IDENTIFYING AND MINIMIZING THE EFFECTS OF MALICIOUS BEHAVIOR IN SERF

Sandeep Natarajan
School Of Electrical Engineering and Computer Science
Oregon State University.
Major Professor: Dr. Jonathan L. Herlocker

ABSTRACT

Collaborative filtering (CF) algorithms are used in a wide range of internet applications. However the chief objective of using CF algorithms across most of these applications is to discover items that might be of interest to its users. CF algorithms work by obtaining feedback from users on the items that they browse and utilize that feedback to suggest recommendations to other users with similar tastes. CF algorithms rely heavily on input provided by humans and thus it is vital to verify that this information is appropriate. In this paper, we analyze various mechanisms by which users can enter malicious data to a CF system called SERF (System for Electronic Recommendation Filtering). We explore how bad data can be propagated through the system and can be used to manipulate the quality of recommendations. We also explore some techniques to counter the effects of bad data on the system. We report the results of our experiment with two simulated systems - a reputation system that utilizes a user's agreement and disagreement history to predict the trust that can be attributed to a user and a word weighting scheme based on word co-occurrence.

1. INTRODUCTION

The internet has enormous amounts of information and finding information specific to one's need is a difficult task. Search engines play an important role in bridging the gap between an information seeker's quest for information and the actual source of information.

To match an *information need* to the right *information source*, search engines use metadata, information that describes the *information source*. To collect metadata search engines use a variety of methods like explicit link analysis and implicit link analysis [6]. However the internet has sources of information that do not support this kind of metadata. Other means of obtaining metadata for these *information sources* will make finding them easier.

The goal of SERF is to involve humans in the automated process of finding *information sources* to improve the quality of information returned by a search engine. SERF uses feedback provided by users on search results to recommend *information sources* to requests with similar *information need*. SERF does this by integrating functionality commonly associated with a question answering (QA) system and a collaborative filtering (CF) system into an innovative new search engine.

In a QA system the user expresses her *information need* with a detailed and well formed question and the system returns an answer specific to that question. The key advantage a QA system gives its implementers is that the *information need* is specified precisely and grammatically, thus giving more *information context*. Similarly in SERF we encourage the users to specify their *information need* with a detailed question.

In a CF system, users with similar interests are matched and the preferences of one user are recommended to others in the group. This process of explicitly obtaining a users opinion on items in the system is called relevance feedback. The idea is that by getting relevance feedback, the system can easily eliminate the bad items from consideration and promote the good items. In SERF we recommend the *information sources* that have a positive feedback to other members with similar *information need*. The notion is that if most users find an *information source* to be relevant to an *information need* then a new user might also find it relevant to a similar *information need*.

In SERF we store both the *information need* (question) specified by the user and her feedback on the results that are generated for that *information need*. When some other user asks a question with similar *information need* we use the feedback provided by the previous user to either recommend interesting *information sources* or to remove the sources of information that are not useful from the search results. Here we have assumed two things. First is that the users provide genuine and meaningful questions using

the QA interface. Second is that the users provide honest feedback on the *information sources* that they are looking at.

It turns out that these assumptions do not always hold true. We have cases when users provide questions that are inappropriate for a number of reasons. We also have cases where users provide feedback for their own personal benefit or mistakenly provide misleading feedback. The chief objective of this paper is to identify what the goals might be for such a behavior and use this knowledge to minimize the effect of such actions on the system.

In this paper, we outline the nature of data that lends to manipulated recommendations. We classify them into different categories based on their effect on the system. We believe that studying and discretizing malicious user behavior will illustrate the complexities of building a robust system. We also initially explore solutions to different types of attacks on the system.

1.1 Terminology

In the rest of this paper we will use the following terminology:

i) *Question*: The *information need* of a user represented as a meaningful question. These questions are better expressed than the keyword based searches often used in search engines.
ii) *Document*: The *information source* that provides answers to some question.
iii) *Rating*: The relevance feedback given by a user denoting if a document answered a question.
iv) *Metadata/Meta-information*: Information that describes information. In this paper we consider the questions and ratings to be the meta-information about the document.

1.2 Interaction Interface

For the purposes of this paper we will consider those interfaces of SERF that can be used by the user to input information into the system. We identified two components in the interface that directly affected the recommendation algorithm.

i) *Question box*: Users provide their questions in a question box as shown in Figure 1. The interface encourages the user to provide a detailed question through the use of a large text box.
ii) *Rating buttons*: Users rate the documents as relevant or irrelevant to the question that they provided. Thus each rating corresponds to a question-document pair i.e., a document is either relevant or irrelevant to the question. We use a binary rating system to keep the interface simpler. Figure 2 shows the rating interface.

Figure 2 also shows the results page that is generated in response to a user's question. The results page contains documents generated both from the search engine and the documents recommended by SERF. The documents recommended by SERF are marked using a star in front of them.

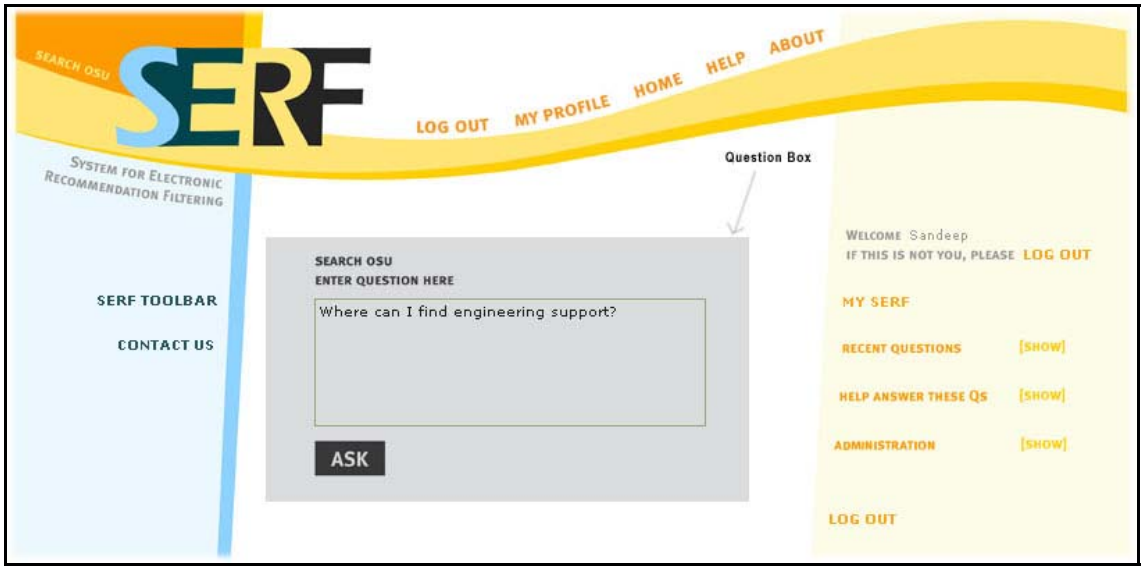

Figure 1: SERF Question Box

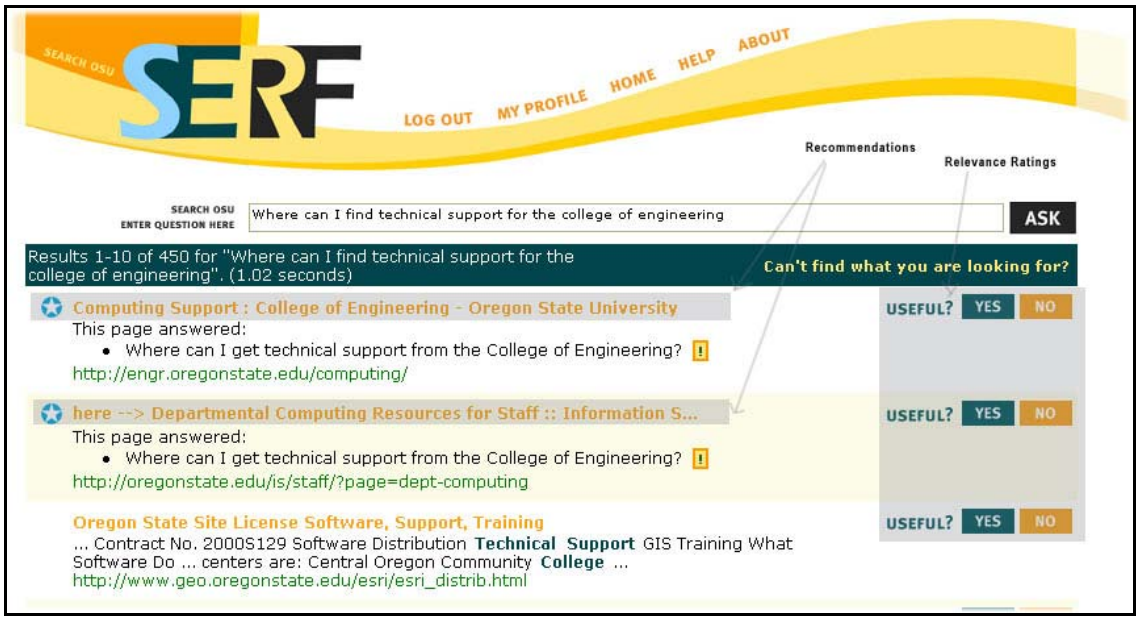

Figure 2: SERF Rating Buttons

1.3 Challenges

For SERF to work efficiently, we need users to actively interact with the various components of the system. In our attempt to convince the users to continuously interact with the system, we faced the following challenges:

