A STUDY OF ATTACKS ON COLLABORATIVE SPAM FILTER

by

XIAOHU FAN

(Under the Direction of Kang Li)

ABSTRACT

Collaborative filtering is a widely used technique to make classifications by using distributed feedback from all users. Recently, collaborative filtering has been proposed and used to detect spam messages. Compared to individual spam filters, collaborative spam filtering has the advantage of potentially accessing large datasets and the effect of crowd sourcing. However, the benefit of the collaboration also comes along with vulnerabilities of false collaborative information. In this thesis, I studied the effect of various possible attacks on a collaborative spam filtering system. We built a platform to simulate a collaborative spam filtering system, and use this system to answer questions such as what attacks cause the most damage and the cost of launching such attacks. The result of this study suggests that collaborative spam filtering is vulnerable to attacks from false collaborators and we identified the most damaging strategy that should be addressed first by the defense of collaborative spam filtering.

INDEX WORDS: Collaborative spam filtering, Bogofilter, Malicious attack

A STUDY OF ATTACKS ON COLLABORATIVE FILTER

by

XIAOHU FAN

MS., Central South University, China, 2007

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial

Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2010

A STUDY OF ATTACKS ON COLLABORATIVE FILTER

by

XIAOHU FAN

DEDICATION

To my mother and my father; for all the love they showered on me, and for supporting me in the hardest of times.

ACKNOWLEDGEMENTS

I would like to thank Dr. Kang Li, Dr. Lakshmish Ramaswamy, and Dr. Ismailcem Budak Arpinar for their willingness to participate in my defense committee. Many thanks to Mr. Jason Lang for his kind suggestions on improving the thesis’s writing. I would also like to thank the Department of Computer Science at The University of Georgia, for providing me the opportunity to pursue the Master's program and the assistantship they granted me during the course of the program.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Spam Problem Description

As more and more people use email as their major communication tool, email spam, i.e. unsolicited commercial email, has become a severer problem. A spammer tends to reach as many email boxes as they can and filled them with ads or fraud information. Think about the unbelievable low cost of sending email, even if only 1 in 100,000 users followed the advertisement and purchased a product, the spammer may still make a profit. Compared to printed material or ads on TV, the cost of spamming is unbelievable low, typically, spammers charge less than a hundredth of a cent per recipient. It is foreseeable that combating against spammers is a long-term problem for security researchers. As anti-spam techs evolve, to survive and profit, spammers not only tends to adapt more sophisticated measures, but also try to become more familiar about the internal working mechanisms of anti-spam systems. In this paper, we are going to record and analyze a simulated battle between spammer and the collaborative spam filter.

1.2 Description of Spam Filter and Collaborative Spam Filter

Email filtering is the processing of email in a MTA (Mail transfer agent) to classify it according to some specified criteria. The filtering of incoming messages is often carried out automatically, but it can also apply to the semi–automatic process--where user gives feedback to help the filter's training.

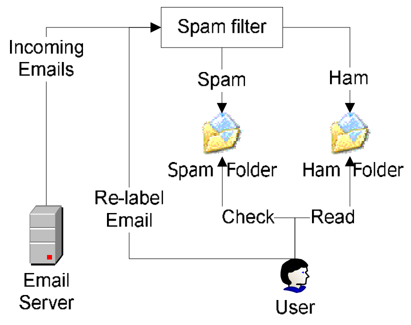


Figure 1.1 Overview of email filter

Figure 1.1 shows the work flow of a typical email filter. Incoming email messages are received from the mail server, and then handled by the filter at the MTA. The filter classifier them into either ham or spam files (quarantine). Only ham emails are presented to the user, while spam messages will normally be blocked. Many filters allow users to report the misfiled spam to the filter, thus improve the filter's performance and effectiveness.

Given the assumption that spam/ham classification criteria between different users are similar, a Collaborative email filter could not only allow feedback to the filter, and enable the sharing of feedback information within the system.

While correct user feedback has been proven to benefit the filter in boosting its accuracy, malicious user's feedback or an unintended mistake can greatly harm a filter's performance [1]. The situation becomes even worse when similar things happened to a collaborative filter — the unique feature of sharing renders a collaborative filter more vulnerable to a malicious user/spammer's attack. i.e., a wrongfully classified email will not only affect the filter itself, it will also pollute other filters within the system by propagating the information.

1.3 Attacking Collaborative Filters

Collaborative filtering (CF) technology has been proved to be a viable and effective solution, thus being widely used. However, for malicious users, or a group who is interested in promoting their product, there is an incentive in poisoning the collaborative filter to their advantage [2]. Such attacks have been referred to as shilling attacks [3].

Shilling attacks can be classified into two basic categories[4]:malicious actions that which will raise a particular item's rate higher are called push attacks, while those aimed at downgrading the popularity/rate of an item are called nuke attacks. In our research, we will address the effects of both push and nuke attacks on a collaborative system.

There are certain forces driving behind people who try to manipulate the collaborative system. Most are likely because of profit. i.e. since CF are widely deployed in commercial systems, the producers/manufacturers may want to promote their products in an "fast and efficient way". The producers/manufacturers can either do a push up attack to raise their own item's rating, or nuke down a competitor's item's rating. Another possible intention is to bring down the target CF as a whole; this can be carried out either by malicious users who want users to abandon the CF system or by other CF system builders who want users to switch to their CF product.

The real attacks, however, are always carried out by a group of hired people, A.K.A. shills. The shils either manually or use agent/bot to provide false options to mislead the real users. Shills pose a serious threat to users and the CF system, as we are going to demonstrate in our experiment section.

1.4 Thesis Contributions

The existing research [2,3,4] about shilling attacks on collaborative filters has demonstrated that shilling could harm a recommender systems, but we believe there are still several important questions that need to be answered.

Our first question is: how effective is shilling on CF Systems? The second is how does the pre knowledge of the CF system affect the attack effect. The third is how does the CF system's scale affect the attack effectness

To answer these questions, first, we proposed and built a scalable, efficient and effective platform that simulated various types of attacks against a collaborative system; second, we discussed several types of attack strategies that described and categorized a wide variety of shill attacks, and performed a series of experiments to study their effectiveness. In our experiments we fully evaluated the shilling attack with different sets of parameters, i.e.: the number of attackers, the strategy of the attack, etc. Finally, we present the results of our experiments and evaluate them against the theoretic prediction. We conclude that the unique vulnerability of collaborative filters should be addressed, and our research can not only help improve our system but also help other collaborative filtering applications.

CHAPTER 2

RELATED WORK

2.1 Related Work on Collaborative Filtering System

Collaborative filtering (CF) is the process of filtering information among different systems, data sources, etc. The technique always involves collaboration. Our collaborative system is used for spam filtering. However, the collaborative filtering method can be applied to many other different kinds of applications, like: educational applications [5], communication applications [6], financial data applications [7], musical data applications [8], movie data applications [9], web applications [10], etc.

2.1.1 Memory-based VS Model-based

Most existing collaborative filtering systems are classified into 2 categories: memory-based collaborative filtering [11,12,13,14] , or model-based collaborative filtering [15,16,17,18]. The memory-based approaches do the recommendation is by computing the similarity between users, i.e. comparing their past ratings on similar items. The memory-based CF group users with similar taste and assume they are likely to make similar ratings in other items. Thus the CF system can uses user's existing rating data to predicate other group member's rating on certain items.

The memory-based approach has achieved great success. Many commercial websites like Amazon [19] adopted this technique due to its highly effective and easy-to-implement nature. However, researches show the memory-based approach suffers from two fundamental problems[18]: 1) data sparsity, the performance degrades as the user data sparse, and 2) scalability: when the users and the large number of products/data grows tremendously [14], the executive cost and excessive storage become unbearable.

Recent researches on the memory-based approach have been focused on how to wisely extract a limited number of data to represent the whole dataset, i.e. instance reduction techniques. For example: user profile analyzing [11], dynamic Weighting and Selection [20], probabilistic selection [14], etc. Although the proposed approaches have alleviated the problem, the nature of memory base approach has limited its use.

On the other hand, model-based approaches use machine-learning or data mining techniques to develop a model for the filtering. Researches prove it is less vulnerable to data sparsity or scalability problems [15]. The model-based approach use various techniques brought from the machine learning realm, like latent semantic models [21], bayesian active learning [22], and techniques from data mining realm like association rule mining [23], nearest neighbors [24], etc. In [25], the author proposed a combined model from latent factor model and neighborhood model. The trained model can reflects the pattern of the training dataset. Once the training is over, the trained model is fast in making predictions on unseen instances.

The so-called hybrid approach has also gain growing attentions. Representative hybrid approaches involve collaborative filtering techniques and makes use of the content-based filtering techniques [26]. The difference between collaborative filtering systems and content-based filtering is that the collaborative filtering system recommends items by grouping users with similar taste and uses their opinions to make recommendation for other users. The content-based recommender system recommends items based on the items' own information. Both systems suffer from scalability and data sparsity problem. Hybrid recommender systems combine individual systems to avoid certain aforementioned limitations of the memory-based system and model based systems. for example, [27] combines Naive Bayes Classifier and Collaborative Filtering, Content-Boosted Collaborative Filtering [28].

2.1.2 Representative Collaborative Systems:

1) Generating personalized recommendations.

Commercial websites like Amazon, Youtube use personalized recommendations [29,30] very heavily. Collaborative filtering for personalized recommendations generate recommendations based on users' rating on items.

For example, if we have a list of movies which a group of users prefer to watch, a collaborative filtering system for movies tastes can make predictions about which movie a user would like[31].

2) Collaborative Search Engine

Collaborative Search Engines [32,33] (CSE) allow users share information resources collaboratively using knowledge tags, and allow search experts help new users search what they are looking for.

2.2 Related Work on Spam Filter

Machine learning researchers view email filtering as a process of text classification. Machine learning techniques, like bayes classifier, have been proofed to be very effective when against spam email. A supervised email filter can achieve extremely high accuracy better than 0.9999 [34].

Paper [35] suggests using support vector machines (SVM) to distinguish spam emails from ham emails. They also compared their SVM approach with other approaches like boosting of C4.5 trees, RIPPER etc. They conclude that SVM's is slightly better in distinguishing the two types of misclassification. Paper [36] presents acceleration techniques that speed up bayesian filter based on approximate techniques like hash-based lookup and lossy encoding. Paper [37] presents a spam filter using statistical data compression models. The model transforms the spam detection problem into a probabilistic text classifying problem based on character-level or binary sequences.

While statistic based spam filters can achieve impressive accuracy rates, they remain prone to false positives. Collaborative spam filters can achieve better filtering through the collaboration of users, i.e. when a user classifies an email as spam, the knowledge can be shared and added to the collective knowledge base. Paper [38] presents a collaborative spam filter which preserves user's privacy by sharing the fingerprint rather than content. Paper [39] focused their research on how to customize a collaborative filter according to user's unique definition of spam.

2.3 Related Work on Attacking Spam Filter

While researchers try to upgrade their filter to stop spammer, spammers do not give up easily--they also study the filters' internal mechanism to avoid their spams being blocked.

To our best knowledge, most existing researches on attacking spam filter focused on attacks on single filter. Paper [40] presents an evaluation against several filters from TREC [1, 41, 42] on dataset with synthetic noise. The research confirms that noisy feedback can greatly impact a filter’s performance. Paper [43] describes and evaluates the effectiveness of active and passive good word attacks against two types of statistical spam filters: the naive bayes filter and the maximum entropy filter. The difference between these researches and us is that we focused our research on the impact on collaborative filters, and we also considered “bad words attack”. Paper [44] is another attacking strategy paper on statistical based filter.

Our research is among the few which focused on the impact of attacks on collaborative filter. Paper [45] presents a collaborative spam filter that learns global tendencies across all users thus it can absorb the influence of emails labels that are very differently from the general public. A similar approach is given in [46]. They assume that the feedbacks given by malicious users are always very different from the other users. They give a supervised machine learning scenario where labels (feedback) are provided by a heterogeneous set of teachers. They implemented a SVM framework to reduce the weight of trainer who has too much impact on the result, thus avoided the potential threat of malicious users.

CHAPTER 3

BOGOFILTER FILTER AND SPAMMER’S APPROACH

Before describing the possible approaches that spammers could use to poison/penetrate a spam filter, we believe it’s necessary to take a close look at how spam filter works, especially how Bogofilter works.

Bogofilter [47] is an open source mail filter written in c that classifies mails as spam or ham (non-spam) by a statistical analysis on both the email's header and body. The program can learn from users’ feedback.

Bogofilter implements a very unique tech. The main idea is: it assumes every email is a bag of random words, which contains 50% spam and 50% ham words. Then it calculates the possibility of how likely it is a spam email based on how many spam words are shown in this email. Note the assumption is not build to stand, as we know each real email is not a bag of random words. Usually the real ham email contains more ham words and a spam email contains more spam words. If Bogofilter has enough knowledge about the ham words and spam words, it can correctly classify the emails.

3.1 The Workflow

Figure 3.1 shows the workflow of training bogofilter. At the beginning of the training process, the filter accepts a set of emails. The emails will then get parsed into tokens. The count of tokens will be added to bogofilter’s database.

