E-mail Spam Filter

By
Kalamkar Niraj N. (Y3111020)
Rohan Shetty (Y3111041)
Atul Gupta (Y3111064)

Department of Computer Science and Engineering,
Indian Institute of Technology Kanpur
Problem Definitions

1. Build an email spam filter, using a single or multiple learning techniques. Carefully create your learning set. At the end of the course your filter will be tested against an independent set of emails.

2. Document your design, reasons for choosing the learning technique(s), choice of learning set, testing, and references.

3. Do not borrow existing filters and adapt them. Build yours from scratch.

Contents

<table>
<thead>
<tr>
<th>S. no.</th>
<th>Particulars</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Introduction</td>
</tr>
<tr>
<td>2.</td>
<td>Anti-spam filters</td>
</tr>
<tr>
<td>3.</td>
<td>Paul Graham’s Approach</td>
</tr>
<tr>
<td>4.</td>
<td>Gary Robinson’s Approach</td>
</tr>
<tr>
<td>4.1</td>
<td>Generating Word Probabilities</td>
</tr>
<tr>
<td>4.2</td>
<td>Dealing with Rare Words</td>
</tr>
<tr>
<td>4.3</td>
<td>Combining the Probabilities</td>
</tr>
<tr>
<td>4.4</td>
<td>The Indicator of Hamminess or Spamminess</td>
</tr>
<tr>
<td>5.</td>
<td>Our Approach</td>
</tr>
<tr>
<td>5.1</td>
<td>Calculating Word Probabilities</td>
</tr>
<tr>
<td>5.2</td>
<td>Corpus Data</td>
</tr>
<tr>
<td>5.3</td>
<td>Program Description</td>
</tr>
<tr>
<td>6.</td>
<td>Performance Measures used</td>
</tr>
<tr>
<td>7.</td>
<td>Observations &amp; Tables</td>
</tr>
<tr>
<td>8.</td>
<td>Conclusions</td>
</tr>
<tr>
<td>9.</td>
<td>Further Scope</td>
</tr>
<tr>
<td>10.</td>
<td>References</td>
</tr>
<tr>
<td>11.</td>
<td>Appendix</td>
</tr>
</tbody>
</table>
1 Introduction

Electronic mail is an efficient and increasingly popular communication medium. Like every powerful medium, however, it is prone to misuse. One such case of misuse is the blind posting of unsolicited e-mail messages, also known as spam, to very large numbers of recipients. Spam messages are typically sent using bulk-mailers and address lists harvested from web pages and newsgroup archives. They vary significantly in content, from vacation advertisements to get-rich schemes. The common feature of these messages is that they are usually of little interest to the majority of the recipients. In some cases, they may even be harmful, e.g. spam messages advertising pornographic sites may be read by children. Apart from wasting time and bandwidth, spam e-mail also costs money to users with dial-up connections.

Attempts to introduce legal measures against spam mailing have had limited effect. A more effective solution is to develop tools to help recipients identify or remove automatically spam messages. Such tools, called anti-spam filters, vary in functionality from blacklists of frequent spammers to content-based filters. The latter are generally more powerful, as spammers often use fake addresses. They use patterns those appear in the mails and based on that the mail may be classified as being spam or ham. These patterns need to be crafted by hand, and to achieve better results they need to be tuned to each user and to be constantly maintained [1], a tedious task, requiring expertise that a user may not have.

A number of content based filters have been designed based on Bayesian statistics since Paul Graham article posted on the web "A Plan for Spam"[2]. It is now believed that the Bayesian approach is a better approach for designing content based filters. In our project, we have also used an approach which is similar to Bayesian but an improved one which tries to overcome some of the difficulties of the earlier approaches based on the Bayesian statistics.
2. Anti-spam filters

A spam filter is a computer program that scans incoming e-mail and sends likely spam to a special spam folder. Spam filters can make two kinds of mistakes. A filter might fail to detect that a particular message is spam and incorrectly let it through to the inbox (a false negative). Equally a filter might incorrectly route a perfectly good message to the spam folder (a false positive). An effective spam filter will have a low false negative rate, and perhaps more critically, a low false positive rate.

The spam filter may be designed based on:

1. Hand Crafted Rules

Early spam filters used hand-crafted rules to identify spam. Here are the antecedents for some typical spam rules:
- <Subject> contains “FREE” in CAPS
- <Body> contains “University Diploma”
- <Body> contain an entire line in CAPS
- <From:> starts with numbers

This approach can produce elective spam filters. In fact, merely looking for the word “FREE” for instance catches about 60% of the e-mails in a collection of spam, with less than 1% false positives. However, the hand-crafting approach suffers from three significant drawbacks. First, creating rules is a tedious, expensive, and error-prone process. Second, humans are unlikely to spot more obscure indicators of spam such as the presence of particular HTML tags or the use of certain font colors. Third, the spammer may cleverly modify their mails to defy these rules.

2. Supervised learning methods
   - Naive Bayesian Classifiers
   - Memory based
   - Improved Bayesian Methods like proposed by Paul Graham, Gary Robinson, Used Here, etc.
3. Paul Graham Approach

In his approach [2], Paul calculates the probability of a given mail being spam, if a particular word appears in that. Likewise it calculates the word probabilities for spamminess and hamminess for each word that appears in the mails comprising training set. The calculation is done as follows:

Let $g$ be the number of times a particular word occurred in all ham mail, and $b$ be the number of times that word occurred in all the spam mail. Let the number of ham mails are $n_{good}$ and number of spam mails are $n_{bad}$.