i) Motivating users to provide questions that precisely and completely reflect their *information need*.
ii) Correctly matching *information needs* of different questions to make good recommendations.
iii) Motivating the users to rate documents as relevant or irrelevant to their *information need*.
iv) Maintaining the correctness of meta-information obtained from the users over a period of time.

It can be seen that each of the above mentioned challenges deals with a different aspect of SERF. The chief goal of this paper is to analyze the last challenge– maintaining the authenticity of the meta-information provided by the users. We try to capture and eliminate data that falls in to one of the following categories:

i) *Tampered questions*: Some users try to fiddle with the system by providing questions that either have content that is undesirable for others to see or by providing questions that have been tampered to manipulate the recommendations that other users get.
ii) *Misleading ratings*: The ratings provided by a user can be incorrect either because the user tried to cheat the system or because the user made a mistake. We analyze the incentives and strategies available to the users who try to deceive the system.
iii) *Time sensitive data*: Both questions and documents may lose their relevance as time passes. Keeping track of what information is still valid and what is obsolete is challenging. We provide an interface that would help the system to capture time sensitive data.

In this paper we analyze each of these issues in detail. We also propose a simple meta-rating system that promises better system performance compared to the automated defense mechanisms that we will discuss later in this paper.

2. RELATED WORK

There are a large number of internet portals that use CF algorithms and Reputation Systems to improve and regulate the quality of their services. Some popular online portals that use these techniques include eBay, Amazon, Yahoo Launchcast, Epinions, Bizrate and Slashdot. There is a significant amount of research on these systems.

These systems can be classified based on the type of values they accept for the ratings. Systems that allow only two values for the ratings are called *binary rating systems* [3]. These ratings are usually in the form of true/false, good/bad, relevant/irrelevant etc. These systems sometimes also have a third level of rating which signifies that the rated item is neither good nor bad. eBay is a typical *binary rating system* which allows its buyers and sellers to give positive, negative or neutral ratings.

These online portals can also be classified as *unidirectional* and *bidirectional systems* based on whether or not they allow their users to be rated by other users [1]. Systems that allow the users to be rated are called *bidirectional rating systems* while systems that allow only the services/products to be rated are called *unidirectional rating system*. eBay is a *bidirectional rating system* and Amazon is a *unidirectional rating system*. There are also systems like Epinions that allow both the users and their services to be rated. SERF uses *unidirectional rating* mechanism.

2.4.1 Attacks on Collaborative Filtering Systems

Any attempt by a user to manipulate the order of items in the recommendation list is considered to be a potential attack on the system. O'Mahony et al. [10] have done a robustness analysis on CF algorithms and proposed metrics to measure the effects of attacks on the system. They have also proposed two types of attacks viz. *product push/nuke attacks* and *random attacks* on CF Systems.

Riedl et al. [8] have built on the work by O'Mahony et al. [10] and experimentally explored affects of different attacks on various well known CF algorithms. They have also identified properties of items that are easier to attack and are more frequently attacked.

While these studies concentrate on the effects of various anomalies on the system, we study how these anomalies can be introduced into the system. We use this knowledge to propose solutions to minimize the effects of these attacks.

2.4.2 Reputation Systems for Collaborative Filtering Systems

Every CF System has an inherent Reputation System in it. The Reputation System is the one that collects ratings on the items and decides on how these ratings are employed in the recommendation process. It makes decisions like what weight can be given to each user's rating and how these are combined to evaluate the true worth of an item.

The reputation systems can either be automated or non-automated. In automated reputation systems, the system analyzes all the ratings given by a user and builds a trust value for the user that signifies how much the user can be trusted. This trust value is then used to weigh the user ratings while providing recommendations to other users. In non-automated reputation systems, the system just displays the feedbacks gathered on a user or service/item/product and the active user makes a decision whether the resource under question is useful or can be trusted based on viewing the feedback.

Dellarocas [3] has analyzed the behavior of sellers and buyers in a *binary rating system* namely eBay and has shed light on how effective the binary rating system is in inducing efficient outcomes. While Dellarocas studied this in the eBay market place which does not predict the user trust automatically, we study how an automated reputation system i.e., SERF responds to a *unidirectional rating system* which employs a *binary rating mechanism*.

Chen et al. [1] have proposed a hierarchical reputation system and used the reputations generated by this system to evaluate the rated objects. Donovan et al. [4] have argued that profile similarity alone cannot yield good recommendations and proposed a reputation system based on profile-level and item-level trust and used this to weigh the ratings provided by a user. Massa et al. [9] have proposed a mechanism of propagating the trust values between users and analyze the potential contribution of trust metrics in increasing the performances of Recommender Systems. Dellarocas [2] has proposed and evaluated a set of mechanisms which eliminates or significantly reduces the negative effects of fraudulent behavior.

All of these studies concentrate on developing reputation systems to generate trust values for the user which can be used to weigh the ratings provided by them. We are concerned about how a reputation system reacts to attacks from malicious users.

3. UNDESIRABLE DATA

The main goal of SERF is to help users find *information sources* that are difficult to locate using regular search engines, by obtaining meta-information about these sources of information from the user community. Thus SERF maintains a lot of user provided meta-information. Filtering out those data which do not truly describe the *information source* is important to ensure the robust working of the system. To identify what constitutes ambiguous data let us first look at how SERF works.

3.1 SERF Architecture

SERF has two important components: a question matching system and a recommendation system. The question matching system identifies questions with similar *information need*. The recommendation system collects feedback in the form of ratings from the users and uses it to recommend documents to questions that were identified as similar by the question matching system.

3.1.1 Question Matching System (QMS)

To match questions based on *information need* we require a procedure to identify these *information needs*. In the absence of context or any such information about the questions, the best approach to identify the *information need* would be to recognize those words in the question that convey the *information need* more precisely than other words in the question. One approach to doing this is to use word frequencies to develop a weighting scheme on the words. In SERF we use *normalized term frequency* and *inverse question frequency* to weigh words. These word weights are then incorporated into the widely known vector-space model to calculate the similarity between two questions. That model works as follows: let $w_{i,q}$ be the weight of a word i in question q. Then

$$w_{i,q} = tf_{i,q} * iqf_i$$

Here $tf_{i,q}$ is the *normalized term frequency* of word i in question q and idf_i is the *inverse question frequency* for word i across all the words that have ever appeared in a question. *Term frequency (tf)* measures the importance of a word relative to other words in the question. *Inverse question frequency (iqf)* measures the importance of a word relative to all the words that have been seen in questions recorded by the system.

$$tf_{i,q} = \frac{freq_{i,q}}{\max_l freq_{l,q}} \qquad iqf_i = \log \frac{N}{n_i}$$

$freq_i$ is the frequency of word i in question q, $freq_{l,q}$ is frequency of the word that occurred most frequently in question q. N is the number of words seen across all observed questions and n_i is the frequency of word i across all the questions recorded by the system so far. Figure 3 shows the *inverse question frequency* of all the words captured by the system.

