Detecting spam blogs from blog search results

Linhong Zhu\textsuperscript{a}, Aixin Sun\textsuperscript{a,*}, Byron Choi\textsuperscript{b}

\textsuperscript{a}School of Computer Engineering, Nanyang Technological University, Singapore
\textsuperscript{b}Department of Computer Science, Hong Kong Baptist University, Hong Kong

\textbf{Abstract}

Blogging has been an emerging media for people to express themselves. However, the presence of spam blogs (also known as splogs) may reduce the value of blogs and blog search engines. Hence, splog detection has recently attracted much attention from research. Most existing works on splog detection identify splogs using their content/link features and target on spam filters protecting blog search engines' index from spam. In this paper, we propose a splog detection framework by monitoring the on-line search results. The novelty of our splog detection is that our detection capitalizes on the results returned by search engines. The proposed method therefore is particularly useful in detecting those splogs that have successfully slipped through the spam filters that are also actively generating spam-posts. More specifically, our method monitors the top-ranked results of a sequence of temporally-ordered queries and detects splogs based on blogs' temporal behavior. The temporal behavior of a blog is maintained in a blog profile. Given blog profiles, splog detecting functions have been proposed and evaluated using real data collected from a popular blog search engine. Our experiments have demonstrated that splogs could be detected with high accuracy. The proposed method can be implemented on top of any existing blog search engine without intrusion to the latter.

\textbf{1. Introduction}

\textbf{1.1. Motivation}

Blogging has now become a popular media for people to express themselves, share information, and communicate among each other. According to Technorati,\textsuperscript{1} a popular commercial blog search engine, the number of blogs doubles about every six months. Such an exponential growth of blogs could be partly explained by a few factors, such as the ease of publishing and the free hosting services (e.g., blogspot.com, livejournal.com and others). However, all these factors have also greatly contributed to a cost reduction of creating and maintaining spam blogs (or splogs). Formally defined in Lin, Sundaram, Chi, Tatemura, and Tseng (2008), a splog is “a blog created for any deliberate action that is meant to trigger an unjustifiably favorable relevance or importance, considering the blog’s true value”. In another words, splogs are often purposely created to attract traffic from blog search engines so as to promote themselves or their affiliates. As the presence of a large number of splogs would greatly reduce the quality of search results and waste network resources, splog detection has recently attracted much attention from research.

Three types of splog detection methods have been proposed in literature (see Section 7 for a more detailed discussion). Content-based detection methods aim to identify features which could distinguish splogs from authentic blogs, such as...
Kolari, Java, Finin, Oates, and Joshi (2006), Lin et al. (2008), Mishne and Carmel (2005). These features are mostly derived from snapshots of either the home page or all published posts of a blog. However, given the dynamic nature of blogs (e.g., publishing new posts and hence changing homepages frequently), content-based splog detection technique becomes less effective. For example, in Katayama et al. (2009), it is reported that the detection model trained with the data collected in 2007–2008 performed worse on the data collected in 2008–2009, compared to the test data collected in the same period as the training data (i.e., 2007–2008). Link-based splog detection techniques are mostly originated from web spam detection. However, as stated in Agarwal and Liu (2008), although web models seem to be an appropriate choice for modeling the blogosphere but there exists certain key differences. For example, web graph is usually quite dense due to the large number of interconnecting hyperlinks. The hyperlink structure in blogosphere, however, is very sparse (Kritikopoulos, Sideri, & Varlamis, 2006). Moreover, in social media (e.g., blogs, comments, and forums), readers have unprecedented freedom to create their content as well as links. As a result, links to splogs may appear in many reader’s editable areas, such as comments. These noisy links reduce the effectiveness of link-based splog detections unless the quality of links are ensured. It is also one of the reasons that Technorati opted not to index the links appear in comments (Sifry, 2005). Collaborative filtering technique has also been proposed to detect splogs (Han, Ahn, Moon, & Jeong, 2006). This method requires manual spam identification from users and the reported spam blogs are shared within a network of trust.

1.2. Research objectives and contributions

In this work, we study the problem of splog detection by monitoring on-line search results. We aim to detect splogs that have successfully slipped through the splog filtering by the search engine (Kolari, Finin, Java, & Joshi, 2007; Sifry, 2005). In particular, we propose a technique that exploits the user search queries and their results returned from search engines to detect splogs. Recall that one main objective of splogs is to attract traffic from blog search engines. This becomes possible only if the posts from the splogs surface in the top-ranked search results of some queries. We hence propose to detect splogs by monitoring the search results of some selected queries. Next, we present some design issues of our splog detection below.

- **Query set selection.** Given the volume of queries processed by a search engine each day, it is not feasible to monitor every single query for splog detection. The proposed solution therefore has to sample a small subset of queries that are more likely to be spammed than other queries. In other words, a set of queries that are more likely to direct traffic to splogs need to be determined and then monitored and analyzed. This reduces the computational and storage costs of splog detection.
- **Blog temporal behavior monitoring.** Due to the dynamic nature of blogs, a splog may behave normally for some time but later shows its true color to the search engine. The proposed solution therefore has to profile blogs with their temporal behavior and determine the splogs based on their temporal behaviors.
- **Flexibility and complexity.** Moreover, the solution shall be generic and be able to work on top of existing blog search engines without modification to them. In our case, the proposed solution would only take information from the search results (e.g., title, snippet, URL, publication time) as well as the information already available in the search engine index. At the same time, as splog detection analyzes the results of blog search engines, both the time and space complexities become major design issues.

Prior to the discussion of our solution, we describe two key assumptions that we have made: (i) among the blogs indexed by a search engine, the number of authentic blogs always dominates the number of splogs, i.e., we always assume that a blog search engine has done a reasonably good job in splog filtering, and (ii) posts from splogs frequently appear in the top-\(k\) hits of their targeted queries (\(k = 50\) throughout this paper). Justifications of these two assumptions are given in Section 3.

To address the aforementioned issues, we have proposed a framework for splog detection. Our proposed framework mainly consists of two modules, namely, spam-post detection and splog detection. The spam-post detection module computes a spam-post score for each blog post in the top-ranked search results of a monitored query and the score is stored in a blog state tuple together with the time. Based on the blog profile (i.e., the sequence of its state tuples), splog detection module computes a splog score indicating its likelihood of being a splog. Splogs are those received high splog scores. The splog detection is online as a splog score could be computed whenever a new blog state tuple is inserted into a blog profile. In all, we summarize the contributions of this research:

1. We have proposed a splog detection framework which is capable of detecting splogs online without involving training data and human inputs. To the best of our knowledge, this is the first effort detecting splogs through search results.
2. We have proposed the notion of blog profile. A blog profile records the temporal behavior of a blog with a sequence of blog state tuples. Four splog scoring functions are investigated for splog detection, taking blog profiles as input. To compute the splog score required in blog state tuples, we have proposed a set of features to model blog posts returned in search results.

