters and frequent clicks are malicious, since the upper bounds were setting based on the propriety of distribution. Thus, the rule-based filter can be chosen as the basic filter with absolute high confidence to meet basic requirement of ad agency. For classification stage, we choose Random Forest as our model. This stage can discover stealthy clicks with suspicious patterns derived from domain knowledge-based labeling. However, for the limitation of supervised method, false positive is inevitable. Our aim is to get the classifier with maximum precision. As to the clustering stage, we evaluate the precision on three different advertisers' click log. If the number of final result groups is n, precision metric in this case is:

$$Precision = \frac{1}{n}\sum_{i}^{n}\frac{|C_i \cap L|}{|C_i|}$$

where L is the cheating click set in result, and C_i is the total click set from each of the advertisers. Figure 5 shows our test results of clustering. Three advertisers' dataset achieve different precision performance, which indicates that the confidence partially depends on the distribution of real click data. Therefore, we set the clustering-based filters in the last stage. Indeed, even this filter has a lower confidence, it

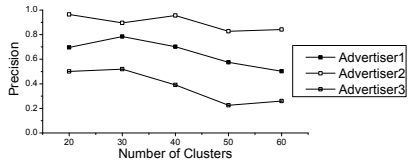


Figure 5: Precision performance of clustering method on evaluation data from three advertisers.

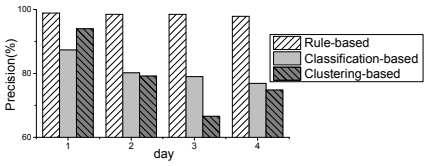


Figure 6: Precision performance of three stages on sampled result.

is important for ad agency to assess the traffic quality from publishers by evaluating the malicious group.

It is worth pointing out that our framework built on the top of Hadoop platform achieved high efficiency for processing click log. Due to the as-many-as computation in one-pass design, each stage could be finished in minutes for both daily and monthly filtering.

To further validate the stage-wise precision, we pick 4-day predicated malicious clicks for human judgement. We invite domain expert to inspect these clicks and the comparison results were shown in Figure 6. We see that the rule-based methods achieve high precision, which verifies our assumption. However, false positives are inevitable in any unsupervised learning algorithms. It is interesting that most of the false classified results are from mobile applications, especially on the 10th day. The missing of referrer field in these clicks is the root cause, which makes the patterns of these clicks similar to those of malicious clicks.

As a comparison, we check the diversity ratio on three features: hash of cookie, hash of user agent, hash of referrer, by sampling clicks from positive and negative results respectively. The diversity is defined as the ratio of distinct items to total samples. Figure 7 shows the result: diversities of normal clicks are relatively higher than these of malicious groups. For example, a group of intentional clicks with similar UA may come from one commander. The obvious difference between the malicious groups and normal groups suggests that the identified ones are indeed very suspicious.

5. CONCLUSION

Fraudulent click is a malicious behavior which threatens the healthy development of Internet ecosystem. In this work, we propose a stage-wise click fraud filtering architecture which effectively identifies the fraud clicks for ad agency with different prediction confidence. The stages in this work can be further divided into a set of modules, which consist of one or several rounds of Map/Reduce using parallel computing. We performed an in-depth analysis on one month click log using the proposed framework and evaluated our results by different metrics.

6. ACKNOWLEDGMENTS

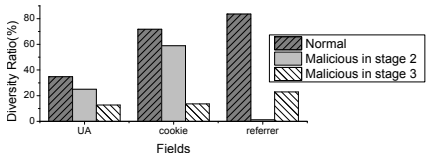


Figure 7: Diversity comparison on three features.

This work is partially supported by the Key Program of National Natural Science Foundation of China grant No. 61232002, National Science Foundation of China under grant No.60925008, No.61103039, No.61021004 and the Key lab Project of Wuhan University.

7. REFERENCES

[1] O. Duskin and D. G. Feitelson. Distinguishing humans from robots in web search logs: preliminary results using query rates and intervals. In *Proceedings of the 2009 workshop on Web Search Click Data*, WSCD '09, pages 15–19, 2009.

[2] A. Gates, O. Natkovich, S. Chopra, P. Kamath, S. Narayanam, C. Olston, B. Reed, S. Srinivasan, and U. Srivastava. Building a highlevel dataflow system on top of mapreduce: The pig experience. *PVLDB*, 2(2), 2009.

[3] H. Haddadi. Fighting online click-fraud using bluff ads. *Computer Communication Review*, 40(2):21–25, 2010.

[4] M. Hager and T. Landergren. Implementing best practices for fraud detection on an online advertising platform. Master's thesis, Chalmers University of Technology, 2010.

[5] B. J. Jansen. Click Fraud. *Computer*, 40(7):85–86, July 2007.

[6] H. Kang, K. Wang, D. Soukal, F. Behr, and Z. Zheng. Large-scale bot detection for search engines. In *WWW*, pages 501–510, 2010.

[7] B. Lahiri, J. Chandrashekar, and S. Tirthapura. Space-efficient tracking of persistent items in a massive data stream. In *DEBS*, pages 255–266, 2011.

[8] A. Metwally, D. Agrawal, and A. El Abbadi. Duplicate detection in click streams. In *WWW*, pages 12–21, 2005.

[9] C. Phua, D. Alahakoon, and V. C. S. Lee. Minority report in fraud detection: classification of skewed data. *SIGKDD Explorations*, 6(1):50–59, 2004.

[10] A. Tuzhilin. The lane's gifts v. google report. *http : //googleblog.blogspot.in/pdf/Tuzhilin$_R$eport.pdf*, 2007.

[11] C. Wei, Y. Liu, M. Zhang, S. Ma, L. Ru, and K. Zhang. Fighting against web spam: a novel propagation method based on click-through data. In *SIGIR*, pages 395–404, 2012.

[12] F. Yu, Y. Xie, and Q. Ke. Sbotminer: large scale search bot detection. In *WSDM*, pages 421–430, 2010.

[13] L. Zhang and Y. Guan. Detecting click fraud in pay-per-click streams of online advertising networks. In *ICDCS*, pages 77–84, 2008.