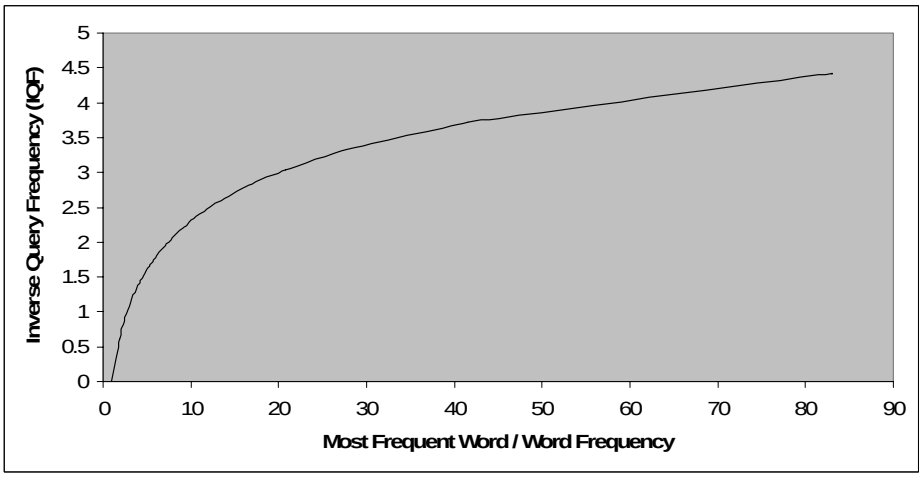

Figure 3: Inverse Question Frequency of words in SERF

The value of *tf* lies between 0 and 1, 0 being least significant and 1 being highly significant. A word that occurs most frequently in a question has a *tf* of 1. This approach rewards words that occur more frequently in a question. The *iqf* rewards those words that help in differentiating the questions in our collection. It penalizes the words that appear in many questions – thus words that have little value in differentiating between potentially similar questions. The *log* term in *iqf* is used to account for the fact that words that occur in 10 questions are not 100 times more significant than words that occur in 1000 questions (Figure 3 In other words it reduces the range of values that the *iqf* takes from [*0, N*] to [*0, log N*]. The word that has occurred only once across all questions recorded so far has an *iqf* of *log N*. In section 3.2 we will discuss how these factors can be used to manipulate recommendations.

The *QMS* treats each question as a vector in an *n*-dimensional space, where *n* is the number of words that have been recorded by the system so far. The component of the vector along each axis is the word weight for the word that axis represents. The similarity between two questions is calculated by calculating the angle between the vectors representing those questions. A smaller angle between the vectors signifies that the questions are similar. The angle between the vectors is calculated using dot product. If we have two questions q_1 and q_2, then the similarity between these questions is calculated as:

$$sim\ (q_1, q_2) = \frac{q_1 . q_2}{|\vec{q}_1||\vec{q}_2|} = \frac{\sum_{i=1}^{t} w_{i,q_1} * w_{i,q_2}}{\sqrt{\sum_{i=1}^{t} w_{i,q_1}^2} \sqrt{\sum_{i=1}^{t} w_{i,q_2}^2}}$$

Here w_{i,q_1} represents the weight of the i^{th} word in question q_1. This equation gives a scalar value in [*0, 1*] for the similarity between two questions. A value of 1 denotes that the questions are exactly the same.

3.1.2 Recommendation System

The Recommendation System takes in two questions: an *active question* - the question asked by the current user - and a *past question* - the question which was previously asked by someone and is computed to be similar to the *active question*. It takes the documents that were rated as relevant to the *past question* and recommends them for the *active question*. The Recommendation System calculates the relevance of a document to the *active question* as the average of the ratings given to that document by users who asked that *past question*. If the number of users who rated the document as relevant to the *past question* is more than the number of users who rated the document as irrelevant to the *past question*, then the document is considered relevant to that *past question*. The ratings have to be on the exact same question.

To deal with malicious data in SERF we have to deal with the vulnerabilities of both the *QMS* and the *RS*. In the rest of section 3 we look at what constitutes a malicious data.

3.2 Misleading Questions

Questions represent the *information need* of a user. A malformed question does not denote the right *information need* and can lead to poor recommendations both to the current user and to future users. Maintaining the quality and precision of questions is important to better match *information needs* and to make recommendations. A malicious user can tamper with the contents of a question to promote or demote a specific document or to damage the system or just arrange for certain questions to be placed in front of people. In this section we discuss what constitutes a misleading question.

In the word weighting scheme that we discussed earlier, though there are two factors, namely *normalized term frequency* and *inverse question frequency* - the first factor almost always takes on the value 1 in SERF. The reason is that the words in most of the questions do not repeat in that question. If the question had just one repetitive word then the *tf* for that word would be 1 and the *tf* for all the other words in the question would be 0.5. If the question had two repetitive words then that word would have a *tf* of 1 and all the other words would have a *tf* of 0.34. Though *tf* takes a value of one in most of the cases we still include it in our formulation to serve questions that have repetitive words.

Since most questions do not have repetitive words, the question matching system relies heavily on the second factor to distinguish or equate two questions. A word that has occurred only once in the history of the system has the highest *iqf* of *log N*. A word that has occurred just two times in the history of the

system has the next highest *iqf* of *log (N/2)*.Someone who understands this mechanism can easily manipulate the output of the *QMS*. We will show how this can be done in the next section.

For the purposes of this paper we will consider an intuitive measure called entropy for each question. A question that describes its *information need* precisely has high entropy and a question that only gives a vague description of its *information need* has low entropy.

3.2.1 Low Entropy Questions

A Low Entropy question is one which comprises mostly of words that appear semi-frequently across all questions and have moderate *iqf*. Moreover, these words are generic and communicate little meaning on their own but enhance the meaning of the words with which they occur. Since these words are generic and semi-frequent, the probability that they will match with a good number of questions and information needs is high. It is undesirable to recommend questions that match only the generic words. Examples of these words are *website (iqf=2.16), information (iqf=0.79), help (iqf=3.03), time (iqf=2.3394)* etc. Consider the question "*Where can I find information on website help*". This question has three generic words *help, information* and *website*. This question partially matches the *information need* of any question containing one of these words. Any document rated relevant to this question is recommended to active questions with one of these three words even though the question is meaningless. Since these questions neither contain stop words nor derogatory words, automatic detection of these questions is a challenging task.

3.2.2 High Entropy Questions

High Entropy questions are those that are targeted towards a precise *information need*. They contain words that have high *iqf*. For example if the *information need* under consideration is *library*, then a high entropy question would contain words like *library (iqf=1.28), book (iqf=3.32), article (iqf=3.72), journal (iqf=2.62)* etc. When someone asks a question on that *information need*, these high entropy questions provide a better match because they contain words with high *iqf* for that information need. An example question is "*Where can I find books and journal articles in the library*". Any document that is rated relevant to this high entropy question would appear in the recommendation list for all the questions within that *information need*.

3.2.3 Multiple Questions

In both the previous categories of questions, a single question and the knowledge about how the system works can be used to make the question match with all the questions related to a specific *information need*. Another scenario that might reduce the performance of the system is multiple questions with slight variations in their contents. For example consider the three questions "*When does recreational center open*", "*What time does the recreational center open*" and "*What is the opening time for recreational center*". The advantage of having such multiple questions is that we now have multiple representations of the same *information need* and thus have a higher probability of matching a question with this *information need*. The disadvantage is that it leads to duplicate questions recommended to users.

3.2.4 Inappropriate Questions

The system is also prone to questions that either have inappropriate language or content. These questions are detrimental because questions are displayed to the user and if inappropriate questions are displayed then the user might lose trust from past uses with the system. Automatic identification of some questions with inappropriate language is possible by matching regular expressions. But identifying questions with inappropriate content is challenging. For example, consider the question "*Is Dr.Galor a loser*". This question will match with all the questions that have the words "*Dr.Galor*" and would be displayed to all the users who ask a question about "*Dr.Galor*". To identify such questions we need an admin to review all the questions before they are entered to the system, which is infeasible.

Table 1 shows a list of examples for the different types of problematic questions.

	Question Type	Problem	Example	Matches
1	Low Entropy Questions	– Contain generic, moderate *iqf* words. – Match with questions with different information needs.	– Where is the information on contact numbers?	– What is Mary's phone number? – Where can I find information on contacts in business school?
2	High Entropy Questions	– Contain specific, high iqf words. – Match question with a precise information need.	– Where can I find information on books and journal articles in library?	– Where is Linus Pauling's journal articles located in the library? – Are science fiction books available in the library?
3	Multiple Questions	– Multiple representations of the same information need – Duplicate recommendations	– When does Dixon open? – What time does Dixon open? – What are the hours at Dixon?	- What time does Dixon open? This question matches all the three questions.
4	Inappropriate Questions	– Derogatory content. – Inappropriate to display to the users.	– Is Mary a loser?	– Who is Mary? – What is Mary's phone number? – Where is Mary's office?

