Combating Spam in Tagging Systems

Georgia Koutrika, Frans Adjie Effendi,
Zoltán Gyöngyi, Paul Heymann, Hector Garcia-Molina
Computer Science Department
Stanford University
koutrika@stanford.edu

ABSTRACT

Tagging systems allow users to interactively annotate a pool of shared resources using descriptive strings, which are called tags. Tags are used to guide users to interesting resources and help them build communities that share their expertise and resources. As tagging systems are gaining in popularity, they become more susceptible to *tag spam*: misleading tags that are generated in order to increase the visibility of some resources or simply to confuse users. Our goal is to understand this problem better. In particular, we are interested in answers to questions such as: How many malicious users can a tagging system tolerate before results significantly degrade? What types of tagging systems are more vulnerable to malicious attacks? What would be the effort and the impact of employing a trusted moderator to find bad postings? Can a system automatically protect itself from spam, for instance, by exploiting user tag patterns? In a quest for answers to these questions, we introduce a framework for modeling tagging systems and user tagging behavior. We also describe a method for ranking documents matching a tag based on taggers' reliability. Using our framework, we study the behavior of existing approaches under malicious attacks and the impact of a moderator and our ranking method.

1. INTRODUCTION

Tagging systems allow users to interactively annotate a pool of shared resources using descriptive strings, which are called tags. For instance, in Flickr [4], a system for sharing photographs, a user may tag a photo of his Aunt Thelma with the strings "Thelma", "Aunt", and "red hair". In Del.icio.us [3], users annotate web pages of interest to them with descriptive terms. In these and other tagging systems, tags are used to guide users to interesting resources. For instance, users may be able to *query* for resources that are annotated with a particular tag. They may also be able to look at the most popular tags, or the tags used by their friends, to discover new content they may not have known they were interested in. Tagging systems are gaining in popularity since they allow users to build communities that share their expertise and resources.

In a way, tags are similar to links with anchor text on the web. That is, if page p contains a link to page q with the anchor text "Aunt Thelma", this implies that somehow page q is related to Aunt Thelma. This would be analogous to tagging page q with the words "Aunt Thelma" (in a tagging system where web pages were the resources). However, a tagging system is different from the web. The latter is comprised of pages and links, while the former is comprised of resources, users and tags. These resources can be more than web pages, e.g., they can be photos, videos, slides, etc. Typically, in a tagging system, there is a well defined group of users and resources that can be tagged.

As we know, the web is susceptible to *search engine spam*, that is to content that is created to mislead searching engines into giving some pages a higher ranking than they deserve [13]. Web spam is a big problem for search engines, as well as a big opportunity for "search engine optimization" companies that for a fee generate spam to boost the customer's pages. In an analogous fashion, tagging systems are susceptible to *tag spam*: misleading tags that are generated to make it more likely that some resources are seen by users, or generated simply to confuse users. For instance, in a photo system, malicious users may repeatedly annotate a photo of some country's president with the tag "devil", so that users searching for that word will see a photo of the president. Similarly, malicious users may annotate many photos with the tag "evil empire" so that this tag appears as one of the most popular tags. In a system that annotates web pages, one shoe company may annotate many pages (except the page of its competitor) with the string "buy shoes", so that users looking to buy shoes will not easily find the competitor's page.

Given the increasing interest in tagging systems, and the increasing danger from spam, our goal is to understand the problem better and to try to devise schemes that may combat spam. In particular, we are interested in answers to questions like the following:

- How many malicious users can a tagging system tolerate before results significantly degrade? One or two bad guys are unlikely to bias results significantly, but what if 1% of the users are malicious? What if 10% are malicious? What if the malicious users collude? The answers to these questions, even if approximate, may give us a sense

of how serious a problem tag spam is or could be. The answers may also help us in deciding how much effort should be put into the process of screening bad users when they register with the system.

- What types of tagging systems are more prone to spam? Is a tagging system with a large number of resources less susceptible to spam than a system with a limited number of resources? A popular system attracts more users and possibly more spammers. Is popularity a blessing or a curse for tagging systems? Revealing and understanding weaknesses of existing systems is a step towards designing more robust systems.
- What is the impact of encouraging users to tag documents already tagged? Does encouraging people to post more tags help a system better cope with spam? In other words, assume users are discouraged from tagging a document with a tag, if it already has that tag. Will things then be easier for spammers?
- What can be done to reduce the impact of malicious users? For example, one could use a moderator that periodically checks the tags of users to see if they are "reasonable." This is an expensive and slow process; how effective can it be? How much effort would the moderator need to place in order to achieve a positive impact? How many users, documents or tag postings would it need to check?
- Is there a way to use correlations to identify misused tags? For instance, if we notice that a user always adds tags that do not agree with the tags of the majority of users, we may want to give less weight to the tags of that user. Would this make the system more resilient to bad users? What would be the downside of using correlations to detect bad users?

As the reader may suspect, answering these questions is extremely hard for a number of reasons. First, the notion of a "malicious tag" is very subjective: for instance, one person may consider the tag "ugly" on Aunt Thelma's photo to be inappropriate, while another person may think it is perfect! There are of course behaviors that most people would agree are inappropriate, but defining such behaviors precisely is not easy. Second, malicious users can mount many different "attacks" on a tagging system. For instance, if they know that correlations are being used to detect attacks, they can try to disguise their incorrect tags by posting some fraction of reasonable tags. What bad users do depends on their sophistication, on their goals, and on whether they collude. Bad users being a moving target, it is hard to know what we need to protect against.

Given these difficulties, our approach here is to define an *ideal tagging system* where malicious tags and malicious user behaviors are well defined. In particular, we generate tags with a synthetic model, where some fraction of the tags are malicious, and there are no ambiguous tags. Based on this model, we study how tag spam affects tag-based search and retrieval of resources in a tagging system. Of course, our query answering algorithms will not directly know which tags are misused, but when we evaluate query answering schemes we will know which answers were correct and which were not. Similarly, we will assume that malicious users use a particular, fixed strategy for their tagging. Again, the protection schemes will be unaware of the malicious user policy. A proper understanding of tag spamming can guide the development of appropriate countermeasures. For this purpose, we start with simple user models and see how a system behaves under naive attacks and how it can protect itself against them. Once we introduce safeguards against these naive bad users, the bad users may respond with more sophisticated attacks. But these sophisticated attacks can only be defined once we know the safeguards.

Given that we are using an ideal model, our results will *not* be useful for predicting how any one particular tagging system may perform. Nevertheless, our results can yield insights into the relative merits of the various protection schemes we study. That is, if scheme A is significantly better than scheme B at protecting against tag spam in the ideal system, then it is reasonable to expect that system A will perform better in practice. Similarly, understanding the level of disruption malicious users can introduce in an ideal system, may provide insights into what they can do in a real system: That is, one can interpret the ideal results as an "upper bound" on disruption, since in a real system the distinction between an incorrect result and a correct one will be less clear cut.

In summary, the contributions of this paper are:

- We define an ideal tagging system that we believe is useful for comparing query answering schemes and we model user tagging behavior (Section 3).
- We propose a variety of query schemes and moderator strategies to counter tag spam (Sections 4 and 5).
- We define a metric for quantifying the "spam impact" on results (Section 6).
- We compare the various schemes under different models for malicious user behavior. We try to understand weaknesses of existing systems and the magnitude of the tag spam problem. We also make predictions about which schemes will be more useful in practice (Section 7).

2. RELATED WORK

We are witnessing a growing number of tagging services on the web, which enable people to share and tag different kinds of resources, such as: photos (Flickr [4]), URLs (Del.icio.us [3]), blogs (Technorati [8]), people (Fringe [10]), research papers (CiteULike [2]), slideshows (Slideshare [7]), and so forth. Reference [1] provides links to several systems. Companies are also trying to take advantage of the social tagging phenomenon inside the enterprise [17, 9, 20, 5].

The increasing popularity of tagging systems has motivated a number of studies [25, 22, 11, 18, 19] that mainly focus on understanding tag usage and evolution. An experimental study of tag usage in My Web 2.0 has shown that people naturally select some popular and generic tags to label Web objects of interest [25]. In general, three factors seem to influence personal tagging behavior [22]: *people's personal tendency* to apply tags based on their past tagging behaviors, *community influence* of the tagging behavior of other members, and the *tag selection* algorithm used by the system for recommending "good" tags for a document to the candidate tagger. Community influence has been shown in

experimental studies of Del.icio.us [11] and Flickr [19]. In this paper, we take a first step towards understanding the magnitude and implications of spamming in tagging systems. Although spamming is directly related to tag usage, existing studies have not explicitly dealt with it. We believe that this fact underlines the importance and uniqueness of our study.

Harvesting social knowledge in a tagging system can lead to automatic suggestions of high quality tags for an object based on what other users use to tag this object (*tag recommendation* [25]), characterizing and identifying users or communities based on their expertise and interests (*user/community identification* [17]), building hierarchies of tags based on their use and correlations (*ontology induction* [21]), and so forth. We argue that leveraging social knowledge may help fighting spam. The Coincidence-based query answering method, which we will describe in Section , exploits user correlations to that end. To the best of our knowledge, only reference [25] takes into account spam by proposing a reputation score for each user based on the quality of the tags contributed by the user. Reputation scores are used for identifying good candidate tags for a particular document, i.e., for automatic tag selection. This problem is somehow the inverse of ours, tag-based searching, i.e., finding good documents for a tag.

