Cost-Effective Spam Detection in P2P File-Sharing Systems

Dongmei Jia
Illinois Institute of Technology
10 W 31st Street
Chicago, IL 60616
1-312-567-5330
jia@ir.iit.edu

ABSTRACT

Spam is highly pervasive in P2P file-sharing systems and is difficult to detect automatically before actually downloading a file due to the insufficient and biased description of a file returned to a client as a query result. To alleviate this problem, we propose probing technique to collect more complete feature information of query results from the network and apply feature-based ranking for automatically detecting spam in P2P query result sets. Furthermore, we examine the tradeoff between the spam detection performance and the network cost. Different ways of probing are explored to reduce the network cost. Experimental results show that the proposed techniques successfully decrease the amount of spam by 9% in the top-200 results and by 92% in the top-20 results with reasonable cost.

Categories and Subject Descriptors

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

General Terms

Measurement, Experimentation, Security

Keywords

P2P search, spam, detection

1. INTRODUCTION

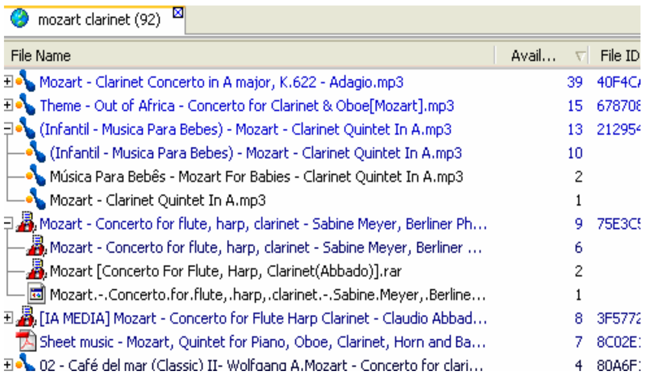
Spam is a well-known problem in P2P file-sharing systems, due to their anonymous, decentralized and dynamic nature [1][2][3][8]. A 2005 study observed that more than 50% of the matching results of popular recent songs were spam [3]. To improve the usability of P2P file-sharing systems, it is important to develop effective spam detection techniques.

Spam is defined as any file that is deliberately misrepresented or represented in such a way as to circumvent established retrieval and ranking techniques. One reason spam is so prevalent in P2P file-sharing systems is most shared files are not *self-describing*. Shared files are often binary media files that are identified by users by their filenames. A spammer can easily rename a file to manipulate how it is retrieved and ranked. For example, the music/movie industry has been injecting large amounts of spam into the network by naming them after real songs/movies in the battle against the illegal distribution of copyrighted materials [2][3].

Spam is harmful to P2P file-sharing systems in several ways. First, it degrades user experience. Second, spam may contain malware that, when executed, could destroy a computing system. Third, its transfer and discovery waste a significant amount of network and computing resources.

The naïve approach for identifying spam is to download the file and then examine its contents. If the file turns out to be spam, it can be reported on centralized databases (e.g., Bitzi [17]). The obvious problems with this approach are that it consumes time and computing resources and can release malware onto the client machine.

P2P spam is difficult to detect automatically before actually downloading a file due to the insufficient and biased description of a file returned to a client as a query result. Consider a search on Emule for "mozart clarinet" as shown in Figure 1. Returned are references to files whose descriptors contain both query terms. The information of matched replica returned to client include file name (labeled as File Name in Figure 1) as part of its descriptor, unique file hash key (labeled as File ID). Replicas are identified and grouped by hash key and results are ranked simply by group size – the number of replicas in a grouped result (labeled as Avail…). From this screenshot, we can clearly see that it is difficult to identify if a returned result is spam based on only its descriptor and group size ranking. It is possible that the result ranked at the top is actually a spam result (i.e., a totally different song) though its filename matches with the query.


mozart clarinet (92)

File Name	Avail…	File ID
Mozart - Clarinet Concerto in A major, K.622 - Adagio.mp3	39	40F4C
Theme - Out of Africa - Concerto for Clarinet & Oboe[Mozart].mp3	15	67870
(Infantil - Musica Para Bebes) - Mozart - Clarinet Quintet In A.mp3	13	21295
(Infantil - Musica Para Bebes) - Mozart - Clarinet Quintet In A.mp3	10	
Música Para Bebês - Mozart For Babies - Clarinet Quintet In A.mp3	2	
Mozart - Clarinet Quintet In A.mp3	1	
Mozart - Concerto for flute, harp, clarinet - Sabine Meyer, Berliner Ph…	9	75E3C
Mozart - Concerto for flute, harp, clarinet - Sabine Meyer, Berliner …	6	
Mozart [Concerto For Flute, Harp, Clarinet(Abbado)].rar	2	
Mozart.-.Concerto.for.flute,.harp,.clarinet.-.Sabine.Meyer,.Berline…	1	
[IA MEDIA] Mozart - Concerto for Flute Harp Clarinet - Claudio Abbad…	8	3F577
Sheet music - Mozart, Quintet for Piano, Oboe, Clarinet, Horn and Ba…	7	8C02E
02 - Café del mar (Classic) II- Wolfgang A.Mozart - Concerto for clari…	4	80A6F

Figure 1. Results for "mozart clarinet" on eMule

The ultimate goal of this work is to improve the search result quality in P2P file-sharing systems by automatically identifying spam in query result set without having to download candidate files. To this end, we propose feature-based ranking techniques based on our observations on real P2P spam data crawled from Gnutella network, and probing techniques to gather more complete feature information of query results so as to assist the proposed ranking functions for better detecting spam results.

