Classification of Automated Web Traffic

Greg Buehrer, Jack W. Stokes, Kumar Chellapilla and John C. Platt

As web search providers seek to improve both relevance and response times, they are challenged by the ever-increasing tax of automated search query traffic. Third party systems interact with search engines for a variety of reasons, such as monitoring a web site's rank, augmenting online games, or possibly to maliciously alter click-through rates. In this paper, we investigate automated traffic (sometimes referred to as *bot* traffic) in the query stream of a large search engine provider. We define automated traffic as any search query not generated by a human in real time. We first provide examples of different categories of query logs generated by automated means. We then develop many different features that distinguish between queries generated by people searching for information, and those generated by automated processes. We categorize these features into two classes, either an interpretation of the physical model of human interactions, or as behavioral patterns of automated interactions. Using the these detection features, we next classify the query stream using multiple binary classifiers. In addition, a multiclass classifier is then developed to identify subclasses of both normal and automated traffic. An active learning algorithm is used to suggest which user sessions to label to improve the accuracy of the multiclass classifier, while also seeking to discover new classes of automated traffic. Performance analysis are then provided. Finally, the multiclass classifier is used to predict the subclass distribution for the search query stream.

Greg Buehrer
Microsoft Live Labs, e-mail: buehrer@microsoft.com

Jack W. Stokes
Microsoft Research, e-mail: jstokes@microsoft.com

Kumar Chellapilla
Microsoft Live Labs, e-mail: kumarc@microsoft.com

John C. Platt
Microsoft Research, e-mail: jplatt@microsoft.com

1 Introduction

The Web has quickly become the *de facto* method for general information gathering. This transition has allowed web search to grow into a multi-billion dollar industry in only a few years. In addition, the proliferation of the web has allowed web search to become an environment for cultivating advancements along many dimensions of research. One such dimension is adversarial informal retrieval, and in particular spam detection. Two common types of spam are email spam and link spam. Email spam is designed to return the receiver to a location in which he would then be coaxed into purchasing a product, relinquishing his bank passwords, etc. This type of email is almost always automated. One study suggested that 85% of all email spam, which constitutes well more than half of all email, is generated by only 6 botnets [1]. With link spam, the generator is attempting to manipulate the search engine towards the end goal of improving its rank in the search results. The generator adds hyperlinks on web pages so as to imply the target page is important, since many since engines use PageRank [4] as an importance metric. For example, a high number of automatically generated web pages can be employed to redirect static rank to a small set of paid sites [7].

In this paper, we focus our attention on an understudied form of automation, namely automated web search engine queries. We define legitimate search queries as those queries entering the system which are typed by a human to gather information. Then, all other traffic is deemed automated traffic. Automated search traffic is of significant concern because it hampers the ability of large scale systems to run efficiently, and it lowers patron satisfaction by hindering relevance feedback. Because search engines are open for public consumption, there are many automated systems which make use of the service.

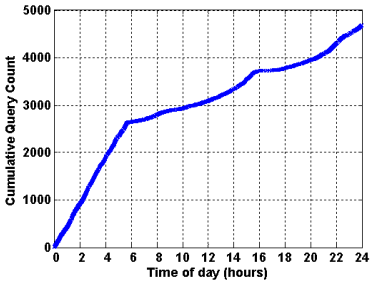

Fig. 1 A graph depicting *time of day* vs. *aggregate queries* for a typical bot.

A *bot* - the entity generating the automated traffic - may submit queries for a variety of reasons, most of which are benign and not overly monetizable. As an example, rank bots periodically scrape web pages to determine the current ranking for a <query,URL> pair. A Search Engine Optimization company (SEO) may employ a rank bot to evaluate the efficacy of his web page ranking optimizations for his clients. If a client's current rank is low, a user may need to generate many NEXT PAGE requests to find it in the

[1] http://computerworld.co.nz/news.nsf/scrt/C70ED4E3A608806CCC25740100186FC6

search engine's results. Since SEOs can have many clients, this practice can result in a significant amount of traffic for a single userId. The traffic of a typical bot is depicted in Figure 1. This bot queries approximately every seven seconds from midnight to about 6A.M., then at a slightly slower rate (approximately every 30 seconds) for the rest of the day. The total number of queries is about 4,500, far more than a human would do in normal browsing.

Correctly separating legitimate queries from automated queries can improve the end user experience in a number of ways. First, search latency can be reduced for legitimate queries; the search engine company may wish to throttle users to improve the Quality of Service (QoS) for interactive users. By reducing the total traffic serviced, or by reordering requests, response times for human users could be lowered (or maintained with less hardware). In addition, some search engines may consider click-through data implicit feedback on the relevance of a URL for a given query [15, 16]. This feedback may then be used to modify the rankings of the associated URLs. This may extend beyond explicit clicks, and include the absence of a click, such as to demote the URL of all results which were not clicked. Conversely, if the fourth result is clicked three times as often as the first result for a given query, it may imply that the fourth result is ranked too low. However, this form of ranking is as susceptible to malicious behavior as link ranking algorithms - an SEO could easily generate a bot to click on his clients' URLs. This form of automatically clicking links is commonly referred to as click fraud. Click fraud for paid search results has been a challenge for some time [5, 13, 15]. This activity may involve rival companies automating click activity on paid search results and banner ads in an attempt to increase an opponent's marketing costs. Another source of click fraud occurs when illegitimate businesses attempt to pose as intermediate search engines who host ads and forward illegitimate click traffic. Recently, a study by Click Forensics reported that click fraud for paid results in the 4th quarter of 2007 represents over 16% of all ad click traffic, up 15% from the same quarter of 2006 [2]. In particular, the report notes that search engine ads experience a fraud rate of 28.3%. Other reports suggest a lower fraud rate, closer to 6% [5], which is still rather high.

In this paper, we target automated traffic and clicks for unpaid results, which do not have the potential benefit of using conversion rates (e.g. Cost-Per-Action metrics) as secondary indicators [13] for legitimate activity. Detecting automated traffic can be difficult for several reasons. To begin with, the identification of user sessions is not trivial. One method to achieve this is through the use of cookies. A cookie is placed on the user's machine whenever the browser visits the site. Some users do not allow cookies, so each visit to the site appears to be a new user. In this case, the IP address of the request can be used, if there is sufficient confidence that the IP address is not a shared resource. In this work, we assume the ability to identify sessions. A second challenge in detecting automated traffic is that it may not be

[2] http://www.clickforensics.com/Pages/Releases.aspx?r=01312008

clear, even to a panel of experts viewing the queries, whether the source is automated. Although current techniques used to automatically query search engines can be relatively simple, sophisticated botnets certainly improve the ability of the programmer to mimic human traffic patterns [6]. Finally, as with most adversarial challenges, the behavior of the adversary changes over time. This suggests that specific signature-based solutions are only effective in the near term. For example, a bot may use the same IP address for an extended period of time, permitting a short term solution of ignoring traffic from that address. These types of practical solutions lead to black lists for IP addresses, user agents, referrers, etc., which must be constantly updated by the search engine. However, if the the bot is run from a different IP address, may be incorrectly labeled as normal traffic.