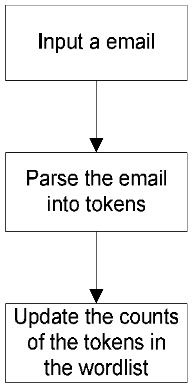

Input a email
Parse the email into tokens
Update the counts of the tokens in the wordlist

Figure 3.1 Training process

For testing, the figure 3.2 shows how the bogofilter evaluates an email: at first the email was parsed into tokens, then bogofilter looked up these tokens in it's database, if a token has been seen by bogofilter before, bogofilter will assign a value of how possible this token is a spam token. All tokens' possibilities are then merged into a final value, which shows how much does bogofilter think the email is a spam/ham:

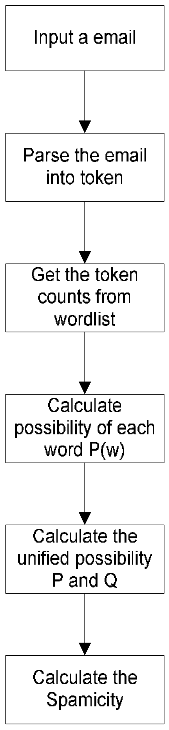

Input a email
Parse the email into token
Get the token counts from wordlist
Calculate possibility of each word P(w)
Calculate the unified possibility P and Q
Calculate the Spamicity

Figure 3.2 Classifying process

3.2 How Bogofilter Calculate the Spamicity of Emails

According to [48], the bogofilter calculates the spamicity based on these steps:

PW: Possibility a document contains the token is a spam

b (w) = (the number of spam containing the word w) / (the total number of spam).

g (w) = (the number of ham containing the word w) / (the number of ham).

$$p(w) = b(w) / (b(w) + g(w))$$

FW: Take the consideration that a (rare) token can impact the result.

$$f(w) = \frac{(s * x) - (n * p(w))}{s - n}$$

Here s is the strength of how much we want to weigh our background information. The default value is 0.0178. x is our assumed probability, based on our general background information, that a word we don't have any other experience of will first appear in a spam. Default is 0.52. n is the number of e-mails we have received that contain word w.

Combined possibility H as for Ham: C-1 is a inverse Chi Square function, H is linear to f (w) and more sensitive when f(w) is close to 0:

$$H = C^{-1}(\ 2 \ln \prod_{w} f(w), 2n)$$

At last, we have the combined possibility I for Spamicity:

$$I - \frac{1 + H - S}{2}$$

3.3 3 Types of Attack Strategies

Since spammers know how the filter works, they can design the attack against the specified filter. By poisoning/penetrating a filter, the spammer can influence people to purchase a product. Thinking about the unbelievable low cost of sending email, even if only 1 in 100,000 users followed the advertisement and purchased a product, the spammer can still make a profit.

We categorized 3 types of possible attack strategies that can be used by a spammer:

1) All Spam: spammer in the collaborative filter system marks all emails he receives as spam emails and propagates the classifications to other clients in the system.

2) All Ham: spammer in the collaborative filter system marks all emails he receives as ham emails and propagates the classifications to other clients in this system.

3) Flip: spammer in the collaborative filter system marks all ham emails he receives as spam emails and all spam emails as ham emails. Then propagate the classifications to other clients in this system.

CHAPTER 4

THE DESIGN OF OUR EXPERIMENT PLATFORM

How to simulate the whole scenario effectively and efficiently is a big challenge. In this section, we will describe the design and implementation of our experiment platform. The first part describes our system structure, followed by another part that explains the python script which simulates user's actions/ interactions.

4.1 System Overview

The system is designed to provide an experiment platform for testing the vulnerability of a collaborative spam filter. It can help improve the spam filtering and other applications that use the feedback mechanism.

What does the system look like? The system is composed of multiple instance of bogofilter. Each instance represents a filter at a mail transfer client.

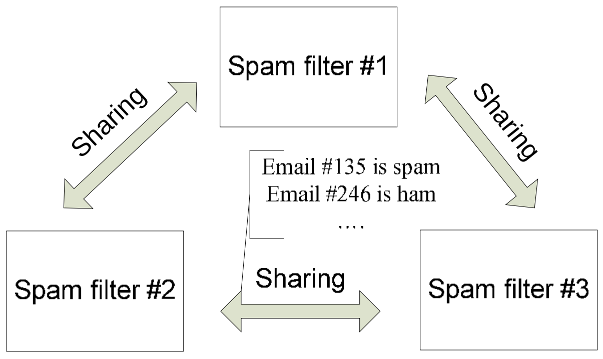


Figure 4.1 Overview of a collaborative filter

As shown in Figure 2, after a filter classified incoming emails, it will propagate this info to other filters in the collaborative system. Nowadays most filters support feedback and retraining. Bogofilter, can retrain the classified ham/spam based on users' feedback.

A straight forward way to build such platform needs several machines, real network and email servers. However, there are several drawbacks of this approach:

1) Extra cost

It will require multiple machines and extra setup work.

2) Unnecessary noise.

The network can be unstable; email servers can fail in sending/receiving emails. Though they happen in real world scenarios, they have little to do with the essentials of our problem.

3) Not scalable

To scale the research, if the number goes up, it will be very difficult to carry out the experiment.

Due to these considerations, we designed our platform in a different way. We believe the platform we are looking for should have the following properties:

1) Easy to set up and scale

2) Only focused on the essential of our problem;

3) Sending/receiving email should be easy and fast;

To get a good simulation and performance, we extract the model and implement our platform in only 1 machine. We will describe it in later part of this section. Before that, we first describe the user representation and email sending/receiving.

4.2 User Representation

The first task to carry out the simulation is to represent users. We believe the users should be represented effectively so that:

1) Emails can bouncing between users easily;

2) Sending/receiving emails should be very efficient;

3) Simulations can be done with only a few machines.

To simulate users in the real world, we made the following assumptions:1) Users are simply classified as good user/malicious user; 2) The good users always provide the proper feedback, i.e. they can either follow bogofilter's suggestion or give the right classification of an email, but never twist the result; while the malicious users classify emails according to their attacking strategy.

These assumptions may look strong at the first place. In real world, good user may misfile a ham to spam, or fail to detect a wrongly classified spam. However, we assume that in this system good user never made mistake. This assumption enables us to conduct our experiment, and inspires the design of our experimental platform. On the other hand, we assume a malicious user always use a specific approach to attack the filter., and the approach doesn't always have to be giving wrong feedback. For example, if a malicious user chooses the all ham strategy and the testing email is compound of half spam and half ham, and then half of the malicious user's feedback is actually correct.

4.3 Email receiving and sending

In our system, we use a Linux file to represent an email. The file has to follow the "mbox" format, i.e. "The beginning of each message is indicated by a line whose first five characters consist of "From" followed by a space (the so-called "From_ line" or "'From ' line") and the return path e-mail address. A blank line is appended to the end of each message."[49]. Multiple emails which follow the mbox format can appear in one Linux text file. Our system can parse it into separated emails.

The process of delivering an email is simulated by copying an email file from one user's folder to another user's folder. Each user has his/her own bogofilter instance. Each instance has a Berkeley database file to store all ham/spam words count information. Our experiment platform can then simulate the whole process by using Linux command to copying file around and calling bogofilter to process the emails. One of the benefits of this design is that we ignored all issues like network, SMTP protocol, etc. thus the whole process is greatly simplified, and can happen in a local machine. On the other hand, our test against bogofilter has no difference with running the experiment in a real world scenario.

4.4 Platform Command

In this section we are going to introduce the commands of this platform, the purpose of these commands is to help to organize the simulation so we can run a complex experiment much easier. Generally speaking, each command has a sequence of parameters. The parameters will later be used to specify a Linux command.

For each experiment, we provide a configure file. A typical configure file will contain several lines of commands, each with its parameter list. There are mainly 5 type of commands:

1) Distribution

The distribution command starts with –D, and is always followed by a filename and numbers. Its purpose is to simulate the email server sending a email to a user. A distribution command execution contains two steps: step1) Extract certain number of emails from a big email dataset, and step2) Copy the extracted emails into the user’s folder, there could be only 1 receiver, or a list of receivers. The system can use this command to distribute new emails and propagate the sharing info between users. We will discuss this command later with register command.

2) Training

This command will call bogofilter to train one or multiple users’ bogofilter instance with the emails they received. This command will generally come after the distribution command, as the distribution step will copy the emails to a default folder of a receiver. Unless specified with an external path, the training command will train the bogofilter on all the email files in the user’s default folder.

3) Register

This command is a combination of the distribution command and the training command. It is used to propagate the shared information, i.e. after a user has received and relabeled some emails,, the register command will distribute these emails to other users and request their filter to train on them based on the user’s label.

The register command is always followed by some special parameters to indicate a user's strategy, i.e. if the user is a good user, he will always apply the "normal" strategy. On the other hand, a malicious user can choose a strategy from All Spam/All Ham/Flip/Random. One thing needs to be noticed is that, in our system, when propagating user's feedback, the system doesn't choose the receivers, i.e. the receiver can be a malicious user or a good user.

4) Test

The purpose of test command is to evaluate the effect of the malicious attack on a bogofilter instance. A typical experiment configuration is like the following: 1) Initialize several users. 2) Distribute certain number of training emails to each user. 3) Each user's bogofilter train on these emails. 4) Choose certain percentage users as malicious users; the remaining users are good. 5) Distribute some extra emails to the malicious users., The malicious users use bogofilter to filter them and twisted the results based on a strategy. propagate the twisted results to other users. 6) Since in step 5 the results can be twisted, the bogofilter instance which trained on these results is polluted. The test command will first use a polluted bogofilter instance to filter a set of pre-classified emails, and then run a clean bogofilter instance to filter a same set of emails, after that it generate a file containing the results. We can then evaluate the damage. This command always follows the register command.

There are some other utility commands: command "initialization" will create a user's folder structure. Command "clean" will delete all results and temporary files. The picture generating command will visualize the result generated by the test command. The test result of the bogofilter contains a list of numbers between 0 and 1, which indicates how likely the email is a spam, i.e. when it equals 1, the filter think the possibility is 100%. To visualize the result, we use python's gnuplot package to draw the graph from the extracted result data.

4.5 System Structure

Python is a highly modularized language. Every module written in python can contain several functions. Our system contains the following 6 modules: 1) Main Module; 2) User Module; 3) Email Module; 4) Bogofilter Module; 5) Gnuplot Module; 6) Configure Module.

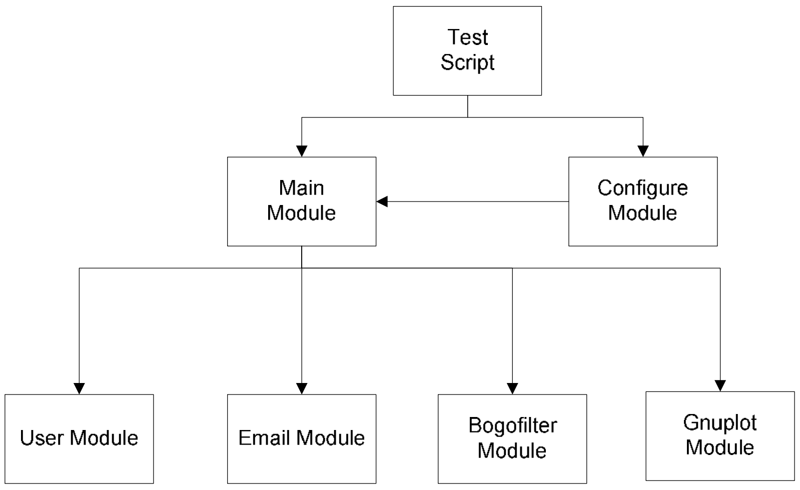


Figure 4.2 System structure

The function of each module is quite self-explained, the main module will first take several parameters from the user, then feed the parameters to the configure module.

The purpose of the configure module is to handle very complex experiment design, for example, if we want to run experiments on different sets of parameters like the total users number, the percentage of the malicious users, or a different workflow etc. Using hard coded configure file is time-consuming and error-prone. Instead, we can feed the range of every changeable parameters into the configure module. The configure module contains several predefined templates. It takes the parameters and generates a configure file. It then feed the configure file back to the main module.

The main process of the main module is a switch statement. The process parses the command lines and then calls corresponding functions in different module. To provide more flexibility, we introduced another level of abstraction--the test script. Every experiment has its own unique test script. The test script will initialize the configure file for us and some special preparation works like extracting partial emails for training dataset. We will discuss the experiment workflow in chapter 5.

4.6 3 Types of Attack Approaches

Since spammers know how the filter works, they can design attack specified against the filter. We categorized 3 types of possible attack approaches that can be used by a spammer:

1) All Spam: where a spammer in the collaborative filter system marks all email he received as spam and propagate this classification to other filters.

2) All Ham: where a spammer marks all emails he received as ham and propagates this classification to other filters.

3) Flip: where a spammer in the collaborative filter system marks all ham he receives as spam and all spam as ham. Then propagate this classification to other filters in this system.

CHAPTER 5

EXPERIMENT RESULTS

This chapter describes the datasets used in this study and our experiment result. Our findings indicate that, in a collaborative spam filter system, not only malicious users can cause damages, good users, in this case, could also unintentionally help the malicious user to spread the false information. The first section describes the dataset followed by the section describing the obtained results. Finally we explain the results in detail.

5.1 Dataset and Metrics

In order to study the effect of malicious attacks on the collaborative spam filter system, we believe we should use real world datasets. Most recent studies on spam filter use the Enron dataset [50]. This dataset corpus contains about 0.5M messages, which bounced between about 150 users, mostly of whom are senior management of Enron. All their emails are organized into folders.

The explanations of results obtained by conducting experiments on the mentioned dataset are in subsequent sections. In case we mention t emails, we always mean t/2 ham, t/2 spam unless explicitly noted. To compare the effects of malicious attacks, for good users, we name their strategy as normal. The normal strategy gives feedback of the bogofilter’s classification of a email. We also introduced a strategy called random, which means a spammer randomly classify a email he receives, and propagate this classification to other filters in this system.

In our experiments, we mainly use spamicity as the metric to evaluate the impact of different malicious strategies. In experiment 3, we also use the CDF graph of spamicity to help us evaluate the damage.

5.2 Experiment 1 Comparison of Different Malicious Strategies

To analyze the possible damage caused by a "bad guy", we must first think and act like a "bad guy". Our first experiment is thus designed to find out which strategy does the most damage to a collaborative filter system.