---

2 We make the claim based on the research papers that have been published in academia. Nevertheless, it is well-understood that most blog search engines have their proprietary techniques for splog detection.
3. We have conducted experiments using real data collected from a blog search engine over about 1.5 years. Our experimental results have demonstrated the effectiveness of the proposed technique.

1.3. Paper organization

The remainder of the paper is organized as follows. In Section 2, we present the observations made from popular blog search queries and their results. Section 3 presents our framework for detecting spam blogs. Sections 4 and 5 discuss the details of spam-post detection and splog detection. The experiments are reported in Section 6. Section 7 reviews the related work of spam blog detection. We conclude the paper in Section 8.

2. Observations on blog searches

As mentioned in Section 1.2, one of the challenges is to select (and monitor) a subset of search queries which are likely to be the targeted queries for splogs. Recall that the main objective of a splog is to attract traffic from search engines. To achieve this, its posts have to reach a large number of users by appearing in the top search results of some queries. As the popular queries largely represent the common interests of web users and are heavily searched, spammers may subscribe the popular queries and generate fresh spam content containing the popular keywords. Their posts could easily appear among the top search results as recency is an important ranking factor in blog search. Similar assumptions were also made in Katayama et al. (2009), Markines, Cattuto, and Menczer (2009). Even in tagging systems (e.g., CiteULike and Delicious), spammers can use popular tags to drive traffic to their sites (Koutrika, Effendi, Gyöngyi, Heymann, & Garcia-Molina, 2008). To be reported in the sequel, the popular queries are not as dynamic as expected, making it even simpler for the spammers to subscribe. In literature, spam-prone queries have also been identified in other ways. For instance, in a recent study on web spam detection, Wang et al. studied two sets of query keywords heavily targeted by redirection spammers (Wang, Ma, Niu, & Chen, 2007). The first set was a manually compiled list of 323 keywords, each of which led to spam URLs among the top-50 results at one of the three major search engines. The second set was the 5000 most-bid keywords from an authentic ads syndication program. Nevertheless, both sets are not publicly accessible.

In the following, we report statistics derived from the popular queries and their search results to support our decision on monitoring popular queries. From 08 Nov 2006 1am to 31 Mar 2008 10pm, we crawled the top-15 most popular queries published by Technorati at the frequency of every three hours. Each of the 15 popular queries was submitted to Technorati and the top-50 results were obtained through Technorati’s search API. Note that the same query keyword may be popular at various time points and hence has been evaluated multiple times, as the search results could be different each time. After punctuation removal and lower-case cast (e.g., “Second Life” becomes second life), the number of distinct queries is 2301.

**Observation 1.** Many popular queries have not been dynamic. In addition, a small set of queries tends to be popular for a long period of time.

A heavily searched query (hence becoming popular query) at a given time often represents the information needed by a large number of users at that time. One may expect that the popular queries at a given time are often related to some events or hot topics; hence, expect that all popular queries are rather dynamic. **Observation 1** states the opposite.

Fig. 1 plots the popular query distribution. We define query frequency to be the number of times that a query appears in the collected dataset. Fig. 1 shows that query frequency roughly follows a power-law distribution: A large number of queries appears just once or twice whereas a few queries tend to be popular for a long time, even up to the entire period of our data collection (e.g., youtube). The top-20 queries with the highest query frequencies are listed in Table 1 and manually grouped into three categories. Many popular queries are website names and person names. That is, the spammers could improve its visibility in search results by simply targeting on those extremely popular queries.

**Observation 2.** Among the top search results of the popular queries, a few blogs appear much more frequently than others.

Fig. 2 plots the distribution of blogs which ever appeared in top-50 results of any popular query. The plot again shows a power-law distribution. About 1 million blogs appeared just once in the collected search results. In contrast, there are 1562 blogs which appeared more than 100 times. From this observation, we argue that splog detection does not need to keep track of every blog that appears in search results. It is sufficient to focus on those highly frequent ones. Furthermore, suppose a splog only appears once or twice a year in the search results. The damage of such a splog to the search engine’s user experience is negligible. On the bright side, by monitoring those frequently appeared in top search results, a splog detector does not necessarily require a large storage space.

---

Observation 3. Blogs appearing in many top search results do not imply they may attract a large number of inblogs.

As part of the search results by Technorati’s search API,4 the number of inblogs of a blog at each search is present for further analysis. The number of inblogs is defined by the number of blogs each having one or more links to the current blog. Therefore we are able to derive the number of inblogs a blog has attracted from its first appearance to its last appearance, namely inblog increment, in our dataset. Fig. 3 plots the averaged inblog increment against the number of times a blog appeared in top search results. It shows that when blogs do not appear frequently in top search results (e.g., fewer than 100 times), their inblog increment grows linearly. However, for those blogs that frequently appear in top search results, most of them do not attract a high inblog increment. Most specifically, some blogs appear more than 1000 times but attract a very small inblog increment (e.g., fewer than 10). One possible reason is that those blogs are splogs. In our proposed framework, we therefore use temporal inblog increment as a feature for splog detection.

In summary, we have presented some observations derived from real data collected from a commercial blog search engine. Based on these observations, we believe that it is highly possible for spammers to target on some popular queries, leading to spam-posts appearing in top search results for these queries.5 In addition, our statistics on the collected data also suggests that a splog detector does not need to keep track of every blog ever appears in top search results, but only the frequent ones. This observation makes it possible for a splog detector to operate on a relatively small amount of data. All these observations further motivate our work on splog detection and also provide some hints on the features that can be exploited in splog detection.

Table 1
The top-20 most popular queries.

<table>
<thead>
<tr>
<th>Category</th>
<th>Popular queries (query frequency)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Website</td>
<td>YouTube (3356), MySpace (2286), Google (1542), Bebo (917), Facebook (801), StudiVZ (809), Joost (563), Second Life (497)</td>
</tr>
<tr>
<td>Person</td>
<td>Ron Paul (2210), Noelia (1503), Paris Hilton (1391), Britney Spears (1068), Galileo Montijo (1038), Tammy NYP (431)</td>
</tr>
<tr>
<td>Others</td>
<td>iPhone (1380), Web 2.0 (571), Music (361), Authority (507), Video (486), Naruto (427)</td>
</tr>
</tbody>
</table>

Fig. 1. Popular query distribution.

Fig. 2. Blog distribution.

Table 1
The top-20 most popular queries.

http://technorati.com/developers/api/.