In this paper, we investigate both binary classification as well as multiclass classification, via subgroups of both normal and bot traffic (e.g. *SpamBot*, *AdultBot*). The goal of this work is to quickly discover new classes of query streams while simultaneously improving the accuracy of the resulting classifiers. Active learning is combined with multimodal anomaly detection to achieve this goal. In active learning for classifier training, a ranked list of items is provided for the analyst to label; labeling the samples at the top of the list improves the performance of the algorithm by some measure. In the past, active learning has been to shown to significantly increase the accuracy of a classifier by having the analyst label samples which are closest to one of the decision boundaries. A classifier is least certain of the class label of these samples - labeling them provides the most information for the next round of classifier training.

Instead of trying to learn a classifier to detect automated bot traffic, we could instead use anomaly detection. In typical anomaly detection algorithms, a model of the "Normal" behavior is learned. For the problem addressed in this paper, traffic which does not belong to the normal class is assigned to the automated bot traffic class. Typical anomaly detection algorithms often fail when it is difficult to learn a model which adequately describes all normal traffic. Instead, we learn separate models for each class of search query traffic. To discover new classes, active anomaly detection is also used. In active anomaly detection, samples are labeled which are the most anomalous for each class of traffic. By searching for new classes of traffic, we seek to quickly discover new ways that automated bots are querying the system. For malicious bot traffic, query patterns can change quickly. In the final algorithm (presented in Section 6), we combine active learning for classifier training and active anomaly detection to yield an algorithm which both discovers new classes of search query traffic and also produces individual classifiers which are highly accurate.

This paper makes the following contributions.

- A large-scale study of search engine traffic (100M requests) is performed.
- Several real-world bot patterns are described.

- Based on the study, a set of discriminating features is presented, designed to separate automated traffic from human traffic.
- A preliminary evaluation is performed, using both binary classification as well as multiclass classification.

The remainder of the paper is organized as follows. Related work is provided in Section 2. In Section 3, we describe the search query data used in this study. Section 4 describes behavioral features of current-day bots. In Section 5, we provide details of our proposed features. We partition the features into two groups, namely physical model features and behavioral features. In Section 6, we describe an algorithm based on active learning which can be used to label search query traffic in order to both increase a multiclass classifier's accuracy as well as discover new types of traffic. Experimental results are provided in Section 7. We then conclude in the final section.

2 Related Work

Relatively little work has specifically targeted classification of automated traffic in query logs. Agichtein, Brill, Dumais and Ragno developed models depicting user behavior for web search [1]. In this work, the authors are primarily interested in modeling users to guide relevance rankings, but some of these features could be used to partition humans from automated traffic as well. They point out that users provide more than click-through data when interacting with search engines. The authors consider deviations from normal behaviors, such as large increases in click-through rates for <query,URL> pairs. In addition, they incorporate page dwell time, query reformulation, and query length, among other features.

Research which studies click fraud in sponsored search results has examined traffic patterns and user behavior. These works do not address bot traffic with respect to organic results, but they do offer insight into the nature of the query stream. Daswani, et al. [6] dissect a large click botnet called ClickBot.A and describe its functionality and technique in detail, with accompanying source code. The botnet is of particular interest because it exhibits controlled execution so as to avoid detection, while still generating significant fraudulent impact. It replicates client bots on over 100,000 machines, each of which have a separate IP address and only click on at most twenty items. The authors do not provide a detection method.

A report by Tuzhilin [15] describes the challenges and issues with click fraud detection. In the report, the author concludes that Google, Inc is taking sufficient steps towards mitigating click fraud. Techniques include both static analysis and dynamic analysis, although exact measures are not described. The report also discusses an alternate reward system, in which rather than employing a system based on click-through rates, it is more advantageous for both parties if conversion rates were employed instead. Schluessler, Goglin

and Johnson [13] develop a client-side framework for detecting whether input data has been automatically generated. The technique targets online gaming, but also mentions that it can be used to address some forms of click fraud in online advertising.

Fetterly, Manasse and Najork [7] perform a similar study to our work to discover web link spam. They illustrate that statistical analysis of web page properties, in particular features such as out degree distributions, host-machine ratios, and near duplicate document clusters can provide significant lift in labeling portions of the web as spam or legitimate material. Anick [2] removes both known and suspected bots coming from internal AltaVista addresses for a study on web searcher behavior using terminological feedback. To eliminate bots traffic from a study on mobile web search, Kamvar and Baluja only considered traffic from a single large wireless carrier [11]. Karasaridis, Rexroad and Hoeflin [12] analyze the transport layer to discover IRC-based botnets attempting Denial-of-service attacks, among other malicious behavior. The method does not use signatures, instead monitoring ports for controller traffic patterns. The work does not investigate botnets attacking search engines.

The active learning algorithm used to label subclasses of search query traffic was originally proposed by Stokes *et al.* in [14]. In this paper, subclasses of network traffic were analyzed using a multiclass classifier in order to discover new forms of malware running on computers in a large corporate network: the malware is detected from anomalous network packet transmissions to machines located on the global internet.

3 Data Set Description

In this section, we describe the data used in our study. We obtained a random sample of approximately 100M requests to a popular search engine from a single day (August 7, 2007). We sampled user queries, such that if a user is in our sample, then all his queries from that day are also included the sample. For this study, we further prune the data to include only users who query at least five times in the day, resulting in 46M requests.

In this study, users are sessionized with a cookie and assigned a userId. Thus, at times we will refer to a userId and the underlying query stream interchangeably. The same applies for the term *bot*, although in this case the stream has been deemed automated. It is also common to sessionize by the requesting IP address. Although in some cases a single IP will service multiple users (i.e. proxies), and in some cases a single user may request from several IPs (see Figure 3), the technique of sessionizing by IP can be the basis for useful sessionization. It is possible for a single machine to produce both bot and human search engine traffic. In these cases, we do not attempt to separate multiple signals from a single cookie.

Finally, we offer some nomenclature. A query is an ordered set of keywords sent to the search engine. The engine then provides as a response an impression set (or simply impressions), which is a set of displayed results (both sponsored and organic). A query may have multiple requests, such as for result page two, upon which the engine will respond with additional impressions, e.g. the results for page two. Thus, the total number of requests may be more than the total number of queries. A click is always with respect to the impression presented to the user.

4 Qualitative Analysis

We now describe several bots discovered through the course of studying the query stream. While these are not inclusive, they are meant to present the reader with common forms of automated traffic. The first bot that we present scans the index for top spam words. Typically, the goal of improving a web site is to offer goods or services for financial gain; thus, a metric relating the query term to the other terms often found in email and/or web spam may be an indication that the query is generated by a bot. This class of bot rarely clicks, often has many queries, and most words have high correlation with typical spam. An example of 12 queries from one particular spam bot are presented in Table 4.

Queries	
Managing your internal communities	find your true love
mailing list archives	book your mountain resort
studnet loan bill	agreement forms online
your dream major	based group captive convert video from
computer degrees from home	products from thousands
free shipping coupon offers	mtge market share slips

Table 1 An example of a simple spam bot.