Our proposed spam detection also requires little new functionality in existing P2P file-sharing systems. Rather, it relies on captured statistics to detect spam. Our results on Gnutella trace data show that we can decrease the amount of spam by 9% in the top-200 results and by 92% in the top-20 results compared with the base case. Furthermore, the factor increase in the network cost introduced by the proposed probing technique can be successfully reduced from 6.7 to 2 over the base case of no probing.

Compared with our previous work [9] that focuses on spam and spammer characterization, this paper talks about using the P2P features that are strongly correlated with spam as ranking functions assisted with probing technique for automatically detecting spam in P2P query result sets. The tradeoff between the spam detection performance and the network cost introduced by probing is closely examined. Different ways of probing are explored to reduce the network cost in this work.

2. RELATED WORK

The most widely recognized form of spam is email spam, also known as junk emails, which are unsolicited, nearly identical messages sent in large quantities to numerous recipients with the purpose of commercial advertising or spreading viruses or other malware. Many popular automated email spam detection methods filter and block email spam by analyzing their content and/or syntax, storing DNS-based blacklists of known spammers' IP addresses or constructing social networks for email addresses. For instance, [4] presents an approach that identifies semantic patterns in emails and then classifies them by applying a back-propagation neural network. [5] proposes MailRank, a social network-based approach, to rank and classify emails according to the address of email senders. It collects data about trusted email addresses from different sources and then creates a graph for the social network via email interactions. However, global information is needed to compute a ranking score for each email address. It is not resilient to attacks from cooperating spammers who build their own social networks. In general, these techniques are not applicable to P2P spam detection because P2P query results are often hard to distinguish (their filenames are relatively short and contain all query terms) and they require global information and the tight integration of users, which is assumed to be infeasible in general P2P environments.

Another well-known type of spam is Web spam – Web pages that are unrelated to the query that appear in search engine results. Many studies have been conducted on detecting Web spam. A variety of methods identify spam pages by analyzing either the content or link structure of Web pages [6][7][25][27]. Again, due to the dynamic and distributed nature of P2P file-sharing systems and the fact that shared files are only represented by small, user-defined file descriptors, Web spam detection techniques are not applicable in the P2P scenario.

The P2P spam detection technique proposed in [3] identifies shared music files as spam if the files are either non-decodable (unplayable) or their lengths are not within +10% or -10% of the official CD version. These techniques require downloading the file and only works for commercial music files whose official CD length is known. Judgments of shared music files encoded in other formats cannot be made. The general idea of using a file size, however, is similar to our feature-based spam detection. Because the file size feature has already been considered, we do not consider it in this work.

A spam filter was introduced to LimeWire's Gnutella [10] at the end of 2005. A user can mark a search result that is not relevant to his query or appears to be a virus as junk. Over time, the filter learns from peers that mark search results as junk, and updates the 'rating' of each result accordingly. A result with a high rating is more likely to be considered spam [24]. Compared with this user-controlled approach, our work does not rely on previous user judgments and takes a different approach on automatically detecting spam results.

Several works rely on the experience of other peers with shared files to detect spam without having to download the files. [11][12][16] build reputation systems to allow peers to rank each other, so that peers identified as malicious are less able to share files. However, the success of this mechanism is determined by the honesty level of peers. Instead of judging peers, [13] proposes that individual files be judged by users. The authenticity of a file is evaluated by having the client collect its judgments and evaluate them based on a credibility judgment of the client from which the judgments come. This system requires each peer maintain a vote database for the purpose of vote matching, which may not be scalable in a large system, is resource–intensive, and may be unreliable in environments where peers anonymously join the network for only short periods of time.

3. QUERY PROCESSING SPECIFICATION

In typical P2P file-sharing systems (e.g., Limewire's Gnutella) peers collectively share a set of (binary) files by maintaining local replicas of them. Each replica is represented by a user-tuned descriptor, which includes a filename, some embedded descriptive information (e.g., ID3 data embedded in mp3 files [19]) as well as an identifying *key* (e.g., a SHA-1 hash on the file's bits). All replicas of the same file naturally share the same key. The query processing includes the following major steps:

1. A client issues a query and routes it to all reachable servers until the query's time-to-live expires.
2. A server compares the query to its local replicas' descriptors; a query *matches* a replica if its descriptor contains all of the query's terms. (This is known as "conjunctive" query processing or query matching.)
3. On a match, the server returns its system identifier and the matching replica's descriptor to the client.
4. The client groups individual results by key. Each group is represented by a group descriptor, which is the aggregation of all the result descriptors the group contains.
5. The client ranks each group in the result set by a specific ranking function – generally by the number of results in the group (numRep) in decreasing order.

6. The client becomes a server for the file that is downloaded. The new file is a replica of one of the servers that returned a result in the result set.

4. A CLASSIFICATION OF SPAM

We classify P2P spam to organize our approaches for their detection. Each class of spam is distinct in how their creators attempt to disseminate them. These differences allow us to tailor the various techniques used to detect them.

To classify P2P spam and design and evaluate our spam detection algorithms, we use a collection of "metadata" from 25,137,217 P2P audio files, of which 9,575,113 are unique, shared by 226,786 peers in Gnutella network. The shared data were collected by browsing peers' shared folders using our IR-Wire crawling tool [14] in the Spring of 2007. The information (i.e., metadata) we recorded for each file includes the filenames, unique identifiers (or *keys*) of files (i.e., SHA1 hash on file's bits), peer identifications (i.e., IP addresses) and file types.