A tagging system is comprised of resources, users and tags. These elements have been studied independently in the past. *Link analysis* exploits the relationship between resources through links and is a well-researched area [16]. Analysis of social ties and *social networks* is an established subfield of sociology [23] and has received attention from physicists, computer scientists, economists, and other types of researchers. Recently, the aggregation and semantic aspects of tags have also been discussed [17, 25]. To what extent existing approaches may be carried over to tagging systems and, in particular, help tackle tag spam is an open question. For instance, link analysis has been suggested to help fight web spam [14, 24] by identifying trusted resources and propagating trust to resources that are linked from trusted resources. An alternative way is to identify spam pages [15]. However, in a tagging system, documents are explicitly connected to people rather than other documents. Moreover, due to this association, tags have the potential to be both more comprehensive and more accurate than anchor-text based methods. Alternatively, tagging systems could utilize the information and trust in the social network, as in [12]. Again, they may still need to take into account the links from users to resources to reason about the importance and trust of users and resources and make the system more resilient to spam.

3. TAGGING FRAMEWORK

In this section, we present our ideal tagging system model. In this model, we assume that malicious tags and malicious user behaviors are well defined. In particular, we assume that users use a particular, fixed strategy for their tagging and we provide models of user tagging behavior.

3.1 System Model

A tagging system is made up of a set $\mathcal{D}$ of documents (e.g., photos, web pages, etc), which comprise the *system resources*, a set $\mathcal{T}$ of available tags, which constitute the system *vocabulary*, a set $\mathcal{U}$ of users, who participate in the system by assigning tags to documents, and a *posting relation* $\mathcal{P}$, which keeps the associations between tags, documents and users. We call the action of adding one tag to a document a *posting*. Given our goals, we do not need to know the content of the documents nor the text associated with each tag. All we need is the association between tags, documents and users. Therefore, all entities, i.e., documents, tags and users, are just identifiers. We use the symbols d, t and u to denote a document in $\mathcal{D}$, a tag in $\mathcal{T}$ and a user in $\mathcal{U}$, respectively. We consider that a posting is a tuple $[u, d, t]$ in $\mathcal{P}$ that shows that user u assigned tag t to document d. Note that we have no notion of when documents were tagged, or in what order. Such information could be useful, but is not considered in this paper.

To capture the notion that users have limited resources, we introduce the concept of a *tag budget*, i.e., a limit on how many postings a user can add. For simplicity, we assume that any given user makes exactly p postings.

Each document $d \in \mathcal{D}$ has a set $\mathcal{S}(d) \subseteq \mathcal{T}$ of tags that correctly describe it. For example, for a photo of a dog, "dog", "puppy", "cute" may be the correct tags, so they belong to the set $\mathcal{S}(d)$. All other tags (e.g., "cat", "train") are incorrect and are not in $\mathcal{S}(d)$. We are using strings like "dog" and "cat" in the example above, but we are not interpreting the strings, they are just tag identifiers for us.

3.2 Basic Tagging Model

To populate a particular instance of a tagging system, we need to: (i) populate the $\mathcal{S}(d)$ sets and (ii) generate the actual postings of users. The members of each $\mathcal{S}(d)$ are *randomly* chosen from $\mathcal{T}$. In order to populate the posting relation, we need to define bad and good user tagging models to simulate user tagging behavior. For our purposes, we assume that there is a clear distinction between malicious and good users and that both good and malicious users use a particular, fixed strategy for tagging. That is, we consider good users in set $\mathcal{G}$ and bad (malicious) users in set $\mathcal{B}$, such that $\mathcal{U} = \mathcal{G} \cup \mathcal{B}$ and $\mathcal{G} \cap \mathcal{B} = \emptyset$. In the subsequent subsections, we define several models of good and bad taggers.

Assuming that users randomly pick documents (*uniform document distribution*) and tags (*uniform tag distribution*) for their postings, we define the following random models for good and bad users.

Random Good-User Model:
  for each user $u \in \mathcal{G}$ do
    for each posting $j = 1$ to p do
      select at random a document d from $\mathcal{D}$;
      select at random a tag t from $\mathcal{S}(d)$;
      record the posting: user u tags d with t.

Random Bad-User Model:
  for each user $u \in \mathcal{B}$ do
    for each posting $j = 1$ to p do
      select at random a document d from $\mathcal{D}$;
      select at random an incorrect tag t from $\mathcal{T} - \mathcal{S}(d)$;
      record the posting: user u tags d with t.

The random bad user model assumes that each user acts independently, that is, the bad users are "lousy taggers" but not malicious. However, in some cases malicious users may collude and mount more organized attacks. We consider a particular form of targeted attack behavior assuming that colluding users attack a particular document d_a with some probability r. This model is defined as follows.

Targeted Attack Model:
  select a particular document d_a from $\mathcal{D}$;
  select a particular incorrect tag t_a from $\mathcal{T} - \mathcal{S}(d_a)$;
  for each user $u \in \mathcal{B}$ do
    for each posting $j = 1$ to p do
      with probability r record the posting:
        user u tags d_a with t_a;
      else:
        select at random a document d from $\mathcal{D}$;
        select at random an incorrect tag t from $\mathcal{T} - \mathcal{S}(d)$;
        record the posting: user u tags d with t.

Observe that for $r = 0$, the targeted attack model coincides with the random bad user model. Also note that both good and bad users may submit duplicate tags: Even if document d already has tag t, a user can tag d with t (and even if the first t tag was added by the same user). Some systems may disallow such duplicate tags. We have experimented with a no-duplicates-per-user policy but do not report the results here (the conclusions are not significantly different). Moreover, a person may sign in the system using different usernames and express a number of duplicate opinions.

One can extend this basic tagging model we have presented in many directions, e.g., changing the distributions that are used to select tags, queries, documents, and so on; or by introducing non-determinism in parameters such as the tag budget or the size of the $\mathcal{S}(d)$ sets; or by defining additional good/bad user models. We have experimented with a number of these variations. In the following subsection, we discuss one interesting variation considering tag popularity.

3.3 Skewed Tag Distribution

People naturally select some popular and generic tags to label web objects of interest [25]. For example, the word "dog" is more likely to be used as a tag than "canine", even though they may be both appropriate. In a tagging system, popular and less frequent tags co-exist peacefully. Therefore, we consider that there is a set $\mathcal{A} \subseteq \mathcal{T}$ of popular tags. In particular, we assume that popular tags may occur in the postings m times more often than unpopular ones. However, when we generate the appropriate $\mathcal{S}(d)$ set per document d, we disregard popularity, because an unpopular tag like "canine" has the same likelihood to be relevant to a document as a popular tag like "dog". So, members of each $\mathcal{S}(d)$ are chosen *randomly* from $\mathcal{T}$.

A *Biased Good User* selects a correct tag for a document d taking into account tag popularity. For instance, for a cat photo, the set of correct tags may be $\mathcal{S}(d)$={"*cat*", "$feline$"}, with "cat" being more popular than "feline". Thus, "cat" is more likely to be selected for a posting. This good user model is defined as follows:

Biased Good-User Model:
  for each user $u \in \mathcal{G}$ do
    for each posting $j = 1$ to p do
      select at random a document d from $\mathcal{D}$;
      select a tag t from $\mathcal{S}(d)$ w.r.t. tag popularity;
      record the posting: user u tags d with t.

Then, for bad users, we consider three different models.

Biased Bad Users may try to disguise themselves by acting like normal users, i.e., using more often popular tags and less frequently unpopular ones, but for mislabeling documents. This bad user model is defined below.

Biased Bad-User Model (The Imitator):
  for each user $u \in \mathcal{B}$ do
    for each posting $j = 1$ to p do
      select at random a document d from $\mathcal{D}$;
      select a tag t from $\mathcal{T} - \mathcal{S}(d)$ w.r.t. tag popularity;
      record the posting: user u tags d with t.

Extremely Biased Bad Users use only popular tags for the wrong documents. For instance, in a particular tagging system, the tag "travel" may be very popular. This means that this tag will also appear in tag searches often. Then, these bad users may use this tag to label particular documents in order to make them more "viewable." This behavior is captured by the following model.

Extremely Biased Bad-User Model (The Exploiter):
  for each user $u \in \mathcal{B}$ do
    for each posting $j = 1$ to p do
      select at random a document d from $\mathcal{D}$;
      select a *popular* tag t from $\mathcal{T} - \mathcal{S}(d)$;
      record the posting: user u tags d with t.

Outlier Bad Users use tags that are not very popular among good users. For instance, in a publications tagging system, these users may try to promote their pages selling particular products, so they may use tags such as "offer" or "buy", which are not popular among good users. This model is defined as follows.

Outlier Bad-User Model (The Atypical):
  for each user $u \in \mathcal{B}$ do
    for each posting $j = 1$ to p do
      select at random a document d from $\mathcal{D}$;
      select an *unpopular* tag t from $\mathcal{T} - \mathcal{S}(d)$;
      record the posting: user u tags d with t.