A second bot, which has a similar pattern of a large number of queries without clicking, but a different bag of words is a finance bot. Eighteen sample queries are presented in Table 2. Most of the keywords in the query are associated with mortgages, credit, money and the housing industry in general. The goal of this bot is to ascertain which web sites are most correlated with these finance terms in the search index.

Some bot activity implies less benign intent. The bot whose queries appear in Table 3 seems to be querying the system for various URLs which are either web sites owned by spammers and operated as spam sites (e.g. http://adulthealth.longlovetabs.biz/cialis.htm) or web sites on legitimate, hijacked domains (e.g. http://astro.stanford.edu/form_1/buy_cialis_oneline.html)

Queries		
2ndmortgage	bestmortgagerate	2ndmortgage
1sttimehomebuyer	badcreditloan	equity
1sttimehomebuyer	badcreditrefinance	equityloans
financinghouse	debtconsolidation	banks
badcredithomeloan	debtconsolidationloan	financing
badcreditmortgage	financinghouse	firstmortgage

Table 2 An example of a finance bot.

created to host spam. Presumably, the bot is attempting to boost its search engine rank. In the next example in Table 4, the bot queries the search engine for mortgage broker keywords. The bot is attempting to find the top ten broker results for a large set of cities. This bot queried the search engine over 2,000 times in our one day random sample.

Queries
http://astro.stanford.edu/forum/1/buy.cialis.online.html
http://adulthealth.longlovetabs.biz/cialis.htm
http://www.bigdrugstoreforyou.info?Viagra.cialis
http://www.cheap.diet.pills.online.info/drugs/pagemaker.html
http://dosap.info/d.php?search=ed,viagra,levitra,cialis
http://www.generic.viagra.cialis.levitra.info/index/cialis.php
http://www.pharmacydirectory.biz/submitlink5.html
http://www.get.prescriptions.online.biz/buy.viagra.online.htm
http://www.redloungebiz.section.gb?page=5

Table 3 An example of a URL bot.

Queries	
maricopa kern broker	monrovia los angeles broker
martinez contra costa broker	montague siskiyou broker
mcfarland kern broker	moorpark ventura broker
mendota fresno broker	moreno valley riverside broker
menifee riverside broker	Moreno valley riverside broker
menifee riverside broker	newport beach orange broker
merced merced broker	norwalk los angeles broker
mill valley marin broker	orange orange broker
millbrae san mateo broker	orland glenn broker
milpitas santa clara broker	oroville butte broker

Table 4 An example of a real estate bot.

Some bots not only repeatedly query the system for information with respect to a particular category, but query in such a way that provides an

unnatural signature. For example, the stock bot queries in Table 5 almost all are single keywords, and those keywords are primarily of length three or four. This bot appeared to be searching for financial news related to particular companies.

Queries								
pae	cln	eu3	eem	olv	oj	lqde	igf	ief
nzd	rib	xil	nex	intc	tei	wfr	ssg	sqi
nq	trf	cl	dax	ewl	bbdb	csco	pl	idti
nesn	edf	intl	spx	ewj	tasr	ibkr	lat	hb1
mesa	edl	dram	iev	sndk	rukn	ifg	igv	ms

Table 5 An example of a stock bot.

Another common bot scenario is when a userId sends queries from many different cities within a short amount of time. An example is shown in Table 6 (IP addresses have been coded to preserve anonymity). This userId sent 428 requests over a 4 hour period, from 38 different cities. Also, the bot always uses the *NEXT_PAGE* button when available and never clicks on algorithmic results (search engine results). The bot's queries had an unusually high number of adult terms. We suspect the userId is automating traffic through anonymous browsing tools, but oddly those tools did not account for machine cookies. It is not uncommon for a source of automated traffic and a legitimate user to originate from the same machine. In some cases, it may be botnet-related activity. However, a second common scenario is that the originator of the bot program is also using the search engine, possibly to set up the program. For example, a particular userId issued 6,534 queries, with only four clicks. Upon inspection, the four clicks were from the first five queries in the day, namely "pottery barn", "pottery barn kids", "pottery barn kids outlet", "pottery barn kids outlet store", and "pier 1". These queries spanned about 7 minutes, which is a typical usage pattern. The userId then issued 6,529 queries over the course of three hours without a click - clearly bot activity.

In a final example, one userId issued the same query 1,892 times over the course of the day. Of those requests, 1,874 had click responses. A possible motive for a very high click rate is to glean why the top results are so ranked. Then, the user can improve the rank of his page by incorporating the discovered attributes. For example, if a user queries the index for "best flowers in San Francisco" and then scrapes the html of the top 1,000 impressions, he can find the most common keywords in those pages, their titles, etc. and incorporate them into his own site.

Time	IP Address	City of Origin
4:18:34 AM	IP1	Charlottesville, Virginia
4:18:47 AM	IP2	Tampa, Florida
4:18:52 AM	IP3	Los Angeles, California
4:19:13 AM	IP4	Johnson City, Tennessee
4:22:15 AM	IP5	Delhi, Delhi
4:22:58 AM	IP6	Pittsburgh, Pennsylvania
4:23:03 AM	IP7	Canton, Georgia
4:23:17 AM	IP8	St. Peter, Minnesota

Table 6 An example of a bot with a single cookie but whose queries originate from many cities.

Name	Description	Type
Number of requests, queries, clicks	Number of requests, queries, clicks	Physical
Query Rate	The max number of queries in any 10 second period	Physical
Number of IPs/location	Number of originating IPs/cities	Physical
Click-Through Rate	Ratio of queries to clicks	Behavioral
Alphabetical Score	Alphanumeric ordering of queries, etc.	Behavioral
Spam Score	Indicator that the keywords are associated with spam	Behavioral
Adult Content Score	Indicator that the keywords are pornographic	Behavioral
Keyword Entropy	Informational entropy of query terms	Behavioral
Keyword Length Entropy	Informational entropy of query term lengths	Behavioral
Request Time Periodicity	Periodicity of requests, queries, clicks	Behavioral
Advanced Syntax Score	Number of advanced syntax terms in requests	Behavioral
Category Entropy	Informational entropy of categories of queries	Behavioral
Reputation	Blacklisted IPs, user agents, country codes, etc.	Behavioral

Table 7 A summary of the proposed query stream feature set.

5 Quantitative Analysis

Table 7 provides an overview of our set of features for detecting automated traffic in the query stream. We generally classify these features into two groups. The first group is the result of considering a physical model of a human. The second group is a set of observed behaviors of current-day automated traffic. In the following two subsections, we investigate each feature in some detail. Histograms are built for each feature, which are normalized to 100,000 users. Areas of high bot class lift in the graphs are then circled. Thus, in the figures presented in this section, the vertical axes are counts of users for the feature, and the horizontal axes are discretized ranges of the values of that feature. In a few cases, we normalized to one million users to allow the areas of interest to be sufficiently displayed.