4.1 Classes of P2P Spam

As stated in Section 1, we consider as spam any file that is altered to manipulate the P2P file-sharing system's retrieval or ranking functions. By this definition, a virus file named "spiderman-movie.dvi" is spam, whereas the same file named "virus.exe" is not. Regardless, spammers find ways to place these results at the top of a user's search results.

Spamming is generally performed by manipulating Steps 1, 3 and 5 of the query processing specification stated in Section 3. These steps control who processes the queries, what results are returned to the client and how they are ranked in the result set. Step 1 can be manipulated by placing highly active peers in the network that actively participate in file sharing (e.g., see Overpeer [22]). Reputation systems discussed in Section 2 address this problem so it is not in the scope of our work. Rather, we focus on the spam that manipulates query processing Steps 3 and 5, which focus on identifying spam that are in the query result sets sent to clients.

Many of the classes of the P2P spam have analogs in Web spam. These analogs are similar in general approach and we point them out where appropriate.

In our data analyses and our experience using P2P file-sharing systems, we have identified four types of spam.

1. Files whose replicas have semantically different descriptors.

In this case, a spammer tries to disseminate widely a file by replicating it and creating different descriptions for each. A spammer might name a file after a currently popular song. An example of this type of spam found in our crawled audio data set is the file with key 26NZUBS655CC66COLKMWHUVJGUXRPVUF. This file's replicas have descriptors that contain various song titles that refer to several distinct songs, including '12 days after christmas.mp3', 'Niche- Oops Oh My.mp3', 'i want you thalia.mp3' and 'comon be my girl.mp3'. Notice that each of these descriptors looks normal in terms of size and combination of terms. It is only when we compare the descriptors of different replicas does it become clear that the shared file is likely spam.

Type 1 spam has many analogs to Web spam, including Web "content spam" techniques, such as keyword stuffing [25] or Web site scraping [26]. These techniques add popular terms to Web sites to increase their visibility in search results. Type 1 spam is also similar to the Web link spam practice of "page hijacking," where a well-known Web site is copied, but then redirects a user to spam content [27].

2. Files with long descriptors that contain semantically non-sensical terms combinations.

Here, a spammer creates a single file that matches a large class of queries by putting popular terms in their descriptors. This type of spam is different from the first type, as terms in the descriptor together do not attempt to represent an existing file.

For example, the descriptor of a single replica of file key 1200473A4BB17724194C5B9C271F3DC4 is 'Aerosmith,Van Halen,Quiet Riot,Kiss, Poison, Acdc, Accept, Def Leappard, Boney M, Megadeth, Metallica, Offspring, Beastie Boys, Run Dmc, Buckcherry, Salty Dog Remix.mp3.'

Type 2 spam is similar to Type 1 spam with the difference that the additional keywords used to boost the file's ranking are added to a single descriptor instead of being spread out over the descriptors of several replicas.

3. Files with descriptors that contains no query terms.

A server wishing to share a particular file may return the file regardless of whether it matches the user's query. For instance, the result could be advertisements or a warning on the illegality of downloading of copyrighted materials, such as files with the descriptor, "Can you afford 0.09 www.BuyLegalMP3.com.mp3".

Type 3 spam falls under the category of query-independent spam because it manipulates query results independent of the query. On the Web, link spam does the same thing. For example, link farms' goal is to increase the strength of association that of a Web site has with a particular term set [23].

4. Files that are highly replicated on a single peer.

We assume that normal users do not create multiple replicas of the same file on a single server and that the only reason this is done is to manipulate the "group size" ranking technique used on most P2P file-sharing clients or to retard the query routing techniques used to route queries for hard-to-find content [28]. Although the file may be correctly described, because its replication manipulates the ranking function, it is by definition spam.

For instance, in our dataset, all of the 177 replicas of the file with key 6DY2QXX3MYW75SRCWSSUG6GY3FS7N7YC are shared on a single peer.

Type 4 spam is analogous to "duplicate content" Web spam, which aims to increase a site's association with some content terms by duplicating its instances of this content [18][21]. It is also similar to the link-farming technique used by Web spammers to increase a Web site's PageRank.

Among the 4 types of spam, Types 2 and 3 should be easy for any query-dependent similarity-based ranking function (e.g., Cosine similarity ranking) [20] to identify directly and rank low. Hence, they may not be widely spread in the network and would be less harmful. However, their inclusion in query result sets does degrade the user's search experience and wastes network and computing resources.

Types 1 and 4 spam are more difficult to detect because they appear to match the query and be described with a sensible combination of terms. We propose automatic ways of detecting them in this work so as to avoid downloading files from spammers.

5. "FEATURES" OF P2P SPAM

In our previous work on spam characterization [9], we investigate the correlations between prevalence of spam and several features of shared files and the peers that share them. Here lists several features that are examined to be strongly correlated with the possibility of spam in [9]:

- Vocabulary size of a file's group descriptor (numUniqueTerms): A file group descriptor is the aggregation of all the replica descriptors of the file. The group descriptor's vocabulary size is the number of unique terms it contains. A larger than normal vocabulary suggests that the file has different descriptions that allow it to match unrelated queries.
- Variance of terms in replica descriptors of a file: High variance in the descriptors of different replicas may indicate an attempt to match several unrelated queries.

  Descriptor variance can be measured by average Jaccard or Cosine distance between replica descriptor D_i and file group descriptor G.

