Large-scale Bot Detection for Search Engines

Hongwen Kang*
Carnegie Mellon University
Pittsburgh, PA 15213, USA
hongwenk@cs.cmu.edu

Kuansan Wang
Microsoft Research,
Redmond, WA 98052, USA
Kuansan.Wang@microsoft.com

David Soukal
Microsoft, Redmond, WA 98052, USA
dsoukal@microsoft.com

Fritz Behr
Microsoft, Redmond, WA 98052, USA
fritzb@microsoft.com

Zijian Zheng
Microsoft, Redmond, WA 98052, USA
zijianz@microsoft.com

ABSTRACT

In this paper, we propose a semi-supervised learning approach for classifying program (bot) generated web search traffic from that of genuine human users. The work is motivated by the challenge that the enormous amount of search data pose to traditional approaches that rely on fully annotated training samples. We propose a semi-supervised framework that addresses the problem in multiple fronts. First, we use the CAPTCHA technique and simple heuristics to extract from the data logs a large set of training samples with initial labels, though directly using these training data is problematic because the data thus sampled are biased. To tackle this problem, we further develop a semi-supervised learning algorithm to take advantage of the unlabeled data to improve the classification performance. These two proposed algorithms can be seamlessly combined and very *cost efficient* to scale the training process. In our experiment, the proposed approach showed significant (i.e. 2 : 1) improvement compared to the traditional supervised approach.

Categories and Subject Descriptors

H.2.8 [**Information Systems**]: Data mining; I.5 [**Computing Methodologies**]: Pattern recognition

General Terms

Algorithms, Experimentation

Keywords

Semi-supervised learning, bot detection, search engine, query logs, click logs, CAPTCHA

1. INTRODUCTION

Web search engines play an important role in satisfying users' information needs. However, due to the openness of web search engines and the profit potential in manipulating the search result pages, malicious use of the search engine has been widely observed [11, 30, 31, 45]. Specifically, robots[1], a type of programs that issue queries and clicks automatically to web search engines can consume a large portion of the overall traffic for the search engines. It is crucial to detect and separate these automatically generated search activities from those of genuine human users' for the following reasons. First, program generated traffic can usually peak to a large traffic volume in a very short period of time, causing an increase of the search engine response time that degrades the human user's experience. Second, search logs that record users' interactions with the search engines are often retained for later analysis. Search engine logs corrupted by bot activities can mislead and even cause serious problems when important conclusions are to be drawn from these logs [37]. Third, it is usually very important to preprocess the data logs and filter out irrelevant records [10] before carrying out other tasks such as modeling user search behaviors [1]. Specifically, one example is learning to rank by user clicks, in which case the ranking algorithms learned from polluted data will not be effective and useful [14, 28].

One difficulty in detecting bots in search engine is the diversity in their behaviors. There are bots that behave ethically by clearly identifying themselves in their visits [15, 21, 25]. However, based on our observations on a popular search engine, only a very small fraction of the bot traffics belong to this class. Other types of bots behave very differently. For example, some bots attempting to reverse engineer the search engine index would issue query terms extracted from a dictionary, i.e. consecutive queries that only differ by one or two characters, while some bots submit the same queries and sometimes click on the similar results in an attempt to boost the ranking for some specific keywords or search results. The diverse search behaviors are also mirrored by the genuine human users, one specific reason being different users exhibit various sophistication level in using the search engine [22].

Prior researches in automated web traffic detection mainly focused on detecting bots on websites. For example [40] used behavioral features, such as percentage of multimedia requests, average time between clicks, and total number of page requested to characterize the navigational patterns of users and then apply a decision tree algorithm to learn and determine if the user is a human or not. [39] used sim-

*Part of this work was done when the first author was on a summer internship with Microsoft Research and the Bing Search Data Mining Group.

WWW 2010, April 26–30, 2010, Raleigh, North Carolina, USA.
ACM 978-1-60558-799-8/10/04.

[1]In this paper, we use "robots" and "bots" interchangeably to refer to automated online programs.

ilar user behavioral features but formulated the problem under Bayesian classification framework that demonstrated promising results for detecting crawlers inside Web-server access logs. However, almost ironically, in these prior researches one major type of robots were the web crawlers from search engines, and work focused on the detection of automatically generated traffic targeted at search engines has been limited. [37] classifies search sessions into typical and atypical behaviors, and showed that by filtering out these atypical (outlier) sessions, one can improve the confidence in the click through rate (CTR) estimation. While it is natural that reducing data variance leads to higher confidence of parameter estimation, the classification task itself mixes genuine user behavior with robots' activities. Therefore it sheds limited light on detecting robot behaviors. Our work is largely inspired by and built upon [4] which were based on active learning framework, and [5] which used unlabeled data to help identify the samples that need to be labeled. Compared to these prior work, our contribution is a cost-efficient way of generating large number of initial samples and propose a semi-supervised learning framework that can improve the classification performance using unlabeled data in an iterative EM process.

In this paper, we propose a novel approach of combining the use of CAPTCHA [42] and some simple heuristics to generate large number of training data at essentially no additional labeling cost. Further, we propose a semi-supervised learning approach for the bot detection problem. Our semi-supervised learning approach is advantageous in handling the sampling bias issue during the initial training dataset generation. This is achieved by introducing a large number of unlabeled data from randomly sampling the whole dataset, again at no labeling cost. We demonstrate the effectiveness of our semi-supervised learning approach in distinguishing bot traffic from genuine human user traffic. Our comparison to a fully supervised learning algorithm shows that the semi-supervised learning approach performs significantly better under human judgments.

2. 0-COST TRAINING DATA GENERATION

For the scale of modern web search engines, it is costly to organize human judges to manually inspect and label the user search logs as training set for bot detection. What is practically possible is often a tiny fraction of one day's search log. However, since there are millions of online users, researchers have made successful attempts in utilizing these valuable resources, such as asking them to recognize scanned documents that are extremely challenging for current computer programs to handle [42] and, at the same time, use the process as a verification technique to determine if the user is a genuine human or some automated program.