5.1 Physical Model Feature Set

In this section, we discuss several features which are designed to model the interaction of a user and the search engine. Humans have physical limitations for entering queries, reading the results, and clicking on URLs. For example, a typical person can only issue and absorb a few queries in any ten second period. A user with 100 distinct requests in ten seconds would lie outside the boundary of normal use. Search query traffic entered by automated means are not subject to these physical limitations. Thus, the following features may be used to discriminate between web search traffic from humans and automated bots.

5.1.1 Number of Queries, Clicks, Pageviews, Etc.

A strong first indicator of automated traffic is volume. Bots often submit many more queries (and possibly clicks) in a given day than the typical person. Volume represents a class of features for which simple aggregate statistics can provide insight into the class of a userId. For example, in Figure 2 (left) we plot the distribution of the number of search queries from each unique user in our sample. While it is possible that a human user submits more than 200 queries in a given day, the histogram suggests it occurs with an unnatural probability. Upon inspection, we found that most of the traffic at this volume appeared automated. As an example, one userId queried the search engine for the term *mynet* 12,061 times during this day.

Fig. 2 Number of requests (left), and maximum requests in any 10 second interval (right).

5.1.2 Query Rate

Since bots are automated, they often enter queries at a much higher rate than queries which have been entered on a keyboard by a human. Various statistics of the query rate such as the average, median, and maximum can help distinguish queries generated by bots versus humans. We studied the

query rates for human traffic and concluded that humans rarely submit more than 7 requests in any 10 second interval. In Figure 2 (right), we plot the distribution of the maximum queries for a user in any 10 second interval over the course of the day. The users falling into the circled area were by and large bot traffic.

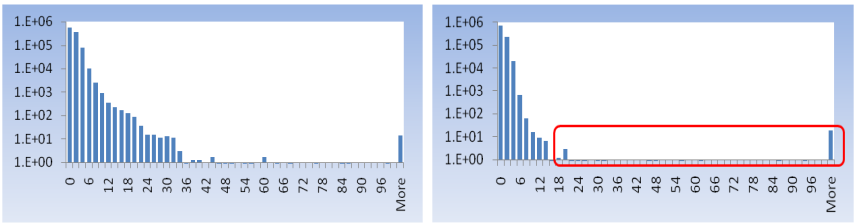

Fig. 3 Distinct IP address (all four octets) (left), and distinct IP address (first two octets) (right).

5.1.3 Number of IP Addresses / Locations

A human cannot be in two distant places at the same time. We maintain a list of requester IP addresses used by each userId. The motivation is to discover potential bot nets. If a user's cookie is compromised by miscreants and is used to make queries from two or more IP addresses, possibly located across large geographical distances, or is used in an interleaved fashion from two IP locations again separated by significant distances, then the unique Id likely belongs to two or more computers each of which are owned by a botnet[3]. A second usage scenario is when a userId is querying the system through an anonymous browsing tool, but has not disabled cookies. When correlating IP addresses, care must be taken to allow for mobile computers and devices which are used in the morning in one city, but later in the day at one or more additional cities. Also, users accessing the internet via a dial-up modem are often assigned a new IP address by the internet service provider (ISP) each time the user logs into the internet service. As a result the feature must ignore small variances in geographic location. In Figure 3 (left), we show a histogram of the number of users employing Multiple IP addresses (normalized to one million users). Figure 3 (right) depicts the same users wherein only the first two octets of an IP address are considered. This allows for multiple IP addresses in the same geographical region. We have highlighted the region where we moderate significant lift in bot classification. As an example, the bot in Table 6 would be flagged by this feature.

[3] http://www.hitslink.com/whitepapers/clickfraud.pdf

5.2 Behavioral Feature Set

The previous subsection introduces physical features that attempt to discriminate traffic generated by humans from that produced by automated means. However, automated search query traffic can be modeled to mimic human input. For these reasons we now provide additional features which seek to classify legitimate web search traffic generated by typical users from illegitimate traffic generated by automated means. In many cases, we will illustrate the efficacy of the feature with an example of a discovered bot.

5.2.1 Click-through Rate

Much of automated traffic is likely used for informational gathering purposes, either to examine the search engines index, or to collect data for self-use, and thus exhibits lower than typical click-through rates. Previously published click-through rates for humans vary, but most show that most users click at least once in ten queries. Our own inspection of the data suggests that many of the zero-click users are automated. Further, when used in conjunction with the total number of queries issued over the day, the feature provides very good lift. We illustrate this principle with two distributions, Figures 4 (left and right, neither are log plots). In Figure 4 (left), we plot click-through rates for all users in the sample with at least a modest number of queries. We then further prune the data to those users with ten times as many queries, shown in Figure 4 (right). Clearly, as the number of queries increases, the percentage of zero-click users increases. This is counter-intuitive if we limit the study to human users, since each query has a non-zero probability for clicking. However, if we consider automated traffic, we can reason about this increase; most bots do not need to click on the results. Even in the case where the bot requires extended information about the URL target, the bot can be programmed to load this URL directly. Thus there are three typical bot click through rates; a bot that clicks on no links, a bot that clicks on every link, and a bot that only clicks on targeted links. Of these, the first is the most common by a wide margin.

As an example, one userId queried for 56,281 times without a single click. On the other extreme, a second userId made 1,162 requests and clicked each time. Upon inspection of the queries, it appeared the userId was downloading the html for each impression in the index for the keywords *168.216.com.tw*. Also, the userId in Section 4 who clicked on 1,874 out of 1,892 requests would also be discovered by this feature.

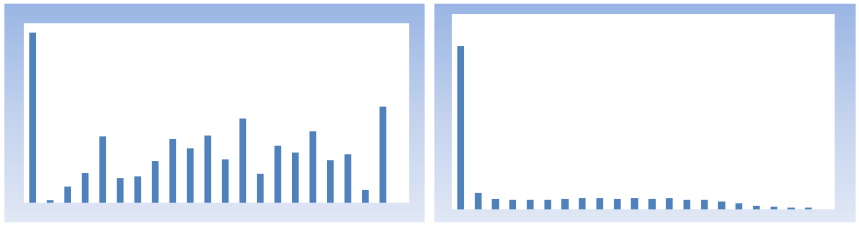

Fig. 4 Click-through rates, low minimum queries (left), and Click-through rates, 10X minimum queries (right).

5.2.2 Alphabetical Ordering of Queries

We have identified a number of instances of bot-generated queries which have significant alphabetical ordering. It may be that the authors of the programs use the alphabetical ordering for improved searching or analyzing. When submitted to the search engine, it is quite detectable. Returning to the bot in Table 4, we witness this behavior. To calculate an alphabetical score for a user, we order the queries chronologically and for each query pair $< i, i+1 >$, we add 1 if $i+1$ sorts after i, and subtract 1 if $i+1$ sorts before i. This number is then normalized by the total number of queries. In the majority of cases, the alphabetical score is near zero, as shown in Figure 5 (left). The discretization [-0.05,+0.05] has contains more than 50% of the mass in the distribution. In almost all cases where the user Id has more than a couple queries and the alphabet score was outside [-0.30,+0.30], we believe the traffic to be automated.