  – Jaccard: The Jaccard distance between a single replica descriptor D_i and file group descriptor G is defined as:

  $$1 - |D_i \cap G| / |D_i \cup G|$$

  Since the term set of replica descriptor D_i is always a subset of the group descriptor G, the second term is equal to $|D_i|$ / $|G|$, which can be interpreted as the ratio of number of terms in replica descriptor to total number of terms in group descriptor, without considering term frequencies. (Term frequency consideration is a major difference between Jaccard and Cosine distance.)

  – Cosine: The cosine distance between replica descriptor D_i and file group descriptor G is defined as:

  $$1 - (V_G \cdot V_{Di}) / (|V_G|\ |V_{Di}|)$$

  where G and D_i are modeled as term frequency vectors (V_G of length $|V_G|$ and V_{Di} of length $|V_{Di}|$). This indicates the degree of dissimilarity between the two term frequency vectors. A high Jaccard or Cosine distance indicates a high variance in the descriptors of different replicas of a file. Notice that Jaccard and Cosine distance only apply to files with multiple replicas (numRep>1).
- Per-host replication degree of a file (repPerHost): This represents how replicas of a file are distributed among peers. We consider a file with a high repPerHost to be abnormally distributed, which may indicate an attempt to manipulate group size ranking on the client. repPerHost is computed as numRep / numHost for a file.

Since these P2P features are good indicators of spam, they are used as ranking functions to detect spam, which will be explained in the next section.

6. FEATURE-BASED SPAM DETECTION

Our goal is to improve P2P search accuracy by automatically identifying spam in P2P query result sets without requiring a user to download any files. We take the basic query processing steps outlined in Section 3 and make modifications to them to accommodate spam detection.

6.1 Detecting Types 2 and 3 Spam

Current P2P clients rank search results typically based on server or file quality such as available bandwidth or relative popularity of the file (i.e., group size). Recent work demonstrates that randomly ranking search results is substantially more reliable than these ranking functions [15].

Group size ranking is not effective on identifying spam results, as the number of copies of a spam file may exceed the number of an authentic file in result set. The experiments on the numRep feature conducted in [9], which show the unreliability of replication degree in identifying spam, is more evidence of the failures of group size ranking in this regard.

To identify Types 2 and 3 spam, we propose a straight-forward application of query-dependent IR-ranking techniques. Types 2 and 3 spam are characterized by several terms that are irrelevant to the user's query. Group size ranking, by being query independent, does not identify such spam. However, query-dependent ranking functions [20], such as Cosine similarity or Okapi BM25 naturally identify Types 2 and 3 spam.

The modification we make to the steps of query processing to detect Types 2 and 3 spam is replacing step 5 with the following:

5a. Groups are ranked by cosine similarity (or some other query-dependent ranking function) in decreasing order.

6.2 Detecting Types 1 and 4 Spam

Types 1 and 4 spam are not detectable with query-dependent because they naturally resemble the query. In this case, we make use of the features identified above to identify spam.

Recall that Type 1 spam is characterized by variance among the descriptors of its replicas. This type of spam is identifiable by the following two features:

- Vocabulary size of a group (numUniqueTerms).
- Variance of replica descriptors of a group (Jaccard or Cosine distance).

Type 4 spam is characterized by its high replication degree per peer. This type of spam is identified by the following feature:

- Per-host file replication degree (repPerHost).

To integrate these features into the query processing steps, we propose the following steps after Step 5a and before Step 6:

5b. Identify the top-*M* results as *candidate* results.
5c. Re-rank the top-*M* results by either NumUniqueTerms or Jaccard/Cosine distance in increasing order. The results that are low in the order are more likely to be Type 1 spam than those higher up.
5d. Identify the top-*N* results, where *N* < *M* as the new *candidate* results.
5e. Re-rank the top-*N* results by their per-host file replication degree in increasing order. The results that are low in the order are more likely to be Type 4 spam than those higher up.

Step 5b isolates candidate results from the first ranking and Step 5c re-ranks them to identify Type 1 spam in within the candidates. Similarly, Steps 5d and 5e identify Type 4 spam within these results. The values of N and M are system-defined parameters, where smaller values result in lower cost, but also lower recall. Anecdotal evidence suggests that N=20 is a reasonable constraint [30].

6.3 Probe Queries to Enhance Spam Detection

One of the challenges in detecting spam is that the query results will tend to look alike due to the conjunctive matching condition. For example, one of our proposed methods for detecting Type 1 spam is to identify variance among the replicas' descriptors. Yet, conjunctive matching only retrieves the replicas of a file with descriptors that resemble the query (and therefore resemble each other), while not retrieving replicas of the same file with very different descriptors.

To solve this problem created by conjunctive matching, we propose the use of "probe queries," [29] which, given a file's key, searches for its feature information relevant to spam detection from other peers in the network.

A probe query contains only the key of a result file and is sent to peers who share this file in the network. A peer responding to a probe query sends back local descriptor(s) of the probed file, the total number of replicas, the number of unique files shared on this peer and the identifier of the peer.

By issuing a probe query for a file, we create a more complete view of how a file is shared. This information (e.g., descriptors that do not match the original query, servers who act like spammers) is used to identify it as spam.

To integrate probing into the query processing steps described above, we insert the following step after Step 5b:

5b'. Issue probe queries for the top-*M* results.

The information collected from the probe query issued for the candidate results will help in determining whether they are Types 1 or 4 spam.

6.4 Simulating P2P Search

To evaluate the effectiveness of the proposed techniques, we simulate a P2P search on a client's perspective using the set of data we crawled from Gnutella network as described earlier in Section 4. On these data, we issue the top 50 most popular queries for audio files that we identified from our crawled data. We use these queries as they are representative of the most users and likely targets for spam.