5.2.1 Experiment Set 1 Workflow

In chapter 4, we have briefly introduced the 3 malicious strategies that could be used on a collaborative spam filter system. We've designed the following experiment:

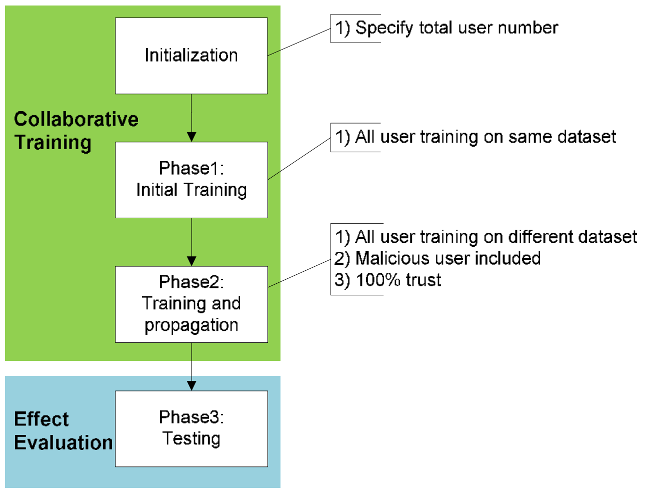

Initialization
1) Specify total user number
Collaborative Training
Phase1: Initial Training
1) All user training on same dataset
Phase2: Training and propagation
1) All user training on different dataset
2) Malicious user included
3) 100% trust
Effect Evaluation
Phase3: Testing

Figure 5.1 Workflow of experiment 1

The workflow of experiment 1 is: we first create n users using the initialization command, where n is the number of total users. Then in phase 1, we do an initial training for all users' filter on the same pre-classified dataset of size t_1. This step will initialize every bogofilter instance with same initial knowledge. After that, in phase2, we choose n_1

users as malicious users, and all remaining users as good users. Both good users and malicious users will receive different t_2 pre-classified emails. The difference of phase2 with phase1 is that the filters will not only train on these emails, but also propagate their training results to every other filter in the system. Here is when the malicious user can apply their malicious strategy to poison the collaborative filter. For example, if a malicious user adapts the all spam strategy, he will mark all emails he received as spam and ask other users to training on these emails as spam. In the last step, every user will test their bogofilter on an email set of size t_3.

In this experiment, the total number of users is $n = 100$. We choose the number of malicious user n_1 from the following numbers: {1, 6, 12, 25, 50, 75, 100}. Obviously when $n_1 = 100$, we are saying all users in the system are malicious users. The training size t_1 is chosen from the following numbers: {5000 (2500ham/2500spam), 4000 (2000ham/2000spam), 3000 (1500 ham/1500spam), 2000 (1000ham/1000spam), 1000(500ham/500spam), 2(1ham/1spam)}. The number of emails used in phase 2 t_2 varies from {50(25 ham/25spam), 40(20 ham/20 spam), 30 (15 ham/15 spam), 20(10 ham/10 spam), 10(5 ham/5 spam)}, the number of emails we use for testing t_3 is always 10000(5000ham/5000spam).

5.2.2 Experiment 1.1

Experiment 1 is designed to tell us which strategy is the most effective one, which, at the same time, is more likely to be used by a malicious user. In this experiment we mainly focused on the impact of the different strategies. We expect that the all spam strategy should have the most impact on the collaborative filter since it tries to put all words in phase 2 into the spam list. The result confirmed our guess. Here we have n = 100, n_1 = 50, t_1 =5000(2500ham/2500spam), t_2 = 50(25ham/25spam), t_3=10000(5000spam/5000ham)

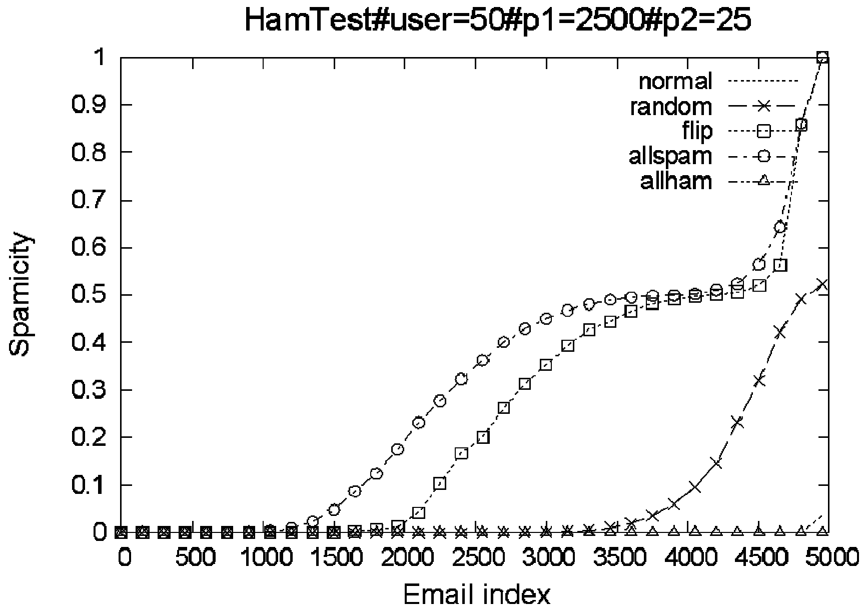


Figure 5.2 Result on ham test

Figure 5.2 shows the impact of different strategies on ham email filtering, the right most line is the normal strategy, which mean for the 5000 ham test email, most of them can pass bogofilter, i.e. the filter has a very low false negative. The left most line is the all spam strategy, which shows after the malicious user poisoned the filter, about 10% ham emails are now wrongly classified as spam, also about 85% ham emails have increased spamicity. Only 5% ham emails' spamicity remain unaffected. The line on the second left most is the flip strategy and the one on the middle is the random strategy. The all ham strategy, in this case, has impacted the fewest on the ham testing set.

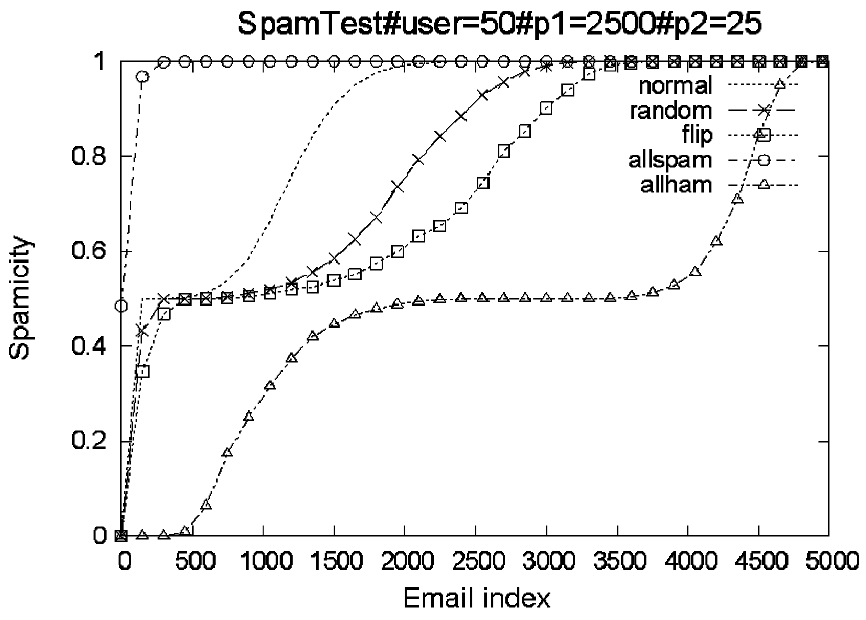


Figure 5.3 Test result on spam test

Figure 5.3 shows the same filter test on spam dataset. The left most line represents the all ham strategy, this makes sense because more tokens are added to the ham list, so the whole dataset has lower spamicity. The line in the middle is the normal strategy, which shows in normal case, about 80% of the spam emails can be correctly detected, while about 20% are unsure. The right most line is the all ham strategy. It has the most impact on the spam test. About 20% spam emails can pass the filter because of the attack.

5.2.3 Why Allspam Strategy Can Outperform Flip Strategy.

One interesting thing is in our experiment allspam strategy outperformed flip strategy. This can be validated with equations presented in chapter 3. An important equation is:

$$p(w) = b(w) / (b(w) + g(w))$$

where p(w) is the possibility a document contains the token is a spam, and b (w) = (the number of spam containing the word w) / (the total number of spam).

g (w) = (the number of ham containing the word w) / (the number of ham).

Since the bigger p(w) value is, the bigger the spamicity of an email is, we will just discuss how allspam strategy and flip strategy can influence the p(w) value. Suppose we have 2 spam emails: S1,S2 and 2 ham emails:H1,H2, To simplify the discussion we assume each email only contains a single word, “ws1”,”ws2”,”wh1”,”wh2” respectively. If we train the filter correctly, the wordlist should look like the following:

Table 5.1 Word count in bogofilter, after training

	ws1	ws2	wh1	wh2
Numble of Spam Email Contain this word	1	1	0	0
Numble of Ham email contain this word	0	0	1	1

Now suppose if we adopted flip strategy and use the same emails to attack the filter, the bogofilter will now looks like:

Table 5.2 Word count in bogofilter, after flip attack

	ws1	ws2	wh1	wh2
Number of Spam Email Contain this word	1	1	1	1
Number of Ham email contain this word	1	1	1	1

We can then calculate the p(w) value of each word as: p(ws1) = (1/4)/(1/4+1/4) = 1/2, and p(ws2) = 1/2, p(wh1) = 1/2, p(wh2) = 1/2

If we adapted the allspam strategy, we have the wordlist looks like:

Table 5.3 Word Count in bogofilter, after allspam attack

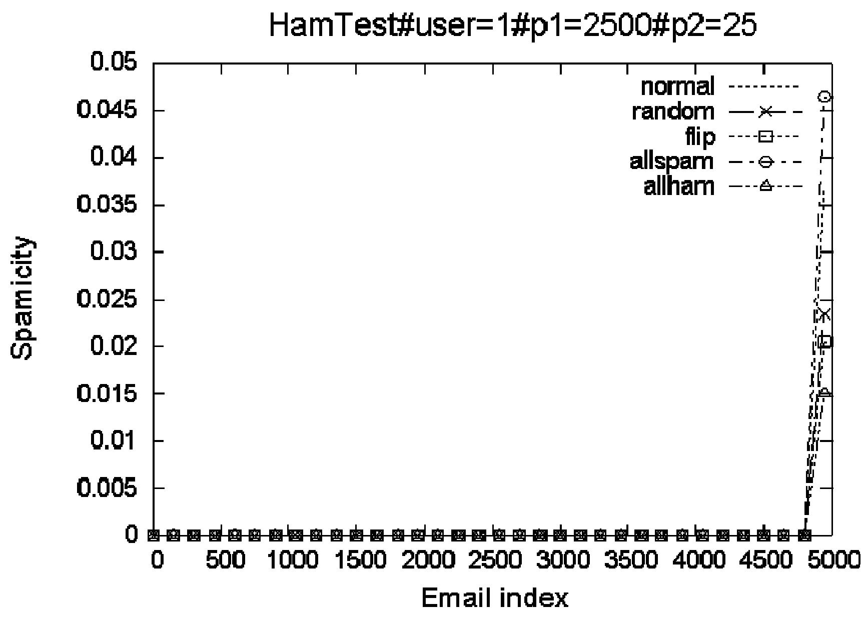
	ws1	ws2	wh1	wh2
Number of Spam Email Contain this word	2	2	1	1
Number of Ham email contain this word	0	0	1	1

And now p (ws1) = (2/6)/ (2/6+0) = 1, p (ws2) = 1, p (wh1) = (1/6)/(1/6+1/2) = 1/4, p(wh2) = 1/4. Now we see under flip attack, the sum of all pw values is 0.5 + 0.5 + 0.5 +0.5 = 2, and under all spam attack, the sum of all pw values is 1 + 1 + 0.25 + 0.25 = 2.5 We see the all spam attack add more pw value to this settings. It’s reasonable that all spam strategy can outperform flip strategy in our experiment.

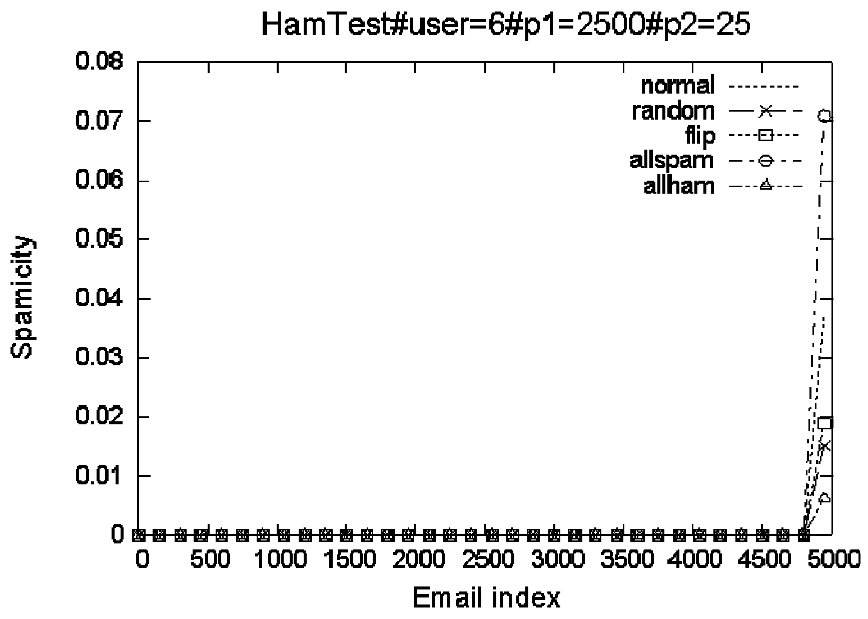
5.2.3 Experiment 1.2

Now we know the allspam strategy is effective in poisoning ham email filtering and allham strategy is effective in poisoning spam filtering. The next question is how parameters like percentage of malicious users will affect the experiment result. We compare the different results when the number malicious users varies from {1, 6, 12, 25, 50, 100}. It seems that the more malicious users there are, the more impact they could create. We verified our intuition by the following experiment.