Than the probability that a mail containing a particular word being an spam is calculated as

$$\frac{\min(1, \frac{b}{n_{bad}})}{\max(0.01, \min(0.99, \frac{g}{n_{good}} + \min(1, \frac{b}{n_{bad}})))}$$

Here in the above formula, the probabilities are made to lie in the range 0.01 to 0.99. The word probabilities for all the words present in the training set are calculated. Paul has used a naïve Bayesian approach to combine the word probabilities which makes the assumption of these probabilities being independent to each other.

When new mail arrives, it is scanned into tokens, and the most interesting fifteen tokens, where interesting is measured by how far their spam probability is from a neutral .5, are used to calculate the probability that the mail is spam. If $<a_1, a_2, \ldots, a_{15}>$ is a list of the fifteen individual probabilities, you can calculate the combine these fifteen individual probabilities as:

$$\frac{a_1 \cdot a_2 \cdot \ldots \cdot a_{15}}{a_1 \cdot a_2 \cdot \ldots \cdot a_{15} + (1-a_1)(1-a_2)\ldots(1-a_{15})}$$

One question that arises in practice is what probability to assign to a word you've never seen, i.e. one that doesn't occur in the table of word probabilities. In his approach he has found, by trial and error that .4 is a good number to use. If you've never seen a word before, it is probably fairly innocent; spam words tend to be all too familiar.

Difficulties with the Paul’s Approach
Following are the difficulties with the approach [3]:

i. \textit{Assumption of Independence}: It assumes that the word probabilities are independent which are not as an e-mail having a word “porn” has got high probability of having “sex” as one of the words.

ii. \textit{Dealing with Rare Words}: When we encounter a word in a test mail which has not been encountered during training, then what should be the probability of that word. Here in this case he considers the probability of that word being 0.4. Now, this is some sort of discrepancy as a word which occurred few times has less probability than the word which does not occurred at all!

iii. \textit{Computing of word Probabilities}: The Probabilities as calculated using the above formula doesn’t seem to be probabilities in the strict sense. Since it considers the number of occurrence of a particular word in all the mails, the word may be repeated in a mail as well. So based on its multiple occurrence, in the same mail or the different mails does not really provide a probability of the kind we are interested in.

iv. \textit{Asymmetric}: Paul's approach is subtly asymmetric with respect to how it handles words that indicate "spamminess" compared to how it handles words indicating "non-spamminess". It is less sensitive in one case than the other. Some people feel that this is good for reducing false positives. But since there is evidence both in favor and against, a spam classification for any email, we should get best performance by not lessening our handling both kinds of evidence equally well.
4. Gary Robinson’s Approach

In his approach [4], he tried to address the issues highlighted and take some measures to counteract the effect of these. It first calculates the word probabilities (including the words that hadn’t appeared very often). Then it uses the Fisher’s inverse chi-square test for combining the individual word probabilities and obtains a measure $H$. Since this probability combination inherently favors the hammy words with near 0- probabilities, not effectively considering the about spammy words with probability near 1, he proposes to calculate another measure $S$ which again combine the words probability but this time considering word probabilities as $1 - f(w)$. Finally it uses a useful spamminess indicator $I$ based on these combined probabilities to identify whether a given mail is a spam.

4.1 Generating Word Probabilities

It assumes the existence of a body of e-mails (the corpus) for training, together with software capable of parsing each e-mail into its constituent words. It further assumes that each training e-mail has been classified manually as either "ham" (the e-mail you want to read) or "spam" (the e-mail you don't). Using this data and software, it trains the system by generating a probability for each word that represents its spamminess.

For each word that appears in the corpus, calculate:

- $b(w) = \frac{\text{the number of spam e-mails containing the word } w}{\text{the total number of spam e-mails}}$.
- $g(w) = \frac{\text{the number of ham e-mails containing the word } w}{\text{the total number of ham e-mails}}$.
- $p(w) = \frac{b(w)}{b(w) + g(w)}$

$p(w)$ can be roughly interpreted as the probability that a randomly chosen e-mail containing word $w$ will be a spam. Spam-filtering programs can compute $p(w)$ for every word in an e-mail and use that information as the basis for further calculations to determine whether the e-mail is ham or spam.
4.2 Dealing with Rare Words

There is a problem with probabilities calculated as above when words are very rare. For instance, if a word appears in exactly one e-mail, and it is a spam, the calculated \( p(w) \) is 1.0. But clearly it is not absolutely certain that all future e-mail containing that word will be spam; in fact, we simply don't have enough data to know the real probability.

*How this issue is handled in Robinson’s approach is as follows:*

When exactly one e-mail contains a particular word and that e-mail is spam, our degree of belief that the next time we see that word it will be in a spam is not 100%. That's because we also have our own background information that guides us. We know from experience that virtually any word can appear in either a spam or non-spam context, and that one or a handful of data points is not enough to be completely certain we know the real probability.

The Bayesian approach lets us combine our general background information with the data we have collected for a word in such a way that both aspects are given their proper importance. In this way, we determine an appropriate degree of belief about whether, when we see the word again, it will be in a spam.

We calculate this degree of belief, \( f(w) \), as follows:

\[
f(w) = \frac{(s \times x) + (n \times p(w))}{s + n}
\]

Where

- \( s \) is the strength we want to give to our background information.
- \( x \) 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.
- \( n \) is the number of e-mails we have received that contain word \( w \).

This gives us the convenient use of \( x \) to represent our assumed probability from background information and \( s \) as the strength we will give that assumption. In practice, the values for \( s \) and \( x \) are found through testing to optimize performance. Reasonable starting points are 1 and \(.5\) for \( s \) and \( x \), respectively.
4.3 Combining the Probabilities

So each e-mail is represented by a set of probabilities. To combine these individual probabilities into an overall combined indicator of spamminess or hamminess for the e-mail as a whole, he uses Fisher’s chi square distribution calculations.