5.2.3 Spam Score

Spam bots submit spam words to a search engine such as the queries shown in Table 4. Consequently, a feature which estimates the amount of spam words in the search queries can be useful for detecting queries from spam bots. We compute a spam score as a feature using a bag of <spam word,weight> pairs for all queries for each userId. The weight assigns a probability that a given keyword is spam. For example, the term *Viagra* has a higher probability of being spam than the term *coffee*. In Figure 5 (right) we show a normalized histogram of the spam score for queries received from individual cookies. The circled region in the histogram indicates userIds submitting queries containing large numbers of spam terms. Per user scores are generated by summing the keyword spam score for their queries.

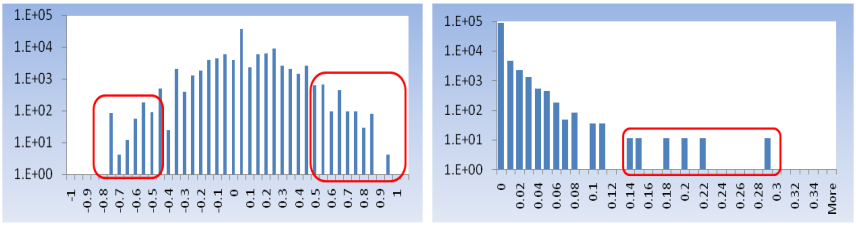

Fig. 5 Alphabetical scores (left), and spam scores (right).

5.2.4 Adult Content Score

The adult entertainment industry has taken to the web with vigor. Many in this industry attempt to attract new customers by directing users to web sites containing pornography. Adult content enterprises may employ bots to measure the ranking of their web site or try to boost their web site's rank in the search engine. Although it is also a common human query space, there is lift in relative adult query counts. Thus, bot generated queries often contain words associated with adult content. As with the spam score, we use another bag of <adult word,weight> pairs to compute an adult score for each userId. A normalized histogram is presented in Figure 6 (left) where we have circled the region in the figure which offers significant lift for bot detection. Examples of discovered bots for this feature are omitted due to space constraints.

5.2.5 Query Keyword Entropy

Many bots enter queries that are extremely redundant; as a result, bot queries tend to have keyword entropies which fall outside normal usage patterns. We calculate a map of <word,count> pairs for each userId. We then use traditional informational entropy, H(k), to assign a score to each user

$$H(k) = E(I(k)) = -\sum_i \sum_j p(k_{ij}) \log_2 p(k_{ij}) \quad (1)$$

where k_{ij} is the jth keyword (i.e. query term) in the ith query submitted by a single userId. In Figure 6 (right) , we plot the distribution of the entropy of keywords in the set of queries issued by users. In one example of a low keyword entropy bot, a user queried *mynet* 10,497 times, generating an entropy of zero.

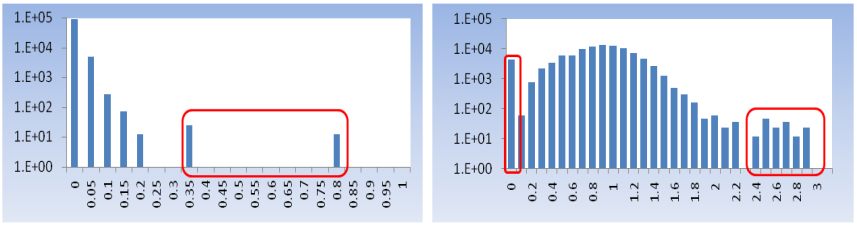

Fig. 6 Adult content scores (left), and Query term entropy (right).

5.2.6 Query Word Length Entropy

Typical query terms have a natural word length entropy distribution, as does the length of a typical query. Some bots query for specific classes of words which are outliers of this distribution. For example, the word length entropy for the stock bot shown in Table 5 will have a lower word length entropy compared to that for a typical human. The word length entropy WLE is calculated as

$$WLE(l_{ij}) = -\sum_i \sum_j l_{ij} \log_2(l_{ij}) \tag{2}$$

where i is the index for each separate query submitted to the search engine by a single userId and l_{ij} is the length of the individual query term j in the ith query. The word length entropy is shown in Figure 7 (left, normalized to 1M users). One could also have as a feature the longest query in the session.

5.2.7 Query Time Periodicity

It is not uncommon for a bot to generate traffic at regular intervals, such as every 15 minutes [6]. To capture this property, we sort requests by request time for each user, and calculate the difference in time between successive entries. For each observed delta, we record the number of occurrences for each user. We then calculate the informational entropy of the deltas (a second option would be to calculate an FFT score for each user). This can be done at a variety of granularities for time deltas (seconds, 10 seconds, minutes, etc). The distribution for seconds can be seen in Figure 7 (right). This feature can be used to investigate dwell time (the number of seconds spent on a clicked URL) [1]. When combined with other features, such as the number of requests, it has the potential to provide significant lift. For example, a userId with 30 queries may not appear automated based on request count alone, but if the entropy for time deltas is zero, it is much more likely to be a bot.

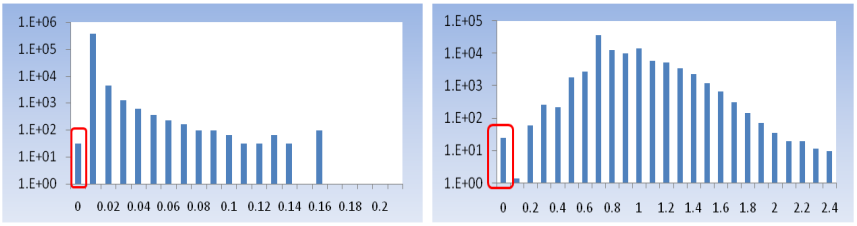

Fig. 7 Keyword length entropy (left), and query period entropy (right).

5.2.8 Advanced Query Syntax

Some bots use advanced syntax to probe particular features of the index. For example, prefixing a query with *intitle:* for many search engines will force results to have the listed keywords as part of the title of the web page. Similarly, *inURL:* will restrict results to those URLs which have the keywords embedded into the URL. To discover bots which use advanced query syntax, we keep a total count of all advanced terms for each user throughout the day. A histogram is shown in Figure 8. Less than 1/10th of one percent of users use more than 5 advanced terms in the sample. As an example of a bot, one user had 110 queries, all of which requested the terms appear in the title of the web page.

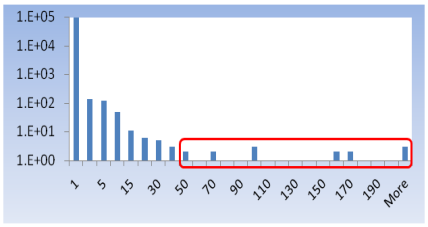

Fig. 8 Advanced query term scores.

5.2.9 Category Entropy

As a generalization of both adult content score and spam score, we can define a feature which captures the number of distinct categories associated with a userId. We use a category hierarchy to assign a category to each query. We then track the category entropy for each userId.