Table 1: Examples of different types of misleading questions

3.3 Misleading Ratings:

Relevance ratings serve as a means by which a user can express her satisfaction or dissatisfaction on the documents that are being presented to her with respect to her current *information need*. A document can either be rated relevant to a question or irrelevant to a question. A document can also be rated relevant to one question and irrelevant to some other question. The relevance ratings represent a mapping from questions to documents. To maintain the quality of the recommendations, the recommendation system has to be robust to incorrect ratings. The relevance ratings can be interpreted in one of the following ways:

i) *True relevance*: The document is relevant to the question and the user rates it as relevant.
ii) *False relevance*: The document is irrelevant to the question but the user rates it as relevant.
iii) *True irrelevance*: The document is irrelevant to the question and the user rates it irrelevant.
iv) *False irrelevance*: The document is relevant to the question but the user rates it irrelevant.

True Relevance and *True Irrelevance* ratings result in desired behavior but *False Relevance* and *False Irrelevance* ratings result in undesired behavior. *False Relevance* ratings cause a document to appear lower in the recommendation list than its desired position and *False Irrelevance* ratings cause a document to appear at a higher position.

To understand what comprises misleading ratings let us consider the different rating scenarios possible. Table 2 shows the input to the recommendation system and its interpretation. We consider pairs of contiguous ratings given to a question-document pair.

	Successive Ratings	Interpretation
1	Relevant, Relevant	Consistent behavior.
2	Irrelevant, Irrelevant	Consistent behavior.
3	Relevant, Irrelevant	If both the ratings are true, it represents a subjective difference in opinion. Otherwise it represents an incorrect rating by one of the users.
4	Irrelevant, Relevant	If both the ratings are true, it represents a subjective difference in opinion. Otherwise it represents an incorrect rating by one of the users.

Table 2: SERF Relevance rating scenarios

Here cases 1 and 2 represent desired behavior. If most of the successive ratings for a question document pair are consistent then it suggests that all the users who are viewing the question-document pair agree on the relevance. But if we get a significant variance in the successive ratings it means that we either have a subjective difference in opinion, an error, or a malicious rating. If it is a subjective difference in opinion it might signify that the question does not clearly state the *information need*. But if it is not a subjective difference in opinion then it represents a malicious rating or an error and we would like to flag it or remove it from the system.

In our discussion on successive ratings, we have assumed that the relevance ratings we get are on the exact same question-document pair. But in reality the chances that we will get the exact same question from two users is small. In most of the scenarios users provide the same *information need* using different question representations. Thus even if the ratings are on a bunch of similar questions and a document, it is hard to tell if the questions represent the same *information need*.

3.4 Time Sensitive Data:

There is a separate class of data that we need to account for. Some ratings data are time sensitive and can go obsolete over time. We call this time sensitive data. It is important to distinguish between those ratings data that are malicious and those ratings data that are time sensitive.

For example consider the question "*Who teaches CS 411*". The answer to this question would change based on the term the question is asked in. Thus a document which is relevant to the question in *fall* might be irrelevant in *spring* and become relevant again the following *fall*.

4. MALICIOUS BEHAVIOR

In CF Systems there may be some users who try to meddle with the working of the system to attain some gain from the system. In this section we try to identify those incentives and strategies that might motivate a user to tamper with the system. We expect that understanding the attack and the attacker will help us lead to identifying a defense mechanism against the attack.

4.1 Attack Incentives

O'Mahony et al. [10] have discussed the various attacks on CF Systems. They have proposed three different types of attacks viz. *product push/nuke* and *random attacks*. We adapt and redefine their classification to match SERF. We propose that there are three types of attack based on the potential incentives: *boost a document*, *sink a document* and *damage the system*.

We redefine these attacks here because the systems in [10] and [8] are significantly different from our system. In SERF the user inputs two types of data viz. *questions* and *ratings* that the system processes to generate recommendations. In the systems defined in [10] and [8], only the user ratings are processed to generate recommendations. Though these categories signify the same type of attacks the actual formulation of these attacks in our system would be significantly different from theirs.

For the purpose of further discussion we will use the terms *target question* and *target document* to refer to a question which is being attacked i.e., the recommendation list is being modified and a document which is being attacked i.e., the relevance value to the *target question* is being modified respectively.

4.1.1 Boost a Document:

The core product of SERF is a document or *information source*. The goal of the system is to recommend documents to questions that meet the *information needs* of those questions. Attackers might want to have a document recommended to a question to which the document is not the most relevant source of information. Thus the system has to face scenarios where the predicted relevance of the document might be amplified. We call this type of attack boosting a document because it involves increasing the predicted relevance of a document through illicit means.

To boost a *target document*, the relevance of the *target document* to the *target question* has to be manipulated to a higher value relative to other documents. This relevance value has to be higher than the relevance values for other documents that are computed to be relevant to the *target question.* Such a scenario involves a *false relevance* for the *target question-target document* pair or a *false irrelevance* rating for the *target question* and other competing documents.

4.1.2 Sink a Document:

Attackers might also want a document to not show up in a recommendation list when it is a relevant source of information for a question or a group of questions. The attacker might use techniques to lessen the predicted relevance of a document. We call this type of attack sinking a document. It can be seen that boosting a document automatically causes some other document or documents to sink and vice versa.

To sink a document, the relevance of the *target document* to the *target question* has to be modified to a lower value relative to the relevance values for other documents that are computed to be relevant to the *target question*. This involves either a *false irrelevance* for the *target question-target document* pair or a *false relevance* to the *target question* and other documents.

In both boosting a document and sinking a document, in a system that gives equal weight to all the ratings, every false rating can be counter balanced by a true rating and vice versa.

4.1.3 Damage the System:

While there are attacks that are motivated to modify the predicted relevance of a document, there are other attacks that can reduce the performance of the system. One approach to doing this is by tampering the questions displayed to users, which has a direct impact on the performance of the system since we use questions to match *information needs*.

4.2 Attack Strategies

Given that we know what comprises malicious data and why someone might be motivated to attack the system, the task now is to identify strategies that an attacker would use to seed malicious data to the system to achieve one of the discussed incentives. Let us look at some of the possible strategies that an attacker could use to attack the system. Here we are concerned only about those types of attacks which involve an attacker or a group of attackers using the system interface. We are not concerned about network level attacks like denial of service.

4.2.1 Multiple Profiles

CF systems work by delegating the task of filtering out unwanted items to the users of the system. Each user provides feedback on the items that she comes across which is then stored in her profile and used both to recommend items to her and to other users. Thus in order to get meaningful results from the system the user needs to create a profile. In the absence of a definitive way of authenticating the user as who she claims to be, it is possible that a user might create multiple profiles. Since each user's profile is used to recommend items to other users, an attack can be scattered across multiple profiles.

4.2.2 Questions as a Tool

In SERF we recommend documents based on questions and not profiles. A question describes an *information need*. Through ratings, questions are linked to documents to become meta-information about the *information sources*. Thus the recommendation a user gets depends on how the user formulates her *information need* as questions and how other users formulated their questions for the same *information need*. If an attacker wanted to boost a document or sink a document for a particular *information need*, the attacker would have to formulate the question in such a way that it matches the potential questions for that *information need* and recommend the *target document* as the relevant source of information to the *target question*. Similarly to show inappropriate questions to other users, the attacker can use similar mechanisms.

We have already discussed the different mechanisms of formulating a question in section 3.2. The combination of a set of malicious questions and multiple profiles can make the task of detecting an attack hard.

4.2.3 Ratings as a Tool

Ratings are the attestation given by users which establishes that the document actually represents the *information need* described by the question. In other words, ratings serve as a mapping from questions to documents. The strength of this mapping depends on who provided the ratings and how many ratings were provided.

The strength of a mapping between a question and a document denotes how close a document is in answering the question. We assume that if many users rate a document as relevant to a question then other users would also find the document to be relevant to that question. But considering the number of ratings to be an indicator of the relevance of a document to a question involves a risk factor.

An attacker could give multiple ratings on a question-document pair in order to boost a document or sink a document. Thus ratings can be used to modify the mapping between a question-document pair. A motivated attacker could also use a combination of multiple profiles, question variations and multiple ratings to design an attack. This type of attack is very hard to detect.