The null hypothesis considered is “The $f(w)$s are accurate, and the present e-mail is a random collection of words, each independent of the others, such that the $f(w)$s are not in a uniform distribution”. Then it uses the Fisher calculation to compute an overall probability for the whole set of words. If the e-mail is a ham, it is likely that it will have a number of very low probabilities and relatively few very high probabilities to balance them, with the result that the Fisher calculation will give a very low combined probability. This will allow us to reject the null hypothesis and assume instead the alternative hypothesis that the e-mail is a ham.

Let us call this combined probability $H$:

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

Where $C^{-1}()$ is the inverse chi-square function, used to derive a p-value from a chi-square-distributed random variable.

The explanation of the above exercise is as follows [4]:

We know from the outset the null hypothesis is always false. Virtually no e-mail is actually a random collection of words unbiased with regard to hamminess or spamminess; an e-mail usually has a telling number of words of one type or the other. And certainly words are not independent. Also the $f(w)$s are not in a uniform distribution. But for purposes of spam detection, those departures from reality usually work. They cause the probabilities to have a nonrandom tendency to be either high or low in a given e-mail, giving us a strong statistical basis to reject the null hypothesis in favor of one alternative hypothesis or the other. Either the e-mail is a ham or it is a spam.

The individual $f(w)$s are only approximations to real probabilities (i.e., when there is very little data about a word, our best guess about its spam probability as given by $f(w)$ may not reflect its actual reality). But if you consider how $f(w)$ is calculated, you will see that this uncertainty diminishes asymptotically as $f(w)$ approaches 0 or 1, because such extreme
values can be achieved only by words that have occurred quite frequently in the training
data and either almost always appear in spams or almost always appear in hams. And,
conveniently, it's the numbers near 0 that have by far the greatest impact on the
calculations. To see this, consider the influence on the product .01 * .5 if the first term is
changed to .001, vs. the influence on the product .51 *.5 if the first term is changed to
.501. The Fisher technique is based on multiplying the probabilities. So the null
hypothesis is violated by the fact that the $f(w)$s are not completely reliable, but in a way
that matters vanishingly little for the words of the most interest in the search for evidence
of hamminess: the words with $f(w)$ near 0.

4.4 The Indicator of Hamminess or Spamminess

The calculation described above is sensitive to evidence of hamminess, particularly when
it's in the form of words that show up in far more hams than spams. This is because
probabilities near 0 have a great influence on the product of probabilities, which is at the
heart of Fisher's calculation. However, very spam-oriented words have $f(w)$s near 1, and
therefore have a much less significant effect on the calculations.

An effective technique that has been identified in recent testing efforts to deal with the
above problem is as follows [4]:

First, “reverse” all the probabilities by subtracting them from 1 (that is, for each word,
calculate $1 - f(w)$). Because $f(w)$ represents the probability that a randomly chosen e-mail
from the set of e-mails containing $w$ is a spam, $1 - f(w)$ represents the probability that such
a randomly chosen e-mail will be a ham.

Now do the same Fisher calculation as before, but on the $(1 - f(w))$s rather than on the
$f(w)$s. This will result in near-0 combined probabilities, in rejection of the null hypothesis,
when a lot of very spammy words are present. Call this combined probability $S$.

Now calculate:

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

$I$ is an indicator that is near 1 when the preponderance of the evidence is in favor of the
conclusion that the e-mail is spam and near 0 when the evidence points to the conclusion
that it's ham.
So it a useful characteristic of $I$ that it is near .5 in such cases, just as it is near .5 when there is no particular evidence in one direction or the other. When there is significant evidence in favor of both conclusions, He advocates taking the cautious approach. In real-world testing, human examination of these mid-valued e-mails tends to support the conclusion that they really should be classified somewhere in the middle rather than being subject to the black-or-white approach of most classifiers.

5. Our Approach

In our approach, we have experimented with the above methods and worked out for useful modification in that. Important improvements are as follows:

- In our approach, while calculating the word probabilities, we use stemming to remove redundancies of the words, thus improving the efficiency of the approach.

- Another improvement that has been worked out is elimination of the most commonly occurring words from the training as well as test mails.

- From experimentation, we found that the best results are obtained when we consider a combination of both word and document frequencies for the words to calculate word probabilities.

The basic approach is based on the Gary Robinson’s approach with the above modifications. The word probabilities for spamminess and non-spamminess are been computed more effectively and then they are combined using chi inverse function. Finally the indicator of the spamminess is calculated.

5.1 Calculating the word probabilities

**Computing document frequencies**

- $b(w) = (\text{the number of spam e-mails containing the word } w) / (\text{the total number of spam e-mails})$.

- $g(w) = (\text{the number of ham e-mails containing the word } w) / (\text{the total number of ham e-mails})$.

Then, $d(w) = b(w) / (b(w) + g(w))$
Computing word frequencies

- Let \( x(w) = (\text{the frequency of the word } w \text{ in spam e-mails}) / (\text{the total number of spam e-mails containing } w) \).
- \( y(w) = (\text{the frequency of the word } w \text{ in ham e-mails}) / (\text{the total number of ham e-mails containing } w) \).

Then, \( z(w) = (x(w) - y(w)) / (x(w) + y(w)) \)

Combining above two probabilities to obtain word probability

- \( p(w) = w1 \cdot d(w) + w2 \cdot z(w) \),

Where \( w1, w2 \geq 0 \), \( w1 + w2 = 1 \)

Here \( w1 \) and \( w2 \) are the weights assigned to the \( d(w) \) and \( z(w) \) respectively.

Calculating believed word probability

- \( F(w) = (s \cdot x + n \cdot p(w)) / (s + n) \)

Rest of the calculations for combining probabilities using chi square inverse function and calculation of Indicators are same as that of the Robinson’s approach.