In some systems, every user must pass the CAPTCHA challenge in order to access some specific service, such as web email account application, online banking, etc. For web search engines, this strategy is not viable because the goal of a web search engine is to help users retrieve useful information as quickly as possible, that one of the most important design objectives is to minimize the efforts a user spends on the search engine. For this reason, blindly applying CAPTCHA challenge for every user is apparently not acceptable. On the other hand, it is also necessary to send out CAPTCHA challenges to some users selectively to guard against malicious use.

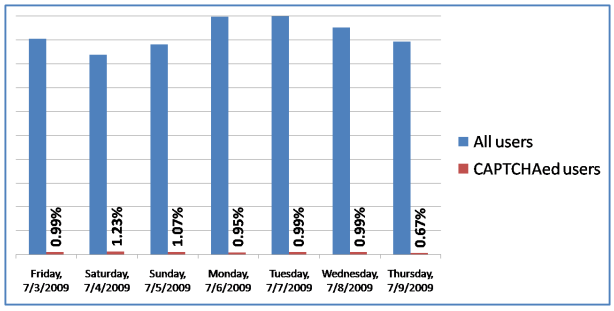

Figure 1: The number of all users and users who were present with CAPTCHA pages. Numbers of all users are normalized to the maximum number during the week. Numbers of CAPTCHAed users are normalized to each day respectively and percentages are shown here.

Our current search engine implements a mechanism to send CAPTCHA challenges to a small fraction of users [2] mainly based on the following criteria: 1), server load status, i.e. CAPTCHA challenges are sent out when some servers are experiencing high traffic volume; 2), user behavior, when a user, tracked by a unique identifier, is behaving abnormally, such as sending in large number of queries in a very short interval; 3), IP block, such as when the traffic from a certain IP address exceeds a high threshold that all users from this address will be asked to verify their identities; 4), random selection, especially when the server continues to experience heavy traffic volume; and 5), some other thresholds have been exceeded, such as those from the baseline algorithm described later in the paper. After correctly responding to the challenge, the user will be exempted from further verification for a period of time. We have found this mechanism a good balance between user experience and system stability.

Figure 1 shows the overall traffic and the percentage of users who were presented with the challenges in the data collected during the period of July 3 to July 9, 2009. The number of overall users fluctuates periodically on a weekly basis and peaks on Monday and Tuesday. On average, less than 1% of overall users are requested for verification.

When presented with a CAPTCHA challenge, a user can either disregard the challenge ("no response") by closing the browser or exiting the search session, or attempt to answer the challenge. Every time a wrong answer is submitted, the server changes the image shown in the verification web page. As long as the user answers correctly in one of the challenges, the user is regarded as having passed the challenge and labeled "correct response" . If the user fails to correctly answer the challenges after all trials, the user is marked as "wrong response". Figure 2 shows the response categorization of these users. It partially demonstrates the effectiveness of the verification mechanism since, among all

[2] A user is identified by a unique user id stored in the cookies. The algorithm discussed in this paper only handles the cases where users accept cookies. To further simplify the process, a user with the same user id is classified independently in different sessions. Here, a session is characterized as consecutive visits from the same user and can consist of several query actions taking place within a temporal interval, say, 30 minutes.

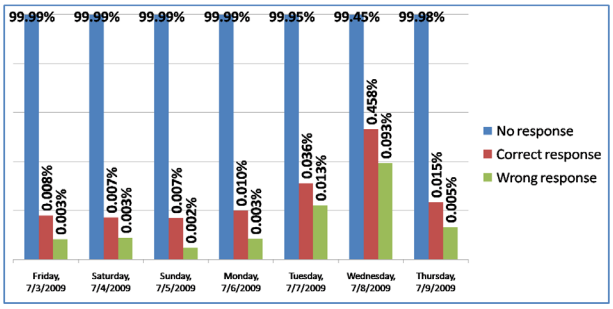

Figure 2: Categorize users based on their response to the CAPTCHA pages. Since the CAPTCHA users were sampled non-uniformly, most of them were likely to be robots, and therefore large portion of the CAPTCHA pages were not responded. For the users who do respond, the correct rate is about 80%.

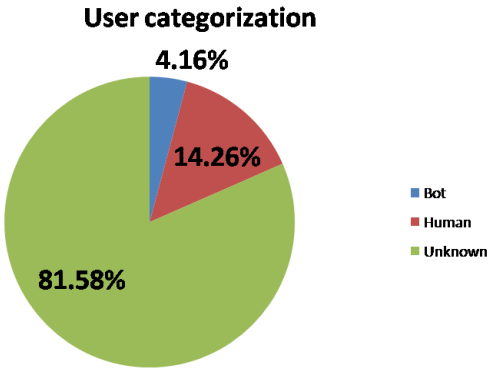

Figure 3: Categorize web search users based on the combination of their response to the CAPTCHA challenge and heuristics. Note the different user response rate on web search vertical compared to that in Figure 2 for all users.

the users who receive the verification requests, over 99.9% of them do not respond at all. This lends to the belief that most current bots do not implement the functionality to respond to the CAPTCHA challenges, let alone the sophisticated algorithms needed to overcome them. For users who do respond to the verification requests, we see that the correct response rate is roughly 80%. The rate depends on many factors, such as the difficulty of different CAPTCHA challenges and the user expertise levels in online usage [22] that make some users more experienced in solving the challenges than others.

For users who correctly answer the challenges, we label them as genuine human users and use their records as the "human" class in our training dataset. Even though we believe that the majority of users who do not respond to the challenges are bots, it is still possible that some human users are turned off by the challenge or simply do not understand how to respond. Therefore, we use some heuristics to select a fraction that are most likely to be bots from the set of "no response" users. The set of heuristics include 1), number of clicks in a time period, 2), number of search result pages browsed, and 3), number of IPs that the user "originates" simultaneously. The definition of these measurement will be further discussed in Section 3. The users who do not respond and exceed the thresholds of the heuristic measurements are initially labeled as "bot".

From the data, we observe that user behaviors appear to be multi-modal and vary notably among the search verticals such as web search, image search, video search, etc. In this paper, we restrict our discussion to web search only. Figure 3 shows the categorization of web search users based on the combination of their responses to the CAPTCHA challenges and our heuristic based classification. The different response rates compared to the overall response rate are shown in Figure 2. Note that in these datasets we use only the most suspicious subset of the users who do not respond as "bot" samples.