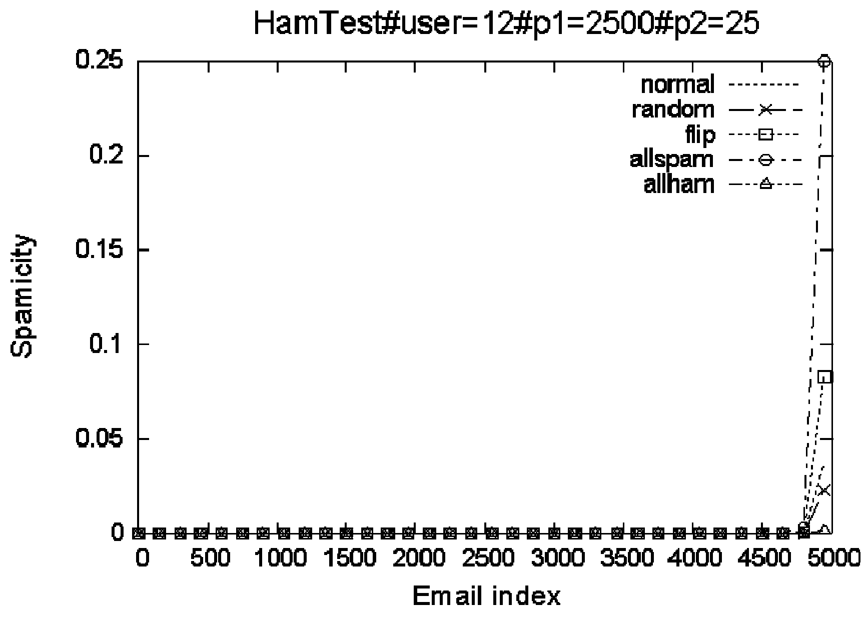
The experiment results are shown below. First we have a group of test results on a pre- classified ham email set.



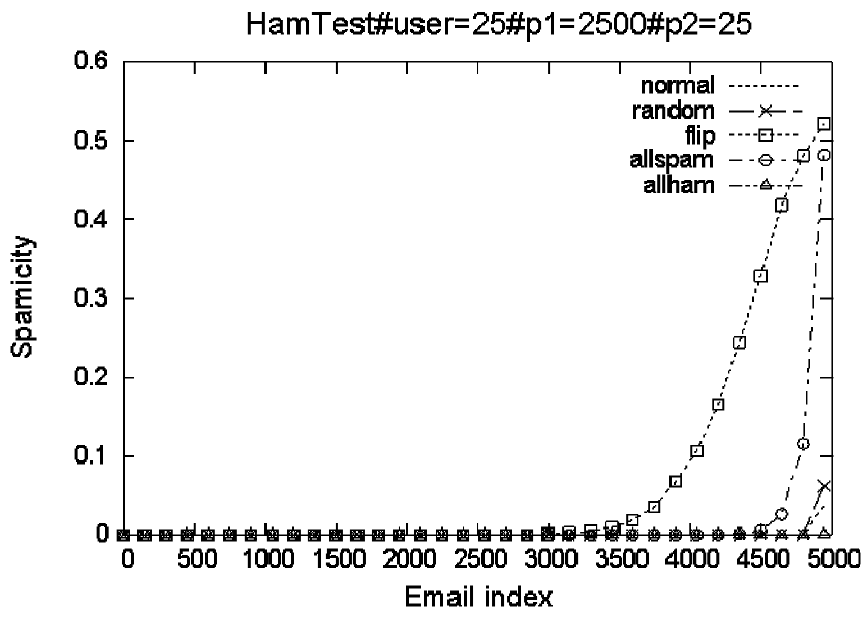
1 malicious user, ham



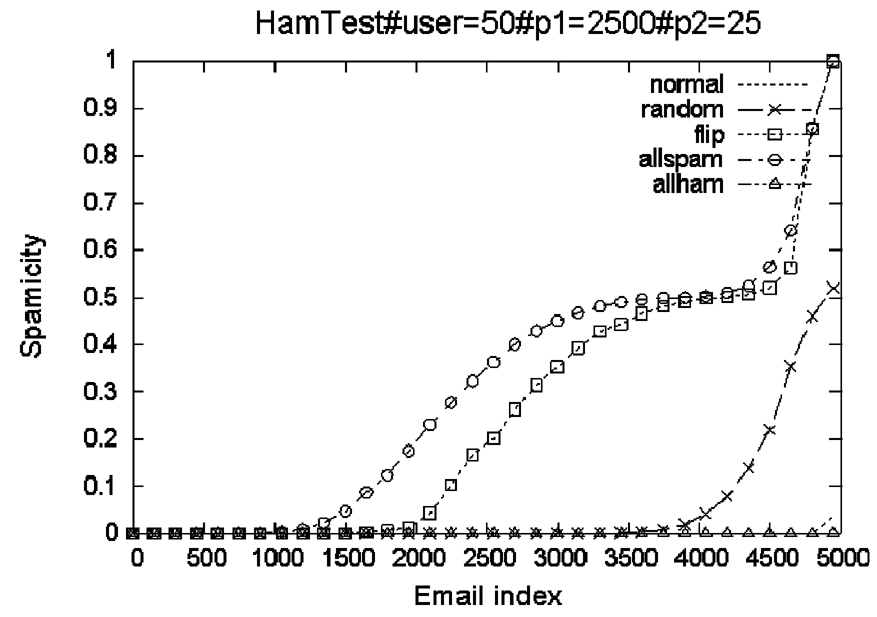
6 malicious user, ham



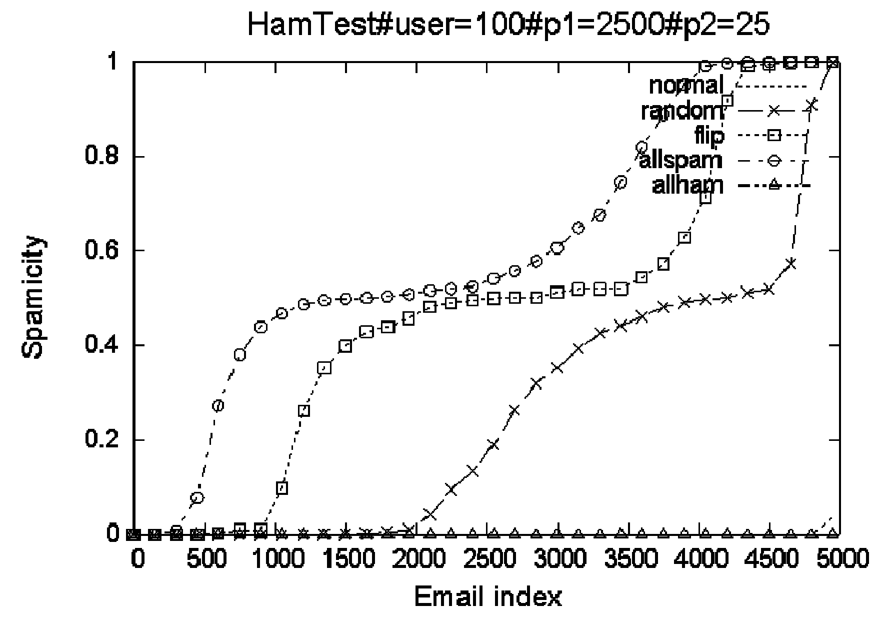
12 malicious users, ham test



25 malicious users, ham test



50 malicious user, ham

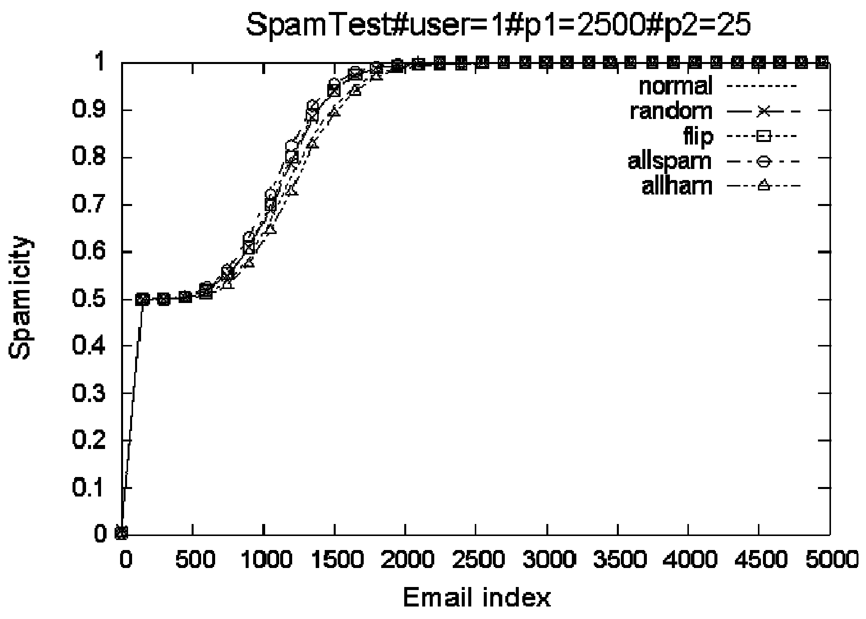


100 malicious users, ham test,

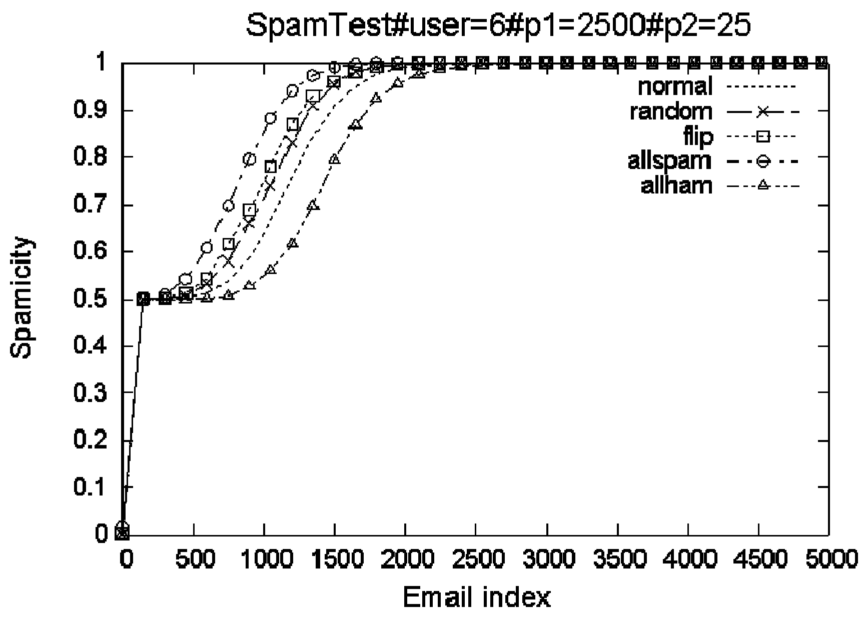
Figure 5.4 Attack effects of different malicious users test on ham email

From the above pictures we have the following observations: For all malicious strategies, the more poisoned emails get propagated, the more effects they cause to the collaborative filter. This follows our intuition. For example, when malicious users are below 10 %(i.e. when the amount of poisoned emails is less than 10 %, the attack has very little effect on the collaborative filter.

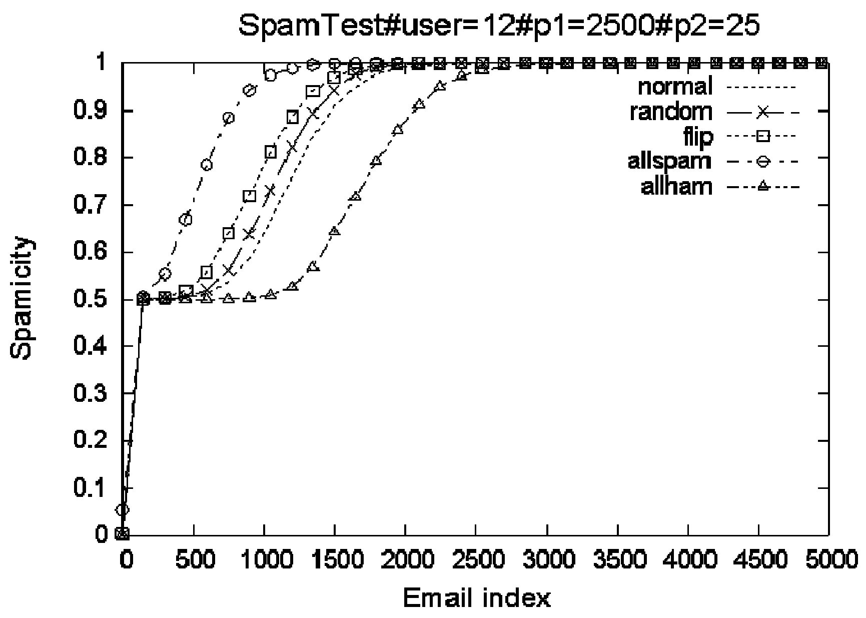
Overall, the all spam strategy caused more ham emails wrongly classified as spam emails (we consider an email is a spam when its spamicity value is above 0.5), however, when the poisoned emails' number is around 25%, the flip strategy significantly outperformed all spam strategy.



