

By Mentor Cana  
May 5<sup>th</sup>, 2004

## **Final Project**

### **Comparing the effectiveness of two Bayesian based spam filtering software packages:**

#### **Bogofilter vs. SpamBayes**

##### **Introduction**

The purpose of this study is to evaluate the effectiveness of two spam filtering software packages in order to decide which one to recommend for further usage. Both Bogofilter (BF) and SpamBayes (SB) are based on the Bayesian probabilistic model (Baeza-Yates & Ribeiro-Neto, 1999, p. 48), as adapted and proposed by Graham (2002; 2003) for spam e-mail identification and filtering. The key to Graham's Bayesian filtering technique is the ability to train the software with known spam and non-spam (i.e. good) e-mail messages on individual user basis. The idea is that with increased and continued training both packages will be more effective in identifying spam messages, while at the same time decreasing the number of false positives and false negatives.

Both BF and SB are open-source software packages available for free download and use. BF is available for Linux/Unix only and it can be integrated with other mail delivery and filtering tools to automatically tag and filter spam e-mail messages to a separate folder. SB is available for multiple operating system platforms and can be configured to work with Unix/Linux command line mail delivery and filtering tools, as well as POP3, IMAP, the Outlook e-mail client, etc. Detailed instructions are provided at <http://bogofilter.sourceforge.net/> and <http://spambayes.sourceforge.net/> respectively.

I have used both of these systems and would like to be able to answer the question as to which one of these two systems is more effective at identifying spam.

The Bayesian technique suggests that with continued training the software packages should become more effective in spam identification. Thus, the first research question:

RQ1: Does the spam filtering effectiveness of BF and SB improve as the amount of e-mail messages used for training increases?

From my personal experience it appears that SB is more effective than BF. Although after good amount of training both SB and BF seem to be very effective in that I rarely get false positives in both of these implementations. Thus, the second research question:

RQ2: Is the spam filtering effectiveness of SB better than BF?

### **Experiment design, sampling and procedures**

The corpus of spam and good (i.e. non-spam) e-mail messages is extracted from my personal account, where I've been using both BF and SB to filter out unwanted spam messages. The corpus of 1500 spam messages is a collection of spam I have received in a period of about two weeks. The corpus of good e-mail messages is extracted from the 'read' set of messages on my account. The corpus of 1500 good messages is a collection of personal e-mails and messages I have received as a participant on multiple electronic discussion and news lists.

To assess the incremental effectiveness due to training, the following sets of training messages are used, where each set contains equal number of good and spam messages: set2, set10, set20, set50, set100, set200, set400, set1000, set3000. The first set (set2) contains one good and one spam message. The last set (set3000) contains the entire corpus of good and spam messages (i.e. 1500 spam and 1500 good messages).

In order to simulate the time sensitive arrival of e-mail messages, the relationship amongst the sets is as follows:  $set2 \subseteq set10 \subseteq set20 \subseteq set50 \subseteq set100 \subseteq set200 \subseteq set400 \subseteq set1000 \subseteq set3000$ ; where set2 starts with the least recent e-mail messages received. It is important to note

that spam messages usually forge their dates, thus it is necessary to sort the spam messages by their arrival and not by their date field. This is easily achievable with the pine e-mail client.

There are three distinct steps (iterated nine times) in this experiment:

- un-train both SB and BF
- training SB and BF each with a set of spam and good messages
- once trained, use SB and BF separately to assess the probability that each message in the corpus of 3000 (1500 spam and 1500 good) is a spam message. The output of this step are the following variables that contain the spam probabilities: sb2, bf2, sb10, bf10, sb20, bf20, sb50, bf50, sb100, bf100, sb200, bf200, sb400, bf400, sb1000, bf1000, and sb3000, bf3000, corresponding to each training iteration.

The ROC curves are plotted for each training iteration to assess the effectiveness between the two systems. Also, the ROC curves are plotted for each systems separately to assess the effectiveness due to increased training.