4. TAG SEARCH

In a tagging system, users may be able to *query* for resources that are annotated with a particular tag. Given a query containing a single tag t, the system returns documents associated with this tag. We are interested in the top $\mathcal{K}$ documents returned, i.e., documents contained in the first result pages, which are those typically examined by searchers. So, although all search algorithms can return more than $\mathcal{K}$ results, for the purposes of our study, we consider that they generate only the top $\mathcal{K}$ results.

4.1 Existing Search Models

The most commonly used query answering schemes are the Boolean (e.g., Slideshare [7]) and the Occurrence-based (e.g., Rawsugar [6]). In Boolean searches, the query results contain $\mathcal{K}$ documents randomly selected among those associated with the query tag, i.e.,:

Boolean Search:
return random $\mathcal{K}$ documents assigned t in $\mathcal{P}$.

In Occurrence-based searches, the system ranks each document based on the number of postings that associate the document to the query tag and returns the top ranked documents. This search model is described as follows:

Occurrence-Based Search:
rank documents by decreasing number of postings in $\mathcal{P}$ that contain t;
return top $\mathcal{K}$ documents.

We have also experimented with variants of this ranking model, such as ordering documents based on the number of a tag's occurrences in a document's postings divided by the total number of this document's postings, i.e., based on tag frequency. In this paper, we consider only the basic occurrence-based ranking scheme, since our experiments have shown that variants of this model exhibit a similar behavior with respect to spamming.

4.2 Coincidences

Common search techniques in tagging systems do not take into account spamming. In particular, in Boolean search, a document that has been maliciously assigned a specific tag may be easily included in the results for this tag. The underlying principle of Occurrence-based search is that a document is relevant to a tag depending on the number of postings that claim so. Although this seems to be quite reasonable and to make searches more "spam-proof", still bad users may easily promote their documents to the top results as the following example shows.

Example. Consider the following postings:

user	document	tag
1	d_1	a
2	d_1	a
3	d_1	b
4	d_1	b
5	d_1	b
3	d_2	a
3	d_2	c
4	d_2	c

We assume that correct tags for document d_1 and d_2 belong to the sets $\{b, c\}$ and $\{a, c\}$, respectively. Different users may assign the same tag to the same document. For instance, users 3, 4 and 5 have all assigned tag b to document d_1. Since we use a small number of documents and postings in order to keep the example compact, let's assume that the system returns the top $\mathcal{K}$=1 document for a query tag. Users 1 and 2 are malicious, since tag a is not a correct tag for d_1, but the system does not know this information. For tag a, based on occurrences, the system will erroneously return d_1.

The example above shows that the raw number of postings made by users in a tagging system is not a safe indication of a document's relevance to a tag. Postings' reliability is also important. We observe that user 3's posting that associates d_2 with tag a seems more trustable than postings made by users 1 and 2, because that user's postings are generally in accordance with other people's postings: the user agrees with user 4 in associating d_2 with tag c and with users 4 and 5 in associating d_1 with b.

Based on the above intuition, we propose an approach to tag search that takes into account not only the number of postings that associate a document with a tag but also the "reliability" of taggers that made these postings.

In order to measure the reliability of a user, we define the *coincidence factor* $c(u)$ of a user u as follows:

$$c(u) = \sum_{d,t:\exists \mathcal{P}(u,d,t)} \sum_{\substack{u_i \in \mathcal{U} \\ u_i \neq u}} \mid \mathcal{P}(u_i, d, t) \mid \qquad (1)$$

where $\mathcal{P}(u_i, d, t)$ represents the set of postings by user u_i that associate d with t.

Example (cont'ed). The coincidence factors for the users are: $c(1) = 1$, $c(2) = 1$, $c(3) = 3$, $c(4) = 3$ and $c(5) = 2$.

The coincidence factor $c(u)$ shows how often u's postings coincide with other users' postings. If $c(u)$=0, then u never agrees with other people in assigning tags to documents. Our hypotheses is that the coincidence factor is an indication of how "reliable" a tagger is. A high factor signifies that a user agrees with other taggers to a great extent; thus, the user's postings are more "reliable." The lower $c(u)$ is, the less safe this user's postings become.

Given a query tag t, coincidence factors can be taken into account for ranking documents returned for a specific query tag. Then, the rank of a document d with respect to t is computed as follows:

$$rank(d, t) = \frac{\sum_{\forall u \in users(d,t)} c(u)}{c_o} \qquad (2)$$

where $users(d, t)$ is the set of users that have assigned t to d and c_o is the sum of coincidence measures of all users. The latter is used for normalization purposes so that a rank ranges from 0 to 1.

Example (cont'ed). The sum of coincidence measures of all users is $c_o = c(1) + c(2) + c(3) + c(4) + c(5) = 10$. Then, the document ranks with respect to each of the posted tags are:

$rank(d_1, a) = (c(1) + c(2))/c_o = 2/10$

$rank(d_1, b) = (c(3) + c(4) + c(5))/c_o = 8/10$

$rank(d_2, a) = c(3)/c_o = 3/10$

$rank(d_2, c) = (c(3) + c(4))/c_o = 6/10.$

In words, a document's importance with respect to a tag is reflected in the number and reliability of users that have associated t with d. A document is ranked high if it is tagged with t by many reliable taggers. Documents assigned a tag by few less reliable users will be ranked low.

Example (cont'ed). For tag a, document d_2 gets the highest rank, $rank(d_2, a) = 3/10$ compared to $rank(d_1, a) = 2/10$, and comprises the system answer.

5. TRUSTED MODERATOR

In order to reduce the impact of bad postings, a trusted moderator can periodically check user postings to see if they are "reasonable." This moderator is a person that can "conceptually" identify good and bad tags for any document in the collection. Search engine companies typically employ staff members who specialize in web spam detection, constantly scanning web pages in order to fight web spam [14]. Such spam detection processes could be used in tagging systems too. The moderator examines a fraction f of the documents in $\mathcal{D}$. For each incorrect posting found, the moderator could simply remove this posting. But she can go a step further and remove all postings contributed by the user that made the incorrect posting, on the assumption that this user is bad. The moderator function could be described as follows:

Trusted Moderator:
  let $\mathcal{D}_f \subseteq \mathcal{D}$ containing a fraction f of $\mathcal{D}$'s documents;
  for each document $d \in \mathcal{D}_f$ do
    for each incorrect posting [u, d, t]
      eliminate all entries [u, *, *].

6. SPAM FACTOR

We are interested in measuring the impact of tag spam on the result list. For this purpose, we define a metric called $SpamFactor(t)$ as follows. Given a query tag t, the system returns a sequence $\mathcal{D}_\mathcal{K}$ of $\mathcal{K}$ documents ranked, i.e.,:

$$\mathcal{D}_\mathcal{K} = [d_1, d_2, ...d_\mathcal{K}]$$
$$\text{where } rank(d_{i-1}, t) \geq rank(d_i, t), \quad 2 \leq i \leq \mathcal{K}.$$

Then, $SpamFactor(t)$ for tag t is given by the formula:

$$SpamFactor(t) = \frac{\sum_{\forall d_i \in D_K} w(d_i) * \frac{1}{i}}{H_K} \tag{3}$$

where

$$w(d_i) = \begin{cases} 1 & \text{if } d_i \text{ is a bad document;} \\ 0 & \text{if } d_i \text{ is a good document.} \end{cases}$$

and H_K is the K^{th} harmonic number, i.e., it is the sum of the reciprocals of the first $\mathcal{K}$ natural numbers, i.e.,

$$H_K = \sum_{i \in [1..K]} \frac{1}{i} \tag{4}$$

In what follows, we explain each part of the definition above. A document d is "bad" if it is included in the results for tag query t, but t is not a correct tag for d, i.e., $t \notin \mathcal{S}(d)$. SpamFactor measures the spam in the result list introduced by bad documents. This is captured by the factor $w(d_i)$ in the formula, which returns 1 if d_i is a bad document and 0 otherwise. SpamFactor is affected by both the number of bad documents and their position in the list. The higher the position of a bad document in the result list is, the higher the numerator in formula (3) is. The maximum numerator value is $1 + \frac{1}{2} + \ldots + \frac{1}{K}$. This is the K^{th} harmonic number and it is used as denominator in the calculation of SpamFactor in order to normalize values between 0 and 1. Higher SpamFactor represents greater spam in the results. In order to illustrate the significance of different SpamFactor values, we consider the following example.

Example. Consider the following postings:

user	document	tag
1	d_1	a
1	d_1	c
3	d_1	c
2	d_1	a
2	d_1	b
1	d_2	a
2	d_2	a
3	d_2	a
3	d_2	c
4	d_2	c
3	d_3	a
6	d_3	a
1	d_3	b
5	d_3	b
6	d_3	b
4	d_4	b
5	d_4	b
5	d_4	c
5	d_5	a
5	d_5	c
1	d_5	b

The sets of correct tags for the documents in this example are: $S(d_1) = \{a, b, c\}$, $S(d_2) = \{a, c, d\}$, $S(d_3) = \{a, c\}$, $S(d_4) = \{b\}$ and $S(d_5) = \{b\}$. We assume that the system returns the top $\mathcal{K} = 4$ documents for a query tag based on the number of occurrences. For query tag a, the system returns d_2, d_1, d_3 and d_5 in that order. Only d_5 is a bad document and it is at the bottom of the result list. So, for this tag, the spam introduced by malicious users is limited. This is indicated by the low value of SpamFactor, which is $SpamFactor(a) = 0.12$. The results for query tag b are d_3, d_4, d_1 and d_5. In this case, d_3 is a bad document, and it is ranked first. This fact results in higher SpamFactor, i.e., $SpamFactor(b) = 0.48$. Finally, for tag c, the system returns d_1, d_2, d_4 and d_5. There are two bad documents in the results, i.e., d_4 and d_5, but these are found at the bottom of the results. So, it is $SpamFactor(c) = 0.28$, i.e., it is between $SpamFactor(a)$ and $SpamFactor(b)$.