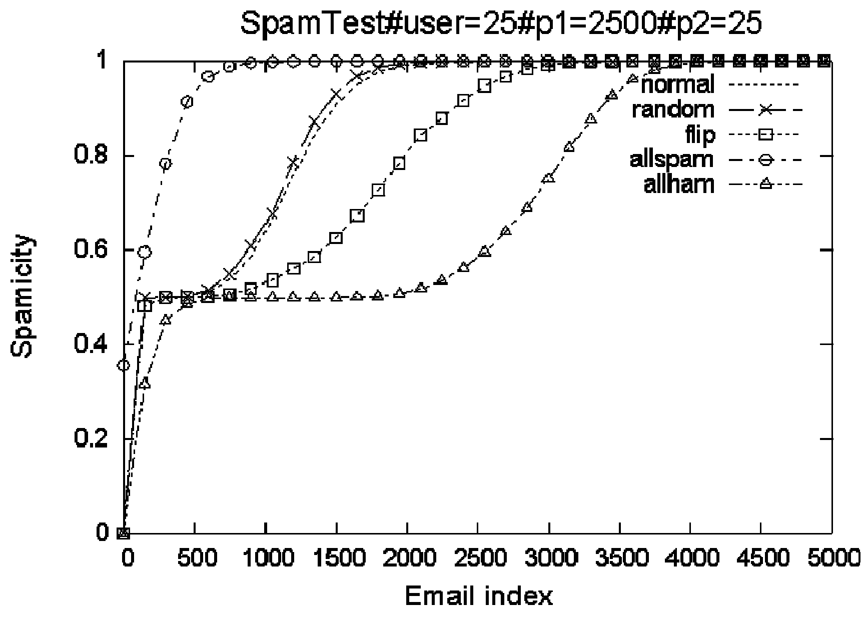
1 malicious user, spam test



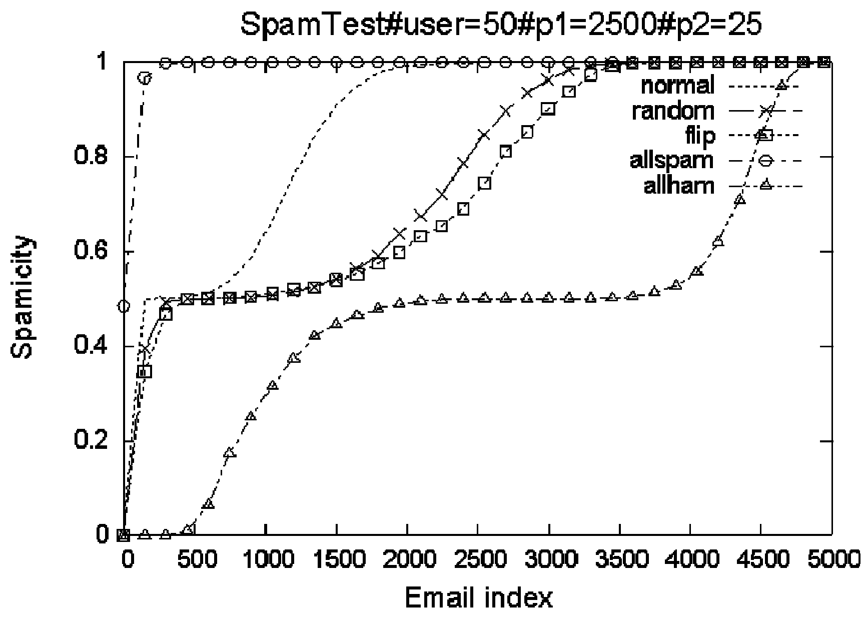
6 malicious users, spam test



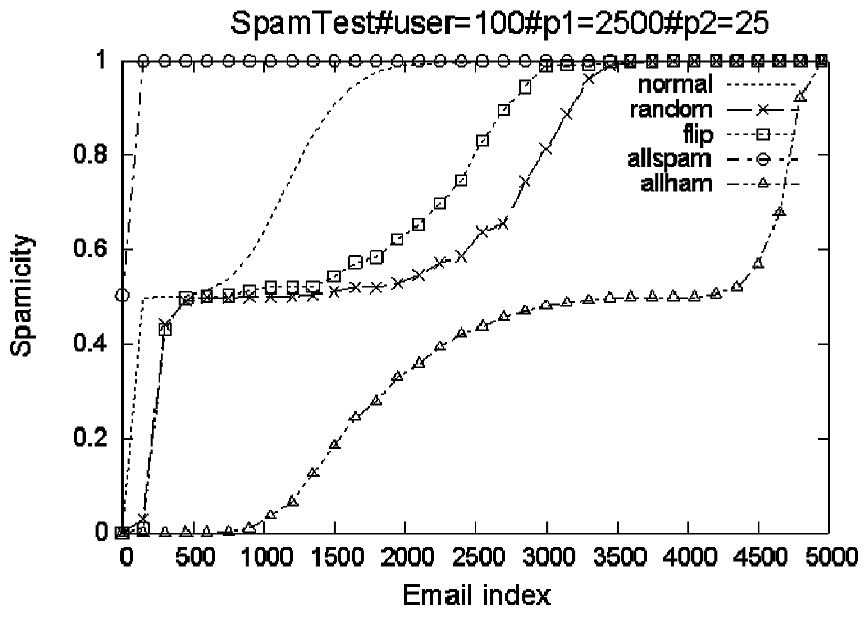
12 malicious users, spam test



25 malicious users, spam test



50 malicious users, spam test.



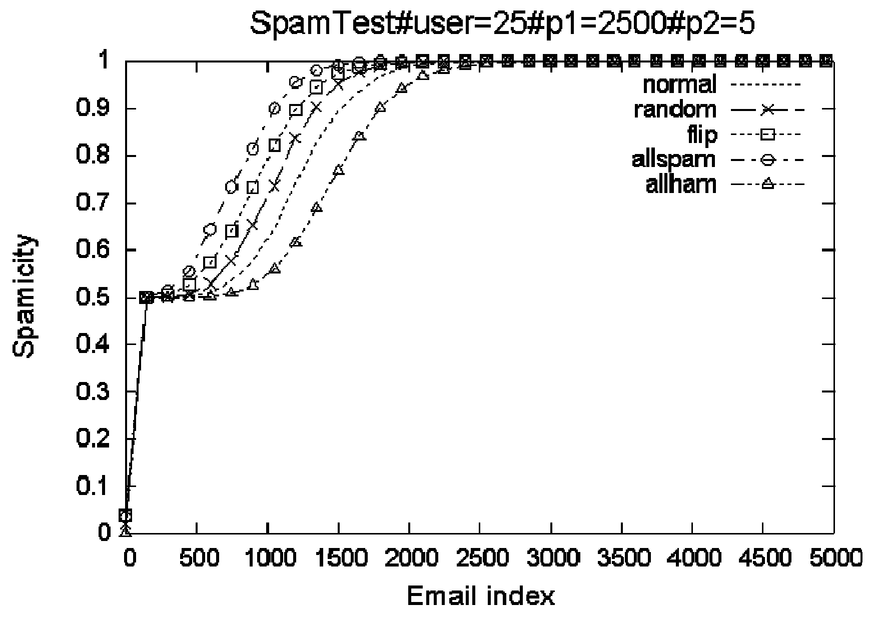
100 malicious users, spam test.

Figure 5.5 Attack effects of different malicious users test on spam email

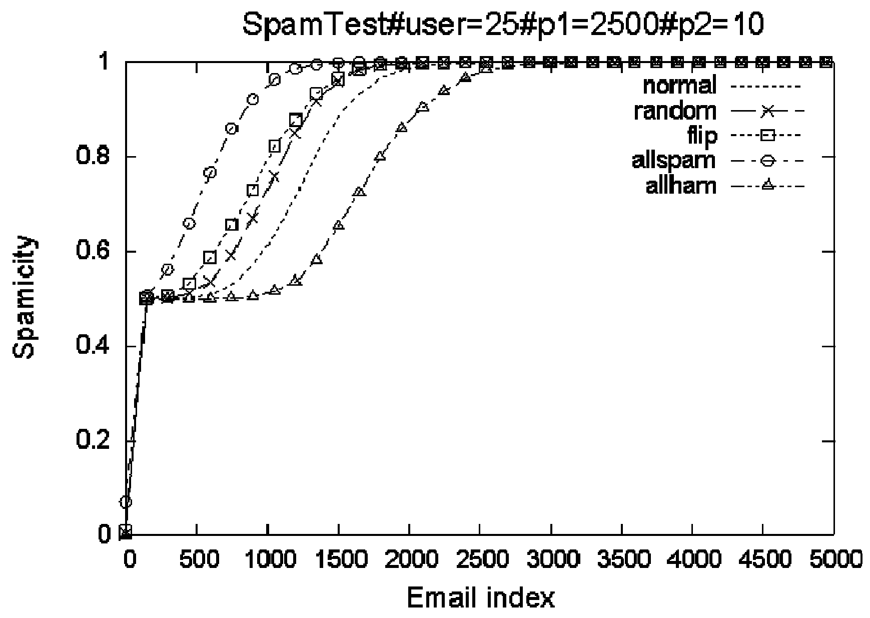
Figure 5.5 shows the same experiment results on pre-classified spam test dataset. The more malicious users there are, the further the line keeps away from the normal strategy, which means the strategy has caused more impact on the filter. This observation is identical to our intuition. Also we noticed that all ham strategy has the most impact on the filter.

5.2.4 Experiment 1.3

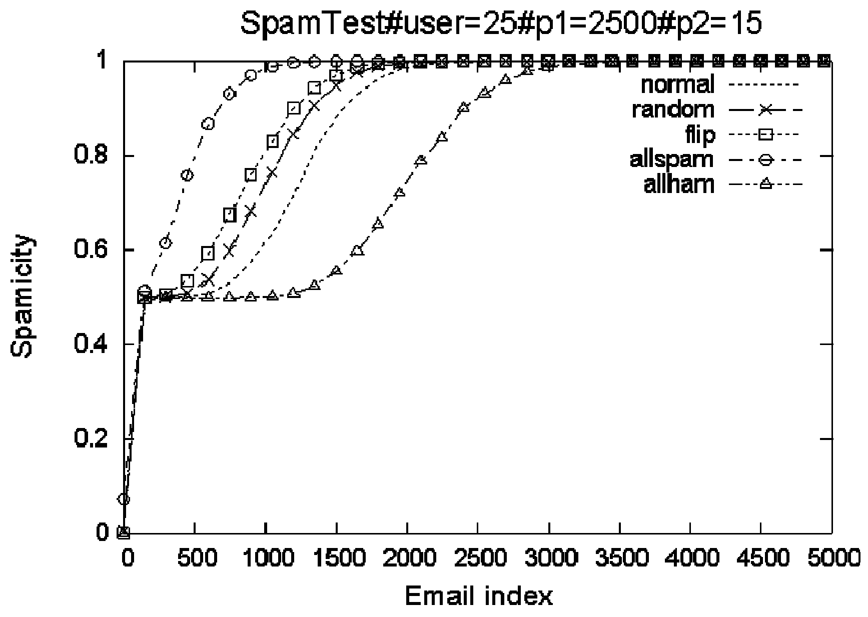
The last experiment in set 1 tell us how parameter like the number of emails used to in phase 2 affects the experiment result. We believe the more emails the malicious users can use in phase 2, the more damage they can cause to the filter.