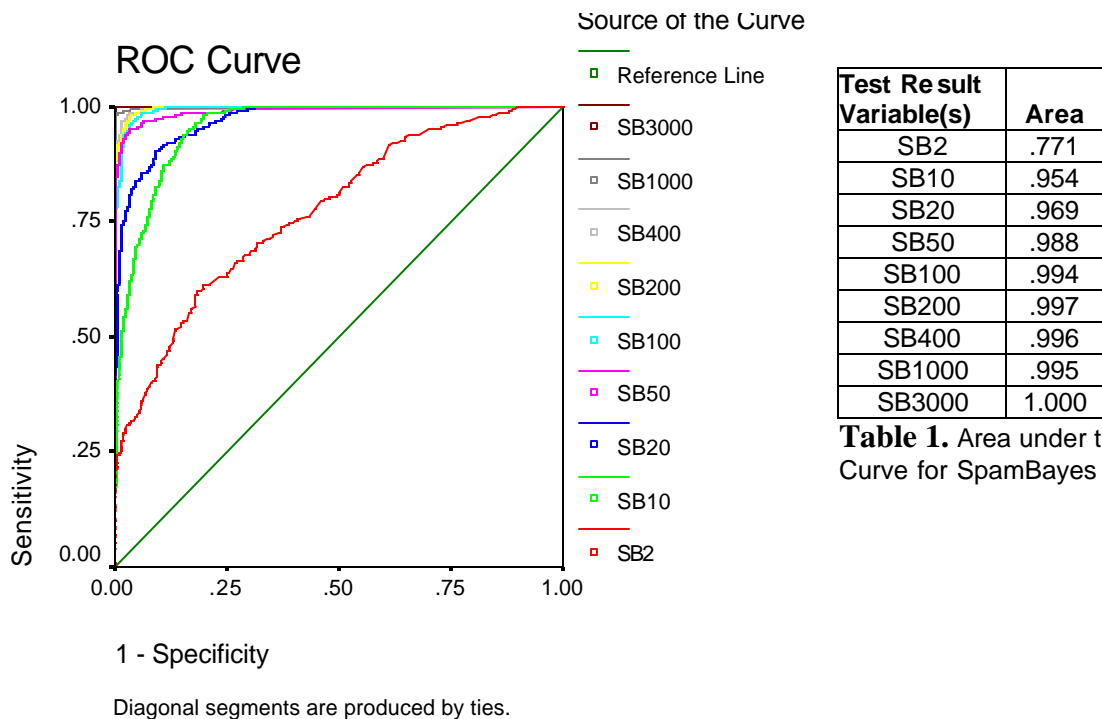
However, the ROC curve does not differentiate between false positives (i.e. good messages

	<b>Spam collection</b>	<b>Good collection</b>
<b>Detected as spam</b>	True positives (TP)	False positives (FP)
<b>Not detected as spam</b>	False negative (FN)	True negative (TN)

filtered as spam) and false negatives (i.e. spam messages not identified as spam) and assigns the same value to both types of misclassifications. Thus, for a better evaluation of these systems, it should be noted that individuals assign higher value to false positives than to false negatives. As is the case with spam filtering, a lost communication due to false positives is much more critical and undesirable than some extra spam messages in individual's inbox. Because of this, the relationship between the false positives and false negatives is tabulated and plotted for each of the iterations at the default cutoff level of 0.9.