By making use of the existing CAPTCHA mechanism and some simple heuristics, our training data generation process has essentially incurred "0-cost" for the developers to extract a large number of labeled data. However, this set of data is not uniformly sampled over the whole dataset and has the following issues to use them directly for supervised training. First, as we discussed before, the users who were selected to present the CAPTCHA challenge are sampled unevenly towards those who have large number of requests. Second, the users who correctly answer the challenges are biased towards the users with more online experience. Third, the samples in the "bot" class is only a partial representative of the whole "bot" class due to the high thresholds in our heuristic rules, and therefore only a small fraction of the users who do not respond are used as the "bot" samples. It is well known that when the labeled training dataset does not reflect the underlying data distribution, the classifiers will have skewed classification boundaries that will lead to poor generalization capability [3, 6, 46, 47].

Obviously, the benefit of generating a large amount of data in a cost efficient way must be accompanied by the data being useful for training. To compensate the data bias issues, we further include a large amount of unlabeled data by uniformly sampling from the whole search logs and use a semi-supervised learning approach to correct the skewed decision boundary. We describe the details of the semi-supervised learning algorithm that we developed from Bayes network [18] (Section 4), and then we demonstrate its effectiveness by comparing the proposal with a supervised learning algorithm based on decision tree [32] (Section 5).

It is important to note that bot detection and the user verification mechanism are closely related. A good bot detection system will make the user verification mechanism more efficient and improve the experience of the genuine human users. On the other hand, selectively verify the identity of certain users can also help a bot detection algorithm learn to better distinguish human user behaviors from those automatically generated by programs. Due to the space limit, this intricacies between these two issues are left out of this article.

3. FEATURE DESCRIPTION

Here we briefly describe the features we use in the bot detection algorithms. Generally speaking, the features can be categorized into two types. First, the numerical user behavior features are the type of summarized measurements

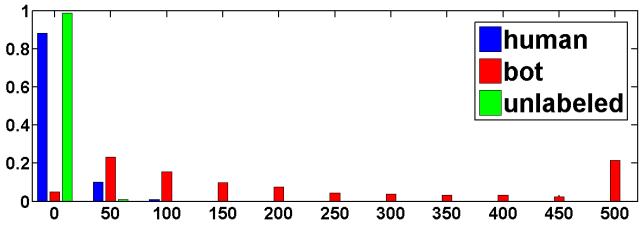

Figure 4: PageTrackedCount (page views).

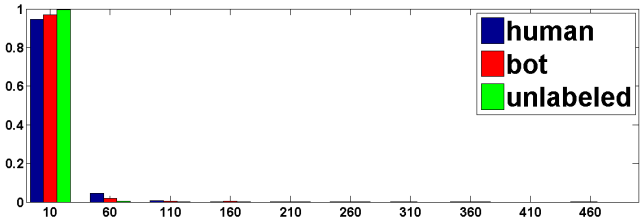

Figure 5: UserClickCount.

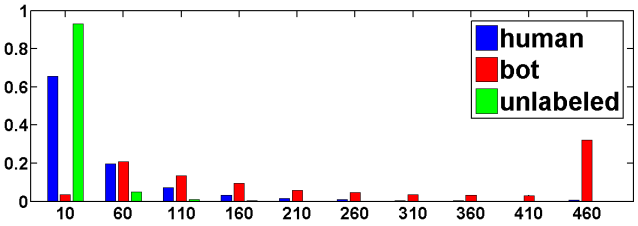

Figure 6: AllHitCount

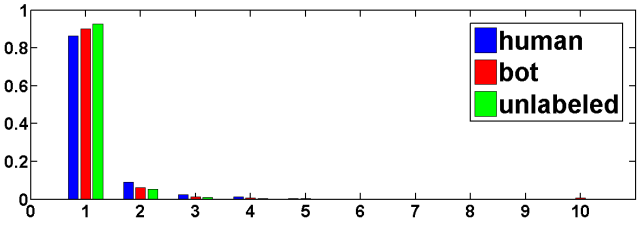

Figure 7: UserUniqueIPs.

derived from the whole, per-user search sessions. The second type of features is the boolean blacklist features that are basically hand crafted rules. The blacklist features are very powerful in identifying bots when triggered, although the frequency of these features being triggered is low in practice. In the following we describe these different types of features and, for clarity, mark their value types in parentheses.

3.1 PageTrackedCount (numeric)

PageTrackedCount measures the number of pages that the user browses. Empirical observations lead to the impression that bots tend to behave in two extremes. Some bots will only submit queries and not browse any of the result pages (except the first one), ostensibly with the intention to increase the query frequency for certain keywords. The other extreme sees the bots fetch all the result pages for each query, probably trying to reverse engineer the index of the search engine, while genuine human users would probably just browse the first few pages of the query results selectively.

Figure 4 shows the *PageTrackedCount* distributions of the different classes. We can see that both "human" and "bot" classes are slightly skewed toward higher *PageTrackedCount* due to the reason we described before.

3.2 UserClickCount (numeric)

UserClickCount measures the number of mouse clicks on the search result pages. This includes clicks on the web search results, i.e. results that are deemed relevant to the query term, and the sponsored items (i.e., advertisements). At this point we do not distinguish these two items and sum the clicks into a single number, although counting them separately can be potentially useful in capturing the ad-fraud bots that intentionally click on advertisements placed on the search result pages. Figure 5 shows the *UserClickCount* distributions of the different user classes.

3.3 AllHitCount (numeric)

AllHitCount measures the overall "impressions" that the user receives in addition to the search results. Since on a web search engine the major contributor to this feature is the page views, this feature (Figure 6) is closely correlated to the *PageTrackedCount* depicted in Figure 4. However, the correlation is weaker for image and video search where the impressions of the media contents are not necessarily bounded by the search result pages.

3.4 UserUniqueIPs (numeric)

UserUniqueIPs measures the unique number of IPs a user is using. [11] and [45] have both reported that a large number of bots can assemble a network to attack online services in a well coordinated manner, and one way to discover these attacks is by counting the number of unique IPs that are associated with each user. Although the IP address of a user could change when the user moves from one place to another, e.g. home v.s. office, the frequency of the IP changes is typically much smaller than that of a bot net. In our experiment we did not discover these malicious networked activities in our search logs, thus the "human" and "bot" classes are less distinguishable on this dimension than others (Figure 7).