5 In our experiments, this hypothesis is partially evaluated with manually labeled data in Section 6.1.
3. Splog detection framework

Prior to our discussion on our framework for splog detection, we present two assumptions made, as discussed earlier in Section 1.

3.1. Assumptions and justifications

Assumption 1. Splogs always try to inject their posts into top search results of the targeted queries.

One main objective of splogs is to attract traffic from blog search engines so as to promote themselves and/or affiliated sites. To achieve this, spammers may exploit search engine optimization techniques to place their posts among the top search results, which have a higher chance to be viewed by users. On the contrary, if the posts from splogs could not frequently appear in top search results, the search quality is less affected for most users; and the problem of splog detection becomes less critical. Based on this assumption, we monitor those blogs whose posts frequently appear in top search results of selected queries (i.e., popular queries in our setting).

Assumption 2. Among the blogs indexed by a search engine, the number of authentic blogs always dominates the number of splogs.

One key factor for the success of search engines is to provide quality search results for users. For this reason, all search engines are fighting hard against spammers, although their techniques are not shared to the public (Kolari et al., 2007; Sifry, 2005). We therefore assume that blog search engines have already done a reasonable job in splog filtering and thus most of the posts in top search results are from authentic blogs. As popular queries at a given time often related to hot events/topics at that time, it may be reasonable to further assume that the posts from authentic blogs, among the top search results, are similar to each other (Sun, Hu, & Lim, 2008). This assumption provides us hints to determine the likelihood of posts from splogs, from top search results, using classical outlier detection techniques.

3.2. Splog detection framework

Illustrated in Fig. 4, the proposed framework consists of two main modules: spam-post detection and splog detection. The spam-post detection module detects those blog posts that are likely from splogs from the search results of popular queries.
The details of the detection is given in Section 4. Each detected spam-post is assigned with a score and the score is stored in the blog’s profile together with other attributes about the blog, such as the number of inblogs to the blog at different time points. The splog detection module computes a score for each blog based on its blog profile with the scoring functions presented in Section 5. Other than blog profiles, the proposed framework also maintains spam-post seeds which are representative spam-posts detected from the search results. The spam-post seeds are used to enhance the spam-post detection, as some spam-posts may be similar to each other. We remark that both blog profiles and spam-post seeds are initially empty and incrementally and automatically maintained with outputs from the spam-post detection module. To sum up, we outline our framework in Algorithm 1.

**Algorithm 1. Splog detection**

```latex
\begin{algorithm}
\begin{algorithmic}[1]
\State **Input:** search results for popular queries at various time points, post-distance threshold $\phi$, seed-post age threshold $\alpha$, and window size $W$
\State **Output:** a set of spam blogs
\State Set of spam-post seeds $S := \emptyset$
\For {each time stamp $t$}
\For {each query $q$}
\State Get search results of query $q$ at time $t$, $R_{qt}$:
\State $P := \text{spam-post-detect}(R_{qt}, S)$; /* see Algorithm 2 */
\State $S := \text{spam-seed-select}(P, S, \phi, \alpha)$; /* see Algorithm 3 */
\EndFor
\State update blog profiles using $P$;
\State compute/update blog spam scores from blog profiles; /* see Section 5.2 */
\EndFor
\State return splogs based on the spam scores of blogs;
\end{algorithmic}
\end{algorithm}
```

4. Spam-post detection

In this section, we describe the technical details of the spam-post detection. Our main technique is to formulate the problem as an outlier detection problem, based on our earlier discussions. The main technical issue of detecting outliers among posts is to derive a set of features to describe each post and a distance measure between posts. These features are then used to compute the distance between a pair of posts, which is required by the outlier detection algorithm.

4.1. Post features and distance measure

Given a query $q$ issued at time $t$ to a blog search engine, the set of the top-ranked blog posts returned is denoted by $R_{qt}$. For each blog post $p_i \in R_{qt}$, we derive eight features to describe it. These features are derived based on the earlier works in spam webpage detection (Ntoulas, Najork, Manasse, & Fetterly, 2006; Svore, Wu, Burges, & Raman, 2007) as well as spam blog analysis (Kolari et al., 2006). Among the eight features, six features are directly derived from the title and content (i.e., snippet or summary in top search results) of each post respectively. They are:

- the length of the title (content) in terms of the number of words;
- the number of query words matched in the title (content); and
- the average word length in the title (content).

We propose two additional features that are derived from a post graph – which is inspired by spam webpage detection (Kurland & Lee, 2005; Kurland & Lee, 2006) – with all the posts in $R_{qt}$. The post graph is used to model the similarity between posts in top search results. Given all posts in $R_{qt}$, a weighted directed graph is constructed where each vertex in the graph is a post $p_i \in R_{qt}$. An edge from post $p_i$ to post $p_j$ is created if and only if $p_j$ is among the $k$-nearest neighbors of $p_i$, with respect to a given similarity measure ($k = 20$ in our experiments). The similarity from $p_i$ to $p_j$, which is also used to weigh the corresponding edge, is defined as follows:

$$w(p_i, p_j) = \exp^{-KL(LM(p_i)||LM(p_j))}$$

(1)

where $KL(P||Q)$ denotes Kullback–Leibler (KL) divergence, which estimates the distance of the two probability distributions $P$ and $Q$ (Kullback & Leibler, 1951); $LM(p_i)$ is the Jelinek–Mercer smoothing language model constructed using the words in the title and content of post $p_i$ (Ponte & Croft, 1998). Note that, KL divergence is directional and $w(p_i,p_j) \neq w(p_j,p_i)$ in general.
The two features, namely, weighted out-going degree and clustering coefficient and then derived from the post graph.

- **Weighted out-going degree** is the sum of the weights of outgoing edges of a post. It is directly proportional to the relevance/proximity of the post to the imprecise ground truth (i.e., all posts in $R_q$). Since all posts in $R_q$ are retrieved for a specific query, it is expected that the posts from authentic blogs have higher weighted out-going degrees than that from splogs.
- **Clustering coefficient** of a post $p_i$ denoted by $c_i$ is computed in the following equation:

$$c_i = \frac{2\sum_{p_b \in N_i} w[p_i, p_b]}{|N_i| \times (|N_i| - 1)}$$

In this equation, $N_i$ is the set of 20 neighbor posts of $p_i$ from the post graph; $p_a$ and $p_b$ are two neighbor posts in $N_i$. The clustering coefficient is a generic metric to quantify how close is the subgraph induced by a post and its neighbors from a clique. Again, since authentic blog posts are all retrieved from a specific query, they are expected to be similar to each other with higher clustering coefficients than that of splog posts. Clustering coefficient was also used as a feature for detecting spam in YouTube (Benevenuto, Rodrigues, Almeida, Almeida, & Gonçalves, 2009).