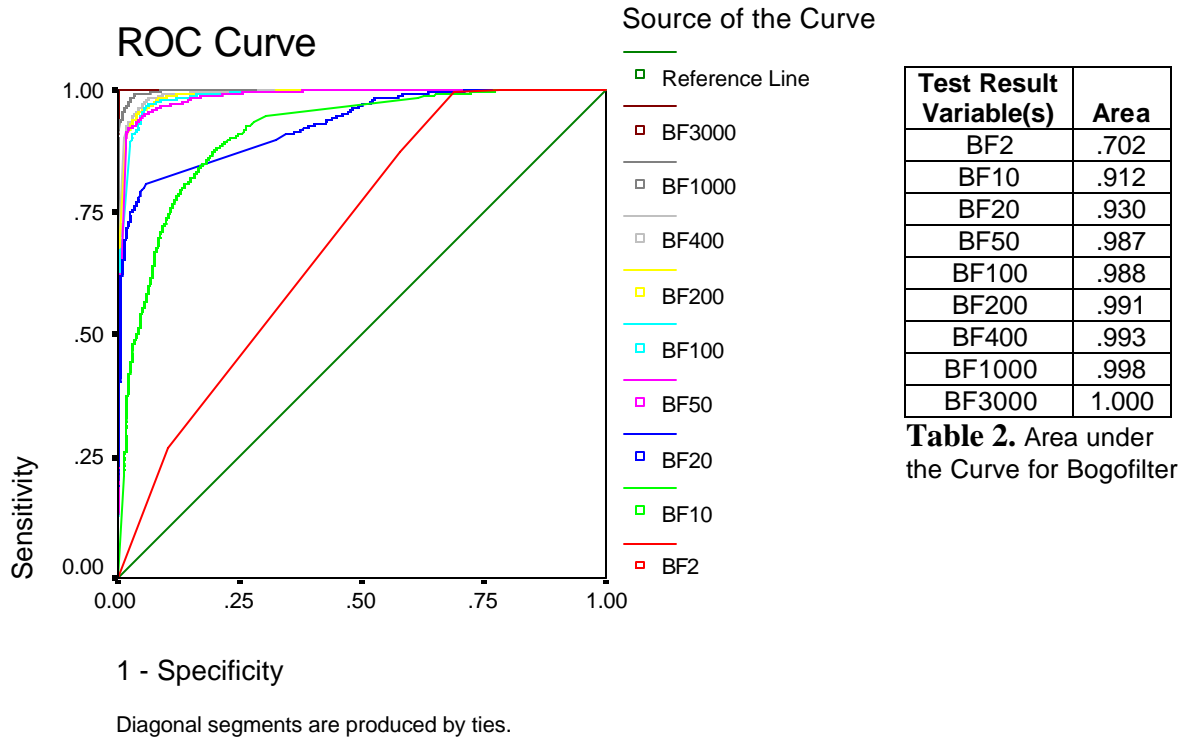
### Data analysis and discussion

As it is shown in Figure 1 and Table 1, and Figure 2 and Table 2 respectively for SB and BF, it is evident that both systems perform more effectively as the amount of training messages increases. With sufficient amount of training, both systems perform comparably as it can be seen from the ROC curves data.



**Figure 1.** SpamBayes ROC curves at different training level

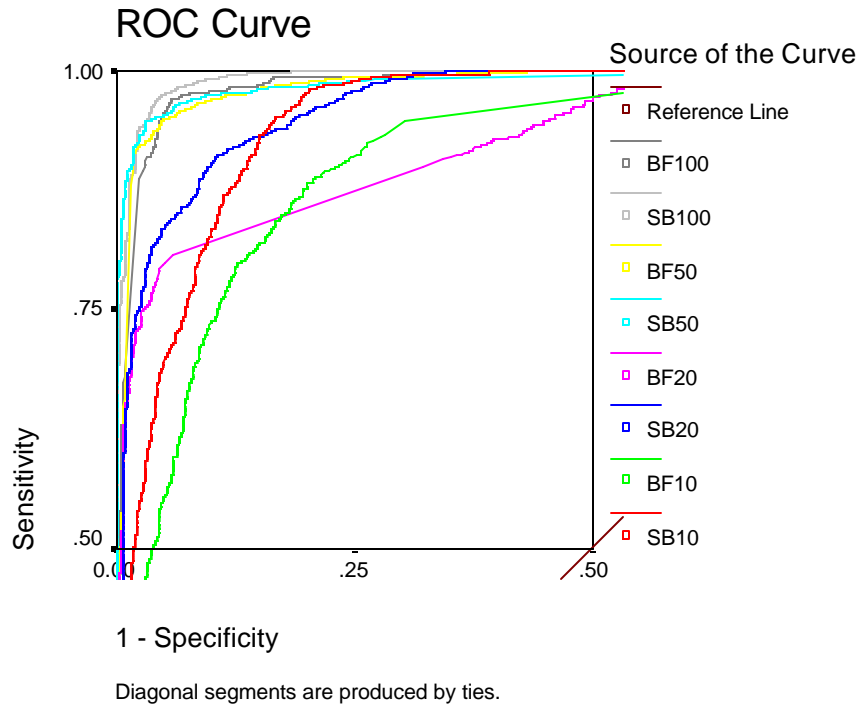
Considering that both systems perform comparably well with sufficient training, we now concentrate on the effort of training required by both systems to achieve the level of effectiveness indicated by SB1000 and BF1000, and beyond.



**Figure 2.** Bogofilter ROC curves at different training levels

With operational spam filtering systems, the incoming e-mails are filtered as spam in real time. In the case of false positives, the messages in question need to be re-trained as good messages. In the case of false negatives, the messages in question need to be re-trained as spam messages. The less retraining required the better the system, keeping in mind that false positives ‘cost’ more due to missed communication, and identifying false positives amongst the true positives is a very time demanding, tedious and frustrating task.

From Figure 3 and Table 3 it can be seen that at each training level (set10, set20, set 50, and set100) SB is more effective than BF.