7. EXPERIMENTS

7.1 Experimental Framework

Table 1: Parameters used in Experiments

Symbol	Description	Value
$\|\mathcal{D}\|$	number of docs. in $\mathcal{D}$	10,000
$\|\mathcal{T}\|$	size of the vocabulary $\mathcal{T}$	500
$\|\mathcal{U}\|$	number of users in $\mathcal{U}$	1,000
$\|\mathcal{B}\|$	number of malicious users	10%
p	tag budget per user	10
s	size of $\mathcal{S}(d)$	25
f	frac. docs. checked by moderator	5%
$\mathcal{K}$	number of docs. in results	10
r	probability of targeted attack	0
$\|\mathcal{A}\|$	number of popular tags	0

We have developed a simulator in Java that simulates the behavior of a tagging system based on the model described in this paper. We have conducted many experiments under several modifications of the parameters involved. Table 1 summarizes all parameters considered and their default values.

The objectives of our study can be summarized in the following: (a) understand the magnitude of tag spam problem and the weaknesses of existing systems, (b) evaluate the effort and effectiveness of a moderator for eliminating spam, (c) evaluate the effectiveness of safeguards we described against the types of malicious attacks we study.

This section is organized into four parts: The first one illustrates how spam factor changes depending on the number of bad documents and their positions in the results. The second one describes the impact of random attacks on tag searches and the effectiveness of a moderator. The third one evaluates the severity of targeted attacks. The last part studies malicious attacks that exploit tag popularity in the system.

7.2 SpamFactor Variation

In the results that follow we will encounter a variety of spam factor values. In order to get a sense of how these factors translate to the "desirability" of an answer, we illustrate how spam factor changes depending on the number of bad documents and their positions in the results. Figure 1 shows SpamFactor for a top $\mathcal{K}$=10 result list. For instance, having two documents at the top two positions in the results ($SpamFactor = 0.51$) is worse than having four documents in the last positions ($SpamFactor = 0.163$). Even for the same number of bad documents, SpamFactor depends on the document positions in the results. For instance, SpamFactor for results containing three bad documents ranges approximately from 0.11 to 0.62. SpamFactor ≤ 0.1 indicates that at most two bad documents exist at low positions in the results, which may be considered as tolerable spam. Greater values of SpamFactor indicate the existence of more bad documents in higher positions in the list. Therefore, such values will be considered excessive in our analysis.

7.3 Random Attacks

For these experiments, we have used a set of $1,000$ tag queries that follow a uniform distribution.

7.3.1 Existing Tag Searches

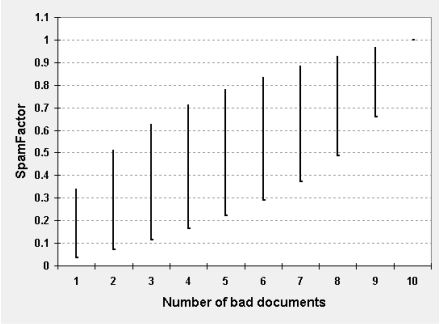


Figure 1: Spam Factor Variation

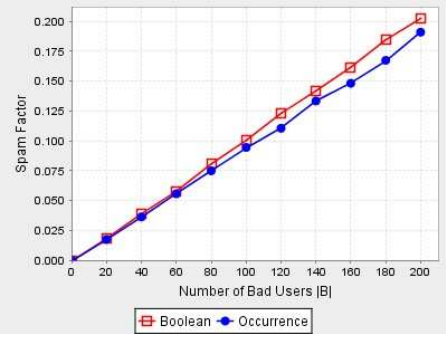


Figure 2: Impact of the number of bad users

In this subsection, we simulate and study how vulnerable existing tag searches may be to malicious attacks.

Number of malicious users $|\mathcal{B}|$. Figure 2 illustrates the effect of varying the number of bad users in the system on Boolean and Occurrence-based tag searches. SpamFactor grows linearly, because the number of bad postings increases linearly with $|\mathcal{B}|$ while the number of good postings decreases. As $|\mathcal{B}|$ approaches $|\mathcal{U}|$, all algorithms converge because the system gets overloaded with bad documents. For presentation reasons, Figure 2 shows SpamFactor for $|\mathcal{B}|$ ranging up to 20% of $|\mathcal{U}|$.

In subsection 7.2, we argue that SpamFactor less than 0.1 is "tolerable" in the sense that the spam documents will be few and towards the bottom of the result list. Thus, looking at Figure 2, we conclude that for Boolean and Occurrence-based searches, a very small percentage of malicious users (e.g., $<$ 2% of $|\mathcal{U}|$) with limited tagging power (p=10) as compared to the document collection size ($\mathcal{D}$=10,000) does not bias results significantly. Excessive SpamFactor is observed for growing bad user populations ($>$ 12%).

Consequently, a moderate number of bad users in the system may significantly degrade results especially for Boolean searches. This observation is critical because in practice many users may accidentally assign incorrect tags (lousy taggers), therefore unintentionally generating spam. The "good news" are that SpamFactor increases linearly, but this is due to the fact that users pick randomly documents and tags and they do not collude.

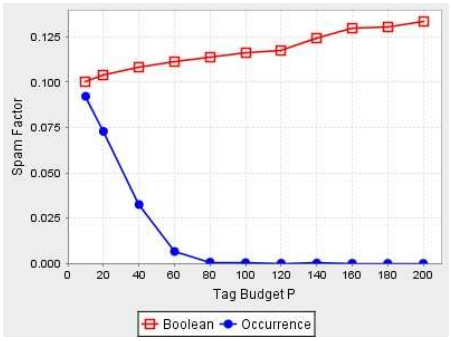


Figure 3: Impact of the tag budget

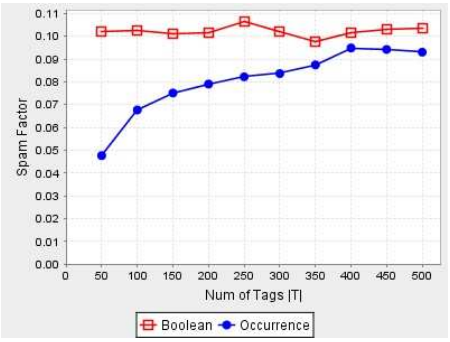


Figure 4: Impact of the vocabulary size

Tag budget per user p. In the previous experiment, users had limited tag budget. The current one shows how tag search results are affected by modifying tag budget from 2 to 500 for a moderate bad user population ($|\mathcal{B}| = 10\%$ of the overall user population). Interestingly, as Figure 3 shows, Boolean and Occurrence-based results are not affected in the same way: SpamFactor increases for the former and decreases for the latter for reasons explained below.

In this experiment, there are $|\mathcal{D}| = 10,000$ documents and $|\mathcal{T}| = 500$ possible tags. The number of correct tags per document is s=25. Consequently, there are 250,000 possible correct $< d, t >$ pairs for good users to use and 4,750,000 possible bad pairs as bad users' options. Since we assume that tags are used following a uniform distribution, the probability that a good pair is selected is 1/250,000, while the probability that a bad pair is selected is 1/4,750,000. Therefore, the chance that two good users will choose the same good pair is much higher than that of two bad users selecting the same bad pair. This means that there will be more duplicate good $< d, t >$ pairs than bad pairs. This fact helps Occurrence-based search select good documents at the top of the result list. As the tag budget increases, more often users will pick the same good pair, thus the number of occurrences of good pairs increases causing more good documents to surface in the results and Occurrence-based SpamFactor to further decrease. On the other hand, Boolean SpamFactor increases with tag budget due to random generation of results. With tag budget growing, distinct bad postings proliferate, thus the probability that a bad document will be randomly picked gets higher.

Consequently, active users are beneficial only for Occurrence-based searches because they can provide more evidence on the goodness of documents.

Size of the vocabulary $|\mathcal{T}|$. Figure 4 shows SpamFactor as a function of the vocabulary size. As $|\mathcal{T}|$ increases and the overall tagging power of good users is constant, the number of tags in the system for which there will be only incorrect postings grows. Given that user queries in the experiment contain random tags, the number of "bad" tag occurrences in the query set will grow. Based on the above, SpamFactor for Occurrence-based searches deteriorates and tends to converge to Boolean SpamFactor. The latter is not substantially affected, because Boolean search randomly picks postings that match a query tag. Hence, since the total number of bad postings in the system does not change with $|\mathcal{T}|$, there is no substantial difference in SpamFactor.