All these eight features are normalized independently with respect to the corresponding maximum value from all posts in $R_q$ (i.e., L-infinity norm). With the 8 features, the distance between $p_i$ and $p_j$ is defined by Eq. (3), where $\cosine$ denotes the cosine similarity measure.

$$\text{dist}(p_i; p_j) = 1 - \cosine(p_i; p_j)$$

We highlight that the eight features describe blog posts from three aspects: (i) the presentation of each post in the search results (e.g., length of the title), (ii) the relationship between each post to the query (e.g., number of matching words), and (iii) the relationship among posts in the top search results (e.g., weighted out-going degree). While the features may seem to be ad-hoc, they have shown their effectiveness in our experiments. Other than the 8 features described above, in our experiments, we have also tried to include a few content features (e.g., words extracted from the post title and content) for the post-distance measure. Nevertheless, the inclusion of content features reduced the accuracy of splog detection. One possible reason is that splogs try to make their posts look similar to those from authentic blogs in terms of semantic. For the succinctness of presentation, we opt not to report the use of content features.

4.2. Incremental spam-post detection

**Algorithm 2. spam-post-detect**

```
Input: a set of blog posts $R_{q,t}$, a set of spam-post seeds $S$
Output: a set of spam-posts $P$
1: $P := \text{outlier}(R_{q,t}, \text{dist});$
2: foreach $p \in P$ do /* $p$ is an outlier in $R_{q,t}$ */
3:     $p.p\text{spam-score} := \text{average}([\text{dist}(p, p') | p' \in R_{q,t}]);$
4: end
5: foreach $s \in S$ do
6:     $s.seed\text{score} := \text{min}([\text{dist}(s, p) | p \in P]);$
7: end
8: foreach $p \in R_{q,t}\setminus P$ do
9:     if $\exists s$, s.t. $s \in S$ and $\text{dist}(s, p) < s.seed\text{score}$ then
10:        $p.p\text{spam-score} := \text{average}([\text{dist}(p, p') | p' \in R_{q,t}]);$
11:        $P := P \cup \{p\};$
12: end
13: end
14: return $P$.
```

The proposed spam-post detection can be summarized as follows: Given the set of top-ranked posts $R_{q,t}$ obtained with query $q$ at time $t$, the first step is to detect the outliers among them, based on the assumption that posts from splogs are likely to be dissimilar to those from authentic blogs with respect to the 8 features described earlier. Each outlier detected from $R_{q,t}$ is associated with a spam-post score representing its likelihood of being a spam-post (or a post from splog). Those posts that are not outliers determined by the outlier detection algorithm, are further compared with earlier spam-post seeds in case the outlier detection algorithm missed some spam-posts. In web spam detection, it is found that many spams share structural similarity (Urvoy, Chauveau, Filoche, & Lavergne, 2008). In splog detection, as spam-posts are generated automatically, we also believe that they share some structural similarity (modeled by the 8 features). Even if a blog post is not an outlier in $R_{q,t}$, but very similar to a spam-post seed detected earlier, it is assigned with a high spam-post score. Spam-post seeds are then maintained with some of the newly detected spam-posts with high scores. In the sequel, we further elaborate the details of spam-post detection and spam-post seeds below.
4.2.1. Spam-post detection

For easy discussion of spam-post-detect shown in Algorithm 2, we assume that a set of spam-post seeds $S$ is already determined. Given $S$ and $R_q$ as input, the algorithm returns a set of spam-posts $P$ as output.

In Line 1, a distance-based detection algorithm outlier is used to detect outliers. Defined in Knorr and Ng (1998), an object $o$ in a dataset $D$ is a distance-based outlier if at least $\theta$ fraction of objects in $D$ lie at a distance greater than or equal to distance $r$, from $o$. In our experiments, we adopted the outlier detection algorithm from Ren, Rahal, Perrizo, and Scott (2004). The algorithm takes the set of posts in $R_q$, a distance function $dist$ to measure the distance between any two posts (i.e., Eq. (3) in our setting) as input and returns the outliers in $R_q$. Following (Knorr & Ng, 1998 & Ren et al., 2004), we set $\theta$ to be 0.8, and distance threshold $r$ to be $d + 1.28\sigma$, where $d$ and $\sigma$ are the mean and standard deviation of the pair-wise distances derived from the given set of objects, i.e., $R_q$.

spam-post-detect first sets the outliers detected to be spam-posts (Line 1) and then computes a score for each spam-post (Lines 2–3). Next, in Lines 5–13, those spam-posts that might have been missed by outlier are further checked. The intuition is that if a post is more similar to some post(s) in the spam-post seeds than the outliers found, then such a post is likely to be a spam. To identify those posts, the minimum distance between a spam-post seed to each outlier is computed (Line 6). For each post $p$, if there is a post $s$ in the seeds such that $p$ is more similar to $s$ than all other outliers to $s$ (Line 9), we include $p$ in the spam-post. In Line 10, the spam score for $p$ is updated. We call our algorithm incremental as the spam-post seeds are continuously updated and maintained with newly detected spam-posts discussed in the next subsection.

4.2.2. Spam-post seeds

The spam-post seeds play an important role in the spam-post detection. As discussed in spam-post-detect, the spam-post seeds are used as pseudo-relevance feedback to detect spam-posts. Note that in our problem, enumerating all spam-posts is not necessarily the primary objective. Rather, selected spam-posts are likely to be used as inputs for a subsequent splog detection step. Hence, a relatively small representative spam-posts, namely spam-posts seeds, may suffice. Ideally, (i) the set of spam-post seeds is a set of non-redundant spam-posts; and (ii) the seeds should be representative. In another words, for every spam-post not reported in the set of spam-post seeds, we want it to have a representative similar to it in spam-post seeds; (iii) the seeds should be a set of fresh posts that reflect to the current spammer’s tactics.

Algorithm 3. spam-seed-select

```
Input: newly detected spam-posts $P$, current set of spam-post seeds $S$, post-distance threshold $\phi$, and seed-post age threshold $\alpha$

Output: updated set of spam-post seeds $S'$

1 $S' := S$
2 $S_\alpha := \{ s \in S, s$.age $\geq \alpha \}$
3 $r := \text{average}(p$.spam_score$| p \in P)$
4 foreach $p$ in $P$ s.t. $p$.spam_score $> r$
5 $S' := S' \cup \{ p \}$
6 $s' := \text{arg } \min \{ \text{dist}(p, s) | s \in S_\alpha \}$
7 if $\text{dist}(p, s') < \phi$ then $/* s'$ is old and similar to $p */$
8 remove $s'$ from $S$
9 end
10 end
11 return $S'$
```

In response to these, we propose spam-seed-select, shown in Algorithm 3. It takes a set of newly detected spam-posts $P$ (e.g., from the top hits of a search), the current spam-post seeds $S$ and two thresholds $\phi$ and $\alpha$ as input and produces the next spam-post seeds as output. Initially, $S$ is empty. As the blog search engine is monitored by spam-seed-select over a period of time, $S$ becomes non-empty and incrementally maintained by spam-seed-select. The age of a post (or a spam-post seed) is defined to be the time difference between the current time and the time of its appearance in the top results of the latest search.