The client issues the basic 6 steps outlined above for query processing, with variations based on the experiment. To simulate P2P query routing, without loss of generality, a query is randomly sent to a given number of peers (i.e., 50 peers) who return matching results. This process repeats until the number of results returned to the client reaches a given threshold (i.e., 200 results) or a threshold number of peers have received the query (i.e., 50,000 peers). Threshold values were chosen based on the specifications of a real-world P2P file-sharing system (i.e., LimeWire's Gnutella [10]).

To the best of our knowledge, no benchmark P2P data set currently exists, esp. the lack of spam judgment, hence, for the purpose of spam evaluation, we manually judge the retrieved results for each of the 50 queries and label them as spam or non-spam based on the collected file information (e.g., terms in file descriptors) from Gnutella network as well as information retrieved by looking up the file (identified by its key) on Bitzi [17]. For instance, one file in the top 50 peer dataset with different descriptors 'jamie kennedy - matress mack.mp3' 'ball busters prank calls.mp3' and key KVGBBGVZYJ7BFPJBFYIVPAWKNEHMPDKX is labeled as a 27-second audio advertisement of 'efreeclub.com' on Bitzi. Another example of spam in our dataset is file with key QFX3NMHJMOGG7VF7IK5AOFST2L3EWKCA and different filenames such as 'brothers boots when she made me promise.wma', 'earth its easy for u to say.wma' is rated on Bitzi as "Dangerous/Misleading" and one user comment for this file is "Downloaded accidentally - clearly a virus".

Performance is measured using a standard metric – the number of the top *N* ranked results that is spam, especially when *N* is small, as a user tends to look at only a few top-ranked results.

6.5 Experimental Results

To test the proposed probing and ranking techniques, we compare them with the two no probing (noprobe) base cases where group size (numRep) ranking and Cosine similarity (CosineQD) ranking are performed.

As discussed earlier, spam Types 2 and 3 (i.e., a file containing many random noisy words in a replica descriptor) are identified by any query-dependent, content-based similarity ranking such as Cosine similarity even with no probing, as terms in this type of spam's descriptors are irrelevant to query. Spam Type 4 (i.e., a file highly replicated on a single peer) is detected by the proposed ranking – per-host file replication degree (repPerHost), which can be easily computed based on the statistics (i.e., number of replicas of a file, number of peers who share a file) obtained by probing. Hence, in our experiments, we focus on examining how the proposed content-based file "quality" ranking functions with the assist of probing perform on the detection of Type 1 spam.

Figure 2 presents the average amount of spam in the top *N* result sets of 35 of the 50 queries. (Fifteen queries returned no spam, so they are excluded from the performance analysis.) Compared with the two no-probing base cases, the proposed probing-based ranking functions (Cosine, Jaccard and numUniqueTerms) are better at ranking spam low in result set, especially when the value of top *N* is small. For instance, compared with the base case, noprobe+numRep, probe+Cosine improves the performance by 9% over all results and by 92.5% for the top-20 results. Compared with the base case noprobe+CosineQD, the two numbers are 21.6% and 97.8% respectively.

We also examine how numRep performs in the case of probing. The results show that probe+numRep performs the worst, which suggests that group size has trouble detecting spam results, especially when such a spam file is widely spread in the network.

As shown in Figure 2, the ranking function numUniqueTerms seems to perform better than Cosine and Jaccard when the value of top *N* is larger. The reason for this is Cosine and Jaccard distance can only be computed for files with multiple replicas; however, the number of unique terms of a file can be computed even if there is only a single replica of a file.

In order to compare Cosine, Jaccard with numUniqueTerms in a fair way, we consider only multi-replica result files in ranking, and recomputed the average number of spam in top-N results. As shown in Figure 3, Cosine and Jaccard performs consistently better than numUniqueTerms in the case of multi-replica files.

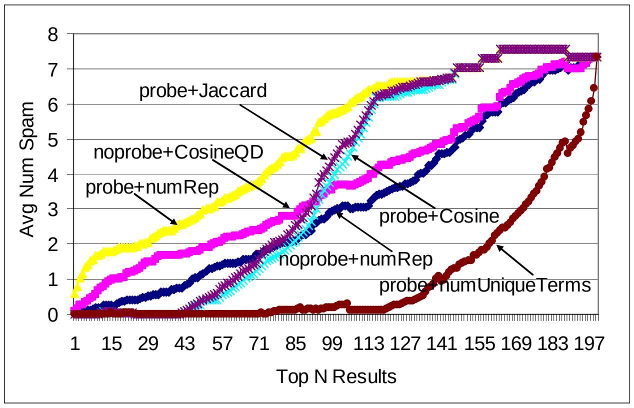


Figure 2. Number of spam in top-N results with various ranking and probing techniques

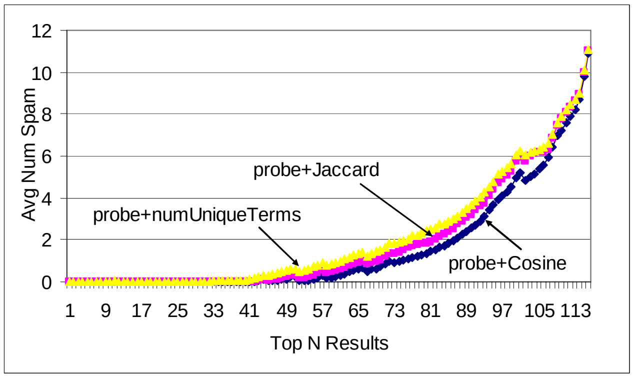


Figure 3. Number of spam in top-N results with various ranking functions (only multi-replica files are considered)

7. Cost Analysis

There is a tradeoff between the cost introduced by probing and the performance of spam detection. Probing on query results dramatically increases the network cost. Because there are two types of queries – regular file queries for files, and probe queries for file feature information, there are two cost components – regular cost and probe cost, which can be roughly measured by the number of query responses for regular file query and probe query respectively. However, we argue that, compared with the cost on downloading large media spam due to user's unawareness, it may worth to apply probing to filter spam out in advance.