3.5 UserUniqueQueries (numeric)

UserUniqueQueries measures the unique number of queries issued by a single user in a search session. There are two key observations from inspecting the distribution shown in Figure 8: 1), "bots" tend to issue either a very small number of repeated queries or a large number of unique ones, and 2), the users in the "human" class tend to have more unique queries than the overall user population. One possibility is that the users labeled as "human" might be more experienced users that are more versed in formulating queries.

3.6 Rules (boolean)

We implement the following rules (boolean) in our current bot detection algorithm: 1) *RuleBlacklistForm*: this rule is triggered when a user includes in the query certain obscure

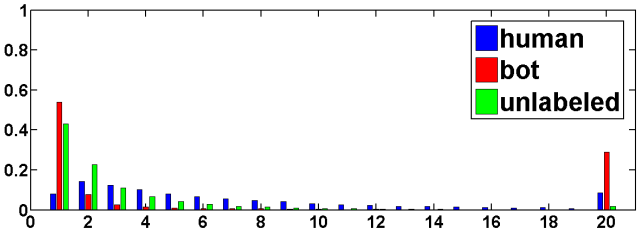

Figure 8: UserUniqueQueries

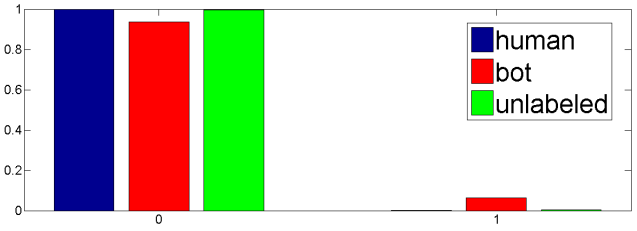

Figure 9: Rules

codes that are designed mostly for internal search engine instrumentation purposes that should be unfamiliar to most genuine human users. 2) *RuleBlacklistIp*: we maintain a list of IPs that are publicly identified as Internet spammers and labeled all the traffic from these IPs as "bot". 3) *RuleBlacklistQuery*: this rule is triggered when the query composition is too complicated to be manually typed in by a human user. In this work, we combine these rules into a single feature (Figure 9) that assumes the value "1" whenever one of the rules is triggered and "0" otherwise.

4. BOT DETECTION ALGORITHMS

4.1 Supervised learning for bot detection

A straightforward way to utilize the labeled data extracted using the techniques described above is to directly learn a bot detector in a fully supervised manner. In our experiment, we choose a specific implementation ($J48$ [43]) of the popular $C4.5$ algorithm [32] as our supervised learning algorithm for bot detection. The details of the decision tree algorithm are omitted here.

Since our initial labeled dataset is extracted using CAPTCHA response and heuristics, the decision tree algorithm can fit very well the labeled training dataset. However, as clearly seen in Figure 4 through 9, the labeled samples we extracted are indeed skewed from the true data distribution, bringing in the question that how well the classification boundaries learned directly from this subset will be able to generalize.

4.2 Semi-supervised learning for bot detection

Indeed, our problem actually matches the strengths of the semi-supervised learning framework [3, 6, 46, 47], in which unlabeled data are included in the classification boundary learning process and the optimization of the learning process is carried out with both labeled and unlabeled data samples. The advantages of applying semi-supervised learning to the bot detection problem are based on the following properties commonly observed in a semi-supervised learning system.

First, compared to fully supervised learning, very few labeled data samples are needed in order to reach comparable performances. This property is especially appealing because, as we have seen, web search users that are presented with the CAPTCHA challenges are only a very small fraction ($\leq 1\%$) of the whole user set, and among these users only a very small fraction of them will respond to the challenges (Figure 2 and Figure 3). Therefore, our labeled data are a very sparse sample of the original data corpus that seem to be more suitable with a semi-supervised than with a fully supervised learning approach.

Second, semi-supervised learning is especially advisable when the labeled data samples' distribution are skewed from the underlying data distribution, e.g. Figure 4 - Figure 9. According to the Bayes rule, $P(Y|X) = \frac{P(Y,X)}{P(X)}$. If the training data distribution, $P(X')$, used in the supervised learning algorithm diverges from the actual data distribution $P(X)$, the learned classification model $P(Y|X')$ will not generalize well over X [6, 46]. Again, our dataset is generated with known biases that make it appealing to avoid supervised learning.

A common way to incorporate the unlabeled data into the learning process is through the Expectation-Maximization (EM) algorithm[12]. The algorithm starts by using the labeled data (X^L) only to bootstrap an initial classifier $P(Y|X^L)$. This initial classifier is then applied to annotate the unlabeled data (X^U) with the posterior probabilities, $P(Y|X^U)$, for each unlabeled observation Y. These probabilistically labeled data are then added to the training dataset, and their posterior probabilities were used as "soft" counts for the purpose of updating the conditional probabilities in the EM iterations. At the end of each iteration, a new classifier, $P(Y|X)$, is obtained and this new classifier is then applied to annotate the unlabeled data. The process is repeated until certain convergence criteria are met. This approach is also known as a self-training algorithm since the learning algorithm is feeding back on its own classification outcomes [35, 36, 44]. A temporal version of this method leads to the Baum-Welsh algorithm for Hidden Markov Model (HMM) training [33].

When the feature dimension is sufficiently large, co-training algorithm [3] splits the features into two subsets, assuming that each subset is distinctive enough to train a good classifier independently. In the co-training algorithm, the labeled data are first used to train each classifier. Afterwards, unlabeled data are classified by each classifier and the samples with high confidence from one classifier are added to the training dataset of the other classifier. Each classifier is trained again with the new training dataset and this process is repeated iteratively. Two assumptions are required for this co-training algorithm to perform well: first, each sub-feature set is sufficiently distinctive and, second, the conditional independence assumption is required for these two feature sets [29]. Relaxations to these strong independence assumptions were explored in [7, 19].