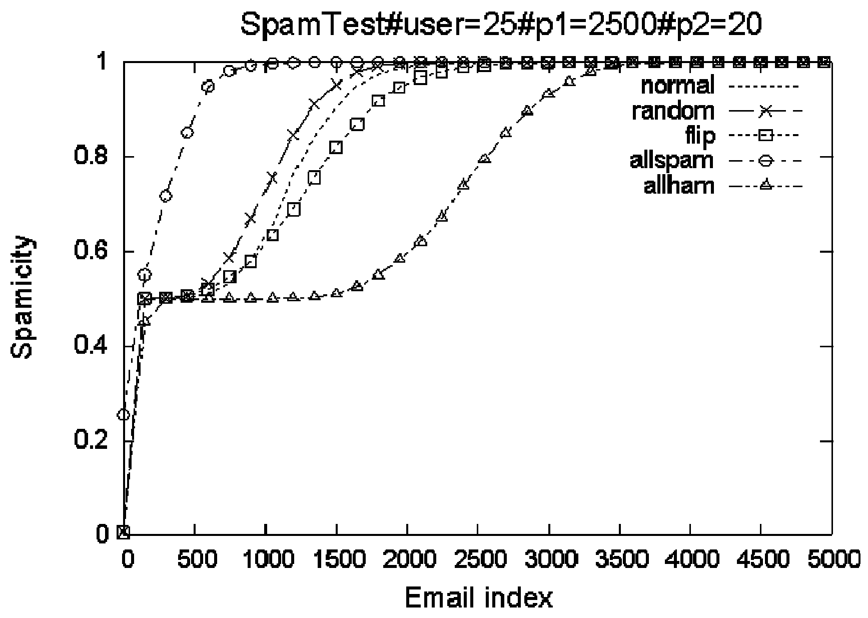
10 emails in parse 2, spam test



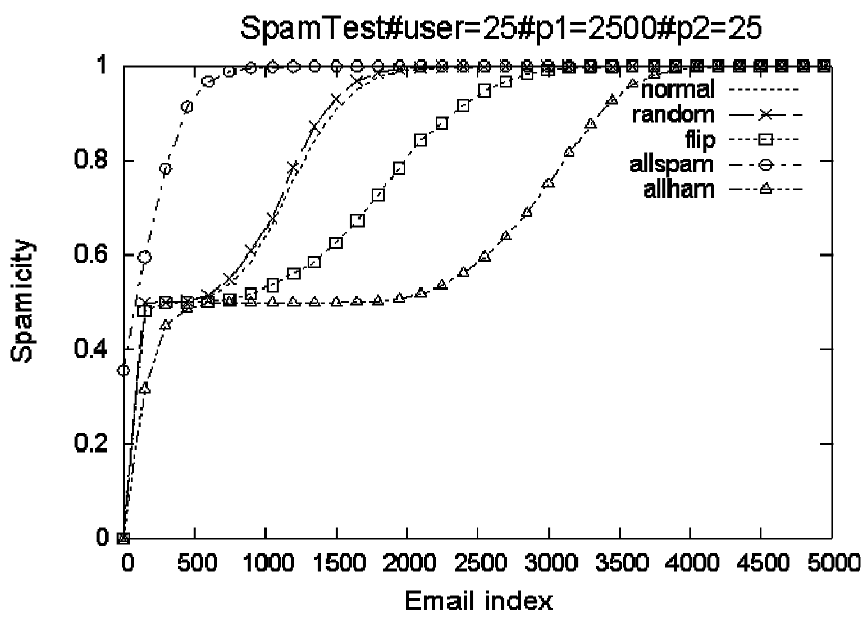
20 emails in parse 2, spam test



30 emails in parse 2, spam test

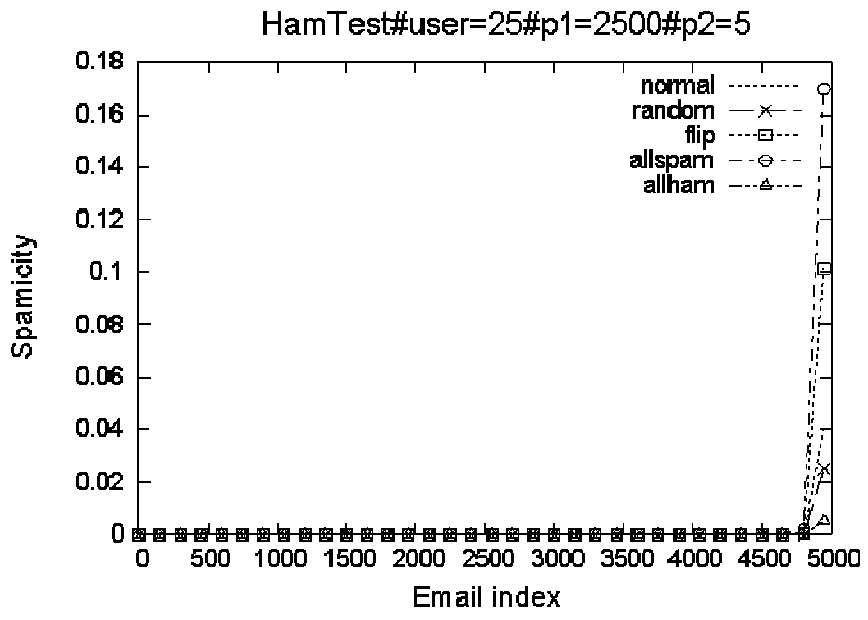


40 emails in parse 2, spam test

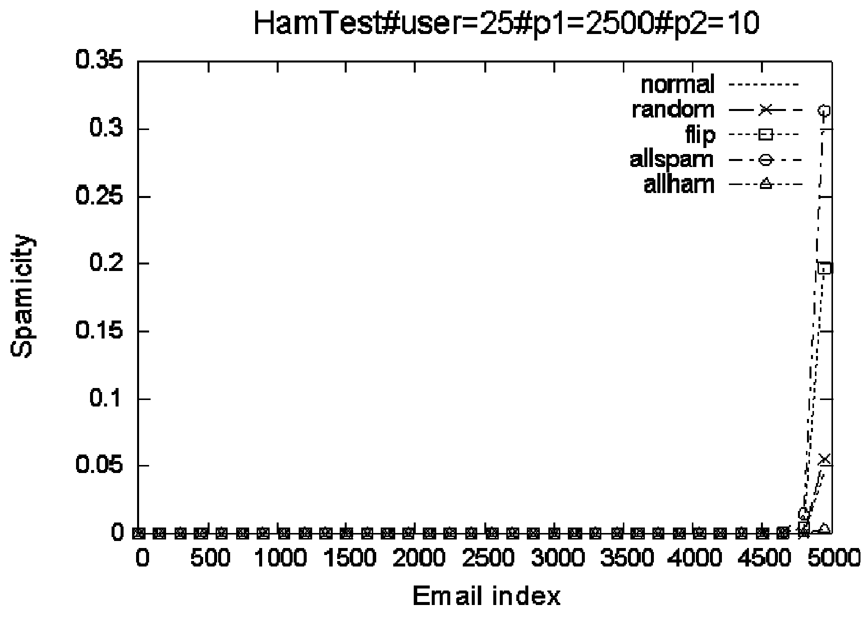


50 emails in parse 2, spam test

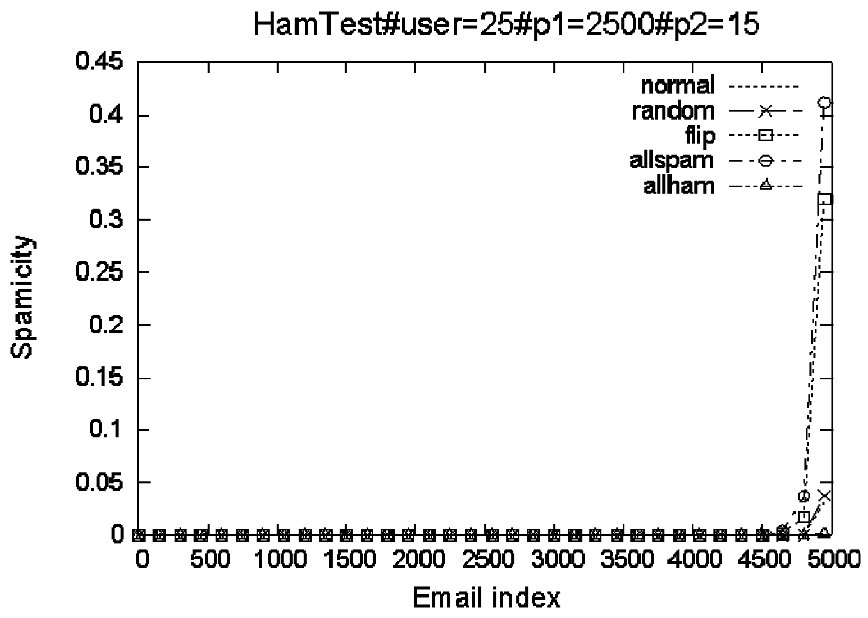
Figure 5.6 Attack effects of different number phase2 training emails, spam test



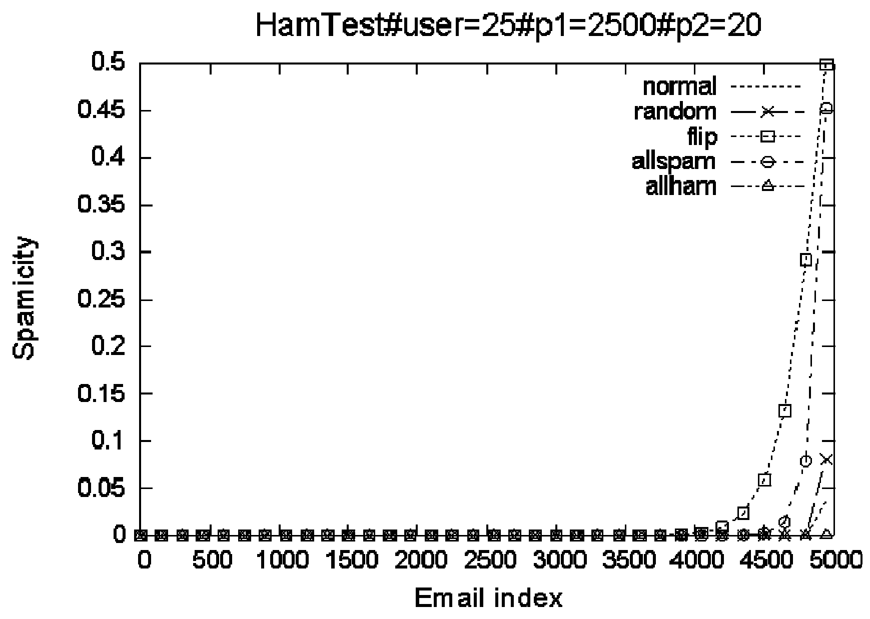
10 training emails in phase 2, ham test



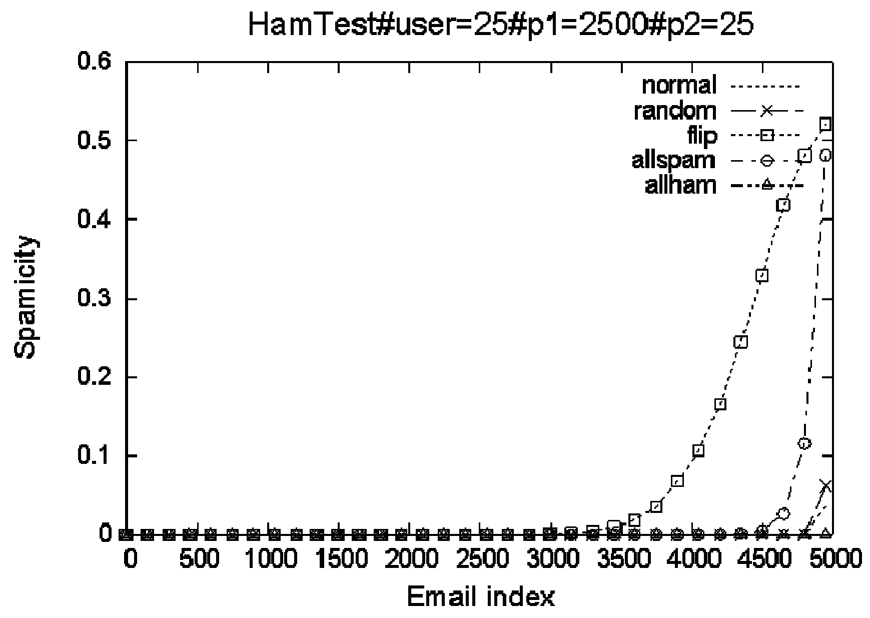
20 training emails in phase 2, ham test



30 training emails in phase 2, ham test



40 training emails in phase 2, ham test



50 training emails in phase 2, ham test

Figure 5.7 Attack effects of different number of phase 2 training emails, ham test

Figure 5.6 and Figure 5.7 confirm our intuitions. Compare the cases when t_2 = 1 to t_2 = 25, lines representing the malicious attack strategies are tends to vary away from the line representing the normal strategy.

5.3 Experiment 2 the Different Percentage of Attacker's Knowledge

From experiment 1 we've learnt how to attack a collaborative spam filter and what's the most effective measure. In this section we will launch an attack. The scenario is like the following: a malicious user is trying to poison the collaborative filter so that emails of certain topic would get blocked. The finding are quite interesting, as we will demonstrate how easily malicious users can break down a collaborative filter system.

5.3.1 Experiment 2 Workflow

The workflow of experiment 2 has 3 phases. There is only 1 user in this system. In phase 1 we train the user's filter with pre-classified dataset t_1 = 5000 (2500ham/ 2500spam). Then in phase 2, we randomly pick an email from the ham training set in phase 1 and call it the target email. We then assume a malicious user has *per* percentage content of this target email, and the malicious user is now trying whatever he can to get this email blocked. Since in experiment set 1 we conclude that all spam strategy has overall the most effect on poisoning the filter, we then assume the malicious user will use this strategy. So in phase 2, the malicious user will repeatedly register the *per* percentage email content he has as spam. At last in phase 3 we evaluate the damage by testing on the target email.

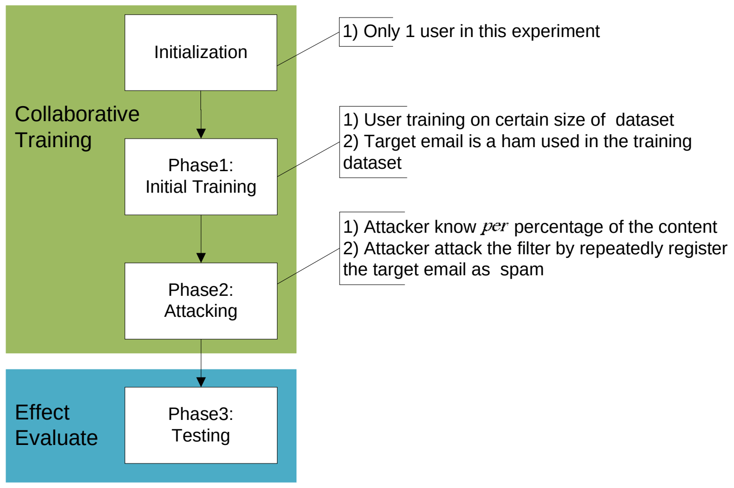

Initialization
1) Only 1 user in this experiment
Collaborative Training
Phase1: Initial Training
1) User training on certain size of dataset
2) Target email is a ham used in the training dataset
Phase2: Attacking
1) Attacker know *per* percentage of the content
2) Attacker attack the filter by repeatedly register the target email as spam
Effect Evaluate
Phase3: Testing

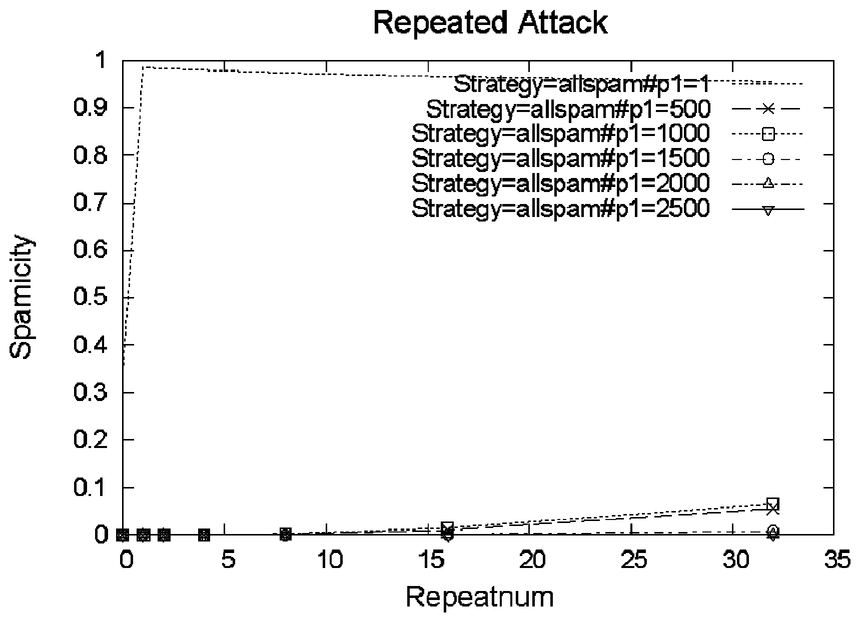
Figure 5.8 Work flow of experiment 2

Figure 5.8 shows the work flow of experiment 2, it is similar with experiment set 1, except instead of propagating feedbacks on phase 2, now the malicious user knows partial content of the target ham email. The way he attacks is to repeatedly register this partial email as a spam and propagate this classification. In next section we are going to introduce the setting of this experiment.