5.2.10 Reputations and Trends

There are several fields in the query logs that can directly identify known bot activity. Examples include blacklisted IP addresses, blacklisted user agents, and particular country codes. Tables are built for each property using domain expertise. For these cases, we simply perform a lookup into these tables at runtime. In less direct cases, query and query-click probability lists are used. For example, some bots search rare queries inordinately often. We often see sessions where each query is nonsensical. To detect these bots, a table of query-frequency pairs can be used to evaluate the popularity of the sessions queries. Finally, a table of <query,URL> click pairs can be stored to evaluate the probability that the user will click on a particular page. Users who often click on very low probability pairs are then deemed suspect. A potential weakness of these last two features is that a separate process is required to update the tables on a regular basis, and the tables can be somewhat large.

6 Active Learning for High Classifier Accuracy and Rare-Class Discovery

In the previous sections we provide numerous physical and behavioral features which can discriminate human from automated search query traffic. In this section, we discuss an algorithm based on active learning which can be used to provide labeled queries to train a multiclass classifier in order to detect subclasses within the search query stream. In general, an active learning algorithm is simply one that can make oracle queries as to the labels of a small number of data points. While it may be expensive to label a "large" set of data, we assume an analyst is available to label a limited number of queries. The proposed algorithm seeks to quickly identify new subclasses of *Normal* and *Bot* traffic while simultaneously improving the accuracy of the resulting multiclass classifier. Using active learning, a ranked list of items is provided to the analyst to label; labeling the samples at the top of the list improves the performance of the algorithm by some measure. Active learning is typically used to reduce the number of samples needed to train a classifier and has been shown to outperform classifier training based on randomly sampled items. The active learning algorithm used in this paper was originally proposed to discover new forms of malware based on outbound network behavior on a large corporate network [14]. In the earlier work, the original goal was to combine active learning and anomaly detection (active anomaly detection) to discover new classes of network traffic which might correspond to new forms of malware transmitting information collected from an infected computer back to the malware host server. While the first iteration of the algorithm was able to quickly identify new forms of network traffic, using active anomaly detection alone did not produce classifiers with high

accuracy for small amounts of labeled data for a particular class. To overcome this problem, active anomaly detection was combined with standard active learning to produce a framework which both discovers new classes in a data set while learning a multiclass classifier with high classification accuracy.

By discovering new classes of queries, we seek to gain the following:

- Discover new forms of automated bot and normal search query activity in the vast amounts of data processed by a production search engine.
- Understand the distribution of different types of bot activity in a typical day of search engine queries.
- Train a multiclass classifier and compare the performance with a simpler, binary classifier (i.e. two-class *Normal,Bot*).

As described in section 3, a day's worth of search traffic was collected from a production search engine and converted into a set of items (feature vectors) corresponding to each userId. Each item, $\mathbf{x}$, is composed of a subset of the individual search query features described earlier. For this problem, all of the features are continuous quantities and the log of each feature is modeled with a univariate Gaussian distribution.

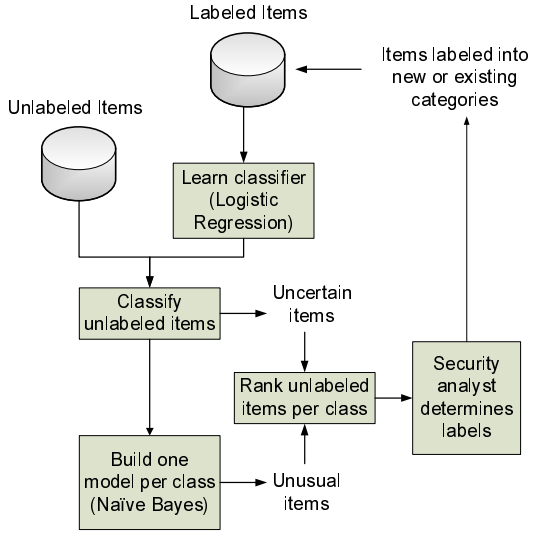

Fig. 9 The algorithmic architecture for discovering new classes of search query traffic.

The algorithm described in Figures 9 and 10 with pseudo code provided in Figure 11 is now briefly reviewed for the problem of discovering new classes of normal and automated search queries; interested readers should consult [14] for additional details. A classifier is first trained using a set of labeled items (i.e. aggregated search query feature vectors) indicated by the larger dots

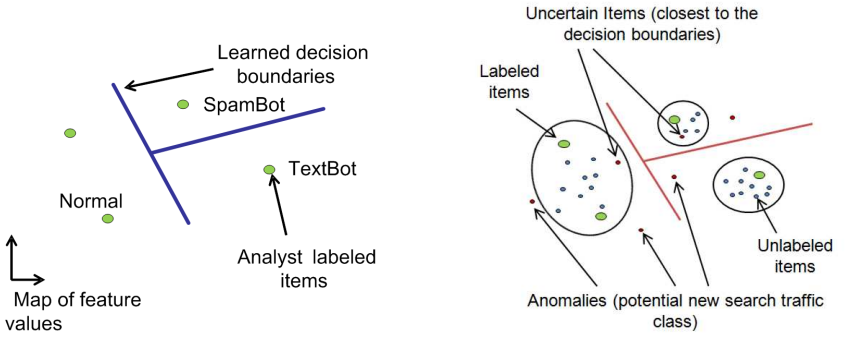

Fig. 10 The left figure shows the result of training a classifier: decision boundaries in input space are hyperplanes. The right figure shows the selection of the anomalies and uncertain items by the per-class models.

in Figure 10. In this work, we use multiclass logistic regression [3] trained with stochastic gradient descent, although other supervised algorithms (e.g. Support Vector Machine (SVM), naive Bayes) could be used for this step. Multiclass logistic regression produces a set of linear hyperplanes which partitions the high-dimensional feature space. Once the multiclass classifier has been learned, the class labels for all of the unlabeled items, indicated by the smaller dots in Figure 10, are then predicted. In addition, an certainty score C (i.e. the margin) is computed for each unlabeled sample which provides a measure of the distance from the labeled sample to the closest decision boundary. The certainty score is given by

$$C = \min_{i,j \neq i} |P(i|\mathbf{x}) - P(j|\mathbf{x})|, \tag{3}$$

for item $\mathbf{x}$ and classes i and j where $i = \arg\max_k(P(k|\mathbf{x}))$.

Afterwards, a generative model is learned *for each class* by combining labeled items with the predicted labels from the unlabeled items belonging to the class. In our work, we learn a naive Bayes model for each class allowing us to compute an anomaly score A which estimates how likely each unlabeled item is to belong to its model. For naive Bayes, the anomaly score can be computed for item $\mathbf{x}$ predicted to belong to class c by taking the negative of the sum of the log of the probabilities (i.e. log-likelihood) for each feature f, as

$$A = -\log P(\mathbf{x}|\text{class } c) = -\sum_f \log P(x_f|\text{class } c), \tag{4}$$

where a large anomaly score indicates a low probability (anomalous) item. The unlabeled items are then ranked for an analyst to label. The ranking is performed in a round-robin fashion where a set containing a single item from each class is chosen during one iteration of the algorithm for an analyst to

review. The round-robin nature of the algorithm (i.e odd, even iterations) is illustrated by the two output arrows emerging from the "Classify unlabeled items" block in Figures 9, and the ranking metrics are illustrated on the right side of Figure 10. For the odd iterations, a set of queries containing one anomaly for each class (the sample with the next highest anomaly score) is presented to the analyst. The anomalies for each class are unlabeled items which may belong to a new class of bot or human search query traffic; this iteration facilitates new class discovery. For the even iterations of the labeling process, the set containing the most uncertain item for each class is then presented to the analyst for labeling. These unlabeled, uncertain items lie closest to one of the decision boundaries and are indicated by the sample with the next smallest certainty score. If the decision boundaries are incorrect, these uncertain items may actually belong to another class. Labeling these items will improve classification accuracy compared to labeling a random unlabeled item predicted to belong to the class.