Test Result Variable(s)	Area
SB10	.954
BF10	.912
SB20	.969
BF20	.930
SB50	.988
BF50	.987
SB100	.994
BF100	.988

**Table 3.** Area under the Curve for few training levels for both SB and BF

**Figure 3.** SB and BF ROC curves at different training levels

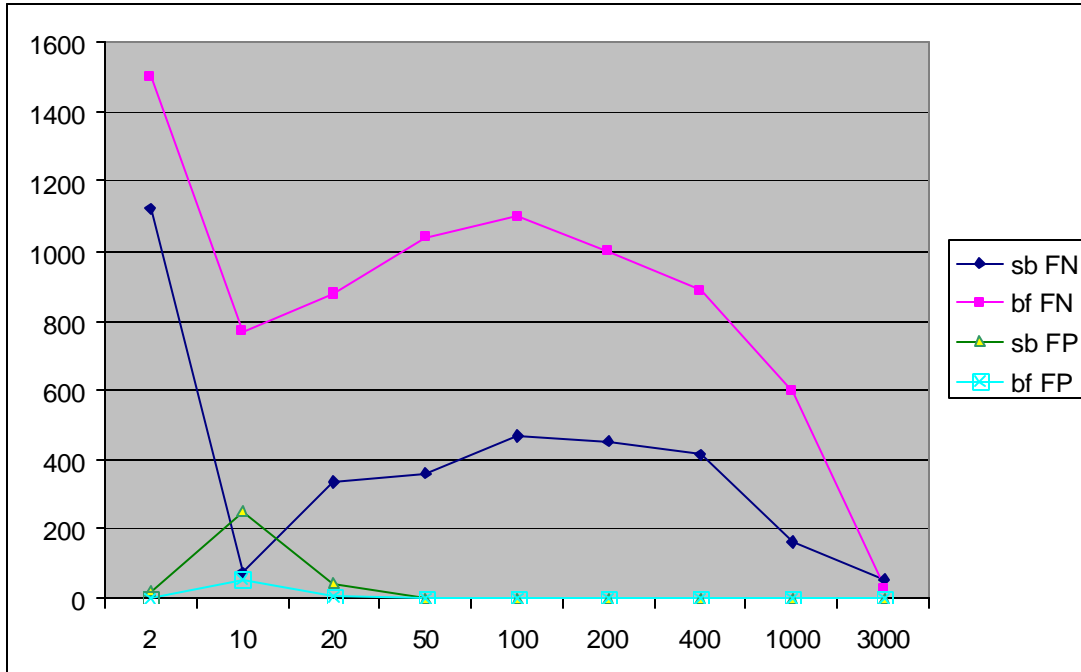
So far it has been shown that with increased amount of training messages both SB and BF are more effective. Also, SB is slightly more effective than BF consistently at each training level.

To determine which of the systems requires less training effort, Table 4 and Figure 4 show the amount of FP, FN, TP, and TN at 0.9 spam cutoff at different training levels.

# number of training messages (half good & half spam)	sb FN	bf FN	sb FP	bf FP
2	1124	1501	18	0
10	75	767	250	52
20	337	877	42	4
50	359	1039	1	2
100	466	1096	1	0
200	454	1001	1	0
400	413	888	0	0
1000	163	595	0	0
3000	52	29	0	0

sb FN %	bf FN %	sb FP %	bf FP %
74.93	100.07	1.20	0.00
5.00	51.13	16.67	3.47
22.47	58.47	2.80	0.27
23.93	69.27	0.07	0.13
31.07	73.07	0.07	0.00
30.27	66.73	0.07	0.00
27.53	59.20	0.00	0.00
10.87	39.67	0.00	0.00
3.47	1.93	0.00	0.00

**Table 4.** The amount of FP, FN, TP, and TN at 0.9 spam cutoff level



**Figure 4.** The amount of FP, FN, TP, and TN at 0.9 spam cutoff level

It is evident that the number of FP for both SB and BF level at zero with as little as 100 or 200 (half spam and half good) training messages. At the same time however, there is an ongoing number of FN for both systems. The amount of FN for SP is less than half of the FN for BF, as it is shown on Figure 4. These ratios hold true for different spam cutoff levels between 0.7 and 0.95. For example, at spam cutoff level of exactly 1.0, the FP level at zero even with 50 training messages. However, the number of FN increases more than two-folds as compared to spam cutoff level of 0.9.