To better control the probing cost, we propose three ways of reducing cost: random sampling of probe query results, piggybacking of descriptor data in probe queries, and limiting the scope of probing to a few top-ranked results in a query result set.

7.1 Random Sampling of Query Results

Our first strategy is to use server-side *Bernoulli sampling* on the result set for probe query. That is, for each matching probe query result, the server decides to return it to the client with a fixed probability P_p, $0 \le P_p \le 1$. Predictably, this type of sampling can reduce the cost by a factor P_p. The question is what impact such sampling has on the effectiveness of spam detection.

Figure 4 and Figure 5 demonstrate the performance and cost of various probe query sampling rates. In the case of probing, we apply Cosine to rank probed results as it performs the best among all the proposed ranking functions in terms of detecting spam as described above. In Figure 4, compared with noprobe base case where retrieved results are ranked by group size (numRep), the overall performance improvement is 9%, 8.4%, 3.7% and 1.7% with sampling rates 1, 0.75, 0.5 and 0.25 respectively. This result indicates that sampling has a negative impact on the performance, because with less description information, it is harder to distinguish between non-spam and spam results based on variance among replica descriptors. However, luckily, for all the sampling rates, the performance with probing and Cosine ranking is always higher than the base case, especially for a small subset of top-N results (i.e., top-20), the performance is improved by 71%-92% with various sampling rates.

Figure 5 shows that the average total cost over the 35 queries is reduced significantly by sampling fewer probe results. However, compared with the noprobe base case, the cost introduced by probing is fairly high. For example, the total cost with sampling rate 0.25 is almost 7 times higher than that of the base case.

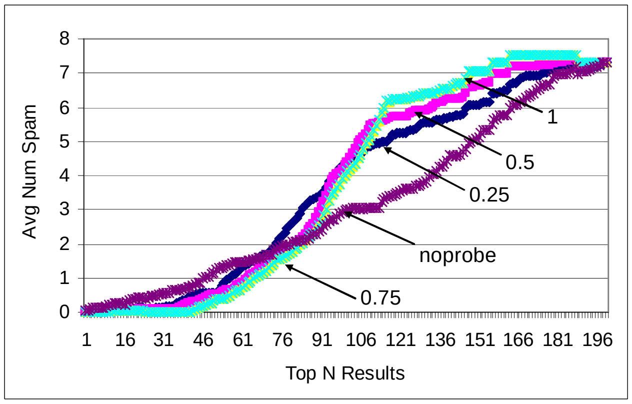


Figure 4. Number of spam in top-N results with various probe query sampling rates

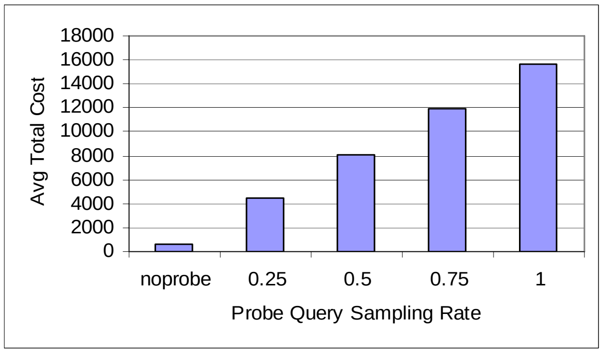


Figure 5. Total cost for various probe query sampling rates

7.2 Piggybacking Descriptor Data in Probe Queries

To further reduce the cost, we propose piggybacking the descriptor of the result file being probed onto its probe query. Hence, this new type of probe query contains both file key and the descriptor of regular query result. A peer in the network who receives such a probe query uses the piggybacked descriptor to determine if a response is appropriate. In particular, if the server's descriptor of the probed file cannot introduce any new term to the descriptor of the result file on the probing peer, then it does not respond to the probe, so as to limit the number of probe results returned to client.

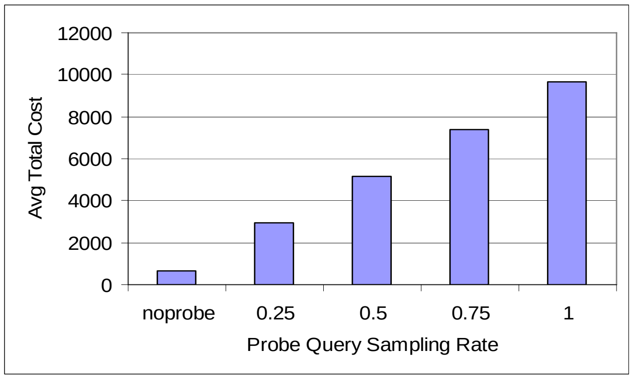


Figure 6. Total cost for probing with piggybacked descriptor with various sampling rates