In addition to the generative models, semi-supervised learning framework has also been enhanced with discriminative methods that optimize $P(Y|X)$ directly without explicitly model the data generation process, i.e. $P(X|Y)$ and $P(X,Y)$. For the unlabeled data to be useful in the discriminative approach, a connection between $P(X)$ and $P(Y|X)$ has to be made [38]. Transductive support vector machines (TSVMs) accomplishes this by requiring that the decision boundary

be away from high density areas with large $P(X)$ [2, 17, 23]. It maximizes the margin on both the labeled data and the unlabeled data, and as a result the decision boundary minimizes the generalization error on unlabeled data [41].

Our work is mostly related to the study that only one class's label is available (in our case the users who pass the CAPTCHA challenge are known to be "human"). [13] assumes that the prior distribution $P(Y)$ is known and showed that it is theoretically possible to estimate $P(Y|X)$ using Bayes rule. [27] address this problem using two rounds of the EM algorithm. In the first round, the labeled data samples are split into two parts unevenly, with the larger split being used on its own, while the smaller split, called the *spy documents*, is mixed with the unlabeled samples to form the opposing class of data. A classifier is learned using the EM algorithm. The posterior probability of the spy documents are considered as a relative standard to determine which data samples have high probability to belong to the other class, after which these data samples and the initial labeled samples are used to train a final classifier through a second EM algorithm. [26] adopts a more aggressive scheme by first treating all unlabeled data as the opposing class to the labeled samples, and using a weighted logistic regression algorithm to learn a linear classification function. As a result, this approach is not suitable for cases where the opposing classes are not linearly separable, or when the unlabeled samples overwhelmingly outnumber the labeled samples.

In this paper we develop a semi-supervised learning algorithm using Bayesian network classifier that has various advantages in handling incomplete data set, encoding causal relationship, and avoiding over-fitting [20]. The training of a Bayesian network is composed of two parts, i.e. the structure learning and the parameter estimation.

4.2.1 Bayesian network structure learning

Given a set of training samples, the goal of structure learning is to generate a graph that best describes the causal relationships in the data. Here, the goodness of the graph structure can be measured in many ways, ranging from the entropy, the posterior probability given the training data, to the minimum description length [34]. Regardless the metric chosen, the structure learning problem amounts to defining a search algorithm that optimizes for the metric by systematically changing the graph structure.

Unfortunately, even for fully labeled dataset, learning the optimal Bayesian network structure is NP-hard [8] that approximation and heuristics are often adopted in practice [9, 16]. In this work, we learn the Bayesian structure using the labeled data only. In addition to the computational complexity concern, it is also because, during the dataset generation process, we heavily utilizes the domain knowledge that is representative of the nature of the problem that it is reasonable to believe the the structure learned from the labeled data can generalize well. We use a structure learning algorithm similar to that of [16] in which a tree structure is formed by calculating the maximum weight spanning tree using the methods described in [9].

4.2.2 Semi-supervised Bayesian network parameter estimation

After acquiring the Bayesian network structure, we learn the conditional probabilities for each node and its parents. We represent each feature (X_i) and the class label (Y) as nodes in the Bayesian network, and the task is to learn the following posterior probability for any given observation [24]:

$$P(Y|X_0, X_1, ..., X_n) = \frac{P(X_0, X_1, ..., X_n, Y)}{\sum_Y P(X_0, X_1, ..., X_n, Y)}. \quad (1)$$

As can be seen in the above equation, the problem of learning the conditional probabilities is equivalent to learning the joint probability of $P(X_0, X_1, ..., X_n, Y)$. Based on the local Markovian assumption and the chain rule of probability, this joint probability can be factorized as

$$P(X_0, X_1, ..., X_n, Y) = \prod_{i=0}^{n} P(X_i|Pa_{X_i}), \quad (2)$$

where Pa_{X_i} represents all the parents nodes of X_i. With this factorization, the parameter learning is further simplified as the process of estimating the conditional probability of a random variable X_i given its parents Pa_{x_i}. Without loss of generality, we explain our parameter learning algorithm in the special case that variable X_i has only one parent node Y (i.e., Naïve Bayes). The terminologies and notations used in the algorithm are listed in Table 1 and, whenever appropriate, we shorthand $P(X = x, Y = y)$ as $P(X, Y)$ for simplicity in notation.

Table 1: Terminologies and notations used in semi-supervised Bayesian network parameter learning

Description	Notation
Observation (features)	X
Class label/prediction	$Y = \begin{cases} 0 & : \text{“}human\text{”} \\ 1 & : \text{“}bot\text{”} \end{cases}$
Bayesian classifier posterior	$P(Y\|X)$
A data sample	$d(X, Y)$
Human data	$\mathbf{H}$
Bot data	$\mathbf{B}$
Unlabeled data	$\mathbf{U}$
Training data corpus	$\mathbf{D} = \mathbf{H} \cup \mathbf{B} \cup \mathbf{U} = \cup d(X, Y)$, $\mathbf{H} \cap \mathbf{B} = \mathbf{0}$, $\mathbf{H} \cap \mathbf{U} = \mathbf{0}$, $\mathbf{B} \cap \mathbf{U} = \mathbf{0}$
Weight for a data sample	$w_d^i \in [0, 1]$, $i \in \{0, 1\}$
Weighted count of a sample with observation x and label y	$\#(X = x, Y = y)$
CAPTCHA trust factor	C
Laplacian smoothing factor	α
Total number of data samples	N
Maximum EM iterations	T

At the core of the parameter learning is the iterative EM algorithm, summarized in Algorithm 1, in which the posterior probabilities of the data samples can be updated using the weights learned in the preceding iteration:

$$P(Y|X) = \frac{P(X, Y)}{P(X)} = \frac{P(X, Y)}{\sum_Y P(X, Y)}, \quad (3)$$

$$P(X, Y) = \frac{\#(X, Y) + \alpha}{N + \alpha}. \quad (4)$$

Algorithm 1: Expectation-Maximization algorithm for semi-supervised Bayesian network parameter learning