To maintain the spam-post seeds, the set of spam-post seeds $S'$ returned is firstly set to be $S$ (Line 1). Then, some out-dated posts among the seeds are replaced with fresh spam-posts. More specifically, in Line 2, all candidate out-dated seeds from $S$ that could be removed are identified since some fresh posts from $P$ will be inserted into $S$. In Line 3, the average spam score of the spam-posts in $P$ is computed, where the spam score has been assigned by Algorithm spam-post-detect (Lines 3 and 10). Spam-posts from $P$ with high spam scores are then added into the spam-post seeds (Line 5). Next, those out-dated seeds that are similar to the newly included spam-posts are removed. That is, we find if there is a seed in $S$, which is very similar to a newly included spam-post $p$ with respect to the given distance threshold (Lines 6–7). If such a seed exists, it can be safely removed. Hence, Line 8 removes the previous seed from $S'$.

6 The detection of spam-post seeds are detailed in Algorithm 3.
5. Blog profiles and splog detection

In this section, we discuss the blog profiles and the scoring functions for splog detection. As discussed in Sections 1 and 2, it is critical to monitor the temporal behavior of a blog for splog detection as a blog may behave normally at the early stage in order to pass any spam filters.

5.1. Blog profile

A blog profile is defined by the blog’s URL and a sequence of blog state tuples recording its temporal behavior. Specifically, when a post $p$ appears in the top search results of a monitored query $q$ at time $t$, a blog state tuple $(t, \ell, p, \text{spam}\_\text{sore})$ is created and inserted into its corresponding blog profile, where $\ell$ is the number of inblogs$^7$ to the blog at time $t$; $p, \text{spam}\_\text{sore}$ is the spam-post score assigned to $p$ if $p$ is a spam-post (see Section 4), and $p, \text{spam}\_\text{sore} = 0$ otherwise.

As the most recent behavior of blogs are more critical for splog detection, we only store the state tuples within a time window $W$ (e.g., one month) with respect to the current time. That is, a state tuple $(t, \ell, p, \text{spam}\_\text{sore})$ is removed from a blog profile if $t < t_c - W$ where $t_c$ denotes the current time. A blog profile is dropped if it does not contain any state tuple; that is, no post from the blog appears in top hits within the past $W$ time period. Fig. 5 illustrates three example blog profiles where $W$ is four time points for clarity. The first and second blog profiles have 4 and 2 state tuples respectively recorded in the past $W$ period, and the third blog profile is dropped since no post from the blog appears in the top hits of any monitored query. With such a time window, only those blogs having posts recently appear in top hits of the monitored queries need to be maintained. Moreover, as each blog state tuple stores three numerical values only, the storage overhead incurred by blog profiles is small.

5.2. Splog scoring functions

Given a blog profile, we have defined 4 scoring functions, denoted by $SF_1$ to $SF_4$. Each of them independently estimates the likelihood of a blog being a splog. For discussion purposes, we use $ST$ to denote all state tuples in a given blog profile $b$.

$SF_1$: The inblog increment over time. Based on a recent study on evolving graph data, we assume links among authentic blogs follow a densification law (Leskovec, Kleinberg, & Faloutsos, 2005). That is, the number of inblogs of authentic blogs grows systematically. We therefore propose a scoring function to detect sudden shrink$^8$ in inblog increment. Specifically, a linear regression model is computed to predict the number of inblogs using $ST$. Upon the creation of a new state tuple, let $\ell'$ be the estimated number of inblogs; $\ell$ be the actual number; and $\ell$ be the average derived from $ST$; The likelihood of $b$ being a splog is given in the following equation:

$$
SF_1(b) = \begin{cases} 
1 & \text{if } \ell' - \ell > \tilde{\ell} \\
(\ell' - \ell) / \tilde{\ell} & \text{if } 0 \leq \ell' - \ell < \tilde{\ell} \\
0 & \text{otherwise}
\end{cases}
$$

$^7$ The number of inblogs is provided by Technorati API as a part of search result.

$^8$ We have also tried the detection of sudden bursts as opposed to sudden shrinks. Nevertheless it was not effective in our experiments.
\(SF_2:\) Correlation between the number of posts appearing in top search results and in blog increment. As shown in Fig. 3, it is expected that a blog would attract more inblogs if its posts frequently appear in top search results, as those posts would reach more users. Hence, we compute the linear correlation coefficient \(\rho\) between the two values derived from \(ST\) and then define \(SF_2(b) = 1 - \rho\).

\(SF_3:\) Average spam-post score. This scoring function is defined based on the spam-post scores obtained from spam-post detection module. As an authentic blog is believed to contain fewer spam-posts, average spam-post score derived from \(ST\) is proposed to detect splogs: \(SF_3(b) = \frac{\sum_{i \in S} \text{spam-score}_{b \in i}}{|S|}\).

\(SF_4:\) Blog \(URL\) length. Reported in Fetterly, Manasse, and Najork (2004), \(URL\) length had been surprisingly effective in spam webpage detection. The observation is that spams often have either abnormally shorter or longer \(URL\)s when compared with authentic webpages. Hence, we simply reuse blog \(URL\) length as a naive metric for spam blog detection. Specifically, we applied zero-mean normalization on blog \(URL\) length, \(z(b) = \frac{burl-\text{mean}}{\text{std-dev}}\), where \(burl\) is the length of blog \(URL\), \(url\) is the expected length, and \(\text{std-dev}\) is the standard deviation derived from the blog \(URL\)s seen in the time window \(W\). In simple word, \(z(b)\) is the number of standard deviations of \(b\)'s \(URL\) length away from the mean. Then \(SF_4(b)\) is the min-max\(^9\) normalized \(z(b)\) such that \(SF_4(b)\) scores are in the range of \([0, 1]\).

From their definitions, the above 4 \(SF\) scoring functions each estimates the likelihood of a blog being a splog from one perspective. Both \(SF_1\) and \(SF_2\) rely on the temporal behavior of increment of inblogs, which can be considered as the global context of the blog (e.g., how many other blogs recommend this blog). The difference is that \(SF_1\) does not consider the frequency of the blog’s appearance in top search results, but \(SF_2\) does. Recall that the spam-post score is computed from the search results of a particular query. \(SF_1\) detects the splog based on the temporal behavior derived from local context (i.e., blog posts returned for the monitored queries). Compared to the first three scoring functions, \(SF_4\) is static. Note that, all these four scoring functions are very easy to compute, reflecting time/space complexity design issue discussed in Section 1.2. The fast computation enables real-time splog detection whenever a blog state tuple is inserted into a blog profile.