4.3 Defense Mechanisms

We now have some insight into the incentives and strategies that would steer an attack on the system. In this section we will discuss what are the possible solutions to each of these attacks and what are the costs associated with these defense mechanisms.

4.3.1 Multiple Profiles

Most CF Systems face the problem of attacks using multiple profiles. Current systems tackle this by increasing the cost of creating multiple profiles by associating the profiles with something unique and authenticable. Some of the things that can be used to authenticate a user are Social Security Number, Driving License Number, Credit Card information etc. We could do this in SERF by requiring users to authenticate using ONID accounts. Anybody associated with OSU has a unique ONID account. A multiple profile attacker would then need to create multiple ONID accounts which is relatively harder. Another approach could be to have each profile creation approved by a moderator.

4.3.2 Defending Against Tampered Questions

Questions are used to match *information needs* and to generate recommendation list. They are the link between a user's *information need* and the recommendation list that they get, and hence are more prone to attacks. We need mechanisms of identifying high entropy questions, low entropy questions and inappropriate questions. If we could identify them, then we could design mechanisms to filter them from the question matching process or the recommendation list. In section 3.2, we saw that the high entropy questions consisted of specific words that had high *iqf* and the low entropy questions had generic words that had moderate *iqf*.

One approach to build a better word matching scheme that is robust to attacks would be by using a thesaurus. A thesaurus can be used to identify words that convey the same information need. For example if we can identify that the words 'situated' and 'located' mean almost the same thing then we can identify that the questions "Where is Library located" and "Where is Library situated" mean the same thing. Thus we could prevent attacks designed using multiple questions.

We could also defend against inappropriate questions using regular expression matching. This is used in many modern systems to filter out inappropriate words. Another approach to defending against tampered questions would be to have moderators review all the questions.

4.3.3 Defending Against Misleading Ratings

In SERF, single ratings are less harmful than multiple ratings because a single *false rating* can be counter balanced by a *true rating* from another user but if there are multiple *false ratings* then we need at least an equal number of *true ratings* to counter the *false ratings*. The challenge is in tackling the situation where an attacker gives multiple *false ratings*. In SERF we limit the number of ratings that a user can give on the same question-document pair to one. But an attacker could give multiple ratings on variations of a question and the target document. One approach to preventing this would be to have an administrator approve all the ratings. But this is infeasible when the size of the ratings received is very high.

In a situation where the attacker is using variations of the same question, an approach where the administrator approves each question may not work because the number of questions is high. Another approach would be to collect and analyze the counter ratings to build a reputation system. A reputation system automatically analyzes the rating patterns and predicts a trust value for the users. These trust values can be used to infer situations when somebody may be trying to attack the system.

To summarize this section Table 3 shows the various attack strategies and defense mechanisms. The defense mechanisms listed in the table are not exhaustive but show the most popular approaches.

	Attack Strategies		Defense Strategies
1	Multiple Profiles	Multiple login ids	– Higher cost for creating profiles. – Approval of account creation by a moderator
2	Tampered Questions	High Entropy Questions	– Thesaurus generation – Regular expression matching – Approval of all the questions by a moderator
		Low Entropy Questions	
		Inappropriate Questions	
		Derogatory Questions	
3	Misleading Ratings	False Relevance	– Reputation Systems – Approval of all the ratings by a moderator
		False Irrelevance	
		Random Ratings	

Table 3: Summary of Malicious behaviors in SERF

5. EXPERIMENTS ON AUTOMATED DEFENSE STRATEGIES

In sections 3 and 4 we discussed the various data vulnerabilities and the various mechanisms by which an attacker could use the susceptible data to attack the system. These vulnerabilities arose due to the weaknesses in the question matching system and the recommendation system. We discussed how the question matching system can be tricked to believe that two questions are similar, by exploiting the word weights in those questions. We also saw how the recommendation system can be tricked to believe that the ratings were not malicious by giving ratings from multiple profiles or on multiple questions. In this section we analyze two automated defense mechanisms that could potentially be used to defend against these attacks. The first one is a word weighting scheme based on word co-occurrence that could be used to identify questions as high entropy or low entropy. The second one is a reputation system that predicts a numeric value for the amount of trust that can be placed on a user and could be used to weigh the user's ratings.

5.1 Word Weighting based on Co-occurrence

The word weighting schemes and the vector-space model that we currently use in SERF are the most popular techniques among information retrieval systems. The vector-space model in general is used to match questions with documents, unlike in SERF where we use it to match a question with other questions. A question typically contains about 5 to 10 words where as a document may contain words in the order of thousands. Since the number of words in a question is very small compared to the number of words in a document, it is relatively easier to manipulate questions using the words they contain.

Another popular approach to content matching among information retrieval systems is to weigh words based on word co-occurrence. The algorithms in this category presume that words which occur together are not independent of each other. The assumption is that occurrence of one word increases or decreases the probability of occurrence of some other words. In this section, we will analyze such an approach for SERF.

In section 3.2, we discussed how an attacker could mislead the system by using the knowledge about word weights to formulate misleading questions. The words in these misleading questions have a distinctive property. The words in high entropy questions are specific, have relatively higher *iqf* and relate to different *information needs*. The words in low entropy questions are generic, have moderate *iqf* and do not relate to any *information need* but tend to occur in a lot of questions.

To withstand attacks designed using misleading questions we design and analyze an approach based on the co-occurrence of words to calculate word weights. We define a *dependence factor* for all the words captured by the system. The objective of using a *dependence factor* is to identify the generic words that appear in low-entropy questions and specific words that appear in high entropy questions. This approach of using a *dependence factor* was motivated from the item-based co-occurrence algorithm discussed in [7]. We take into account the following aspects to calculate the *dependence factor*:

i) Words that co-occur in a large number of questions should have a higher *dependence factor* because the occurrence of one word increases the chances of occurrence of the other.

ii) Words that do not co-occur in a large number of questions should have a lower *dependence factor* because the occurrence of one word decreases the chances of occurrence of the other.
iii) Dependence factor should also take into account the *iqf* of each word, which is a measure of how important a word is in distinguishing questions.

We divide the dependence factor into two sub-factors– a *conditional probability factor (cpf)* and an *inverse question factor (iqf)*. We use the following notations for in our model:

Q_T Target Question. Question for which we want to find similar questions.
Q_C Candidate Question. Questions that might be similar to the Target Question.
w_t Target Word. Word for which we want to find the correlation with other words.
w_c Candidate Word. Words with which w_t co-occurs.

The dependence of a word w_t in the target question to a word w_c in the candidate question is calculated as:

$$df(w_t, w_c) = cpf * iqf$$

$$cpf = P(w_c \mid w_t) = \frac{freq(w_c, w_t)}{freq(w_t)}$$

$$iqf = \log\left(\frac{N}{freq(w_t)}\right) * \log\left(\frac{N}{freq(w_c)}\right)$$

Given the above word weights we could calculate the similarity between two questions as:

$$sim(Q_T, Q_c) = \frac{\sum_{t \in Q_t, c \in Q_c} df(w_t, w_c)}{\sqrt{\sum_{t \in Q_t, c \in Q_c} df(w_t, w_c)^2}}$$

5.1.1 Results:

We simulated this model over the SERF dataset which was obtained over the past 2 years. The dataset consisted of 941 questions with about 1141 keywords. We omitted the stop words like *the, is, a* etc. from our analyses.

Our simulation showed that the *dependence factor* which was based on word co-occurrence, assigned values as follows:

- Higher values to pairs of words that frequently co-occurred and had high *iqf*.
- Lower values to pairs of words that did not co-occur frequently but had high *iqf*.
- Lower values to pairs of words with moderate *iqf*.

This word weighting scheme based on co-occurrence was able to achieve the goals that we set for the *dependence factor*. It was able to distinguish words that co-occur and words that do not co-occur. It was also able to discriminate words based on the *iqf*. Appendix A shows some example questions and *dependence factors* that were calculated by this scheme.

Though the *dependence factor* showed improvement over the previous weighting scheme, using the *dependence factor* to match questions is a challenging task. As can be seen from our formulation, in order to match questions using word correlations we would have to calculate the correlation of each word in the target question to every word in the candidate question. This would be of time complexity ($m*n$) where m and n are the number of words in the target and candidate questions respectively. If there are q questions in the system then to match questions we would have to do ($m*n*q$) operations every time. This is expensive and we would need a better approach to use *dependence factor*.