### Assumptions and limitations

For a more complete analysis of effectiveness, the experiment needs to be repeated with multiple corpuses provided by different individuals due to the uniqueness and various patterns of e-mail use between individuals. The pattern of e-mail messages imbedded in the set3000 corpus is defined by my personal e-mail communication as well as by e-mails I receive at aliases and

forwarding e-mail addresses due to my involvement as moderator and administrator of various electronic news and discussion lists.

Additionally, the cap of 3000 messages in the corpus can be modified to see any potential variability in effectiveness. Although I believe that having 3000 messages tested for spam probability seems sufficiently large amount comparable to real life operational spam filtering systems.

The equal proportion of spam and good messages in the training sets might not resemble real life situations. The rate of spam messages received is much higher than good messages, at least in my case. Accounting for variable proportion would improve this experiment. This might even yield an optimal proportion for a given spam cutoff level.

In this experiment, the issue of performance was not considered. Apart from the effectiveness, if one of the systems is to be used for real time spam filtering on a Unix/Linux server supporting thousands of users, SB might be at disadvantage due to its implementation with the python language. BF on the other side is implemented with c++ and runs significantly faster.

## **Conclusion**

Based on the above analyses, the following can be concluded:

- the spam filtering effectiveness of both SB and BF improves with the increased number of training messages
- at each training level SB is more effective than BF

Recommendation: In conjunction with the results on Figure 4 and Table 4 showing the amount of FP, FN, TP, and TN at 0.9 spam cutoff at different training levels, SB is more effective due to the significantly lower number of FN compared to BF. In order to minimize the

training effort due to false negatives and false positives, it is recommended that once SB is installed for use, it should be trained with at least 200 or 400 (half good & half spam) messages. At this stage, the number of FP is zero. However, the spamming techniques change as fast (and even faster) in comparison with spam filtering packages. This is an ongoing battle and no software packages can identify all spam messages.

For example, my installation of SB rarely identifies false positives. But once in a while it does, especially when the pattern of spam messages changes all of a sudden or a virus with a unique behavior appears on the scene. I also expect to receive false positives when subscribing to a new discussion list, more so if it is in a different language or topic of interest different from the rest of the discussion lists I'm already subscribed.

The battle against spam will continue as long as spammers have incentives to send spam messages. Spam filtering systems are indeed helpful in reducing the false positives and false negatives. Both SB and BF seem to be designed to eliminate the false positives with as little training as possible. In any case, due diligence and patience is needed by the user. For better effectiveness the user should continuously train the system of choice. To aid in this process, both SB and BF allow for good cutoff level in addition to the spam cutoff level. The messages with spam probabilities between the good cutoff and spam cutoff can be filtered in a separate folder (usually called 'unsure') and trained appropriate.

## Appendix A

### Sample ROC curve SPSS code

```
ROC
  sb10 bf10 sb20 bf20 sb50 bf50 sb100 bf100  BY knownst (1)
  /PLOT = CURVE(REFERENCE)
  /PRINT = SE
  /CRITERIA = CUTOFF(INCLUDE) TESTPOS(LARGE) DISTRIBUTION(FREE) CI(95)
  /MISSING = EXCLUDE .
```

### Filtering and training commands

The following are the training and filtering Linux/Unix commands. The e-mail messages are in mbox mail format.

#### Bogofilter (v0.17.5)

- Training

```
spam training: % cat <spam file> | bogofilter -s -M
```

```
spam training: % cat <good file> | bogofilter -n -M
```

- Filtering

```
% cat <set3000 file> | bogofilter -p -t -M
```

The above command outputs the entire set3000 file where each individual message has an additional header line added noting the spam probability. Additional Linux commands and scripts were used to extract the probabilities.

#### SpamBayes (v1.0b1)

- Training

```
% sb_mboxtrain.py -f -d hammiedb -g <good file> -s <spam file>
```

- Filtering

```
% sb_filter.py -d hammiedb <set3000 file>
```

## References:

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval*. New York: Addison Wesley.

Bogofilter (n.d.). *Bogofilter v0.17.5*. Retrieved May 4, 2004, from

<http://bogofilter.sourceforge.net/>

Graham, P. (2002, August). *A plan for spam*. Retrieved May 4, 2004, from

<http://www.paulgraham.com/spam.html>

Graham, P. (2003, January). *Better Bayesian filtering*. Retrieved May 4, 2004, from

<http://www.paulgraham.com/better.html>

SpamBayes (n.d.). SpamBayes v1.0b1. Retrieved May 4, 2004, from

<http://spambayes.sourceforge.net/download.html>