To aggregate the efforts of the four proposed scoring functions in splog detection, there could be many different ways. A straightforward approach is to linearly combine the scores a blog received from the four scoring functions as shown in Eq. (5), where \(w_i (1 \leq i \leq 4)\) is the weight of the corresponding scoring function and \(\sum_{i=1}^{4} w_i = 1\).

\[
SF_a(b) = \sum_{i=1}^{4} w_i \cdot SF_i(b) \tag{5}
\]

The main difficulty here is to learn the four \(w_i\)'s which requires a reasonable number of labeled data with supervised learning. In our experimental setting, initially, we set the four weights to be the same value (i.e., 0.25). Then we apply evolutionary algorithm (EA) (Holland, 1992; Ingo, 1971; Evolutionary, 1996) to learn the weights, with 200 manually labeled blogs. The learned weights is \(w = [0.254, 0.245, 0.265, 0.236]\).

6. Experiments

We designed two sets of experiments to evaluate (i) the effectiveness of scoring functions, and (ii) the impact of varying the parameters including time window size \(W\), age threshold \(\sigma\), and post distance threshold \(\phi\). The two sets of experiments are reported in Sections 6.2 and 6.3 respectively. Before we report the two sets of experiments, we first describe the dataset and the performance measures used in our experiments.

6.1. Dataset and performance measure

The proposed splog detection framework was evaluated using real data collected from Technorati (see Section 2 for the statistics on the dataset). Recall that we collected the top-15 popular queries published by Technorati every three hours from Nov 06 to Mar 08. For each query, the top-50 results were retrieved in XML format through Technorati API. The collected results were parsed and processed using Lucene\(^{10}\) with KStem\(^{11}\) and then indexed using both Lucene and MySQL\(^{12}\). The splog detection framework were implemented in Java.

To measure the effectiveness of the proposed technique, we selected 3000 blogs with the highest frequencies (blog frequency \(\geq 65\)) for manually labeling. The labeling was done by two undergraduates from business school and the students have no knowledge on the algorithms to be evaluated on the labeled dataset. Among the 3000 blogs, 444 were no longer in operation at the time of labeling. In addition, 1102 blogs containing many non-English blog posts. As the result, we obtained 699 splogs from the remaining 1454 blogs that are in English. We call the set of 699 splogs SplogSet-1 (or SS1 for short). Stated in Liu, Cen, Zhang, Ma, and Ru (2008), many blogs frequently change their \(URL\)s to bypass the search engine's

---

\(^{10}\) Note that, in a strict online setting, the max value is unknown. In our experiments, we found that all the \(z(b)\)'s were smaller than 6.

\(^{11}\) http://lucene.apache.org/.

\(^{12}\) http://ciir.cs.umass.edu/.

\(^{12}\) http://www.mysql.com/.
For this reason, we construct SplogSet-2 (or SS2) which is the union of the SS1 and the set of 444 non-operational blogs, with the assumption that all the non-operational blogs were splogs. Nevertheless, we understand that it is a strong assumption and the results on SS2 are therefore mainly for the completeness of the experiments. Our discussion on the experimental results will be mainly based on SS1. With the 699 splogs manually identified, we are able to derive the queries that are more likely to lead to these splogs in their search results. In Table 2, splog-hits is the number of search results that originate from splog(s) at various time points for the query, query-hits is the total number of search results retrieved for the query in our dataset, and percentage is the ratio of the two values. From Table 2, among the top 20 queries with the highest splog-hits, 10 of them are also among the top 20 retrieved most number of search results. This partially supports our Assumption 1 such that splogs target on those popular queries frequently.

Precision and recall are the most commonly used measures in various prediction problems. In our setting, precision is the percentage of true splogs among all blogs detected to be splogs; and recall is the percentage of splogs detected among all true splogs (that should be detected). However, precision and recall are threshold-dependent. In our work, we therefore report the precision-recall curve (PR-Curve) and Average Precision (AP). PR-Curve gives the details of the changes in precision and recall values with respect to different thresholds. AP is a single-value metric for easy comparison of different methods. Given a list of blogs ranked according to their assigned splog-scores in non-increasing order, average precision is the average of the precisions obtained after each true splog is obtained in the ranked list.

### 6.2. Effectiveness of scoring functions

The first set of experiments is to evaluate the effectiveness of the proposed scoring functions in detecting splogs. We fixed time window size for blog profiles at $W = \infty$ for all scoring functions. For $SF_3$, we fixed spam-post seed age threshold $\alpha = 10$...
days and post-distance threshold $\phi = 0.20$. Fig. 6 plots the PR-Curves of detected splogs using the five scoring functions over SS1 and SS2 respectively. APs of the five scoring functions are reported in Table 3.

From Fig. 6a and Table 3, clearly the aggregated scoring function $SF_a$ was the best performing scoring function, followed by $SF_2$ and $SF_3$. To test whether the performance differences were statistically significant, we conducted paired $t$-test on their results. For each scoring function, we sorted the blogs according to their assigned splog-scores in non-increasing order and computed the precisions obtained after each true splog is obtained. Our paired $t$-test on the precision values showed that $SF_a \gg \{SF_2, SF_3\} \geq \{SF_1, SF_4\}$, where $\gg$ denotes significantly better with $p$-value $<0.05$. Despite $SF_2$ outperformed $SF_3$ when recall was in the range $[0.0, 0.4]$, we argue that $SF_3$ is more generic as it requires no information about inblogs to be provided by a search engine. That is, $SF_3$ could be implemented solely based on the search results of any blog search engine. $SF_4$, which detects splogs purely based on URL length, achieved only better precision than random guess. $SF_1$, based on the assumption that the inblogs increase along time, only managed to achieve better precision than $SF_4$, but worse than $SF_2$ and $SF_3$. One possible reason is that most blogs are personal online diaries. They do not reach many readers and do not gain large inblog increment along the time. So the difference in inblog increment from them and the splogs is marginal.

On SS2, shown in Fig. 6b and Table 3, all scoring functions achieved higher precisions than those on SS1. It is also observed that the performance of $SF_3$ increased significantly. One possible reason is that most of those sites in SS2 were no longer in operation during the data collection period, hence gained low inblog increment.