5.2 Corpus Collection:

Corpus collection consists of the mails from our personal mail boxes. Also we have downloaded some of the mails which consist of spams.

5.3 Program Description:

Following are the major components of the spam filter:

Parser: Parser is the component that takes input as e-mail files. While extracting words form the file it ignores the Date field and To field. It also removes the special characters such as double cotes, brackets (<, >, (,), {,}), question mark, asterisks symbols. It considers white spaces as separators for tokenization.
**Stemmer:** The output of the Parser (word extracted) is fed to Stemmer before storing to the dictionary. Stemmer performs stemming on that word. Stemming converts the derived form of words into its basic form. e.g. "working", "worked" are converted to its basic form "work". [6]

**Dictionary:** This is the hash table of words. It contains entries like (Word, spam mails in which word occurred, #ham mails in which word occurred). The hashing is done on the words.

**Tester:** This is the heart of the Spam filter. This takes input as parsed stemmed test mails. Then calculate the probabilities of word being spammy and non-spammmy. It combines the probabilities of all the words. Then it finds the inverse chi-square of the combined probability with 2n degree of freedom. Finally it calculates the indicator I. If I is greater than .55 then it is considered to be spam. I is less than 0.45 then it is considered to be ham. However if I is in range .55 to .45 then we are not sure about that mail and we can not say whether it is spam or ham. We let that mail to be in undetermined category.

**File Reader:** This is the helper class that is used to read various files. This class takes the data from file and stores it as single string, ready to parse.

**Main program: Algorithm**

**Training:**
- Parse the training files and stem the words
- Prepare the Dictionary of these words.

**Testing:**
- Load the dictionary
- For each test file f
  - Stem its words
  - Calculate p(w) for each words
  - Calculate f(w) for each word
  - Calculate Indicators H, S and finally I
  - If (Indicator is >0.55) then test file f is spam
  - If (Indicator is <0.45) then test file f is ham
  - Else test file f is undetermined

End
6. Measures to evaluate classification performance

In classification tasks, performance can be measured by following parameters [5]:

- **Accuracy and Error rate,**

Let $N_{\text{legit}}$ and $N_{\text{spam}}$ be the total numbers of legitimate and spam messages, respectively, to be classified by the filter, and $N_{Y \rightarrow Z}$ the number of messages belonging to category $Y$ that the filter classified as belonging to category $Z$ ($Y,Z \in \{\text{legit, spam}\}$). Then

\[
\text{Accuracy} = \frac{N_{\text{legit} \rightarrow \text{legit}} + N_{\text{spam} \rightarrow \text{spam}}}{N_{\text{legit}} + N_{\text{spam}}}
\]

\[
\text{Error} = \frac{N_{\text{legit} \rightarrow \text{spam}} + N_{\text{spam} \rightarrow \text{legit}}}{N_{\text{legit}} + N_{\text{spam}}}
\]

- **Recall and Precision rate**

Experimental results are also presented in terms of *spam recall* ($SR$) and *spam precision* ($SP$):

\[
SR = \frac{N_{\text{spam} \rightarrow \text{spam}}}{N_{\text{spam}}}, \quad SP = \frac{N_{\text{spam} \rightarrow \text{spam}}}{N_{\text{spam} \rightarrow \text{spam}} + N_{\text{legit} \rightarrow \text{spam}}}
\]

Spam recall measures the percentage of spam messages that the filter manages to block (intuitively its effectiveness), while spam precision measures the degree to which the blocked messages are indeed spam (the filter’s safety).

- **False Positive and False Negative**

\[
FP = \frac{N_{\text{legit} \rightarrow \text{spam}}}{N_{\text{legit}}}, \quad FN = \frac{N_{\text{spam} \rightarrow \text{legit}}}{N_{\text{spam}}}
\]
7. The Observations and Tables

The Training was done on the Dataset having 288 hams and 600 spams

The testing was done on the Dataset having 1200 hams and 1771 spams.

Table 1: Gary Robinson's approach, with common words included in the calculations

<table>
<thead>
<tr>
<th>X</th>
<th>CCI ham</th>
<th>ICI ham</th>
<th>UD ham</th>
<th>CCI spam</th>
<th>ICI spam</th>
<th>UD spam</th>
<th>Accuracy</th>
<th>Error</th>
<th>Spam R</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>951</td>
<td>19</td>
<td>230</td>
<td>1108</td>
<td>30</td>
<td>33</td>
<td>0.965415</td>
<td>0.034585</td>
<td>0.946</td>
</tr>
<tr>
<td>0.35</td>
<td>838</td>
<td>19</td>
<td>343</td>
<td>1116</td>
<td>22</td>
<td>33</td>
<td>0.96879</td>
<td>0.03121</td>
<td>0.95303</td>
</tr>
<tr>
<td>0.37</td>
<td>819</td>
<td>20</td>
<td>361</td>
<td>1085</td>
<td>32</td>
<td>54</td>
<td>0.955293</td>
<td>0.044707</td>
<td>0.92655</td>
</tr>
<tr>
<td>0.4</td>
<td>677</td>
<td>22</td>
<td>501</td>
<td>1117</td>
<td>18</td>
<td>36</td>
<td>0.967946</td>
<td>0.032054</td>
<td>0.95388</td>
</tr>
<tr>
<td>0.45</td>
<td>523</td>
<td>26</td>
<td>651</td>
<td>1121</td>
<td>16</td>
<td>34</td>
<td>0.967946</td>
<td>0.032054</td>
<td>0.95730</td>
</tr>
</tbody>
</table>

[The abbreviations in headings of the table are expanded in Appendix]

Table 2: Gary Robinson's approach, with common words not included in the calculations