5.2 Reputation System

Reputation Systems are used to predict a value for the trust that can be attributed to a user's feedback based on the reliability of feedbacks provided by her in the past. The assumption is that users who provided bad data in the past have a higher probability of providing bad data again either due to malicious intent or due to mistake. We presume that past behavior serves as an indicator of future behavior. We hypothesize that by weighing the input from a user based on her trust calculated from her past behavior, the impact of inappropriate data can be minimized.

Reputation systems can be classified into *automated* and *non-automated reputation systems* based on whether they calculate a trust value from the ratings or not. *Automated reputation systems* can be further classified into two types– *relative trust systems* and *absolute trust systems*, depending on the mechanism they use to calculate the trust values. In *relative trust systems*, the trust is calculated from the perspective of a user. Thus a user might have a high trust value from the perspective of one user and a low trust value from the perspective of some other user. In *absolute trust systems* each user has only one trust value, which is from the perspective of the system.

We simulated an *absolute trust system* to determine if it would be possible to capture the user behavior by predicting a trust value based on her rating pattern. In our model the system assigns a trust value for a user based on how she rates an item and how others rate that item.

We propose that the trust on a user should exhibit the following properties:

i) Trust should be directly proportional to the agreement of the user with the rest of the community.
ii) Trust should be inversely proportional to the disagreement of the user with the rest of the community.

We hypothesize that users with higher agreement with the community, serve the community better than users with lower agreement. In SERF we have the following entities:

U	Set of all users. $\{u_1, u_2 \ldots, u_n\}$
Q	Set of all questions. $\{q_1, q_2 \ldots, q_n\}$
D	Set of all documents. $\{d_1, d_2 \ldots, d_n\}$

We will use the following terms for further discussion:

u_a	The active user for whom we are predicting the trust value.
QD	Set of question-document pairs rated by the user u_a
QD_{owner}	The set of qd's rated by u_a, for which u_a is the first rater.
T	A trust value assigned to each user

Initially we assume that each user has a trust value of 0.5 which means the system neither fully trust nor fully distrusts a user. Given the ratings specified by the user u_a for the set of documents in QD_{owner} and a value for trust for other users in the community, we calculate the agreement and disagreement that the user u_a receives from the rest of the community as follows:

$$Agree = \sum_{qd_i \in QD_{owner}} \frac{Trust \quad of \quad people \quad who \quad agree \quad with \quad user \quad u_a \quad on \quad qd_i}{Total \quad trust \quad of \quad people \quad who \quad voted \quad qd_i}$$

$$Disagree = \sum_{qd_i \in QD_{owner}} \frac{Trust \quad of \quad people \quad who \quad disagree \quad with \quad user \quad u_a \quad on \quad qd_i}{Total \quad trust \quad of \quad people \quad who \quad voted \quad qd_i}$$

These two variables– (*Agree*, *Disagree*) are normalized over the number of documents that the user has rated and the average agreement and disagreement for the user is calculated.

$$Agree_{avg} = \frac{Agree}{|QD|}$$

$$Disagree_{avg} = \frac{Disagree}{|QD|}$$

We use these $Agree_{avg}$ and $Disagree_{avg}$ factors to generate a trust value for a user which reflects the properties of trust that we defined earlier.

$$T = Agree_{avg}^{\ Disagree_{avg}}$$

$Agree_{avg}$ takes values in [*0, 1*] and $Disagree_{avg}$ takes values in [*0, 1*]. Thus the value for trust also lies in [0, 1]. Note that the sum of $Agree_{avg}$ and $Disagree_{avg}$ is equal to 1. In this equation the value of trust would be high if $Agree_{avg}$ is high and the value for trust would be low if $Disagree_{avg}$ is high (Figure 4). The intuition for using this equation is based on the properties of trust that we proposed earlier.

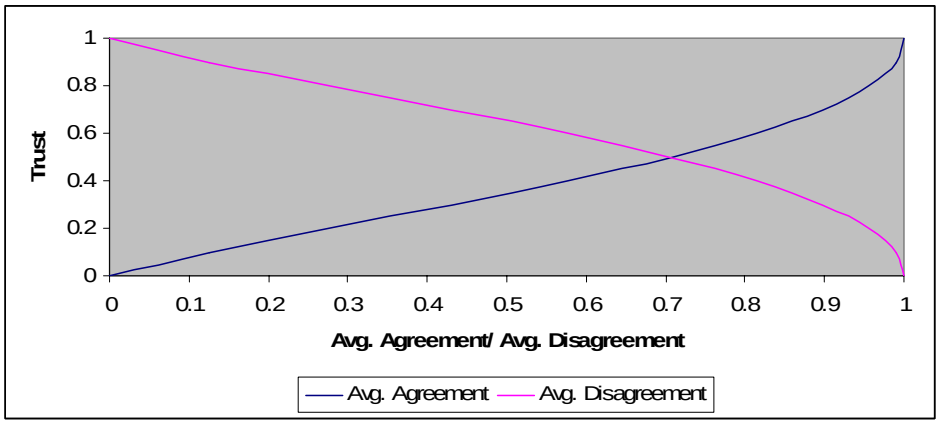

Figure 4: Trust vs. Average Agreement and Average Disagreement

We assume that the trust of a user depends only on the agreement and disagreement on the documents for which the user is the first ratings provider. We do this to avoid users from building a high trust by rating the question-document pairs that are already rated. Thus to build a high reputation with the system a user has to provide new recommendations and get good ratings on them.

5.2.1 Experiment Design

To test the robustness of this model we defined five types of users in the system:

- *Regular Users*: These users always give the true rating for a question document pair with an error rate of 2%.
- *Attackers*: These users are trying to attack the system for one of the incentives already discussed in the earlier sections. In order to make the attack more powerful we assumed an error rate of 2% for these attackers, except on the list of target documents which they are attacking, in which case they give false ratings deliberately. Moreover we assumed that an attacker provides ratings at least four times the ratings of a regular user. We assume four type of attackers:
  - *Boost Attackers*: These users try to boost one or more documents for a question.
  - *Sink Attackers*: These users try to sink one or more documents for a question.
  - *Group Attackers*: These are set of users who are trying to either boost or sink one or more documents.
  - *Random Attackers*: These users give random ratings on all documents. They don't have any specific target documents which they want to attack.

To keep the simulation simple, we assume each question-document pair to be an item. Thus ratings are obtained on items rather than question-document pairs. For each item we randomly define a true rating. True rating is the rating that would satisfy most of the users in the community. Additionally we made the following assumptions for the simulation which were based on the usage statistics for SERF:

- Number of users: 100, 200, 300.
- Number of items: 300, 600, 900 respectively.
- Number of votes: 100, 2000, 3000.

Further we assumed that 1 in every 5 users was an attacker of one of the proposed types and the attacker could attack up to 5 documents. These two assumptions were arbitrary and were made to imitate a realistic situation. We start with a reputation value of 0.5 for each user which reflects the fact that the system neither trusts nor distrusts the user initially.

5.2.2 Results

The trust values predicted for the different types of users are shown in the Figure 5. In the bar graph, the x-axis shows the number of documents that each user attacked. The y-axis shows the trust on the user. Average values of trust for the five categories of users are calculated. The average value of trust for a user with no attacks is 0.91. The following can be inferred from the graph:

- The average trust of random attackers is always less than the average trust of all users in the system.
- The average trust of boost and sink attackers seem to be random but are close to the average trust of all users.
- The average trust of group attackers is almost always higher than the average trust of all users in the system.

In our simulation random attackers where correctly identified. These attackers consistently received a trust value lower than the rest of the community. The reason is that random attacks take place with no knowledge of the system. Since the attacker randomly rates the documents the number of disagreements with other users is almost equivalent to the number of agreements.

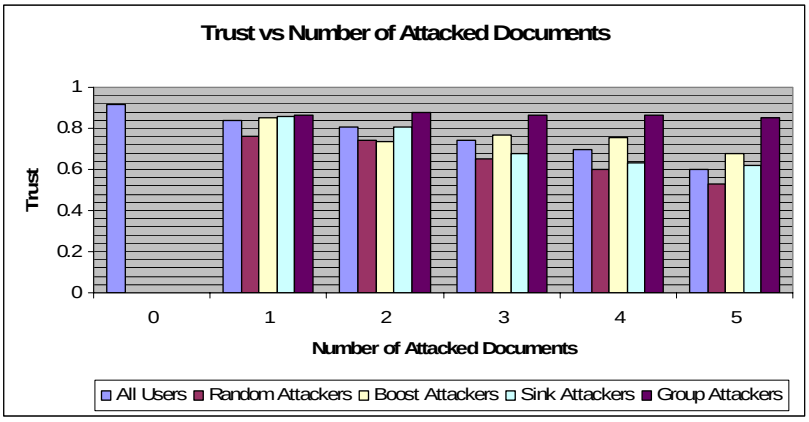

Figure 5: Trust vs. Number of attacks

The reputation system could not identify the boost and sink attackers. Their trust values were as high as the regular users. This happened because these attackers carried out a planned attack. They gave true ratings for all the documents except the documents that were being attacked. Thus they could achieve agreements as high as regular users and received trust values equivalent to the regular users.