In the above experiments, the scoring functions involving inblogs (i.e., $SF_a$ and $SF_2$) have demonstrated their effectiveness. Such a result does not contradict with our earlier discussion in Section 1 that link information is less effective for splog detection. The links used in our experiments are of high quality: (i) the link information is obtained from Technorati where most splogs are already filtered out, and (ii) Technorati does not index comments where links to splog commonly appear in Sifry (2005).

6.3. Impact of varying parameters

This set of experiments aims to find out the impact of varying various parameters to the splog detection performance. Based on the earlier results, we only report the performance of $SF_a$ with different parameter settings as $SF_a$ was the best performer.

6.3.1. Window size $W$

While fixing all other parameters, we studied the impact of varying window size $W$ (see Section 5). We varied window size $W$ from 1 day to $\infty$ and report the PR-Curves and APs of $SF_a$ in Fig. 7 and Table 4 respectively. As shown in Fig. 7 and Table 4, the larger the $W$ the better the detection accuracy among the values evaluated. Specifically, on SS1 the paired $t$-test shows that $W = \infty \gg \{W_{3\text{days}}, W_{15\text{days}}\} \gg \{W_{7\text{days}}, W_{3\text{days}}\} \gg W_{1\text{day}}$, where the subscripts are the values of $W$’s. On SS2, the paired $t$-test shows that $W = \infty \gg W_{3\text{days}} \gg W_{15\text{days}} \gg W_{7\text{days}} \gg \{W_{3\text{days}}, W_{1\text{day}}\}$. That is, on SS2, there was no significant improvement on the results with a larger $W$ than $W_{31}$. Recall that many blogs in SS2 were no longer in operation. Once
Where a relative long time period (e.g., 31 days), the increase of $W$ does not necessarily lead to more blog state tuples in these blog profiles. The detection accuracy then became stable. However, for those blogs remain in operation, the increase of $W$ means more blog state tuples to be maintained in the detection process. In our following experiments, we set $W$ to be 31 days.

6.3.2. Post-age threshold

Next, we studied the impact of varying seed-post age $\alpha$ in spam-post seed selection (see Algorithm 3) while fixing $W = 31$ days and $\phi = 0.2$. Fig. 8 plots the PR-Curves with $\alpha$ varying from one day to two weeks and the corresponding APs are reported in Table 5. On SS1, it is interesting to observe that either a small $\alpha$ (e.g., 1 day or 3 days) or large $\alpha$ (e.g., 2 weeks) would result in poorer splog detection accuracy. In specific, our significant test shows that $\{\alpha_{1\text{day}}, \alpha_{3\text{days}}\} > \alpha_{14\text{days}}$. Again, $>$ means significantly better with $p$-value <0.05. Keeping the spam-post seeds with the age of less than 10 days achieved the best precision in the experiment. Such results partially unveil the temporal phenomena in the blog data. Bloggers may discuss and publish posts referring to an event or a hot topic for a short period then move to some newer topics. On SS2, small $\alpha$ (e.g., 1 day, 3 days) led to a poorer performance. With $\alpha$ larger than one week, there is not much difference in the detection performance.

Similarly, fixing $W = 31$ days and $\alpha = 10$ days, the impact of varying post-distance threshold $\phi$ (see Algorithm 3) is reported in Fig. 9 and Table 6 for PR-Curves and APs respectively. The best performance was achieved when $\phi = 0.2$.

6.4. Spam-post seed accuracy

Recall that in the proposed splog detection framework (see Fig. 4), a set of spam-post seed is maintained, during the detection process, to facilitate the detection of spam-posts for $SF_3$. In this set of experiments, we evaluate the accuracy of the spam-posts detected and maintained by Algorithm 3.

As the set of spam-post seeds is updated through the detection process, we report the averaged detection accuracy (e.g., the percentage of the blog posts that are from splogs) for one run with fixed parameters. Fixing the spam-post seed age $\alpha = 10$ days, the accuracies of the spam-post seeds of varying distance threshold parameter $\phi$ is reported in Fig. 10a. Overall, the detected spam-post seeds have a decent quality. The paired $t$-test gives the following $\phi_{0.20} \gg \{\phi_{0.25}, \phi_{0.15}\} \gg \{\phi_{0.10}, \phi_{0.30}\}$.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$W = 1$ day</th>
<th>$W = 3$ days</th>
<th>$W = 7$ days</th>
<th>$W = 15$ days</th>
<th>$W = 31$ days</th>
<th>$W = \infty$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS1</td>
<td>0.594</td>
<td>0.682</td>
<td>0.696</td>
<td>0.707</td>
<td>0.710</td>
<td>0.751</td>
</tr>
<tr>
<td>SS2</td>
<td>0.685</td>
<td>0.690</td>
<td>0.709</td>
<td>0.763</td>
<td>0.791</td>
<td>0.811</td>
</tr>
</tbody>
</table>

Table 4
Average precision for varying window size $W$.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$\alpha = 1$ day</th>
<th>$\alpha = 3$ days</th>
<th>$\alpha = 7$ days</th>
<th>$\alpha = 10$ days</th>
<th>$\alpha = 14$ days</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS1</td>
<td>0.596</td>
<td>0.612</td>
<td>0.704</td>
<td>0.710</td>
<td>0.660</td>
</tr>
<tr>
<td>SS2</td>
<td>0.685</td>
<td>0.735</td>
<td>0.791</td>
<td>0.791</td>
<td>0.785</td>
</tr>
</tbody>
</table>

Table 5
Average precision for varying seed age $\alpha$.
where the subscripts are the values evaluated, and $\gg$ means significantly better with $p$-value <0.05. Next, fixing distance threshold $\phi = 0.2$, we compared the accuracy of spam-post seeds with varying seed age from one day to two weeks, as shown in Fig. 10b. It is observed that either small $\alpha$ (e.g., 1 day) or large $\alpha$ (e.g., 2 weeks) would result in a poorer splog detection accuracy. The paired $t$-test shows $\alpha_{1\text{day}} \gg \alpha_{2\text{days}} \gg \alpha_{3\text{days}} \gg \alpha_{14\text{days}}$. Again, $\gg$ means significantly better with $p$-value <0.05. Keeping the spam-post seeds with the age of about 10 days achieved the best detection accuracy in the experiment.

6.5. Summary

To summarize, our experiments showed that splogs can be detected from search results with a fairly high accuracy by monitoring popular search queries. We would like to further highlight the following three points.

- The data used in our experiments was collected through 120 searches to the blog search engine each day. Given the fact that a search engine often processes millions of queries daily, the splog detection leads to a negligible overhead to search engines.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$\phi = 0.10$</th>
<th>$\phi = 0.15$</th>
<th>$\phi = 0.20$</th>
<th>$\phi = 0.25$</th>
<th>$\phi = 0.30$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS1</td>
<td>0.617</td>
<td>0.681</td>
<td>0.710</td>
<td>0.690</td>
<td>0.649</td>
</tr>
<tr>
<td>SS2</td>
<td>0.709</td>
<td>0.777</td>
<td>0.791</td>
<td>0.780</td>
<td>0.745</td>
</tr>
</tbody>
</table>

Table 6

Average precision for varying post-distance threshold $\phi$. 