<table>
<thead>
<tr>
<th>X</th>
<th>CCI ham</th>
<th>ICI ham</th>
<th>UD ham</th>
<th>CCI spam</th>
<th>ICI spam</th>
<th>UD spam</th>
<th>Accuracy</th>
<th>Error</th>
<th>Spam R</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>1195</td>
<td>3</td>
<td>2</td>
<td>990</td>
<td>140</td>
<td>41</td>
<td>0.922396</td>
<td>0.077604</td>
<td>0.84541</td>
</tr>
<tr>
<td>0.35</td>
<td>1167</td>
<td>5</td>
<td>28</td>
<td>1020</td>
<td>87</td>
<td>63</td>
<td>0.934205</td>
<td>0.065373</td>
<td>0.87105</td>
</tr>
<tr>
<td>0.37</td>
<td>1104</td>
<td>7</td>
<td>89</td>
<td>1029</td>
<td>63</td>
<td>79</td>
<td>0.937157</td>
<td>0.062843</td>
<td>0.878736</td>
</tr>
<tr>
<td>0.4</td>
<td>900</td>
<td>13</td>
<td>287</td>
<td>1057</td>
<td>44</td>
<td>70</td>
<td>0.946436</td>
<td>0.053564</td>
<td>0.902647</td>
</tr>
<tr>
<td>0.45</td>
<td>454</td>
<td>22</td>
<td>724</td>
<td>1096</td>
<td>17</td>
<td>58</td>
<td>0.959089</td>
<td>0.040911</td>
<td>0.935952</td>
</tr>
</tbody>
</table>

Observations:

1. The accuracy of this approach improves when common words are included.
2. But false positives are increases when the common words are included.
3. Best Accuracy = 96.79 % at 1.8 % False positive (when common words are included), however UD ham is are very high
4. Best Accuracy = 95.9 % at 1.8 % False positive (when common words are excluded), however UD ham is increasing drastically
5. In above both tables we can see that the UD ham values very high and are increasing. This shows that there is formation of another cluster for the undetermined mails in addition to the ham and spam clusters which is not desirable.
Table 3: Our approach, with common words included in the calculations (effect of varying the value of s) x=0.37, w1= .04, w2=0.6

<table>
<thead>
<tr>
<th>S</th>
<th>CCI ham</th>
<th>ICI ham</th>
<th>UD ham</th>
<th>CCI spam</th>
<th>ICI spam</th>
<th>UD spam</th>
<th>Accuracy</th>
<th>Error</th>
<th>Spam R</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>1136</td>
<td>14</td>
<td>50</td>
<td>1107</td>
<td>13</td>
<td>51</td>
<td>0.967102</td>
<td>0.032898</td>
<td>0.945346</td>
</tr>
<tr>
<td>0.2</td>
<td>1142</td>
<td>14</td>
<td>44</td>
<td>1110</td>
<td>13</td>
<td>48</td>
<td>0.968368</td>
<td>0.031632</td>
<td>0.947908</td>
</tr>
<tr>
<td>0.3</td>
<td>1151</td>
<td>15</td>
<td>34</td>
<td>1111</td>
<td>13</td>
<td>47</td>
<td>0.968368</td>
<td>0.031632</td>
<td>0.948762</td>
</tr>
<tr>
<td>0.4</td>
<td>1154</td>
<td>15</td>
<td>31</td>
<td>1111</td>
<td>12</td>
<td>48</td>
<td>0.968368</td>
<td>0.031632</td>
<td>0.948762</td>
</tr>
<tr>
<td>0.5</td>
<td>1156</td>
<td>17</td>
<td>27</td>
<td>1109</td>
<td>12</td>
<td>50</td>
<td>0.966681</td>
<td>0.033319</td>
<td>0.947054</td>
</tr>
<tr>
<td>0.6</td>
<td>1156</td>
<td>17</td>
<td>27</td>
<td>1107</td>
<td>12</td>
<td>53</td>
<td>0.965837</td>
<td>0.034585</td>
<td>0.945346</td>
</tr>
<tr>
<td>0.7</td>
<td>1155</td>
<td>17</td>
<td>28</td>
<td>1105</td>
<td>11</td>
<td>55</td>
<td>0.967994</td>
<td>0.035006</td>
<td>0.943638</td>
</tr>
<tr>
<td>0.8</td>
<td>1155</td>
<td>17</td>
<td>28</td>
<td>1105</td>
<td>11</td>
<td>55</td>
<td>0.964994</td>
<td>0.035006</td>
<td>0.943638</td>
</tr>
<tr>
<td>0.9</td>
<td>1155</td>
<td>17</td>
<td>28</td>
<td>1106</td>
<td>11</td>
<td>54</td>
<td>0.965415</td>
<td>0.034585</td>
<td>0.944492</td>
</tr>
</tbody>
</table>

Observations:
1. Accuracy is maximum for s=0.7 (Accuracy=96.79%)
2. Results do not vary much with value of s.