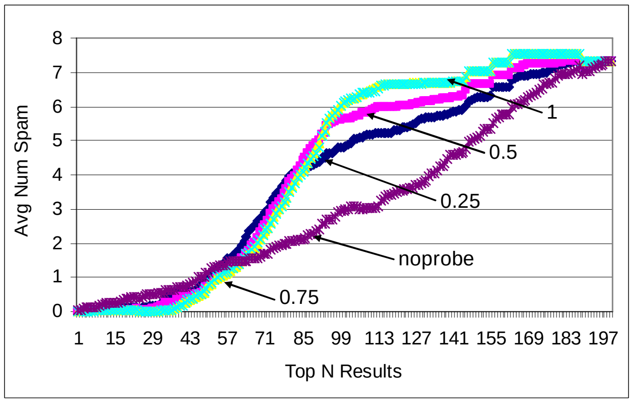


Figure 7. Number of spam in top-N results for probing with piggybacked descriptor with various sampling rates

Figure 6 presents the cost for probe query containing both descriptor data and file key with various probe sampling rates. Compared with probe query containing only file key shown in Figure 5, this new type of probe query successfully decreases the total cost by 35%-39% for a same sampling rate.

Probing with piggybacked descriptor degrades the overall performance by approximately 15% in all the sampling cases in Figure 7. This is not surprising, because with fewer probe responses, the proposed ranking functions (e.g., Cosine) that target on spam with high variance in replica descriptors is less effective. In other words, non-spam results may not be ranked higher than spam results, as in the probing process, replicas of a non-spam file that have consistent descriptors with the piggybacked descriptor will not be returned as probe results. Hence, the Cosine ranking score of a non-spam result may be lower than it is supposed to be.

However, we observe in Figure 7 that probe+Cosine always beats the noprobe base case when the value of top *N* is small. For instance, the performance for the top-20 results is 71%, 82%, 84% and 88% better than the base case when the sampling rate is increased from 0.25 to 1. The performance improvement (71%-88%) is similar to that (71%-92%) of probe query containing only file key in Figure 4, with almost 40% less cost. This is promising, as a user tends to look at only a few top-ranked results in general.

7.3 Limiting Scope of Probing

As mentioned above, a user may only consider downloading a file from a few top-ranked results. We explore another way of reducing the cost by limiting the number of result files to be probed. Previously, probe queries are issued for all the results (i.e., 200 files) for each query. In this set of experiments, probing is restricted to only the top-20 results in regular query result set ranked by group size (numRep), the most popular ranking in commercial P2P file-sharing systems. In other words, the intension of this method is to identify spam in only a subset (e.g., top-20) of regular query results that have high possibility to be downloaded by user, as well as to reduce the probing cost.

We rerun the experiments presented in section 7.1 for only the top-20 results. Similarly, the performance gets better by probing more, as shown in Figure 8. The performance of probing with sampling is always 22%-56% better over the noprobe base case. Notice that queries that do not have spam in the top-20 regular query result sets before probing are excluded for the performance and cost evaluation. As expected, probing only the top-20 results significantly reduces cost. For instance, as shown in Figure 9, the total cost with probe sampling rate 0.25 is only twice as much as the cost in the base case without probing. Compared with that, the factor increase in cost is 6.7 in the case of probing all the results (Figure 5) and 4.4 in the case of probing with piggybacked descriptor data (Figure 6).

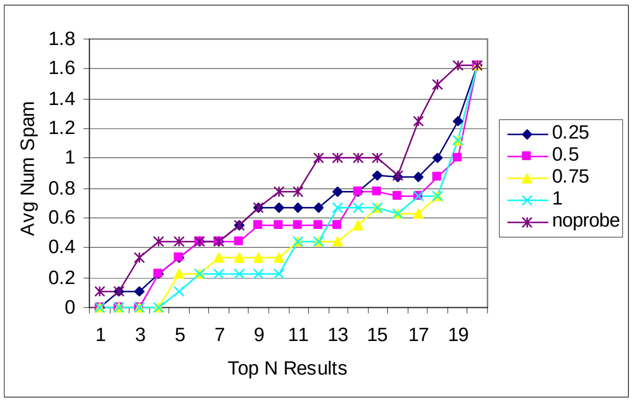


Figure 8. Number of spam in top-20 results with various probe query sampling rates

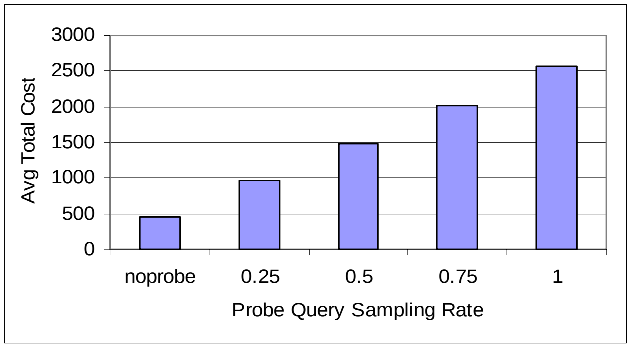


Figure 9. Total cost for probing top-20 results with various sampling rates

8. CONCLUSIONS

Insufficient and biased description of a file returned to a client as a query result makes it difficult to detect spam automatically before actually downloading a file. We propose probing technique that aggregate more descriptive information of result files and statistics of peers and ranking by P2P features that are strongly correlated with spam, such as vocabulary size, variance of replica descriptors, and per-host replication degree of a file. The experimental results show that the proposed rankings assisted with probing improve the ability to detect spam by 92.5% over the top 20 results. Different ways of reducing the cost are explored. The results show that the cost is successfully reduced to a reasonable range (e.g., twice as much as the cost of no probing base case).