Data: $\mathbf{D}(X, Y) = \mathbf{H} \cup \mathbf{B} \cup \mathbf{U}$
Result: $P(Y|X)$
begin
  Initialization:
  Set $\#(X, Y) = 0$;
  for *each* $d \in \mathbf{D}$ **do**
    if $d \in \mathbf{H}$ **then**
      Set $w_d^0 = 1$, $w_d^1 = 0$,
    else if $d \in \mathbf{B}$ **then**
      Set $w_d^0 = 0$, $w_d^1 = 1$,
    else if $d \in \mathbf{U}$ **then**
      Set $w_d^0 = 0$, $w_d^1 = 0$,
  $i = 0$;
  Maximization: learn Bayesian classifier $P^0(Y|X)$ as Equation (3) − (6);
  $i++$;
  while $i < T$ **do**
    Expectation: Update weights
    for *each* $d \in \mathbf{D}$ **do**
      if $d \in \mathbf{B}$ **or** $d \in \mathbf{U}$ **then**
        Set $w_d^0 = P^{i-1}(Y = 0|X)$;
        Set $w_d^1 = 1 - P^{i-1}(Y = 0|X)$;
      else if $d \in \mathbf{H}$ **then**
        Set $w_d^0 = (P^{i-1}(Y = 0|X))^{\frac{1}{C}}$;
        Set $w_d^1 = 1 - (P^{i-1}(Y = 0|X))^{\frac{1}{C}}$;
      Maximization: learn Bayesian classifier $P^i(Y|X)$ as Equation (3) − (6);
      $i++$;
  Return $P(Y|X)$;
end

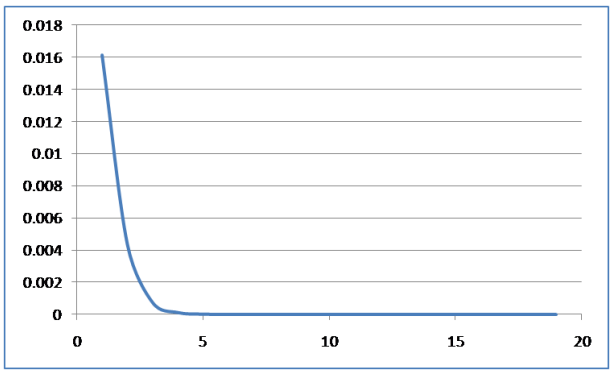

Figure 10: Averaged change of posterior probabilities after each EM iteration.

Combining (3) and (4) we have

$$P(Y|X) = \frac{\#(X, Y) + \alpha}{\sum_Y (\#(X, Y) + \alpha)}, \tag{5}$$

where the weighted count is defined as

$$\#(X = x, Y = y) = \sum_{\substack{d \in \mathbf{D} \\ X = x \\ Y = y}} w_d^{i=y}. \tag{6}$$

Our semi-supervised learning algorithm is slightly different from many other algorithm in that we introduce a CAPTCHA trust factor (C) in the parameter learning process. As in Algorithm 1, the soft count of each data sample is updated based on its previous label. If a sample is initially labeled as "bot" or "unlabeled", we update its soft count with the posterior probability learned from the previous iteration. If the sample is initially labeled as "human", i.e. the user has correctly answered the CAPTCHA challenge, its soft count is the posterior discounted by a CAPTCHA trust factor C. This factor controls to which degree we trust the CAPTCHA system. Even though at this moment no computer program could automatically solve the CAPTCHA challenge problem, it is possible that one can make use of manual labors to cheat CAPTCHA. We use a large trust factor C with a belief that the CAPTCHA system is reliable in most cases. However, when some samples have a very high posterior probability to be "bots", soft counts can still be drawn from these samples and added to the "bots" class. Note that $C \to \infty$ means that we absolutely trust the CAPTCHA system and therefore the identity of the "human" users can not be changed in the EM training process. We tested various C values and found that 10 seems to be a good balance. Figure 11 shows the effect of different C values on the performance. Our current EM algorithm stops after a fixed number of iterations (e.g. 10). Figure 10 shows the average change of posterior probability for the training dataset, which can also serve as a convergence criteria.

5. EXPERIMENT

Our training data is composed of a week's web search log data on a popular search engine from July 03 to July 09, 2009. The initial labeled dataset is generated through the process described in Section 2. For computational considerations, we further sample both the "human" and "bot" dataset from the log, and finally extract 20000 "human" records and 6000 "bot" records that keep the original ratio unchanged. For the semi-supervised learning algorithm, we uniformly sample the whole week's logs to generate an "unlabeled" dataset. The final labeled/unlabeled dataset size ratio is, 1 : 9. The feature distributions of the three different set are shown in Figure 4 - Figure 9.

As for testing, we generated a smaller dataset composed of 170 user sessions. In order to be fair for all algorithms, we invited 10 web search researchers to label the full testing dataset so that each search session is judged by 3 different persons. We developed a visualization tool to render the search log and display a user's search session in its original temporal order. The visualized information includes 1) numeric/boolean features as defined in Section 3; 2) the first 100 raw query terms; 3) referrer's link; 4) clicked links; 5)

Table 2: Testing dataset human judgment

Description	Numbers
Total number of user sessions	170
Number of user sessions with at least 2 judges reach agreement	111
Number of user sessions with at least 2 judges agreed on either "human" or "bot" labels	102 "human": 71 "bot": 31
Number of user sessions with at least 2 judges agreed on "uncertain" labels	9
Agreement percentage	$111/170 \approx 65\%$
"for sure" percentage (i.e. either "human" or "bot")	$102/170 \approx 60\%$

Table 3: Performance comparison for the supervised learning algorithm (D-Tree) and our proposed semi-supervised learning approach

Algorithm	Precision	Recall	F-measure
D-Tree	28%	39%	35%
Semi-supervised Bayesian network	75%	72%	73%

dwell time. For each session, a judge could choose among the three different labels: "human", "bot", or "uncertain".

The statistics of the labels in the test set are summarized in Table 2. Note that even for human, the problem of distinguishing bot from a genuine human user seems extremely difficult[3]. Among all the 170 sessions, only about 60% of them that an agreement is reached by a majority of 2 over 3 judges. This subset on which we have higher confidence with the judgment labels and in total has 71 "human" sessions and 31 "bot" sessions, is used as the test set to evaluate various bot detection algorithms.

In our experiment, we use the $J48$ implementation of [32, 43] as our supervised learning algorithm. All the parameters are set to their default values.