Group attackers were the most difficult to identify. The group attackers all attacked the same set of documents and thus received agreement from each other. Moreover these attackers gave true ratings for the documents that were not the target of the attack. As a result these users had agreement far higher than the rest of the community.

Since the reputation system uses agreement and disagreement history of a user to calculate trust, malicious users with a planned attack were able to extract higher agreement from the community by giving true ratings on most of the items. In a real world scenario this is possible by rating the items are recommended by the system as relevant. In fact this problem is faced by most of the reputation systems where the user behaves honestly over a period of time to build high trust and then uses this trust for malicious activities.

Thus *absolute trust systems* might be helpful in identifying random attackers but do not seem to detect other types of attackers. Weighing the ratings of a user using the trust values returned by an *absolute trust system* does not seem to add much value to the system. We believe the problem is that the reputation system has no means of predicting the true relevance of the items except through user ratings. It would be

interesting to see if the performance of the reputation system can be improved by using it in combination with some other mechanism to predict the true relevance of the question-document pair. One such approach would be natural language processing. It would also be interesting to see if *relative trust systems* are able to perform better than the absolute trust systems in filtering out boost, sink and group attackers.

6. OUR APPROACH

The automated reputation systems that we discussed in the previous section do not seem to be of much help in detecting attackers and undesirable data in the system. A good defense strategy needs to address all the problems that we pointed out in section 4.2. Let us revisit these problems and look at what damages these attacks cause to the system rather than trying to identify the attacks and the attackers. We will try to minimize the damages these attacks cause to the system and to its users rather than trying to eliminate the effects of these attacks.

6.1 Defense against boosting a document:

Assume that an attacker has designed an attack to boost a document. An attacker could seed the system with multiple variations of the *target question* and rate the *target document* as relevant to those questions using multiple profiles to make the attack hard to detect. The damage this attack causes is that all the users who ask a question that is similar to the *target question* would be recommended the *target document*. An ideal system should be able to filter the *target question-target document* pair before any user sees it.

An acceptable defense mechanism to this problem would be to provide a means by which the first user who sees the *target question-target document* pair should be able to flag it as inappropriate. This rating should have a higher weight than the regular relevance ratings given to the system. We call this type of rating a *meta-rating*. Thus in order to boost a document to the recommendation list the user would have to give multiple relevance ratings. But to remove a question-document pair from the recommendation list a user just needs to give one *meta-rating* since the meta-ratings are given more weight than the regular ratings.

If we provide an interface such that the first user who sees the *target question-target document* pair can flag it using a *meta-rating* then the other users wouldn't have to see it.

6.2 Defense against sinking a document:

Consider an attack designed to sink a document. In the worst case, the attacker would provide multiple negative ratings for the *target question-target document* pair. In SERF we take into account negative ratings only for the recommendation list but not on the regular search results. Thus a user can only sink a document from the recommendation list but not from the search results and hence the *information source* is still available to the user. It is probable that this document would receive more relevance ratings and climb up the recommendation list again. The worst damage this type of attack can cause to the system is that it would reduce the quality of the system to its underlying search engine.

6.3 Defense against damaging the system:

The other category of attack consists of those attacks designed to damage the system as a whole. These attacks include either inappropriate questions or malicious ratings or both. Inappropriate questions comprises of high entropy questions, low entropy questions and derogatory questions. The link between a user's current question and her recommendation list is the *target question*. Thus to reduce the damages caused by the *target question* we would have to break the link between the current question and the recommendation list generated due to the *target question*. We can do this by providing a meta-rating system that could be used to rate displayed questions and their associated ratings instead of question document pairs.

Thus when an inappropriate question is displayed to a user the user would flag the inappropriate question using the meta-rating. Since the *target document* is tied to the *target question*, the *target document* would automatically be filtered out.

6.4 Defense against Time-sensitive Data:

In section 3.4, we also mentioned time sensitive data in our discussion about malicious data. The system should be able to differentiate between data which is malicious and data which is time sensitive.

The ideal system should provide an interface to rate time-sensitive question-document pairs differently from other question-document pairs. A question-document pair that is rated time sensitive can then be reviewed by a moderator.

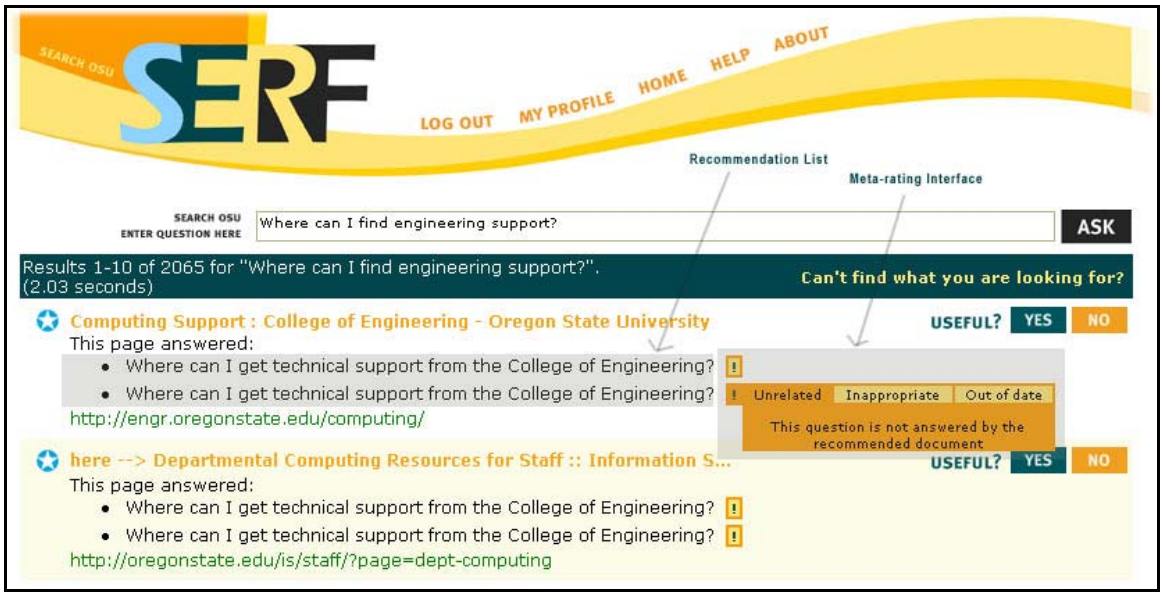

Figure 6: SERF Meta-Rating System

6.5 The Meta-rating System:

To solve these problems we designed a meta-rating system (Figure 6). The meta-rating system provides a separate rating interface to do the following:

i) Rate a question as inappropriate.
ii) Rate a rating as inappropriate.
iii) Flag the mapping between a question and a document as time sensitive.

When a question is rated inappropriate we report the question to a group of administrators who can then review the question for any anomalies. When a rating is rated inappropriate we break the link between the question and the document and remove the question form the recommendation list. When a question-document pair is flagged we have an administrator look at the question and document and take the required action.

Another advantage that meta-rating system has over the regular relevance rating system is that the ratings obtained by a meta-rating system are over the exact same question and document that were rated by someone else. But in a relevance rating system the ratings we obtain are over a document and similar questions.

7. CONCLUSION

The contributions of this paper are three fold. Firstly, we have identified what constitutes a malicious data, what incentives might provoke a user to introduce malicious data into the system and the strategies available at the user's disposal to attack the system. Secondly we simulated two automated strategies – a word co-occurrence based weighting scheme and a reputation system, to find out if it would be possible to defend against these attacks. We observed that while the new word weighting scheme does provide improvement by identifying words that co-occur, using this word weighting scheme to match questions is a difficult task. Our simulations on the reputation system show that while it is possible to identify random attacks, identifying planned attacks is much more difficult. Finally we have proposed a meta-rating system which could minimize the effects of attacks on the system.