Fig. 9. Precision-recall curve for varying distance threshold $\phi$.

Fig. 10. Accuracy of spam-post seed.
• The proposed scoring functions are simple and easy to implement on top of any existing search engine. In particular, our proposed spam-post detection based scoring function $SF_3$ even requires no inblog information to be provided in the search results. The small computation and storage requirements facilitate online splog detection in real-time.
• In our experiments, we monitored the popular queries. However, the proposed framework is generic and can be used to monitor any set of queries that are likely to be targeted by spammers. To evaluate the splog detection accuracy of using other sets of queries is part of our future work.

7. Related work

To-date, spam blog detection is still in its infancy where few previous work has been proposed. Here, we highlight some details of previous spam blog detection. Han et al. proposed a collaborative blog spam filtering technique, which is originated from email spam detection (Han et al., 2006). This method relies on some manual spam identification. Then, the method shares such spam information to others in a network of trust. Mishne et al. proposed a comment spam detection approach based on language modeling (Mishne & Carmel, 2005). They defined a scoring function as the similarity between the language model of the blog posts and that of the related comments and other posts linked by comments. The score range of authentic comments is then derived by sampling and subsequently used to detect spam comments. There is another stream of work on applying supervised machine learning technique for spam-blog detection. Katayama et al. (2009) examined several strategies for selective sampling in active learning against the task of splog/authentic blog detection. Kolari et al. (2006) extracted content features, url (or domain-based) features and link features to build a splog detector by supervised machine learning technique. In Salvetti and Nicolov (2006), the authors proposed a splog filtering technique based on url tokenization and analysis. In Kolari et al. (2007), a framework combining URL based filtering, blacklist based filtering, blog home-page based filtering and feed based filtering is presented. Though the combined filter takes the time-stamp of blog posts as input, the temporal information is not explicitly utilized in splog detection.

Only few work has utilized temporal information on splog detection. In Lin et al. (2008), a method is proposed to extract temporal-regularity features for splog detection. It is reported that compared to authentic blogs, splogs have high autocorrelation, regular post time interval, and consistent link structures. Sato et al. (2008) analyzed splogs based on temporal characteristics of 50 pre-defined keywords contained in the blogs. The keywords are those of public concern (e.g., related to political/economical issues) or private concern (e.g., related to entertainment or celebrity issues). Various features collected from the homepages of blogs were manually examined and the relationship between these features and the selected keywords was studied. It is found that if an unknown blog contains the identified keyword of higher splog rates, the blog is likely to be a splog.

Our technique is different from the existing approaches in two ways: (i) Most of the existing work analyzes the entire content and links of the blogs to be detected, while we detect splog through blog search results and only consider those blogs whose posts frequently appear in top search results. (ii) Previous work, except Lin et al. (2008), Sato et al. (2008), does not exploit the temporal behavior of blogs. Different from Lin et al. (2008), Sato et al. (2008), our technique does not need to analyze the temporal behavior of each blog in a given set but only those frequently appeared in the blog search results. Moreover, our proposed technique does not require more information than that already provided in the search results.

Besides the few spam blog detection discussed above, a large number of spam website detection has been proposed. Webpage spammers often introduce a link farm to boost the PageRank (Brin & Page, 1998) of the spam webpages and thus the traffic to the spammers’ sites. TrustRank (Gyöngyi, Garcia-Molina, & Pedersen, 2004) has proposed an improved PageRank to avoid link farm spams. The idea is to manually select a set of good seed pages. Their observation is that good webpages often link to good webpages but not spam webpages. Hence, their approach ranks the pages that have a good connectivity to good seed pages higher than others. An approach from another direction has been undertaken by BadRank (Wu & Davison, 2005). BadRank proposes an algorithm to generate a set of bad seed pages. Their observation is that pages in a spam link farm are densely connected to each other. A ranking function is defined to identify the spam webpages based on such link structures. Spam mass (Gyöngyi, Berkhin, Garcia-Molina, & Pedersen, 2006) derives a score from PageRank and TrustRank to determine the proportion of PageRank contributed by spam pages (mass). Shen et al. (2006) and DiffusionRank (Yang, King, & Lyu, 2007) are two pioneer works that consider ranking of webpages over a period of time. Shen et al. (2006) defines a set of temporal features such as in-link growth rate and in-link death rate in detecting link spams. The intuition of DiffusionRank is that the energy associated with each link will gradually decrease as time passes by. This method also requires a set of trusted webpages as an input. Castillo, Donato, Gionis, Murdoch, and Silvestri (2007) incorporated web topology into the prediction obtained by their own classifier to improve the web spam detection. Similar to other spam detection approaches, it has a base classifier to label spam and non-spam blogs. Propagation algorithms are then proposed to smooth spam detection. The label of the neighbors of a blog is propagated to the blog. This propagation continuously affects spam detection.

8. Conclusions and future works

Being the first effort in splog detection by monitoring on-line search results, our work has demonstrated that splogs could be detected from search results through their temporal behavior. We have proposed a framework for splog detection consisting of spam-post detection and splog detection modules. For recording temporal behavior of blogs, we propose the notion
of blog profile, which is a sequence of blog state tuples. A blog state tuple stores the time when a blog's post appears in the top search results, the spam score of the post, and the number of inblogs. In our experiments, using real data collected from a blog search engine, we evaluated in total five scoring functions for splog detection and observed promising results. As discussed in our experiments, the proposed technique could be implemented on top of any existing blog search engine without modification to them. The insignificant computational and storage costs of the proposed approach make it ideal for online splog detection in real-time.

We will mainly focus on three tasks in our future research on the topic. The first task is to conduct a detailed study on the effectiveness of the features used for detecting spam-posts with the labeled data obtained. The second task is to research on better ways of aggregating the scoring functions for more effective splog detection. As we have shown that some popular queries are more likely to leading to splogs, we plan to study the spamicity of queries and try to detect splogs based on features derived from both queries and blogs, which is the third task. Finally, given the popularity of tagging systems, tag spam is now becoming an important research issue. As tag search is similar to keyword query search in blogs or other social media, we are also interested in finding out whether the proposed techniques can be adopted for tag spam detection.

Acknowledgement

This work was supported by A-STAR Public Sector R&D, Singapore. Project Number 062 101 0031. The work of the third author is partly supported by FRC/07-08/1-59, Hong Kong Baptist University.

References