Table 4: Our approach, with common words included in the calculations (effect of varying x) x=0.37, w1=0.4. w2=0.6

<table>
<thead>
<tr>
<th>X</th>
<th>CCI ham</th>
<th>ICI ham</th>
<th>UD ham</th>
<th>CCI spam</th>
<th>ICI spam</th>
<th>UD spam</th>
<th>Accuracy</th>
<th>Error</th>
<th>Spam R</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>1177</td>
<td>1</td>
<td>22</td>
<td>997</td>
<td>79</td>
<td>95</td>
<td>0.926191</td>
<td>0.073809</td>
<td>0.85140</td>
</tr>
<tr>
<td>0.2</td>
<td>1174</td>
<td>3</td>
<td>23</td>
<td>1048</td>
<td>57</td>
<td>66</td>
<td>0.946858</td>
<td>0.053142</td>
<td>0.89496</td>
</tr>
<tr>
<td>0.3</td>
<td>1165</td>
<td>7</td>
<td>28</td>
<td>1085</td>
<td>31</td>
<td>55</td>
<td>0.960776</td>
<td>0.039224</td>
<td>0.92655</td>
</tr>
<tr>
<td>0.37</td>
<td>1154</td>
<td>12</td>
<td>34</td>
<td>1107</td>
<td>13</td>
<td>51</td>
<td>0.967946</td>
<td>0.032054</td>
<td>0.943734</td>
</tr>
<tr>
<td>0.4</td>
<td>1151</td>
<td>18</td>
<td>31</td>
<td>1110</td>
<td>10</td>
<td>51</td>
<td>0.966681</td>
<td>0.033319</td>
<td>0.947908</td>
</tr>
<tr>
<td>0.5</td>
<td>1112</td>
<td>38</td>
<td>50</td>
<td>1124</td>
<td>7</td>
<td>40</td>
<td>0.96415</td>
<td>0.03585</td>
<td>0.959865</td>
</tr>
<tr>
<td>0.6</td>
<td>1034</td>
<td>92</td>
<td>74</td>
<td>1130</td>
<td>5</td>
<td>36</td>
<td>0.943906</td>
<td>0.056094</td>
<td>0.964986</td>
</tr>
<tr>
<td>0.7</td>
<td>885</td>
<td>213</td>
<td>102</td>
<td>1135</td>
<td>4</td>
<td>32</td>
<td>0.894981</td>
<td>0.105019</td>
<td>0.969258</td>
</tr>
<tr>
<td>0.8</td>
<td>579</td>
<td>448</td>
<td>173</td>
<td>1138</td>
<td>2</td>
<td>31</td>
<td>0.797132</td>
<td>0.202868</td>
<td>0.971841</td>
</tr>
<tr>
<td>0.9</td>
<td>199</td>
<td>738</td>
<td>263</td>
<td>1143</td>
<td>1</td>
<td>27</td>
<td>0.67693</td>
<td>0.32307</td>
<td>0.97608</td>
</tr>
</tbody>
</table>

Observations:
1. Accuracy is maximum for x=0.37 (Accuracy=96.79%)
2. Results heavily depend on value of parameter x.
Table 5: Our approach, with common words included in the calculations (effect of varying w1 & w2) s=0.7, x=0.37

<table>
<thead>
<tr>
<th>w1</th>
<th>W2</th>
<th>CCI ham</th>
<th>ICI ham</th>
<th>UD ham</th>
<th>CCI spam</th>
<th>ICI spam</th>
<th>UD spam</th>
<th>Accuracy</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>792</td>
<td>28</td>
<td>380</td>
<td>1102</td>
<td>10</td>
<td>59</td>
<td>0.959089</td>
<td>0.040911</td>
</tr>
<tr>
<td>0.8</td>
<td>0.2</td>
<td>1115</td>
<td>19</td>
<td>66</td>
<td>1100</td>
<td>14</td>
<td>57</td>
<td>0.962041</td>
<td>0.037959</td>
</tr>
<tr>
<td>0.6</td>
<td>0.4</td>
<td>1158</td>
<td>15</td>
<td>27</td>
<td>1102</td>
<td>17</td>
<td>52</td>
<td>0.964572</td>
<td>0.035428</td>
</tr>
<tr>
<td>0.4</td>
<td>0.6</td>
<td>1156</td>
<td>11</td>
<td>33</td>
<td>1105</td>
<td>18</td>
<td>48</td>
<td>0.967524</td>
<td>0.032476</td>
</tr>
<tr>
<td>0.2</td>
<td>0.8</td>
<td>1140</td>
<td>10</td>
<td>50</td>
<td>1084</td>
<td>17</td>
<td>70</td>
<td>0.959089</td>
<td>0.040911</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1029</td>
<td>12</td>
<td>159</td>
<td>1079</td>
<td>11</td>
<td>81</td>
<td>0.956137</td>
<td>0.043863</td>
</tr>
</tbody>
</table>

Observations:

1. The Accuracy is not very much affected by varying the value of s.
2. The Accuracy depends heavily on the value of x.
3. Also weight factor is needed to be tuned for better performance.
4. The Accuracy is maximum when w1=0.4 and w2=0.6 with False positive 0.91% (= 96.75% best results when common words included in the calculations) 
5. The Accuracy is maximum when x is set to 0.37.
6. Thus we can say that when parameters are set to following values we can get the best results s=0.7, x=0.37, w1=0.4, w2=0.6

Table 6: Our approach with common words are not included in the calculations with x=0.7, w1=0.8, w2=0.2