1. Learn a classifier on the labeled samples
2. Evaluate the classifier and a certainty score C, (i.e. the margin), for each of the unlabeled samples
3. Assign all unlabeled samples to the most likely category
4. Compute the model parameters (mean, variance, or histogram) for every $P(\mathbf{x}|c)$
5. Compute an anomaly score A, how likely the sample is to belong to the class, for all unlabeled samples
6. Select the next group of samples to be labeled choosing as follows, sweeping through the categories:
   a. select the next, most anomalous unlabeled sample with the highest anomaly score in each class
   b. OR, select the sample with the smallest certainty score for each class
   c. if not enough samples for a class are found from 6a) or 6b) select the unlabeled sample with the next highest output probability $P(c|\mathbf{x})$ corresponding to the desired class c
7. Repeat step 6 until the desired number of samples have been labeled for the iteration

Fig. 11 One iteration of the proposed active-learning training algorithm, as pseudo code.

After the weights for the classifier have been trained using logistic regression, new incoming search queries can be then evaluated. Evaluation for logistic regression is extremely fast and is easily parallelizable. As a result downstream processing can have an accurate prediction of the type of normal or bot traffic received from a userId.

7 Experimental Results

In this section, we investigate the experimental performance of both the binary (i.e. *Normal*, *Bot*) and well as the multiclass search query classifier. We labeled 370 different user sessions using the active learning framework described in Section 6. The process discovered eight classes of search traffic shown in Table 8. Using the 370 labeled user sessions as an oracle, we simulate how fast the active learning system would discover all of the classes in Figure 12. We first randomly select 10 sessions which yields 3 classes of search query traffic. Having the oracle label 20 sessions per iteration, we see that the algorithm has identified the remaining 5 classes in 3 iterations (i.e. 60 labeled sessions).

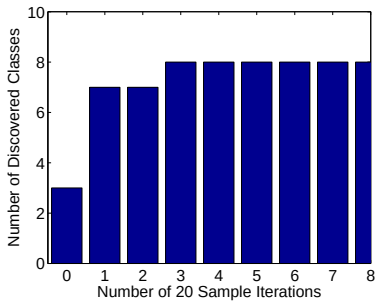

Fig. 12 The number of discovered classes using proposed active learning algorithm.

From Table 8, the *AdultSurfer* class contains search queries from a human searching for adult content; the Adult Content feature is useful for detecting items belonging to the *AdultSurfer* class. The remainder of the search queries entered by typical human users are represented by the *Normal* class. Including the main *Bot* class, there are six bot types. The *AlphaBot* contains queries which have a large number of consecutive searches which are in alphabetical order and use the Alphabet Score as the primary feature for detection. A *TextBot* contains large amounts of random text submitted to the search engine: the keyword entropy feature is highly indicative of a *TextBot*. In addition to being a strong feature for the *AdultSurfer* class, the Adult Content feature is useful for detecting *AdultBots*: bots which typically submit a large number of queries containing adult keywords. Other physical and behavioral features also help to further separate instances of these two classes. The Spam Content feature is helpful for detecting *SpamBots* which submit queries containing many different types of spam terms. A *TrackBackBot* contains queries which usually contain the terms "*trackback address*" or "*trackback from your own site*". Adding additional features to detect these specific keywords would significantly improve the accuracy of detecting the *TrackBackBot* class. Finally, all remaining automated traffic is currently grouped in the *Bot* class. It

is noted that not all of the Bot types described in Section 4 were discovered by this process, most likely due to insufficient labeled data.

We note that this distribution is artificially skewed towards an equal amount of Bot and Normal traffic because the active learning suggests userIds for labeling which do not lie in the mass of previously discovered classes. Also, a larger set of labeled sessions would improve confidence.

Class	Type	Amount
Base Normal	Normal	205
AdultSurfer	Normal	12
Base Bot	Bot	100
TrackBackBot	Bot	7
TextBot	Bot	15
SpamBot	Bot	10
AlphaBot	Bot	12
AdultBot	Bot	9

Table 8 The labeled multiclass data set - note that the amounts are distinct.

7.1 Binary Classification Results

We report binary classification results provided by the publicly available Weka toolset [8] (default options in all cases), as shown in Table 9. In all cases, we used 10-fold cross validation. We consider automated traffic labeled as automated traffic to be a true positive, noted as TP. Most of the classifiers chosen afforded about 90% accuracy on this small labeled, data set. Bagging with trees was the most accurate at 93%.

Classifier	TP	TN	FP	FN	Percent Correct
AdaBoost	123	207	10	30	89
ADTree	131	201	16	22	90
BaggingTrees	135	209	8	18	93
Bayes Net	130	202	15	23	90
Logistic Regression	126	211	6	26	91
Nave Bayes	131	201	16	22	90
PART	129	199	18	24	87

Table 9 The binary classification results using proposed feature set and Weka.

We also used Weka's attribute evaluator to gain insight into the relative benefits of each feature, namely Information Gain using the Ranker search method. The top four features in order were query count, query entropy, max

requests per 10 seconds, click through rate, and spam score, with ranks of 0.70, 0.39. 0.36, 0.32, and 0.29. As suspected, volume is a key indicator of present-day automated activity.

7.2 Multiclass Classification Results

We now employ Weka to classify in the multiclass case. If we consider exact class matches, then the accuracy is lowered considerably, as is shown in Table 10. The highest accuracy is 83% afforded by Logistic Regression.

Classifier	**Percent Correct**
AdaBoost	76
BaggingTrees	81
Bayes Net	78
Logistic Regression	83
Nave Bayes	71
PART	82
RandomForest	82

Table 10 The multiclass classification results using proposed feature set and Weka.

However, as shown in the confusion matrix[4] in Table 11, the accuracy of Logistic Regression is 89% if we consider any Bot userId labeled as any Bot type as correct. For example, examining row E, the Bot row, we can see that of the 29 misclassified elements, 9 are actually classified as a Bot subclass. Similarly, 4 of the 8 misclassified Alphabots were classified as some other Bot type. We then ran the classifier on a large subset of the data described in Section 3. The relative Bot distribution is shown in Figure 13 (scaled to 1M userIds); the vast majority of bots were not classified as a distinct Bot subclass. It is reasonable to argue that for many of the Bot subclasses, there are insufficient samples to predict accurately. One avenue for further study would be to label much more data, and retrain the classifier.