We are currently working on ways of combining features into single "spam probability" scores to boost accuracy. We are also working on identifying other possible types of P2P spam and better controlling the cost.

9. REFERENCES

[1] S. Shin, J. Jung, H. Balakrishnan. Malware Prevalence in the KaZaA File-Sharing. Network. *In Proc. of the Internet Measurement Conference (IMC)*, ACM 2006.

[2] N. Christin, A. S. Weigend and J. Chuang. Content Availability, Pollution and Poisoning in Peer-to-Peer File Sharing Networks. I*n ACM E-Commerce Conference (EC'05)*, 2005.

[3] J. Liang, R. Kumar, Y. Xi and K. Ross. Pollution in P2P File Sharing Systems. *In Proc. of INFOCOM'05*, May 2005.

[4] R. Hashemi, M. Bahar, K. D. Tift, and H. Nguyen. Spam Detection: A Syntax and Semantic-based Approach. *In proc. International Conf. on Information and Knowledge Engineering (IKE'06)*, Las Vegas, Nevada, June 2006.

[5] P. A. Chirita, J. Diederich, and W. Nejdl. MailRank: Using ranking for spam detection. *In proc. CIKM'05*, Bremen, Germany, 2005.

[6] Qingqing Gan and Torsten Suel. Improving Web Spam Classifiers Using Link Structure. *In Third International Workshop on Adversarial Information Retrieval on the Web (AIRWeb'07)*, Banff, AB, Canada, May 2007.

[7] A. Ntoulas, M. Najork, M. Manasse, D. Fetterly. Detecting spam web pages through content analysis. *In Proc. of WWW'06*.

[8] J. Liang, N. Naoumov, K. Ross. The Index Poisoning Attack in P2P File Sharing Systems. *In proc. of INFOCOM*, Barcelona, Spain, Apr. 2006

[9] D. Jia, W. G. Yee, O. Frieder. Spam Characterization and Detection in Peer-to-Peer File-Sharing Systems. In Proc. ACM 17th Conference on Information and Knowledge Management (CIKM'08), Napa Valley, California, Oct. 2008.

[10] Limewire. www.limewire.org

[11] D. Dutta, A. Goel, R. Govindan, H. Zhang, The Design of A Distributed Rating Scheme for Peer-to-peer Systems, *In Proc. of Workshop on the Economics of Peer-to-Peer Systems*, 2003

[12] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. *In Proc. of the Twelfth International World Wide Web (WWW) Conference*, May, 2003.

[13] Kevin Walsh, Emin Gun Sirer. Experience with an Object Reputation System for Peer-to-Peer Filesharing. *In 3rd Symposium on NSDI*, 2006.

[14] L. T. Nguyen, W. G. Yee, D. Jia, and O. Frieder, A Tool for Information Retrieval Research in Peer-to-Peer File Sharing Systems, *In Proc. IEEE ICDE*, 2007.

[15] D. Dumitriu, E. Knightly, A. Kuzmanovic, I. Stoica and W. Zwaenepoel. Denial-of-Service Resilience in Peer-to-Peer File Sharing Systems. *In Proc. Of ACM SIGMETRICS'05*, Banff, AB, Canada, June 2005.

[16] Runfang Zhou and Kai Hwang. Gossip-based Reputation Aggregation for Unstructured Peer-to-Peer Networks. *21th IEEE International Parallel & Distributed Processing Symposium (IPDPS'07)*, Los Angeles, March 26-30, 2007

[17] Bitzi website. www.Bitzi.com

[18] Google Duplicate Content Web Site. http://www.google.com/support/webmasters/bin/answer.py?answer=66359. Accessed May 25, 2008.

[19] M. Nilsson. Id3v2 web site. www.id3.org.

[20] D. Grossman and O. Frieder. Information Retrieval: Algorithms and Heuristics. Springer, second edition, 2004.

[21] Steve Webb, J. Caverlee, and C. Pu. Characterizing Web Spam Using Content and HTTP Session Analysis. In *Proc. 4th Conf. on Email and Anti-Spam (CEAS)*, 2007.

[22] J. Macguire. Hitting P2P Users Where It Hurts, In Wired, Jan. 13, 2003. http://www.wired.com/entertainment/music/news/2003/01/57112

[23] Googlebombing 'failure.' Official Google Blog. Sept. 16, 2005. http://googleblog.blogspot.com/2005/09/googlebombing-failure.html

[24] http://wiki.limewire.org/index.php?title=Junk_Filter

[25] K. Svore, Q. Wu, C.J.C. Burges and A. Raman. Improving Web spam classification using Rank-time features. In Proc. *AIRWeb workshop in WWW*, 2007

[26] http://en.wikipedia.org/wiki/Web_scraping#References

[27] J. Caverlee and L. Liu. Countering Web Spam with Credibility-Based Link Analysis. *In Proc. the 26th ACM Symposium on Principles of Distributed Computing (PODC)*, 2007.

[28] The Gnutella protocol specification v0.6. http://rfc-gnutella.sourceforge.net.

[29] D. Jia, W. G. Yee, L. T. Nguyen, O. Frieder. Distributed, Automatic File Description Tuning in P2P File-Sharing Systems. *Springer Journal of Peer-to-Peer Networking and Applications*, 2008.

[30] W. G. Yee, L. T. Nguyen, and O. Frieder. Improved Result Ranking in P2P File-Sharing Systems by Probing for Metadata. In Proc. IEEE NCA, 2006.