<table>
<thead>
<tr>
<th>S</th>
<th>C</th>
<th>CCI ham</th>
<th>ICI ham</th>
<th>UD ham</th>
<th>CCI spam</th>
<th>ICI spam</th>
<th>UD spam</th>
<th>Accuracy</th>
<th>Error</th>
<th>Spam</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>1161</td>
<td>1</td>
<td>38</td>
<td>1095</td>
<td>1</td>
<td>75</td>
<td>0.967524</td>
<td>0.032476</td>
<td>0.93</td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td>1162</td>
<td>2</td>
<td>36</td>
<td>1102</td>
<td>1</td>
<td>68</td>
<td>0.970055</td>
<td>0.029945</td>
<td>0.94</td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>1160</td>
<td>3</td>
<td>37</td>
<td>1105</td>
<td>2</td>
<td>64</td>
<td>0.970898</td>
<td>0.029102</td>
<td>0.94</td>
<td></td>
</tr>
<tr>
<td>0.4</td>
<td>1159</td>
<td>3</td>
<td>38</td>
<td>1112</td>
<td>4</td>
<td>55</td>
<td>0.973851</td>
<td>0.026149</td>
<td>0.94</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>1154</td>
<td>5</td>
<td>41</td>
<td>1114</td>
<td>4</td>
<td>53</td>
<td>0.973851</td>
<td>0.026149</td>
<td>0.95</td>
<td></td>
</tr>
<tr>
<td>0.6</td>
<td>1154</td>
<td>5</td>
<td>41</td>
<td>1117</td>
<td>4</td>
<td>50</td>
<td>0.975116</td>
<td>0.024884</td>
<td>0.95</td>
<td></td>
</tr>
<tr>
<td>0.7</td>
<td>1155</td>
<td>6</td>
<td>39</td>
<td>1118</td>
<td>4</td>
<td>49</td>
<td>0.975116</td>
<td>0.024884</td>
<td>0.95</td>
<td></td>
</tr>
<tr>
<td>0.8</td>
<td>1156</td>
<td>8</td>
<td>36</td>
<td>1120</td>
<td>3</td>
<td>48</td>
<td>0.975116</td>
<td>0.024884</td>
<td>0.95</td>
<td></td>
</tr>
<tr>
<td>0.9</td>
<td>1154</td>
<td>9</td>
<td>37</td>
<td>1121</td>
<td>2</td>
<td>48</td>
<td>0.975116</td>
<td>0.024884</td>
<td>0.95</td>
<td></td>
</tr>
</tbody>
</table>

Observations:

1. The Accuracy is not much affected by varying the value of s
2. The Best Accuracy is 97.51% at s= 0.6 (with minimum false positive)
Table 7: Our approach with common words are not included in the calculations with $s=0.9$, $w_1=0.8$, $w_2=0.2$

<table>
<thead>
<tr>
<th>X</th>
<th>CCI ham</th>
<th>ICI ham</th>
<th>UD ham</th>
<th>CCI spam</th>
<th>ICI spam</th>
<th>UD spam</th>
<th>Accuracy</th>
<th>Error</th>
<th>Spam Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>1200</td>
<td>0</td>
<td>0</td>
<td>476</td>
<td>287</td>
<td>408</td>
<td>0.706875</td>
<td>0.293125</td>
<td>0.4064</td>
</tr>
<tr>
<td>0.2</td>
<td>1200</td>
<td>0</td>
<td>0</td>
<td>702</td>
<td>226</td>
<td>243</td>
<td>0.802193</td>
<td>0.197807</td>
<td>0.5994</td>
</tr>
<tr>
<td>0.3</td>
<td>1200</td>
<td>0</td>
<td>0</td>
<td>861</td>
<td>170</td>
<td>140</td>
<td>0.869253</td>
<td>0.130747</td>
<td>0.7352</td>
</tr>
<tr>
<td>0.4</td>
<td>1198</td>
<td>0</td>
<td>2</td>
<td>967</td>
<td>121</td>
<td>83</td>
<td>0.91936</td>
<td>0.08604</td>
<td>0.8257</td>
</tr>
<tr>
<td>0.5</td>
<td>1197</td>
<td>1</td>
<td>2</td>
<td>1023</td>
<td>64</td>
<td>84</td>
<td>0.937157</td>
<td>0.062843</td>
<td>0.8736</td>
</tr>
<tr>
<td>0.6</td>
<td>1183</td>
<td>2</td>
<td>15</td>
<td>1092</td>
<td>22</td>
<td>57</td>
<td>0.965837</td>
<td>0.034163</td>
<td>0.9323</td>
</tr>
<tr>
<td>0.7</td>
<td>1154</td>
<td>9</td>
<td>37</td>
<td>1121</td>
<td>2</td>
<td>48</td>
<td>0.975116</td>
<td>0.024884</td>
<td>0.9573</td>
</tr>
<tr>
<td>0.8</td>
<td>798</td>
<td>57</td>
<td>345</td>
<td>1139</td>
<td>0</td>
<td>32</td>
<td>0.962463</td>
<td>0.037537</td>
<td>0.9726</td>
</tr>
<tr>
<td>0.9</td>
<td>18</td>
<td>173</td>
<td>1009</td>
<td>1147</td>
<td>0</td>
<td>24</td>
<td>0.916913</td>
<td>0.083087</td>
<td>0.9795</td>
</tr>
</tbody>
</table>

Observations:

1. The Accuracy is maximum at $x=0.7$ ($=97.51\%$)
2. Variation is $x$ has heavy impact on accuracy, UD ham, UD spam and most importantly on false positive.

Table 8: Our approach with common words are not included in the calculations with $s=0.6$ and $x=0.7$

<table>
<thead>
<tr>
<th>$w_1$</th>
<th>$w_2$</th>
<th>CCI ham</th>
<th>ICI ham</th>
<th>UD ham</th>
<th>CCI spam</th>
<th>ICI spam</th>
<th>UD spam</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1000</td>
<td>77</td>
<td>123</td>
<td>1153</td>
<td>0</td>
<td>18</td>
<td>0.959933</td>
</tr>
<tr>
<td>0.8</td>
<td>0.2</td>
<td>1154</td>
<td>5</td>
<td>41</td>
<td>1117</td>
<td>4</td>
<td>50</td>
<td>0.975116</td>
</tr>
<tr>
<td>0.6</td>
<td>0.4</td>
<td>1159</td>
<td>3</td>
<td>38</td>
<td>1071</td>
<td>8</td>
<td>92</td>
<td>0.956558</td>
</tr>
<tr>
<td>0.4</td>
<td>0.6</td>
<td>1149</td>
<td>2</td>
<td>49</td>
<td>1025</td>
<td>3</td>
<td>143</td>
<td>0.937579</td>
</tr>
<tr>
<td>0.2</td>
<td>0.8</td>
<td>1115</td>
<td>2</td>
<td>83</td>
<td>1016</td>
<td>8</td>
<td>147</td>
<td>0.933783</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1005</td>
<td>3</td>
<td>192</td>
<td>1011</td>
<td>5</td>
<td>155</td>
<td>0.931253</td>
</tr>
</tbody>
</table>