Consequently, occurrence-based tag searches with unrestricted vocabulary are more vulnerable to spam.

Number of users $|\mathcal{U}|$. This experiment measures SpamFactor as a function of the number of users in the system, ranging from 200 to 4200. Since the percentage of bad users is constant, one might not expect significant variation in SpamFactor. Figure 5 reveals a different situation. For a "small" number of users, both Boolean and Occurrence-based results degrade as more users enter the system. For example, with 200 users each making 10 postings, there will be only 2,000 postings, while the number of documents is 10,000. Given the small number of postings, the actual pool of documents that match a particular query tag is undersized. Hence, good candidate documents for the result list will be rarer and bad documents will also surface. As $|\mathcal{U}|$ increases, so does the number of malicious postings, increasing the number of bad documents in the results. Also, few, if any, duplicate $< d, t >$ pairs exist in a small number of postings, making Occurrence-based search to behave similarly to Boolean search. Therefore, the initial parts of Boolean and Occurrence-based SpamFactor curves almost coincide.

As $|\mathcal{U}|$ keeps increasing, good re-occurring postings multiply. There is a point in the figure ($|\mathcal{U}| = 600$), where a sufficient number of (re-occurring) postings is collected. From this point forward, Occurrence-based results increasingly get better. A second interesting point in the figure is around $|\mathcal{U}| = 2400$, where SpamFactor returns to its starting value (~ 0.07), thus the effect of insufficient number of postings in the system is canceled. In order to find this turning point, given that the percentage of good users in the system is $g = 90\%$, a rule of thumb is the following: $g * |\mathcal{U}| * p \geq 2 * |\mathcal{D}|$. In words, the estimated number of good postings in the system should be at least equal to twice the number of documents. Boolean results significantly degrade as more users enter in the system, because the probability that a bad document will be picked gets higher.

Consequently, enlarging the system's user base is good for Occurrence-based searches because more users can help identify good documents. As a rule of thumb, the estimated number of good postings should be at least twice the number of documents in the system. This lower bound is required

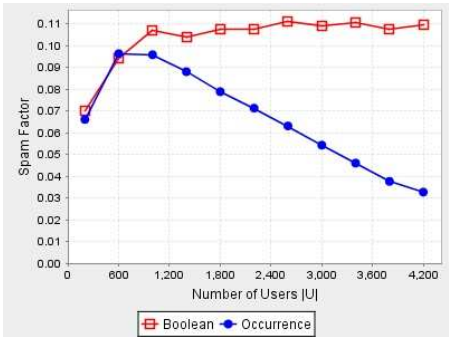


Figure 5: Impact of the number of users

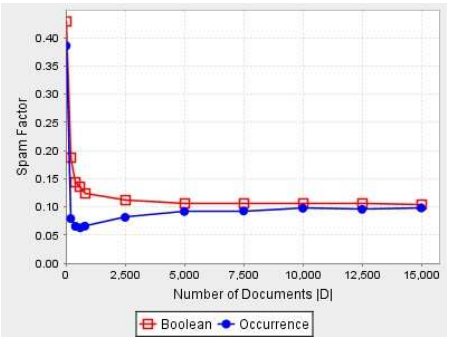


Figure 6: Impact of the number of documents

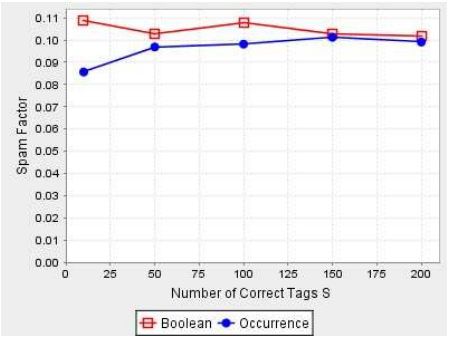


Figure 7: Impact of the number of correct tags

in order to cancel side-effects due to sparse postings.

Number of documents $|\mathcal{D}|$. Figure 6 shows SpamFactor for $|\mathcal{D}|$ ranging up to 15,000. The number of bad and good postings is constant throughout the experiment. We observe that SpamFactor is initially very high. The reason is that for relatively small document collections compared to the number of malicious users, e.g., 100 documents and 100 bad users, and using the random bad user model, it is more likely that bad users will have most documents spammed. Thus, it is very probable that a document will be picked based on bad posting evidence. As $|\mathcal{D}|$ grows, the influence of malicious users is confined, because good and bad postings are uniformly distributed over more documents. Since, good postings outnumber malicious ones (9,000 vs.1,000), it becomes more probable to select a document based on good postings. For this reason, Boolean SpamFactor decreases, initially very quickly then slowly, until it stabilizes. Occurrence-based SpamFactor exhibits the same initial behavior but then slowly increases and converges to Boolean SpamFactor. The reason for this degradation is that fewer re-occurring good postings are generated with $|\mathcal{D}|$ growing. So, the minimum SpamFactor based on occurrences is observed at around $|\mathcal{D}| = 600$. Given that the percentage of good users in the system is $g = 90\%$, a rule of thumb is that $g * |\mathcal{U}| * p \geq 15 * |\mathcal{D}|$. In words, in order to minimize SpamFactor, there should be a sufficiently large number of good postings per document (greater than 15).

Consequently, a balance must be kept: Small document collections, compared to the set of users, are more susceptible to spamming because it is easier to assign bad tags to a greater percentage of documents in the collection. On the other hand, it is difficult to gather duplicate good postings for large document collections.

Size of $\mathcal{S}(d)$ s. In the real world, the set of correct tags per document is usually just a very small subset of the entire vocabulary and cannot be freely varied to an arbitrarily large set; there are usually just few words that properly describe a document and the rest of other words in dictionary are less pertinent. In this experiment (Figure 7), s varies between 10 and 200. Given s correct tags and $|\mathcal{D}|$ documents, there are $s * |\mathcal{D}|$ possible correct $< d, t >$ pairs. Increasing s causes the number of possible correct $< d, t >$ pairs to increase and, since $|\mathcal{T}|$ is constant, the number of possible bad pairs decreases. Since, we assume that tags are used following a uniform distribution, the probability that a particular good pair is selected by two good users shrinks. This fact explains why Occurrence-based SpamFactor increases and approaches Boolean SpamFactor. On the other hand, Boolean SpamFactor slowly decreases because the number of distinct bad pairs decrease, thus the probability that a bad document will be randomly picked is lower.

Consequently, collections of documents with well-defined semantics, i.e., documents that can be correctly described using a small number of tags, are more tolerant to spam.

7.3.2 Trusted Moderator

In the subsequent paragraphs, we present experimental results regarding the effort and effectiveness of a trusted moderator supporting a system that uses either Boolean ($Boolean+TM$) or Occurrence-based searching ($Occurrence+TM$).

Fraction of documents examined f. Figure 8 presents SpamFactor as a function of the percentage f of documents examined by the moderator, for f ranging from 1% to 50% and $|\mathcal{B}| = 10\%$. We observe that spam postings are completely removed from the system by scanning almost half the document collection, but this is impractical. A substantial reduction in spam is achieved after 10% of the documents have been examined. At this point, SpamFactor is around 0.04. Referring to Figure 1, this value indicates the existence of only one bad document in a low position in the results. The gain of having a moderator decreases with f growing. For example, increasing f from 10% to 20% achieves a slower reduction in SpamFactor than increasing f from 1% to 10%. This means that the moderator needs to invest increasingly

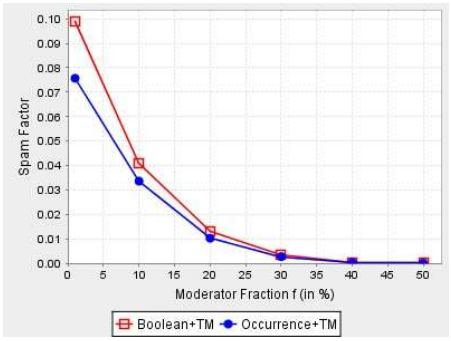


Figure 8: f Impact on Moderator

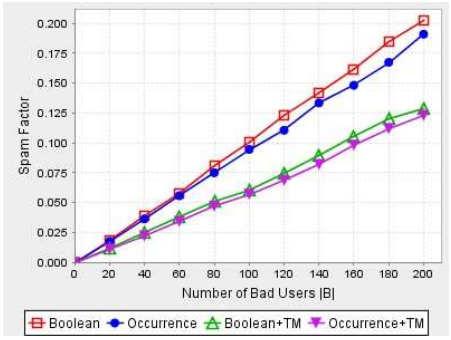


Figure 9: Impact of the number of bad users on Moderator

more effort in order to achieve a steady improvement on SpamFactor. In other words, as bad postings are eliminated from the system, it becomes more difficult to discover the remaining ones.

Consequently, a trusted moderator greatly helps reducing the spam in the system but it takes a significant effort in order to achieve a positive impact. Also, aiming at progressively better results means that the moderator needs to place increasingly more effort.

Number of malicious users $|\mathcal{B}|$. Figure 9 shows that, with the help of a moderator, SpamFactor increases only sub-linearly with the number of bad users. The moderator's intervention results in the removal of the same bad users from the system, so the new SpamFactor curves for both Boolean and Occurrence-based searches almost coincide. Interestingly, with $|\mathcal{B}|$ growing, the moderator achieves proportionally the same improvement in SpamFactor. This is due to the fact that based on the random user models, bad postings are uniformly distributed to all documents. Examining the same percentage of documents, the moderator can identify approximately the same fraction of bad postings.