For the semi-supervised learning approach, we developed our own algorithm based on the implementation of [34]. We first used the labeled data to learn the Bayesian network structure using the built-in Tree Augmented Naïve Bayes structure learning algorithm, and then kept this graph structure fixed for the EM algorithm to learn the conditional probability tables (CPT) for each Bayesian network node. We used standard Laplacian smoothing in CPT estimation, with smoothing factor $\alpha = 1$ throughout the experiment. The inference process used the built-in inference algorithm [34]. To pre-process the original numerical features, we used the built-in supervised quantization algorithm [43]. More specifically, we used the labeled dataset to learn the quantization bins for each feature dimension, and then applied this binning scheme to the unlabeled dataset. Our experiments were conducted on an Intel Xeon workstation with $8GB$ of memory. The structure learning from 26000 labeled

[3]Because we are using user id to track users, it requires the user to enable cookies. In reality, there are a large number of bots that do not accept cookies and are not considered here. As for the bots that do accept cookie, they tend to be more sophisticated and better disguised, which contributes to the difficulty of the detection problem.

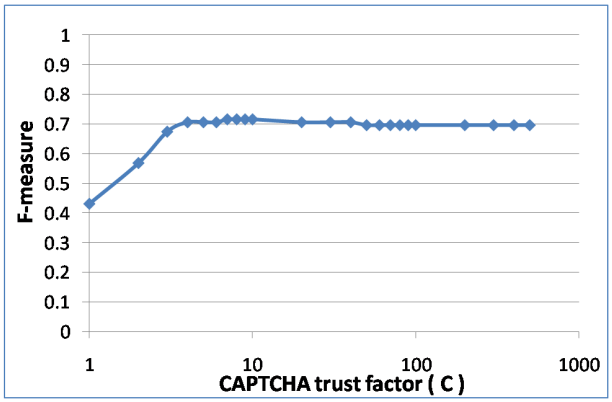

Figure 11: The effect of different CAPTCHA trust factors on the performance, measured by F-measures.

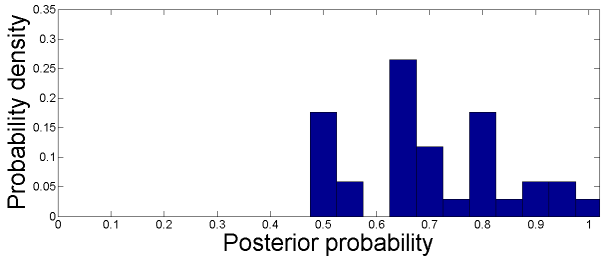

Figure 12: Distribution of the posterior probability for the misclassified samples in our approach.

data samples took about 3 minutes. For the parameter estimation, we fixed the maximum EM iteration to 10, and the training on around 200 thousand samples took about $3-5$ minutes on average.

The performance for each algorithm is summarized in Table 3, where the precision and recall are averages of the two classes. Because the decision tree algorithm was trained with skewed training dataset, its performance on the testing set was far from comparable to the semi-supervised learning algorithm we proposed in this work. For most of the performance measures, the semi-supervised learning approach performs over 2 times better than the supervised learning approach. Figure 12 shows the distribution of the posterior probabilities of the mis-classified samples in the semi-supervised learning approach. Note that a large portion of the errors are made close to the classification boundary. Manual inspection over the mis-classified examples shows that the errors mostly fall under the following two cases, 1), when the user session is short, there is very little information that the algorithm can infer from; 2), there are some cases that a user issues a lot of queries and the algorithm tends to classify such a behavior as a bot. However, human judges can tell these sessions are from genuine human users because "the queries were semantically related and the queries and clicks were coherent with each other".

6. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a semi-supervised learning framework for detecting automatically generated web search traf-

fic. One important issue for applying machine learning algorithms to this type of Internet scenarios is data annotation scalability, to which we propose a *0-cost* approach that makes use of the existing CAPTCHA technique to extract data logs of genuine human users. The advantage of this approach is that it is virtually cost free to harvest large amount of human annotations through the existing user verification mechanism. We also propose using the simple heuristics from domain experts to generate the initial positive ("bot") dataset. We address the skewed sample issue of the proposed approach and formulate the problem under semi-supervised learning framework. The proposed approach makes use of a large number of unlabeled data, evenly sampled from the whole dataset (which again is virtually 0-cost). Compared to the supervised learning algorithms that only base on the labeled dataset, our semi-supervised learning approach performs significantly better in our evaluation ($2:1$).

Many aspects in the proposed approach can be further improved. First, the feature set used in our current algorithm seems to be not distinctive enough for this problem. For example, it may be helpful to learn users' search intention in order to classify if the user is a genuine human or a bot as most genuine users typically use search engines when they have information needs. This could be done by further including the query content and the correlation between query and clicks into the feature set . In essence, our experiment is still preliminary because we use only one week's data that might not be long enough to capture some bot activities. It is therefore not clear if our model can generalize well over a longer time span. Ideally, the learning approach should be adaptive in nature so that the system performance can be automatically improved as more and more new data are collected. In its current form, human judgments are involved only for testing data annotation, but it is straightforward to extend the judgment efforts towards the active learning approach so that new ambivalent data, when detected, can be quickly added into the adaptation data to broaden the scope and enrich the capabilities of the system. As mentioned before, the CAPTCHA technique and our bot detection system could be better combined.

7. ACKNOWLEDGMENTS

The authors would like to thank anonymous reviewers for helpful suggestions. The authors are also grateful to the colleagues at Microsoft who generously offered help in the user judgment study. Hongwen Kang thanks Christos Faloutsos, Nikolas Gloy and Jay Stokes for helpful discussions.

8. REFERENCES

[1] R. A. Baeza-Yates, C. A. Hurtado, M. Mendoza, and G. Dupret. Modeling user search behavior. In *LA-WEB*, pages 242–251. IEEE Computer Society, 2005.

[2] K. P. Bennett and A. Demiriz. Semi-supervised support vector machines. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 368–374, Cambridge, MA, USA, 1999. MIT Press.

[3] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT' 98: Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, New York, NY, USA, 1998. ACM.