Observations:

1. Change in values of weights $w_1$ and $w_2$ changes the results significantly.
2. The Accuracy is maximum at $w_1=0.8$ and $w_2=0.2$ ($=97.51\%$) with False positive 0.3 % (best results in this project)
3. Over all false positive and false negative are minimum in this case.
4. From above 3 results, we can say that the accuracy is highest for following values of the parameters: $s=0.6$, $x=0.7$, $w_1=0.8$, $w_2=0.2$
5. Our approach clearly reducing the indeterminism of classification in spam as well as non-spam mails at the same time improving accuracy compared to Robinson’s approach. This results into significant improvement of the filtering the mails.
Clustering effect in Robinson’s approach Vs. clustering in our approach

Figure 1 Clusters of Gary Robinson’s Method [Not a good clustering]

Figure 2 : clusters of our Method (considering common words) [Red points are still spread]
Observation: we have got very good clustering in third graph (when we do not consider commonly occurring words), thus reducing the data points in undetermined category.

**Comparison: considering commonly occurring words vs. not considering them.**

Following graph shows how the behaviors of the filter works when tested on data in which we have considered commonly occurring words (green curve) and when tested on data in which we have remove commonly occurring words (red). We can observe that we get red curve always bellow. That is the errors made by filter when we removed commonly occurring words lower.
8. Conclusions

Our aim was
1. To improve the accuracy
2. To reduce the false positives
3. To get good and clear clustering., i.e. to reduce the fall of the samples in undetermined category (UD spam and UD ham category)

For this purpose we modified the Gray Robinson’s formula to add the effect of the word frequencies in training data.

The Training was done on the Dataset having 288 hams and 600 spams
The testing was done on the Dataset having 1200 hams and 1771 spams.

The Results
1. The performance increases if we consider both word frequencies and document frequencies for word probability calculation.
2. When commonly occurring words are removed then accuracy increased compared to case when common words are not removed.
3. Clustering, as can be seen from the above figures, shows that we get better clustering effect in our method.
4. False positive are reduced considerably.
5. The procedure needs heavy exercise for tuning the parameters viz. s, x, w1 and w2.
6. The spam precision in our method is better than that of the Robinson’s method.
7. Accuracy has increased by around 1%.
8. Our approach clearly reducing the indeterminism of classification in spam as well as non-spam mails at the same time improving accuracy compared to Robinson’s approach. This results into significant improvement of the filtering the mails.

Facts Experienced
1. The filter when tested on data from same corpus as that of training then the results are significantly better that when we test on some other corpus. This shows that filter built with our method should be trained and tested on same corpus data. In other words the filter should be customized for different mail-boxes differently.
2. As the quantity of the data from which dictionary is built changes parameter s (s, x, w1 and w2 )should be tuned.

9. Future Scope

We have considered individual words as the tokens. However considering phrases can improve the results. Also if context information is present then the performance is expected to improve tremendously.
The filter can be made adaptive. For this user feed-back is considered. User can give feedback on false positive and false negatives. Then using feed-back filter can be made to learn to classify more correctly on its own. For this we can modify the dictionary online rather than offline.
10. References


11. Appendix

How to use the project software

To run the files in the directory, follow given steps

1) To create new dictionary of words
   a) with commonly occurring words (given in "common" file) included
      
      ```perl
      perl parse.pl training_ham_dir_name training_spam_dir_name name_of_dictionary_file
      ```
   b) with commonly words excluded from dictionary
      
      ```perl
      perl rem_common_parse.pl training_ham_dir_name training_spam_dir_name name_of_dictionary_file
      ```
   Note: if you want to use provided files (with_common_dictionary or rem_common_dictionary) then this step is redundant

2) To stem the testing ham and spam files
   a) with commonly occurring words (given in "common" file) included in stemmed data
      
      ```perl
      perl process_test.pl dir stemmed_dir
      ```
   b) with commonly occurring words (given in "common" file) excluded in stemmed data
      
      ```perl
      perl rem_common_process_test.pl dir stemmed_dir
      ```
   Note: apply above step to both test_spam_dir and test_ham_dir to get stemmed data in respective directories

3) To get the Results on the test data

   There is now one dictionary file
   add dictionary files name in the MyFile.java's loadDictionary() functions first line

   There are two directories containing stemmed spam data and stemmed ham data (after step 2)
   in main function of finalTester.java add these names to the two function calls t.testDir()
   Compile both MyFile.java and finalTester.java
   run the finalTester with arguments as follows
   
   ```java
   java finalTester s x w1 w2
   ```
   here s x w1 and w2 are double values.
   s is our belief factor
   x is spamminess probability value to assign to a unseen word
   w1 and w2 are the parameters that need to be given, w1+w2 should be equal to 1 and w1 and w2 should be greater than 0.0
Abbreviations used

S – Strength of belief
X – Prior probability of an unseen word
W1 - Weight given to the probability, calculated using document frequency.
W2 - Weight given to the probability, calculated using word frequency.
CCI ham – Correctly Classified Instances of ham (i.e. non spam mails)
UD ham – Undetermined hams
CCI spam – Correctly Classified Instances of Spam
UD spam – Undetermined spam
Spam R – Spam Recall
Spam P – Spam Precision.