Consequently, the moderator's relative effectiveness is independent of the number of bad users in the system.

Tag budget per user p. Figure 10 shows SpamFactor for the moderated and unmoderated results as a function of tag budget. The use of a moderator has a dramatic impact on Boolean searches: although SpamFactor for the un-

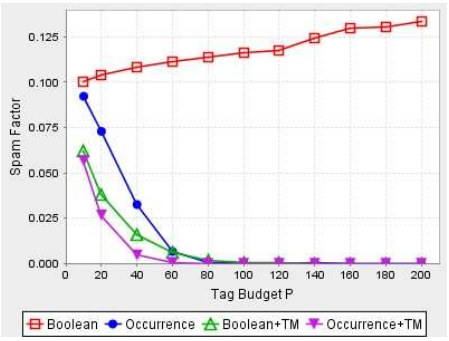


Figure 10: Impact of the tag budget on Moderator

moderated searches deteriorates with tag budget, moderated SpamFactor improves. In addition, the gain of having a moderator for Boolean searches grows with tag budget. This is due to the fact that when users contribute more tags, more bad postings can be found for the same fraction of documents.

Consequently, the moderator's impact is not always the same on all search schemes.

Number of users $|\mathcal{U}|$. Figure 11 shows the impact of varying $|\mathcal{U}|$ on SpamFactor when a trusted moderator is used.

For Boolean results, the moderator can cut SpamFactor almost by a factor of 2. This improvement does not change with the number of users in the system, because with $|\mathcal{U}|$ growing, bad postings are uniformly distributed over all documents. Thus, the number of bad postings coming from different users found in the same fraction of documents does not change significantly. This effect is different from what we observed for the moderated Boolean results in Figure 10, despite the fact that both $|\mathcal{U}|$ and p affect the posting relation in the same way: increasing either of them produces more postings. However, with p growing, once a moderator finds a bad posting, then a possibly larger number of bad postings is eliminated at once by removing the corresponding tagger.

On the other hand, the moderator's relative effectiveness for Occurrence-based searches slowly decreases with $|\mathcal{U}|$. The reason is that, after a certain point in the figure, unmoderated Occurrence-based results greatly benefit from the increasing number of users in the system thus reducing the gap between the moderated and unmoderated curves.

Consequently, the moderator's relative effectiveness is not affected by the number of users in the system for Boolean searches but is limited for Occurrence-based searches.

Number of documents $|\mathcal{D}|$ **and size of** $\mathcal{S}(d)$ s. Figures 12 and 13 show the effect of growing collection size and number of correct tags per document, respectively, on the moderator effectiveness. The shape of the curves for the moderated results are similar to the curves for the unmoderated results, but at a lower position (lower SpamFactor).

Consequently, the moderator's relative effectiveness is not

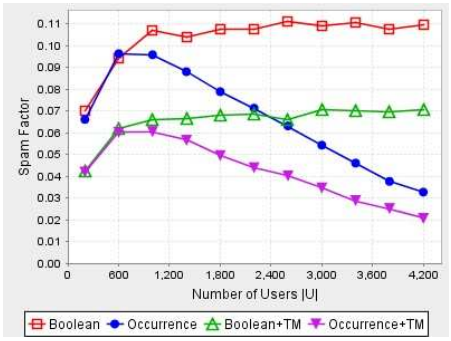


Figure 11: Impact of the number of users on Moderator

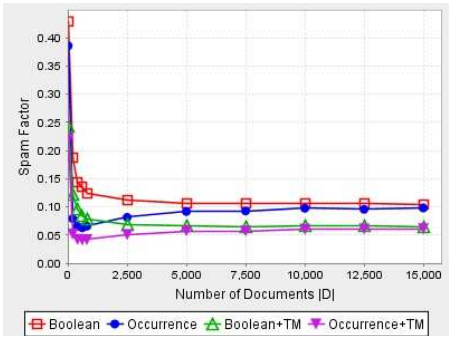


Figure 12: Impact of the number of documents on Moderator

affected by the number of documents in the system or the number of correct tags per document.

7.3.3 Coincidences

In this subsection, we present experimental results showing the effectiveness of using coincidences of postings for reducing spam impact on search results. Since our previous experimental findings have shown that Boolean searches do not cope well with spam, in the following figures, we compare Coincidence-based vs. Occurrence-based results with and without using a moderator. We assume a reasonable amount of effort, i.e., that the moderator examines 500 documents ($f = \%5$).

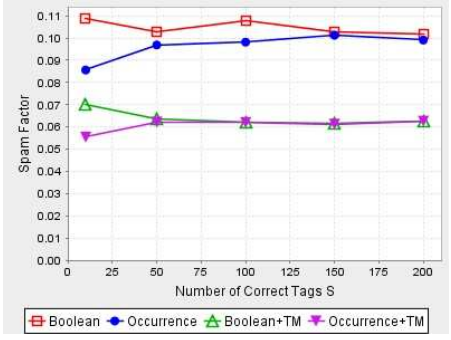


Figure 13: Impact of the number of correct tags on Moderator

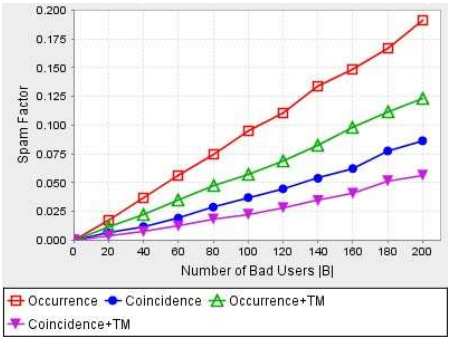


Figure 14: Impact of the number of bad users on Coincidences

Number of malicious users $|\mathcal{B}|$. Figure 14 illustrates SpamFactor as a function of the number of bad users in the system. Using coincidences works substantially better than using occurrences, cutting spam by a factor of 2. The reason behind this improvement is that coincidence factors are computed taking into account not only the postings that associate a document to a query tag, but also information about the users that have made these postings. Thus, they exploit a greater number of postings in order to generate results for a tag. This leads to more informed decisions regarding which documents to return and justifies low Coincidence-based SpamFactor. However, as bad users proliferate, effectiveness of this scheme also deteriorates. A high coincidence factor could actually correspond to a bad user. Still, using coincidence factors retains its factor-of-2 advantage.

Consequently, relying not only postings that associate a document to a query tag but also considering information about the users that have made these postings can help reduce substantially spam impact. Of course, when bad users proliferate all countermeasures become gradually ineffective.

Tag budget per user p. Figure 15 shows that Coincidence-based SpamFactor rapidly decreases with p increasing. As we have explained in Section 7.3.1, when users provide more postings in the system, duplicate good postings accumulate, helping Occurrence-based searches to generate better results. The difference in SpamFactor between Occurrence-based and Coincidence-based searches lies in the way they exploit this increase. The former counts the number of tag occurrences in a document's postings in order to decide whether a document will be included in the results. This number slowly increases as postings are uniformly collected for all documents. On the other hand, the coincidence factors are computed by taking into account common postings over the whole collection. Thus, increasing tag budget boosts coincidence factors.

Consequently, when users contribute many tags in the system, it becomes easier to discover more user correlations for judging tagging behavior.

Size of the vocabulary $|\mathcal{T}|$. Figure 16 shows that Coincidence-based SpamFactor increases at a slower rate than Occurrence-based SpamFactor. This is due to the fact that coincidences take into account not only local information regarding the

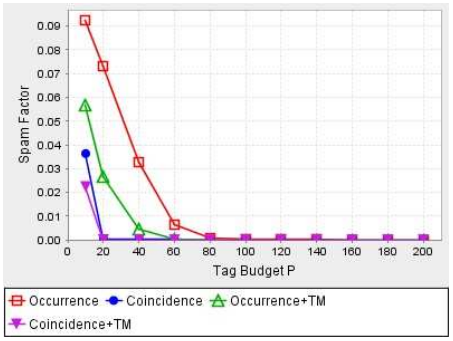


Figure 15: Impact of the tag budget on Coincidences

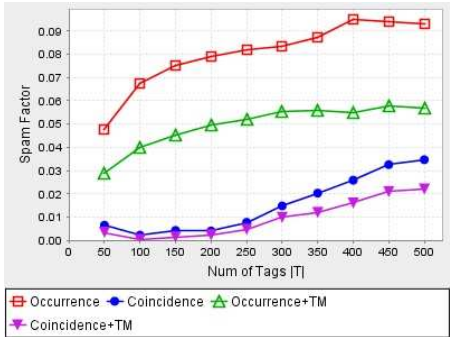


Figure 16: Impact of the vocabulary size

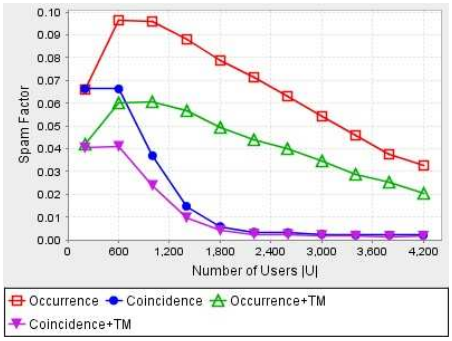


Figure 17: Impact of the number of users on Coincidences

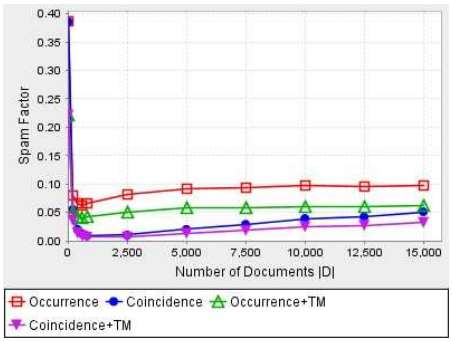


Figure 18: Impact of the number of documents on Coincidences

query tag, but also global information regarding the users that have made the postings.