[4] G. Buehrer, J. W. Stokes, and K. Chellapilla. A large-scale study of automated web search traffic. In *AIRWeb '08: Proceedings of the 4th international workshop on Adversarial information retrieval on the web*, pages 1–8, New York, NY, USA, 2008. ACM.

[5] G. Buehrer, J. W. Stokes, K. Chellapilla, and J. C. Platt. Classification of automated search traffic. In I. King and R. A. Baeza-Yates, editors, *Weaving Services and People on the World Wide Web*, pages 3–26. Springer, 2009.

[6] O. Chapelle, B. Schölkopf, and A. Zien. *Semi-Supervised Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2006.

[7] N. V. Chawla and G. J. Karakoulas. Learning from labeled and unlabeled data: An empirical study across techniques and domains. *J. Artif. Intell. Res. (JAIR)*, 23:331–366, 2005.

[8] D. Chickering, D. Geiger, and D. Heckerman. Learning bayesian networks is np-hard. Technical report, Microsoft Research, 1994.

[9] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968.

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

[11] N. Daswani and M. Stoppelman. The anatomy of clickbot.a. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 11–11, Berkeley, CA, USA, 2007. USENIX Association.

[12] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[13] F. Denis, A. Laurent, R. Gilleron, and M. Tommasi. Text classification and co-training from positive and unlabeled examples. In *Proceedings of the ICML 2003 Workshop: The Continuum from Labeled to Unlabeled Data*, pages 80–87, 2003.

[14] Z. Dou, R. Song, X. Yuan, and J.-R. Wen. Are click-through data adequate for learning web search rankings? In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 73–82, New York, NY, USA, 2008. ACM.

[15] D. Eichmann. Ethical web agents. *Comput. Netw. ISDN Syst.*, 28(1-2):127–136, 1995.

[16] N. Friedman, D. Geiger, M. Goldszmidt, G. Provan, P. Langley, and P. Smyth. Bayesian network classifiers. In *Machine Learning*, pages 131–163, 1997.

[17] G. Fung and O. Mangasarian. Semi-supervised support vector machines for unlabeled data classification, 2001.

[18] Z. Ghahramani. An introduction to hidden markov models and bayesian networks. pages 9–42, 2002.

[19] S. A. Goldman and Y. Zhou. Enhancing supervised learning with unlabeled data. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, pages 327–334, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[20] D. Heckerman. A tutorial on learning with bayesian networks. pages 301–354, 1999.

[21] O. Heinonen, K. Hätönen, and M. Klemettinen. WWW robots and search engines. In K. Oksanen, editor, *Seminar on Mobile Code*, Technical Report TKO-C79. Helsinki University of Technology, Department of Computer Science, May 1996.

[22] C. Hölscher and G. Strube. Web search behavior of internet experts and newbies. In *Proceedings of the 9th international World Wide Web conference on Computer networks : the international journal of computer and telecommunications netowrking*, pages 337–346, Amsterdam, The Netherlands, The Netherlands, 2000. North-Holland Publishing Co.

[23] T. Joachims. Transductive inference for text classification using support vector machines. In *ICML '99: Proceedings of the Sixteenth International Conference on Machine Learning*, pages 200–209, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[24] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[25] M. Koster. Robots in the web: threat or treat ? *ConneXions*, 9(4), April 1995.

[26] W. S. Lee and B. Liu. Learning with positive and unlabeled examples using weighted logistic regression. In T. Fawcett and N. Mishra, editors, *ICML*, pages 448–455. AAAI Press, 2003.

[27] B. Liu, W. S. Lee, P. S. Yu, and X. Li. Partially supervised classification of text documents. In *ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning*, pages 387–394, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[28] T.-Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *LR4IR 2007, in conjunction with SIGIR 2007*, 2007.

[29] K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *CIKM '00: Proceedings of the ninth international conference on Information and knowledge management*, pages 86–93, New York, NY, USA, 2000. ACM.

[30] N. Provos. The reason behind "we're sorry ..." message. `http://googleonlinesecurity.blogspot.com/2007/07/reason-behind-were-sorry-message.html`, July 2007.

[31] N. Provos, J. McClain, and K. Wang. Search worms. In *WORM '06: Proceedings of the 4th ACM workshop on Recurring malcode*, pages 1–8, New York, NY, USA, 2006. ACM.

[32] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[33] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. pages 267–296, 1990.

[34] R. B. Remco. Bayesian network classifiers in weka. Technical report, University of Waikato, 2004.

[35] E. Riloff, J. Wiebe, and T. Wilson. Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pages 25–32, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[36] C. Rosenberg, M. Hebert, and H. Schneiderman. Semi-supervised self-training of object detection models. In *Seventh IEEE Workshop on Applications of Computer Vision*, January 2005.

[37] N. Sadagopan and J. Li. Characterizing typical and atypical user sessions in clickstreams. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 885–894, New York, NY, USA, 2008. ACM.

[38] M. Seeger. Learning with labeled and unlabeled data. Technical report, University of Edinburgh, 2001.

[39] A. Stassopoulou and M. D. Dikaiakos. Web robot detection: A probabilistic reasoning approach. *Comput. Netw.*, 53(3):265–278, 2009.

[40] P.-N. Tan and V. Kumar. Discovery of web robot sessions based on their navigational patterns. *Data Min. Knowl. Discov.*, 6(1):9–35, 2002.

[41] V. N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, September 1998.

[42] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. Captcha: Using hard ai problems for security. In *In Proceedings of Eurocrypt*, volume 2656, pages 294–311, 2003.

[43] I. H. Witten and E. Frank. Data mining: practical machine learning tools and techniques with java implementations. *SIGMOD Rec.*, 31(1):76–77, 2002.

[44] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196, Morristown, NJ, USA, 1995. Association for Computational Linguistics.

[45] Y. Zhao, Y. Xie, F. Yu, Q. Ke, Y. Yu, Y. Chen, and E. Gillum. Botgraph: large scale spamming botnet detection. In *NSDI'09: Proceedings of the 6th USENIX symposium on Networked systems design and implementation*, pages 321–334, Berkeley, CA, USA, 2009. USENIX Association.

[46] X. Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005.

[47] X. Zhu, J. Lafferty, and Z. Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML 2003 workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, pages 58–65, 2003.