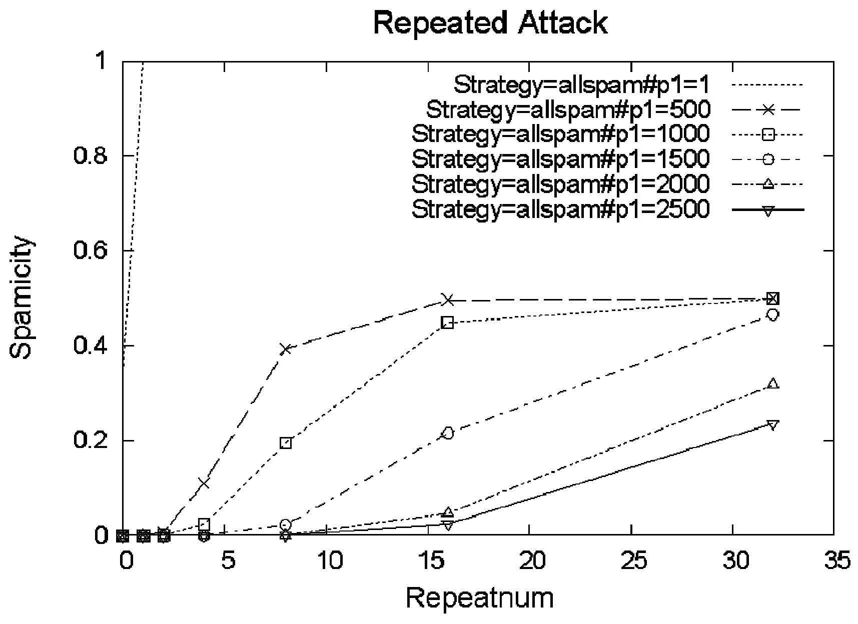
5.3.2 Experiment Setting

This experiment has the following parameters: t_1: the number of emails used in phase 1, $t_1 \in \{1, 500, 1000, 1500, 2000, 2500\}$; *per*: the percentage of content that is known to the malicious user, $per \in \{0.125, 0.25, 0.5, 1\}$; *rpt*: the number of malicious user register the target email as spam, $rpt \in \{0, 1, 2, 4, 8, 16, 32\}$. We believe that the more known content, the more likely the target email will get blocked.

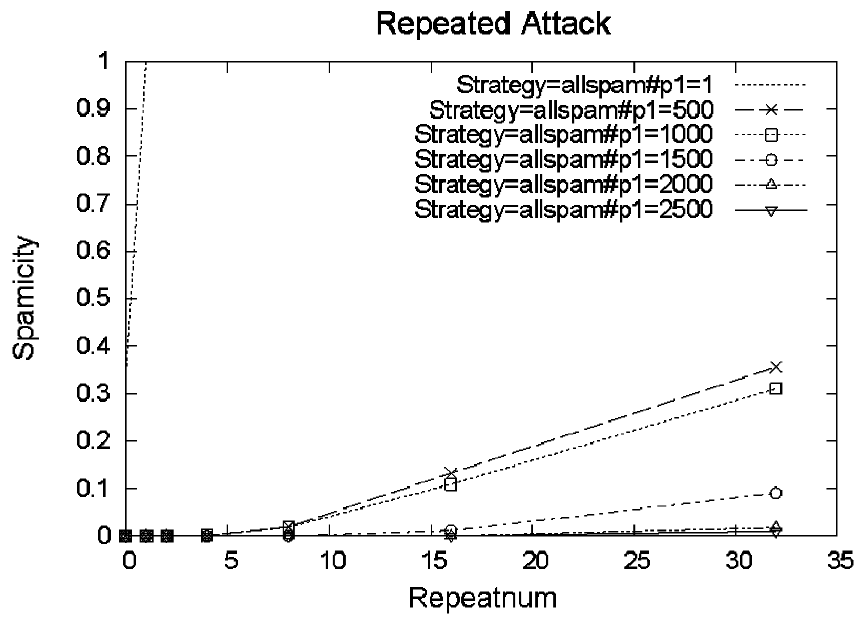
5.3.3 Experiment Results



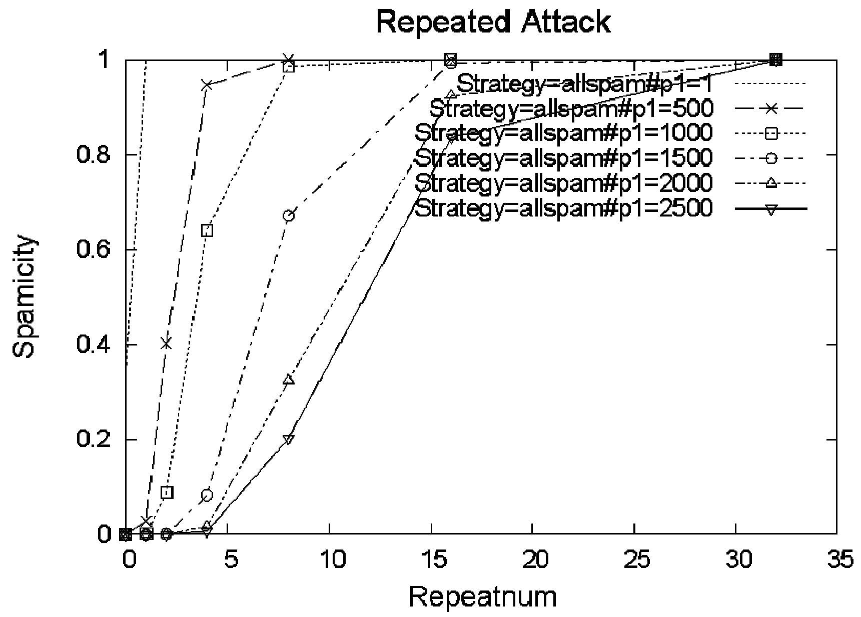
per = 0.125



per = 0.5



per = 0.25



per = 1.0

Figure 5.9 Malicious attacks with *per* percentage known content

The above figures show the results of malicious attacks with different amount of known contents of the target email. In each graph, we also compare the effect against different initial knowledge t_1 which is represented by different types of lines. The y axis stands for spamicity and the x axis is repeat time of the attack. We have the following observations: 1) the more the malicious user knows, the more likely he can successfully block the target email. This can be seen by comparing spamicity(y value) with the same x value in different graph; 2) identical with our experiment set 1, the more initial knowledge, the less effect the malicious attack is. As we can see the line representing 500(500ham/500spam) phase 1 training, has increased most rapidly when the attack number increase and when the malicious user knows more the target email. 3) Similar to our conclusion from experiment 1, the more emails the malicious user used in the attack, the more impact it can cause to the filter, as we can see the spamicity grows as the number of attacks increase.

5.4 Experiment 3 Compare the Impact of Bogofilter Database Size

It is from the previous experiment that we have an intuition that the big the training set is, the more robust a collaborative system will be. A question follows this intuition is will an increased size of bogofilter help us defend/mitigate the malicious user's attack? In this experiment we are trying to see how the total size of the bogofilter's database could affect the different malicious strategies' effect. This experiment is inspired by the conclusion we draw from experiment 2. In experiment 2, we observed that the attack will be less effective as the initial training increase. Our question is that will increasing the database size of the bogofilter a valid way to defend malicious attacks? Are there going to be any side effects?

5.4.1 Workflow of Experiment Set 3

The workflow of experiment 3 is very similar to experiment 2. First we initialize a user. In phase 1 we train the bogofilter instance on some pre-classified email dataset. The difference with experiment 2 is in phase 2 we train some unrelated emails to boost the size of the bogofilter's database, by saying unrelated, here we are saying an artificial email which only contains words that are not in the target email. At last we test the filter on 5000 test emails (2500ham/2500spam).

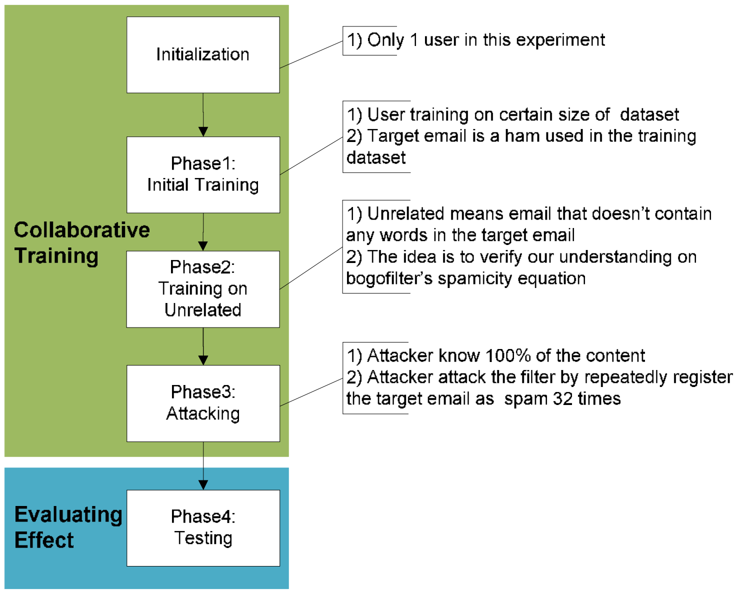

Collaborative Training
Initialization
Phase1: Initial Training
Phase2: Training on Unrelated
Phase3: Attacking
Evaluating Effect
Phase4: Testing
1) Only 1 user in this experiment
1) User training on certain size of dataset
2) Target email is a ham used in the training dataset
1) Unrelated means email that doesn't contain any words in the target email
2) The idea is to verify our understanding on bogofilter's spamicity equation
1) Attacker know 100% of the content
2) Attacker attack the filter by repeatedly register the target email as spam 32 times

Figure 5.10 Experiment 3 work flow

The purpose of our experiment is to see how the size of bogofilter can affect its performance. We will discuss the setting in next section.

5.4.2 Experiment Setting

To simplify the setting, we fix t_1= 2500, *rpt* = 32, *per* = 100%, as we have thoroughly discussed their impact on experiment 2. To generate an unrelated email, we simply create an mbox email with a random string that is not contained by the Enron dataset. Training this email will, on the one hand, increase the size of a bogofilter's database; on the other hand, have no direct effect on the words counting when testing the target email. We have this parameter named *nonsense* and *nonsense* $\in \{0, 500, 1500, 2000, 2500, 5000\}$. Intuitively, we believe that the larger the initial knowledge base is, the less the attacker's effect is. A similar scenario is if we image the malicious attack as salt, the knowledge base as a container of steamed water. To "poison" the water using the same amount of salt, the bigger the container is (i.e. the more water there is), the less salty of the water will be.

5.4.3 Experiment Results

To effectively evaluate the experiment result, we generated a CDF (cumulative distribution function) graph to reflect the impact of different bogofilter database sizes. A CDF is the probability that the variant's value is less than or equal to x:

$$F(x) = P\{X \le x\}$$

By drawing the CDF graph we can observe the trend of the overall spamicity with different parameters.

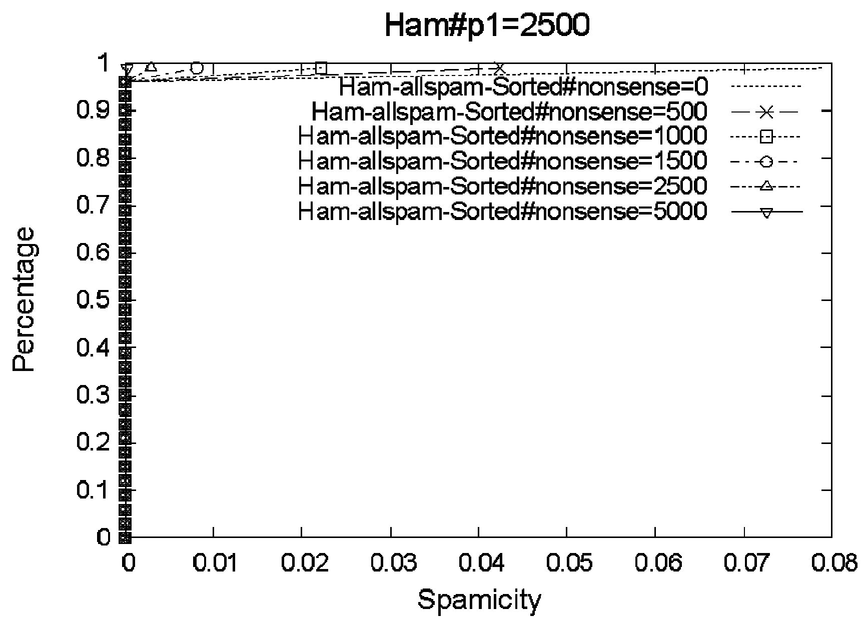


Figure 5.11 Bogofilter with unrelated message training, ham test

One benefit with increased database size is that it will increase its ability to filter ham email. This can be observed from figure 5.11. If we draw a vertical line at spamicity = 0.1, then the line with *nonsense* = 5000 will have the highest percentage value, i.e. it will classify ham emails with lower spamicity overall.