Consequently, coincidence-based tag searches with unrestricted vocabulary are more vulnerable to spam.

Number of users $|\mathcal{U}|$. Figure 17 shows SpamFactor as a function of the number of users in the system. We have discussed in Section 7.3.1 that for relatively small user populations, i.e., ($|\mathcal{U}| < 600$), there are not many duplicate postings. That is the reason we have seen Occurrence-based results initially degrading. In this case, coincidence factors for all users will be almost the same, since each user agrees with no one else. Therefore, Coincidence-based results demonstrate a stable SpamFactor for small user populations. As more users enter the system, more common good postings are generated and coincidence factors of good users are boosted. This effect explains why subsequently Coincidence-based SpamFactor decreases faster than Occurrence-based one.

Consequently, several user correlations can be discovered when there is a sufficiently large number of users and postings in the system. As a rule of thumb, the number of postings should be at least twice the number of documents in the system (both for Occurrence-based and Coincidence-based searches).

Number of documents $|\mathcal{D}|$. Figure 18 plots SpamFactor as a function of the document collection size. We observe that initially both Coincidence-based and Occurrence-based SpamFactor are high but improve very quickly. The explanation is the same as the one given for Figure 6, so we do not repeat it here. The interesting part of this figure is that, after this initial phase, a different situation arises from what we have observed in previous figures: Coincidence-based SpamFactor degrades faster than Occurrence-based SpamFactor. The reason for this degradation is that fewer re-occurring good postings are generated with $|\mathcal{D}|$ growing. This degradation has a greater effect on the coincidence factors that take into account more postings than Occurrence-based search for generating an answer. The minimum SpamFactor based on coincidences is observed at around $|\mathcal{D}| = 800$. Given that $|\mathcal{U}| = 1,000$, the percentage of good users in the system is $g = 90\%$ and $p = 10$, a rule of thumb is that $g * |\mathcal{U}| * p \geq 11 * |\mathcal{D}|$. Thus, for coincidences, in order to minimize SpamFactor, there should be a sufficient number of good postings per document, but the lower bound is smaller than the bound for occurrences (see Section 7.3.1).

Consequently, we have seen that a sufficient number of good postings is required in order to minimize spam impact. However, using coincidences this number can be significantly lower than when using occurrences.

Size of $\mathcal{S}(d)$ s. Figure 19 shows the effect of varying the number of correct tags per document for Coincidence-based SpamFactor. We observe that after a point the coincidence factor for each user is no longer a good indicator of good user tagging behavior because the number of re-occurring postings decreases with s growing. Using a trusted mod-

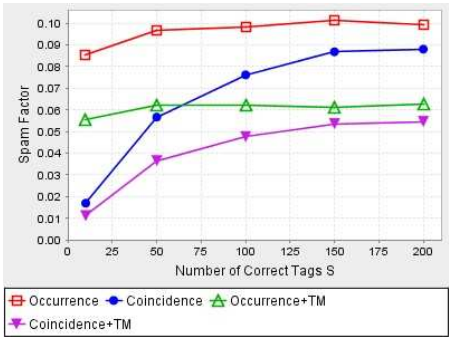


Figure 19: Impact of the number of correct tags on Coincidences

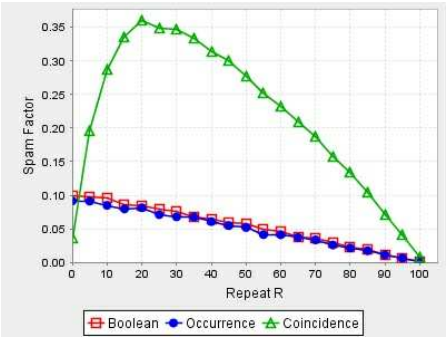


Figure 20: Impact of targeted attacks

erator to filter out some bad postings and then using the Occurrence-based algorithm gives better results. Only moderated coincidence-based SpamFactor is the lowest.

Consequently, it is easier to discover correlations among postings for collections of documents with well-defined semantics, i.e., documents that can be correctly described using a small number of tags.

7.3.4 Targeted Attacks

In this subsection, we study the effect of colluding users. Figure 20 shows SpamFactor as a function of the probability r that bad users attack the same document (Targeted attack model). If $r = 0$, then we observe the random bad user tagging behavior, while $r = 1$ means that all users attack the same document. With r growing, targeted bad postings proliferate resulting in an amplified SpamFactor for the tag used in the targeted attacks. However, the number of bad postings for the rest of the documents and tags is reduced. Consequently, Boolean and Occurrence-based SpamFactor decrease with r. Coincidence-based SpamFactor initially degrades fast with r, because coincidence factors of bad users are boosted, which means that all bad postings (apart from the targeted attack ones) are promoted in searches. However, as r increases, the number of different bad postings decreases, so the influence of bad users is restricted to fewer documents and tags. Therefore, Coincidence-based SpamFactor starts shrinking after a certain point.

Consequently, under targeted attacks, there is little one can do to protect searches for the attacked tag, but all other searches actually fare better. Moreover, we see that while using coincidences was a good strategy with "lousy but not malicious" users, it is not such a good idea with colluding bad users. However, with focused attacks, it may be easier for a moderator to locate spammed documents. For instance, the moderator may examine documents that have an unusually high number of tags, or postings by users with unusually high coincidence factors. We expect such a focused moderator approach to work very well in this scenario.

7.3.5 Attacks Based on Tag Popularity

In this subsection, we study how vulnerable tag searches are to malicious attacks that exploit tag popularity in a tagging system. We studied all meaningful combinations of good and bad user models: (Good = *random*, Bad = *random*), (Good = *biased*, Bad = *random*), (Good = *biased*, Bad = *biased*), (Good = *biased*, Bad = *extremely biased*) and (Good = *biased*, Bad = *outlier*). Also, we considered two different searcher models: A *naive searcher* may use any tag in his searches. We simulate this behavior with a set of random queries. A *community member* may query popular tags more often. We simulate this behavior by a set of queries that follow the biased tag distribution.

Number of popular tags $|\mathcal{A}|$. For each combination of good/bad user models, we study the effect of varying $|\mathcal{A}|$ on SpamFactor. We consider that popular tags may occur in the postings $m = 2$ times more often than unpopular ones. Figure 21 summarizes the corresponding experimental results. Random good and bad user models do not generate popular tags and thus do not depend on $|\mathcal{A}|$. Moreover, for $|\mathcal{A}| = 0\%$ and $|\mathcal{A}| = 100\%$, the biased tag distribution becomes random. Thus, the biased (good/bad) user models become random (good/bad) user models. Therefore, SpamFactor curves corresponding to all combinations but the ones that involve the Extremely Biased and the Outlier bad user models coincide at these two points. For the Extremely Biased behavior, SpamFactor is 0 for $|\mathcal{A}| = 0\%$, because there are no popular tags to use in bad postings. For the Outlier model, SpamFactor is 0 for $|\mathcal{A}| = 100\%$, since there are no unpopular tags to use.

Some general observations can be made on Figure 21. Comparing Figures 21(a) and 21(c), corresponding to random searches, to Figures 21(b) and 21(d) for biased searches, we observe that random searches are more vulnerable to spam. In particular, for random searches, random and outlier malicious attacks are the worst sources of spam. Moreover, comparing Figures 21(a) and 21(b), corresponding to Occurrence-based searches, to Figures 21(c) and 21(d) for Coincidence-based searches, we observe that: For any combination of user models, using coincidences cuts SpamFactor almost by a factor of 2.

Let us discuss each bad user model in more detail.

For the Outlier model, we observe that for all types of searches except for community searches based on occurrences (Figure 21(b)), SpamFactor curves are similar. In particular, with $|\mathcal{A}|$ growing, two conflicting phenomena take place: *On one hand*, unpopular tags receive increasingly more spam postings. This results in re-occurring bad postings multiplying,

and thus in SpamFactor increasing. This effect is more observable in occurrence-based searches than in coincidence-based searches, due to the fact that the former count only the mere number of postings that match a document to a tag. *On the other hand*, with $|\mathcal{A}|$ growing, unpopular tags become fewer. Consequently, spam is confined to a smaller set of tags, while the "healthy" tags proliferate. These two conflicting phenomena reach a balance point, where maximum SpamFactor is observed. From this point forward, SpamFactor decreases to zero. In comparison, SpamFactor for community occurrence-based searches always decreases, because these searches consider spam postings only when unpopular tags are queried and this happens less often as $|\mathcal{A}|$ increases.