The work in this paper can be further extended by simulating a relative trust system and testing it on real dataset. Identification of better robust similarity matching techniques could also be pursued as an extension to this paper.

ACKOWLEDGEMENTS

I am grateful to my major professor Dr. Jonathan Herlocker for the valuable guidance and constant encouragement that he provided over the past one and half years. I would like to express my sincere thanks to Dr. Timothy Budd and Dr. Zhongfeng Wang for sparing their valuable time and accepting to be on my committee. I would also like to thank Seikyung Jung for her valuable feedback during the course of this project.

REFERENCES

[1] Mao Chen and Jaswinder Pal Singh. Computing and Using Reputations for Internet Ratings. In *ACM Conference on Electronic Commerce* October 2001.

[2] Chrysanthos Dellarocas. Immunizing Online Reputation Systems Against Unfair Ratings and Discriminatory Behavior. In *ACM Conference on Electronic Commerce* October 2000.

[3] Chrysanthos Dellarocas. Analyzing the Economic Efficiency of eBay-like Online Reputation Reporting Mechanisms. In *ACM Conference on Electronic Commerce* October 2001.

[4] John O'Donovan and Barry Smyth. Trust in Recommender Systems. In A*CM International Conference on Intelligent User Interfaces* January 2005.

[5] R.Guha and Ravi Kumar. Propagation of Trust and Distrust. In *The International World Wide Web Conference*, May 2004.

[6] Seikyung Jung, Kevin Harris, Janet Webster and Jonathan Herlocker. SERF: Integrating Human Recommendations with Search. In *ACM Conference on Information and Knowledge Management* November 2004.

[7] George Karypis. Evaluation of Item-Based Top-*N* Recommendation Algorithms. Technical report, University of Minnesota, Department of Computer Science / Army HPC Research Center, Minneapolis, USA, 2000.

[8] Shyong K. Lam and Jon Riedl. Shilling Recommender Systems for Fun and Profit. In *The International World Wide Web Conference*, May 2004.

[9] Paolo Massa and Paolo Avesani. Trust Aware Collaborative Filtering for Recommender Systems. In *Springer-Verlag Berlin HeidelBerg*, 2004.

[10] Michael O'Mahony, Neil Hurley, Nicholas Kushmerick and Guenole Silverstre. Collaborative Recommendation: A Robustness Analysis. In *ACM Transactions on Internet Technology* November 2004.

[11] Gail L.Rein. Reputation Information Systems: Reference Model. In *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2005.

[12] Paul Resnick, Richard Zeckhauser, Eric Friedman and Ko Kuwabara. Reputation Systems. In *Communications of the ACM*, December 2004.

[13] Gerald Salton and Christopher Buckley. Term-Weighting Approaches in Automatic Text Retrieval.

[14] Ricardo Baeza-Yates and Berther Ribeiro-Neto. Modern Information Retrieval.

[15] Bin Yu and Muninar P.Singh. Detecting Deception in Reputation Management. In *The International Conference on Autonomous Agents and Multiagent Systems* July 2003.

APPENDIX A: WORD WEIGHTING BASED ON CO-OCCURRENCE

The co-occurrence based word weighting scheme was simulated on the dataset obtained from SERF. Table 4 shows the *dependence factor* of the words in three arbitrarily picked questions:

- Where can I find information about long term research?
- Find Willamette Basin information?
- Where is information services website?

Table 4 shows only those words that do not fall in the category stop-words. The following things can be deduced from the table:

- The word *information* has lower *dependence factor* on other words
- All the words have high dependence factor on the word *information.*

The occurrence of the word *information* in the target question does not increase the weight of any words in the candidate question as much as the occurrence of one of the other words in the target question increases the weight of the word *information* in the candidate question.

Target Word	Candidate Word	Dependence Factor
basin	basin	7.36572
willamette	willamette	7.36572
basin	willamette	3.68286
willamette	basin	3.68286
research	research	1.68941
website	website	1.68941
services	services	1.02233
basin	information	0.608382
willamette	information	0.608382
research	long	0.484177
research	term	0.284161
information	information	0.195976
research	information	0.145682
website	services	0.06571
services	website	0.041069
information	research	0.036421
website	information	0.029136
services	information	0.028332
information	long	0.020876
information	basin	0.01521
information	willamette	0.01521
information	term	0.012252
information	services	0.011333
information	website	0.007284

Table 4: Correlation Index

APPENDIX B: CLASS DIAGRAM FOR REPUTATION SYSTEM

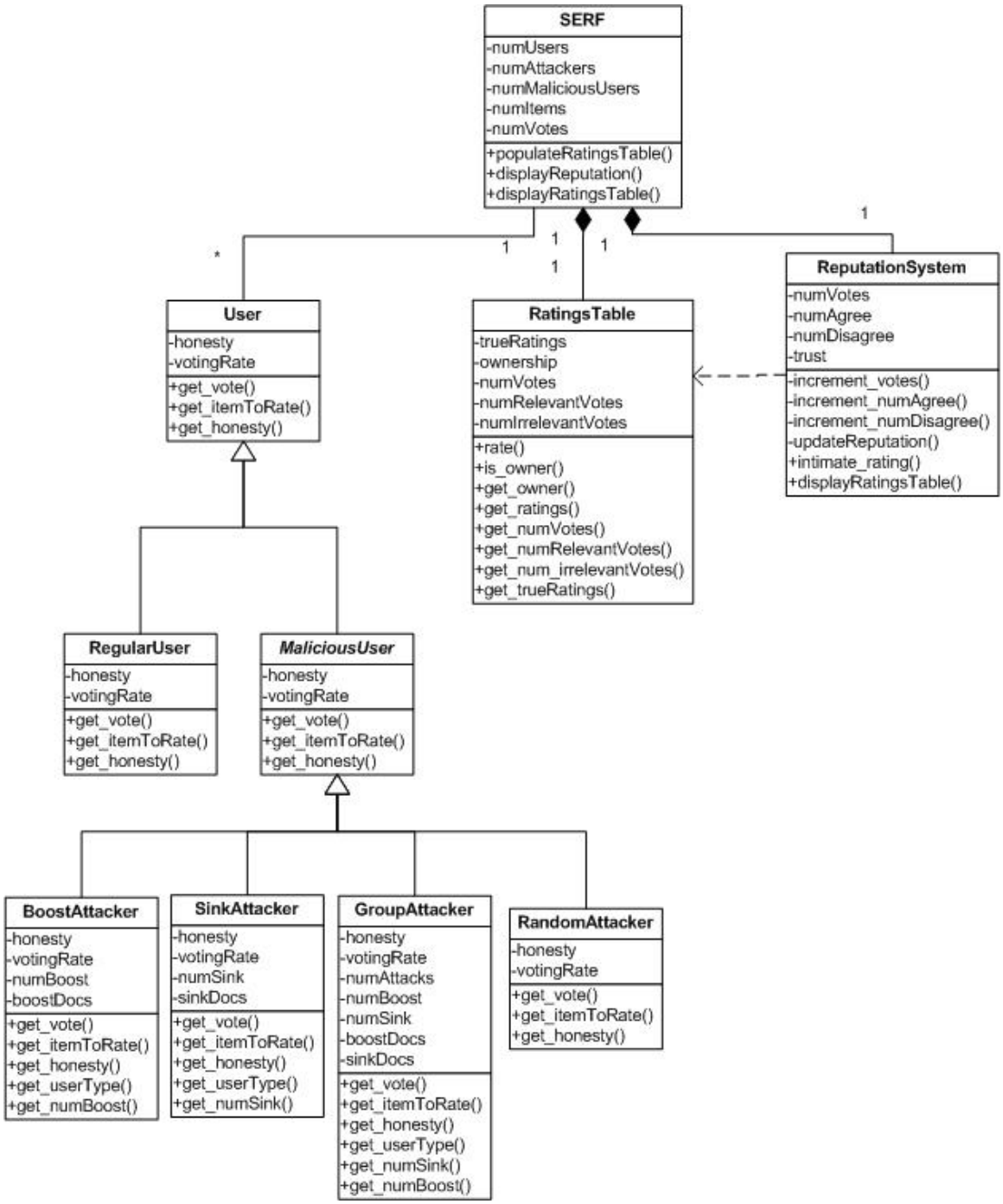

Figure 6: Class diagram for the implementation of Reputation System