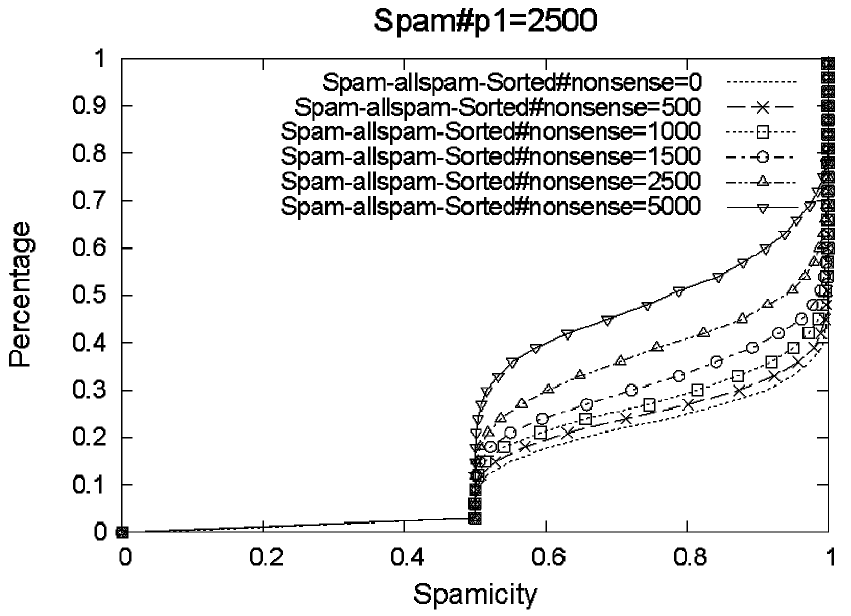


Figure 5.12 Bogofilter with unrelated message training, spam test

However, when testing on the spam dataset, the boosting database size seems causing more trouble than good. From figure 5.12 we can see that, the bogofilter instance behaves less robust when facing malicious attacks on spam test. For example, if we draw a vertical line with spamicity = 0.6, the line represents training with 5000 (2500 ham/2500spam) unrelated messages has the biggest percentage value, i.e. if we view emails with spamicity value bigger than 0.5 as spam, then a bogofilter instance with 5000 extra trainings will capture least number of spam on the spam test dataset. On the other hand, bogofilter instance with no extra training has the highest ability to capture the spam emails.

CHAPTER 6

CONCLUSIONS

Although collaborative filtering systems have many forms, the general workflow of collaborative system is very similar to each other [51], a typical collaborative filter system works like the following:

1) Looking for users who share a similar taste with the active user;

2) Gathering information from each similar group;

3) Use information obtained from Step 2 to calculate prediction/recommendations for the active user

From the workflow, we can see that the shilling attack that we discussed in chapter 5 can be used to against any collaborative system, i.e. malicious users can spread false/wrong information in step2 thus poison the whole system. Our research has been focused on this general mode of collaborative filtering. The conclusion can thus benefit the general collaborative filtering systems.

From the 3 questions we posed earlier in chapter 1, we have the following findings. First question is how effective shilling is on CF Systems. As shown in Experiment set 1, by constantly providing false feedbacks, the collaborative spam filter will be affected when the false feedback takes about 1/4 of the total feedbacks. The filter performs poorly when 1/2 of the total feedbacks are false (figure 5.4/figure 5.5). Thinking about the possibility of using bot to create false accounts, it’s very easy for shills to achieve this goal. Thus we believe shilling is very effective against the CF system.

The second is how the pre knowledge of the CF system affects the attack effect. In experiment set 2 we demonstrate that if attackers know part of the target email, it takes much less effort to block it. Similarly, we believe in general CF systems, knowledge like the items, users, ratings, and algorithms will generally make the attack more effective.

The third question is how the scale of the CF system affects the attack effect. As we have shown in experiment set 3, it will take shills more effort to bring down a CF system with a bigger knowledge base. However, we also observed that simply increasing the filter’s database size is not a good option for defending the shilling attack. It will decrease spamicity for both ham test and spam test;

Although the collaborative filter provides an efficient way for knowledge sharing between different users, based our experiments, we found it also brings vulnerability. Without proper detection/protection mechanisms, the CF system is extremely vulnerable to the shilling attack.

CHAPTER 7

FUTURE WORK

One possible solution to defend the collaborative filter against malicious attacks is to build a user's classification system, where users with similar taste will be grouped into the same group. Thus collaborative operation will only happen within group members. This will require a partition and clustering algorithm to group users.

Another possible solution is to build a user credit system, where every feedback will be assigned a credit score. This approach assumes: 1) the majority users in the system are good users. 2) Good users share a similar opinion on distinguishing spam and ham. Given this system, if a user's feedbacks are significant different from the other users' feedbacks. We'll lower the credit score of this feedback. The credit score can be used to determine the possibility a feedback will be accepted by others. The higher the credit score, the more persuasive the feedback is going to be. This approach will need an algorithm to determine the similarity between emails.

REFERENCE

[1] G. V. Cormack and T. R. Lynam. TREC 2005 spam track overview. In The Fourteenth Text Retrieval Conference (TREC 2005) Proceedings, 2005

[2] Bhaskar Mehta, Thomas Hofmann: A Survey of Attack-Resistant Collaborative Filtering Algorithms. IEEE Data Eng. Bull. 31(2): 14-22 (2008)

[3] Shyong K. Lam , John Riedl, Shilling recommender systems for fun and profit, Proceedings of the 13th international conference on World Wide Web, May 17-20, 2004, New York, NY, USA

[4] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik, Classification features for attack detection in collaborative recommender systems, ACM Press New York, NY, USA, 2006, pp. 542–547

[5] Andreas T¨oscher, Michael Jahrer, Collaborative Filtering Applied to Educational Data Mining. Journal of Machine Learning Research 2010

[6] Husheng Li Learning the Spectrum via Collaborative Filtering in Cognitive Radio Networks, New Frontiers in Dynamic Spectrum, 2010 IEEE Symposium

[7] Larsen, S.a.T., S. “Developing the Personalization-Centric Enterprise Through Collaborative Filtering and Rules-Based,Technologies,” in: *CRM Project*, 1999.

[8] Anderson, M., Ball, M., Boley, H., Greene, S., Howse, N., Lemire, D. and McGrath, S. (2003) ‘RACOFI: a Rule-Applying Collaborative Filtering System’, Proceedings of IEEE/WIC COLA’03, Halifax, Canada, October

[9] S.-T. Park, D. M. Pennock, and D. DeCoste. Applying collaborative filtering techniques to movie search for better ranking and browsing. In AAAI Workshop on Intelligent Techniques for Web Personalization (ITWP 2006), 2006.

[10] T. Anastasakos, D. Hillard, S. Kshetramade, and H. Raghavan. A collaborative filtering approach to sponsored search. Technical Report YL-2009-006, Yahoo! Labs, 2009.

[11] Instance Selection Techniques for Memory-Based Collaborative Filtering (2002) by Kai Yu , Xiaowei Xu , Jianhua Tao , Martin Ester , Hans-peter Kriegel in Proceedings of the 2nd SIAM International Conference on Data Mining

[12] Bin Cao , Jian-Tao Sun , Jianmin Wu , Qiang Yang , Zheng Chen, Learning Bidirectional Similarity for Collaborative Filtering, Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases - Part I, September 15-19, 2008, Antwerp, Belgium

[13] Hyeong-Joon Kwon, Tae-Hoon Lee,Improved Memory-based Collaborative Filtering Using Entropy-based Similarity Measures,Proceedings of the 2009 International Symposium on Web Information Systems and Applications (WISA’09) Nanchang, P. R. China, May 22-24, 2009

[14] Kai Yu, Anton Schwaighofer,Probabilistic Memory-based Collaborative Filtering, Knowledge and Data Engineering, IEEE Transactions Jan. 2004

[15] Robert M. Bell and Yehuda Koren, Improved Neighborhood-based Collaborative Filtering, KDDCup’07, August 12, 2007, San Jose, California, USA.

[16] Abhay S. Harpale and Yiming Yang, Personalized Active Learning for Collaborative Filtering SIGIR’08, July 20–24, 2008, Singapore.

[17] Yehuda Koren. Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model. Proc. Int. Conf. on Knowledge Discovery and Data Mining (2008), p. 426--434.

[18] Collaborative Filtering for Orkut Communities: Discovery of User Latent Behavior Wen-Yen Chen, Jon Chu, Junyi Luan, Hongjie Bai, Yi Wang, and Edward Y. Chang International World Wide Web Conference (WWW)Madrid, Spain, April 2009

[19] J. Y. G Linden, B Smith, “Amazon.com recommendations: item-to-item collaborative filtering,” in IEEE, Internet Computing, vol. 7, 2003, pp. 76–80

[20] Baltrunas, L., Ricci, F.: Dynamic item weighting and selection for collaborative filtering. In: B. Berendt, D. Mladenic, G. Semeraro, M. Spiliopoulou, G. Stumme, V. Svatek, F. Zelezny (eds.) Web Mining 2.0 International Workshop located at the ECML/PKDD 2007, pp. 135–146 (2007)

[21] Thomas Hofmann, Latent Semantic Models for Collaborative Filtering, ACM Transactions on Information Systems (TOIS) TOIS Homepage archive vol. 22 Issue 1, January 2004

[22] Abhay S. Harpale , Yiming Yang, Personalized active learning for collaborative filtering, Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, July 20-24, 2008, Singapore, Singapore

[23] Wen-Yen Chen , Jon-Chyuan Chu , Junyi Luan , Hongjie Bai , Yi Wang , Edward Y. Chang, Collaborative filtering for orkut communities: discovery of user latent behavior, Proceedings of the 18th international conference on World wide web, April 20-24, 2009, Madrid, Spain

[24] R. M. Bell and Y. Koren, "Improved Neighborhood-based Collaborative Filtering", Proc. KDD-Cup and Workshop at the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2007.

[25] Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. Adomavicius, G. and Tuzhilin, A. IEEE Transactions on Knowledge and Data Engineering 06.2005

[26] R.V. Meteren and M.V.Someren. Using content-based filtering for recommendation. Machine Learning in the New Information Age MLnet / ECML2000 Workshop, Spain, 2000.

[27]Ghazanfar, M. and Prugel-Bennett, A. (2010) An Improved Switching Hybrid Recommender System Using Naive Bayes Classifier and Collaborative Filtering. In: The 2010 IAENG International Conference on Data Mining and Applications, 17-19 March, 2010, Hong Kong.

[28] Melville, P., Mooney, R.J., Nagarajan, R.: Content-Boosted Collaborative Filtering for Improved Recommendations. In AAAI/IAAI(2002) 187-192

[29] Abhay S. Harpale , Yiming Yang, Personalized active learning for collaborative filtering, Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, July 20-24, 2008, Singapore, Singapore

[30] Ilaria Bartolini, Zhenjie Zhang, Dimitris Papadias, "Collaborative Filtering with Personalized Skylines," IEEE Transactions on Knowledge and Data Engineering, vol. 23, no. 2, pp. 190-203, Feb. 2011, doi:10.1109/TKDE.2010.86

[31] S.-T. Park, D. M. Pennock, and D. DeCoste. Applying collaborative filtering techniques to movie search for better ranking and browsing. In ITWP, 2006

[32] Balfe, E., Smyth, B.: Case-Based Collaborative Web Search. In: Proceedings of the 7th European Conference on Cased Based Reasoning.

[33] O’Mahony, M.P., Smyth, B.: Collaborative Web Search: A Robustness Analysis. Artificial Intelligence Review, Special Issue on the 18th Artificial Intelligence and Cognitive Science Conference (AICS-07)

[34] G. V. Cormack. TREC 2007 spam track overview. In TREC 2007: Proceedings of the The Sixteenth Text REtrieval Conference, 2007.

[35]H. Drucker, D. Wu, and V. Vapnik. Support vector machines for spam categorization. IEEE Transactions on Neural Networks, 10(5):1048–1054, 1999.

[36] Zhenyu Zhong, Kang Li, "Speed Up Statistical Spam Filter by Approximation," IEEE Transactions on Computers, 14 Apr. 2010

[37] A. Bratko and B. Filipi¨c. Spam filtering using compression models. Technical Report IJS-DP-9227, Department of Intelligent Systems, Jo¨zef Stefan Institute, Ljubljana, Slovenia,2005

[38] K. Li, Z. Zhong, and Lakshmish Ramaswamy, "Privacy-Aware Collaborative Spam Filtering," in IEEE Transactions on Parallel and Distributed systems, Vol. 20, No.5, May 2009

[39] Gray, A. and Haahr, M. Personalised, collaborative spam filtering. In Proceedings of the First Conference on Email and Anti-Spam (CEAS 2004)

[40] D. Sculley and G. V. Cormack. Filtering email spam in the presence of noisy user feedback. In Conference on Email and Anti-Spam, (CEAS 2008), Mountain View, California, August 2008

[41] G. V. Cormack. TREC 2006 spam track overview. In TREC 2006: Proceedings of the Fifteenth Text REtrieval Conference, 2006.

[42] G. V. Cormack. TREC 2007 spam track overview. In TREC 2007: Proceedings of the The Sixteenth Text REtrieval Conference, 2007.

[43] Daniel Lowd Christopher Meek. Good Word Attacks on Statistical Spam Filters, In Proceedings of the Second Conference on Email and Anti-Spam, (CEAS2005)

[44] Wittel, G. L., & Wu, S. F. On attacking statistical spam filters. Proceedings of the First Conference on Email and Anti-Spam (CEAS 2004). Mountain View, CA.

[45] J. Attenberg, K. Weinberger, A. Dasgupta, A. Smola, and M. Zinkevich. Collaborative Email-Spam Filtering with the Hashing-Trick. In Proceedings of the Sixth Conference on Email and Anti-Spam (CEAS 2009)

[46] Ofer Dekel , Ohad Shamir, Good Learners for Evil Teachers. ACM International Conference Proceeding Series; Vol. 382. Proceedings of the 26th Annual International Conference on Machine Learning. archive Pages: 233-240, 2009

[47] http://bogofilter.sourceforge.net/

[48] Gary Robinson, A statistical approach to the spam problem, Linux Journal, v.2003 n.107, p.3, March 2003

[49] http://en.wikipedia.org/wiki/Mbox

[50] Bryan Klimt, Yiming Yang, Introducing the Enron Corpus, In Proceedings of the Sixth Conference on Email and Anti-Spam (CEAS 2004)

[51] X. Su and T. Khoshgoftaar, “A survey of collaborative filtering techniques,” Advances in Artificial Intelligence, 2009.