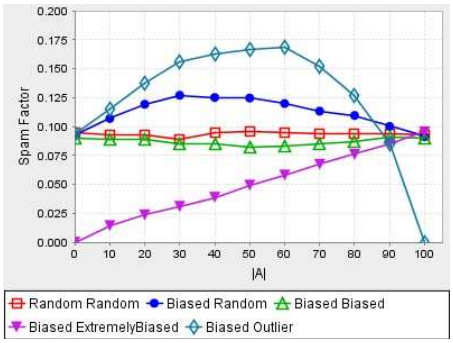
Regarding the Random bad user model, when good users are biased, we observe the following: As biased good users mostly use popular tags, for small $|\mathcal{A}|$, there will be a large number of re-occurring good postings. As $|\mathcal{A}|$ increases, fewer re-occurring good postings are generated causing SpamFactor to increase. At the same time, as biased good users mostly use popular tags, for small $|\mathcal{A}|$, a large subset of tags may be extensively misused by bad users. As $|\mathcal{A}|$ increases, the size of this subset decreases. Thus, the influence of bad users is confined and eventually SpamFactor starts to decrease. Therefore, since random searches may contain any tag, SpamFactor initially increases and then decreases with $|\mathcal{A}|$ (Figures 21(a) and 21(c)). For biased searches, the level of spam is the same irrespectively of the number of popular tags in the system (Figures 21(b) and 21(d)).

The combination of good and bad biased tagging behaviors has a similar impact on a tagging system as the combination of random tagging behaviors. On the other hand, as expected, Extremely Biased users generate more spam as $|\mathcal{A}|$ grows, since their postings are distributed over more tags. However, with $|\mathcal{A}|$ approaching 100%, the Extremely Biased model resembles more the random bad model and thus the corresponding SpamFactors converge. Moreover, coincidence factors prove more spam tolerant, since SpamFactor grows sub-linearly.

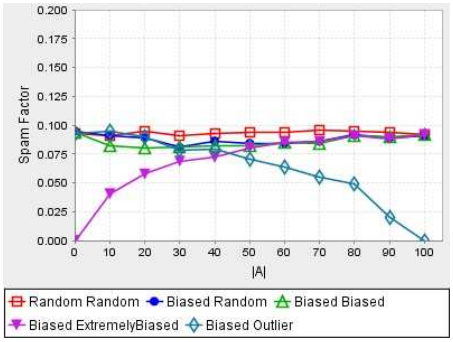
Overall, the existence of popular tags provides many possibilities for malicious users to misuse tags and spam searches. Naive or first-time users are most vulnerable. Random noise in postings as well as misused unpopular tags constitute the worst sources of spam for naive searches. Bad users mimicking good users and using popular tags for their postings can have a smaller impact on the system, in the worst case being as disruptive as lousy taggers (given a moderate number of bad users in the system). Community members may be less confused by spam postings, since they more often query tags contributed by their community. In any case, using user coincidence factors can overall help reducing spam under different bad user attacks.

8. CONCLUSIONS AND FUTURE WORK

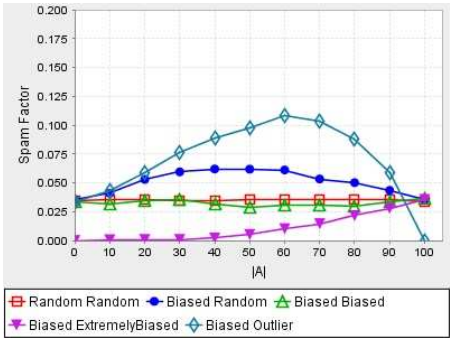
Given the increasing popularity of tagging systems and the increasing danger from spam, we have proposed an ideal tagging system where malicious tags and malicious user behaviors are well defined, and we described and studied a variety of query schemes and moderator strategies to counter tag spam. We have seen that existing tagging systems, e.g.,



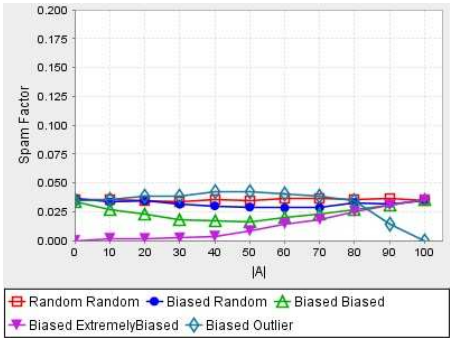
(a) Occurrence-based naive searches



(b) Occurrence-based community searches



(c) Coincidence-based naive searches



(d) Coincidence-based community searches

Figure 21: Impact of the number of popular tags

ones using the number of occurrences of a tag in a document's postings for answering tag queries, are threatened not only by malicious users but also by "lousy" ones. A countermeasure like our coincidences algorithm can be defeated by focused spam attacks. As a countermeasure for that situation, we proposed a focused moderator to detect the focused attacks. This is just an example of the measure-counter-measure battles that must be constantly fought to combat spam. Undoubtedly, the bad guys will counter-attack this proposal, and so on. We hope that the model we have proposed here, and the results it yields, can provide useful insights on how to wage these ongoing "spam wars." We also believe that our approach helps one quantify (or at least bound) the dangers of tag spam and the effectiveness of counter-measures.

There are also other interesting aspects of the problem and possible future directions to look into. For instance, if tags are related, e.g., there is a tag hierarchy, can we devise smart algorithms that take into account tag relationships? If we track the time at which postings are made, can we better deal with spammers? For example, would it help to give more weight to recent tags as opposed to older tags? Also, if users can also use negative tags, e.g., this document is *not* about "cars", what would be the impact on searches?

9. ACKNOWLEDGEMENTS

We would like to thank Manuel Deschamps for taking part in the initial stages of the simulator development.

10. REFERENCES

[1] 3spots: url: http://3spots.blogspot.com/2006/01/all-social-that-can-bookmark.html.

[2] CiteULike: url: http://www.citeulike.org/.

[3] Del.icio.us: url: http://del.icio.us/.

[4] Flickr: url: http://www.flickr.com/.

[5] Ibm's dogear: url: http://domino.research.ibm.com /comm/research-projects.nsf/pages/dogear.index.html.

[6] Rawsugar: url: http://rawsugar.com/.

[7] Slideshare: url: http://slideshare.net/.

[8] technorati: url: http://www.technorati.com/.

[9] L. Damianos, J. Griffith, and D. Cuomo. Onomi: Social bookmarking on a corporate intranet. In *Collaborative Web Tagging Workshop in conjunction with the 15th WWW Conference*, 2006.

[10] S. Farrell and T. Lau. Fringe contacts: People tagging for the enterprise. In *Collaborative Web Tagging Workshop in conjunction with the 15th WWW Conference*, 2006.

[11] S. Golder and B. A. Huberman. Usage patterns of collaborative tagging systems. *Journal of Information Science*, 32(2):198–208, 2006.

[12] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *13th WWW Conference*, pages 403–412, 2004.

[13] Z. Gyöngyi and H. Garcia-Molina. Web spam taxonomy. In *1st Intl. Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, pages 39–47, 2005.

[14] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating spam with TrustRank. In *30th Intl. Conference on Very Large Databases (VLDB)*, pages 576–587, 2004.

[15] Z. Gyongyi, P. Berkhin, H. Garcia-Molina, and J. Pedersen. Link spam detection with mass estimation. In *32nd International Conference on Very Large Databases (VLDB)*, pages 439–450, 2006.

[16] M. Henzinger. Link analysis in web information retrieval. *IEEE Data Engineering Bulletin*, 23(3):3–8, 2000.

[17] A. John and D. Seligmann. Collaborative tagging and expertise in the enterprise. In *Collaborative Web Tagging Workshop in conjunction with the 15th WWW Conference*, 2006.

[18] R. Kumar, J. Novak, and A. Tomkins. Structure and evolution of online social networks. In *KDD*, pages 611–617, 2006.

[19] C. Marlow, M. Naaman, D. Boyd, and M. Davis. Position paper, tagging, taxonomy, flickr, article, toread. In *Hypertext*, pages 31–40, 2006.

[20] D. Millen, J. Feinberg, and B. Kerr. Social bookmarking in the enterprise. *Social Computing*, 3(9), 2005.

[21] P. Schmitz. Inducing ontology from flickr tags. In *Collaborative Web Tagging Workshop in conjunction with the 15th WWW Conference*, 2006.

[22] S. Sen, S. Lam, A. Rashid, D. Cosley, D. Frankowski, J. Osterhouse, F. Maxwell Harper, and J. Riedl. Tagging, communities, vocabulary, evolution. In *CSCW'06*, 2006.

[23] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, Cambridge, 1994.

[24] B. Wu, V. Goel, and B. Davison. Topical trustrank: Using topicality to combact web spam. In *WWW*, pages 63–72, 2006.

[25] Z. Xu, Y. Fu, J. Mao, and D. Su. Towards the semantic web: Collaborative tag suggestions. In *Collaborative Web Tagging Workshop in conjunction with the 15th WWW Conference*, 2006.