We now investigate how well the multiple classes separate when projected onto the two largest basis functions using principal component analysis (PCA). The covariance matrix was first computed using all of the samples from both the labeled and unlabeled collections. The resulting projection of the labeled data is shown in Figure 14. The items from the *Normal* class and the *AdultSurfer* do cluster relatively well on the left side of the figure, but overall there is not significant correlation. For PCA, the different classes of automated traffic tend to overlap.

[4] A confusion matrix shows the count of the correctly classified instances (values on the diagonal) as well as the count of the misclassified instances (values off the diagonal).

A	B	C	D	E	F	G	H	← Classified As
6	1	0	0	3	0	0	0	**A = SpamBot**
1	5	0	0	1	0	2	0	**B = AdultBot**
0	0	3	0	2	0	0	1	**C = TrackBack**
0	0	0	195	6	0	2	2	**D = Normal**
2	1	2	19	71	2	1	2	**E = Bot**
0	0	0	0	2	13	0	0	**F = TextBot**
0	1	0	1	0	1	9	0	**G = AdultSurfer**
0	0	0	3	4	0	1	4	**H = AlphaBot**

Table 11 The confusion matrix for the multiclass case, using logistic regression as the classifier.

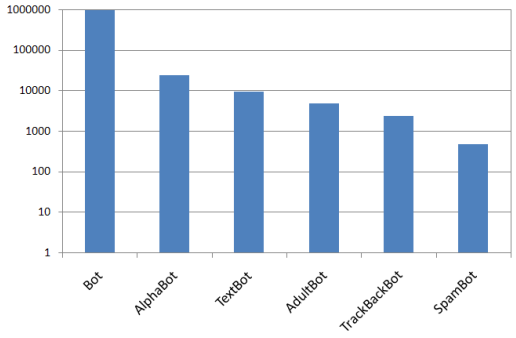

Fig. 13 The predicted distribution of bot traffic for sessionized userIds.

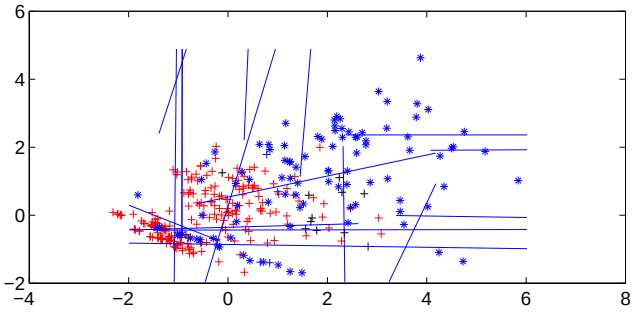

Fig. 14 The multiclass query projection using principal component analysis.

8 Conclusions

In this paper, we provided an investigation of web query traffic received from a large-scale production search engine. Separating automated traffic from human traffic is useful from both a relevance perspective as well as a performance perspective. To this end, a set of repurposeable features has

been proposed to model the physical interaction of a user as well as the behavior of current day automated traffic. An analysis of the distributions of these features indicated they can be used as a basis for labeling search engine query streams accordingly. We then described a framework for active anomaly detection, which can be used to discover distinct subclasses of automated traffic. An empirical study suggests that while classification is more accurate in the binary case, information can be gleaned by investigating subclasses. We are investigating several avenues which may improve the proposed feature set, for example analysis over a longer time range (one month), and the evolution of a user's query stream over time.

References

1. Agichtein, E., Brill, E., Dumais, S., and Ragno, R.: Learning User Interaction Models for Predicting Web Search Result Preferences, In *SIGIR'06*, 29th International ACM Conference on Research and Development on Information Retrieval, 2006. (ACM, New York, NY) pp 3 - 10.
2. Anick, P.: Using Terminological Feedback for Web Search Refinement - A Log-based Study., In *SIGIR'03*, 26th International ACM Conference on Research and Development on Information Retrieval, 2003. (ACM, New York, NY) pp 88 - 95.
3. Bishop, C.: *Pattern Recognition and Machine Learning*. (Springer, New York, NY, 2006).
4. Brin, S., and Page, L.: The Anatomy of a Large-scale Hypertextual Web Search Engine, In *WWW'98*, 7th International Conference on World Wide Web, 1998. (Elsevier Science Publishers B. V.,Amsterdam, The Netherlands) pp 107 - 117.
5. Click Quality Team, Google, Inc. How Fictitious Clicks Occur in Third-Party Click Fraud Audit Reports. http://www.google.com/adwords/ReportonThird-PartyClickFraudAuditing.pdf, 2006.
6. Daswani, N., Stoppelman, M., and the Google Click Quality and Security Teams: The Anatomy of Clickbot.A, In *HOTBOTS'07*, 1st Workshop on Hot Topics in Understanding Botnets, 2007. (USENIX Association, Berkeley, CA) pp 11 - 11.
7. Fetterly, D., Manasse, M., and Najork, M.: Spam, Damn Spam, and Statistics: Using Statistical Analysis to Locate Spam Web Pages, In *WebDB'04*, 7th International Workshop on the Web and Databases, 2004. (ACM, New York, NY) pp 1 - 6.
8. Frank, E., Hall, M., Trigg, L., Holmes, G., and Witten, I. H.: Data mining in bioinformatics using Weka. Bioinformatics , 20(15), 2479 - 2481, 1994.
9. Joachims, T.: Optimizing Search Engines Using Clickthrough Data, In *KDD'02*, 8th ACM International Conference on Knowledge Discovery and Data Mining, 2002. (ACM, New York, NY) pp 133 - 142.
10. Joachims, T., Granka, L., Pang, B., Hembrooke, H., and Gay, G.: Accurately Interpreting Clickthrough Data as Implicit Feedback, In *SIGIR'05*, 28th International ACM Conference on Research and Development on Information Retrieval, 2005. (ACM, New York, NY) pp 154 - 161.
11. Kamvar, M., and Baluja, S.: A Large Scale Study of Wireless Search Behavior: Google Mobile Search, In *CHI'06*, CHI Conference on Human Factors in Computing Systems, 2006. (ACM, New York, NY) pp 701 - 709.
12. Karasaridis, A., Rexroad, B., and Hoeflin, D.: Wide-scale Botnet Detection and Characterization, In *HOTBOTS'07*, 1st Workshop on Hot Topics in Understanding Botnets, 2007. (USENIX Association, Berkeley, CA) pp 7 - 7.

13. Schluessler, T., Goglin, S., and Johnson, E.: Is a Bot at the Controls? Detecting Input Data Attacks, In *NetGames'07*, 6th Workshop on Network and System Support for Games, 2007. (ACM, New York, NY) pp 1 - 6.
14. Stokes, J. W., Platt, J. C., Kravis, J., and Shilman, M.: ALADIN: Active Learning of Anomalies to Detect Intrusions, Microsoft Research Technical Report MSR-TR-2008-24, March 4, 2008.
15. Tuzhilin, A.: The Lane's Gifts v. Google Report. http://googleblog.blogspot.com/pdf/Tuzhilin_Report.pdf.
16. Wu, K.-L., Yu, P. S., and Ballman, A.: SpeedTracer: A Web usage mining and analysis tool. http://www.research.ibm.com/journal/sj/371/wu.html, 1998.