Spam Filtering based on Naive Bayes Classification

Tianhao Sun

May 1, 2009

Abstract

This project discusses about the popular statistical spam filtering process: naive Bayes classification. A fairly famous way of implementing the naive Bayes method in spam filtering by Paul Graham is explored and a adjustment of this method from Tim Peter is evaluated based on applications on real data. Two solutions to the problem of unknown tokens are also tested on the sample emails. The last part of the project shows how the Bayesian noise reduction algorithm can improve the accuracy of the naive Bayes classification.

Declaration

This piece of work is a result of my own work except where it forms an assessment based on group project work. In the case of a group project, the work has been prepared in collaboration with other members of the group. Material from the work of others not involved in the project has been acknowledged and quotations and paraphrases suitably indicated.

Contents

1 Introduction **3**

2 Naive Bayes Classification **5**
2.1 Training data 5
2.2 Filtering process 7
2.3 Application 10

3 Paul Graham's Method **12**
3.1 Filtering process 12
3.2 Application 15
3.3 Tim Peter's criticism 19

4 Unknown Tokens **24**
4.1 Modified sample emails 24
4.2 Problem 24
4.3 Solution 25
4.4 Application 27

5 Bayesian Noise Reduction **33**
5.1 Introduction 33
5.2 Zdziarski's BNR algorithm 33
5.3 Application 35

6 Conclusion **39**

Chapter 1

Introduction

In this project, I investigate one of the widely used statistical spam filters, Bayesian spam filters. The first known mail-filtering program to use a Bayes classifier was Jason Rennie's iFile program, released in 1996. The first scholarly publication on Bayesian spam filtering was by Sahami et al. (1998)[12]. A naive Bayes classifier[3] simply apply Bayes' theorem on the context classification of each email, with a strong assumption that the words included in the email are independent to each other. In the beginning, we will get two sample emails from the real life data in order to create the training dataset. Then a detailed filtering process of the naive Bayes classification will be explained and we will apply this method on our sample emails for testing in the end of the chapter. These sample emails will be tested throughout the project using the other methods we will discuss about later.

Among all the different ways of implementing naive Bayes classification, Paul Graham's approach has become fairly famous[2]. He introduced a new formula for calculating token values and overall probabilities of an email being classified as spam. But there is a unrealistic assumption in his formula which assume the number of spam and ham are equal in the dataset for everyone. Where Tim Peter correctly adjust the formula later to make it fit into all datasets[7], and both methods will be evaluated using our training dataset.

Also two different solutions to the problem of unknown tokens will be discussed in this project, the simple one is introduced by Paul Graham and the more accurate approach is created by Gary Robinson[10]. Where Robinson's method lets us consider both our background knowledge and the limited data we have got from unknown tokens in the prediction of their token values. Again, their performances will be measured against our sample emails.

Then, we will discuss about a major improvement on the Bayesian filtering process: Bayesian noise reduction. Jonathan A. Zdziarski first discussed about the Bayesian noise reduction algorithm in 2004[15], and also include

this idea in his book later[16, p. 231]. The BNR algorithm is particularly useful in making the filtering process more effective and gain a higher level of confidence in the classification.

Chapter 2

Naive Bayes Classification

2.1 Training data

Here I want to introduce some email examples to demonstrate the process of naive Bayes classification. Each of them contains 15 words or tokens in the filtering way of saying, and all of them will be included in the training data table later on.

- Example for spam:

  Paying too much for VIAGRA?

  Now,you have a chance to receive a FREE TRIAL!

- Example for ham:

  For the Clarins, just take your time. As i have advised her to exercise regularly.

The following table is made to show the frequency of a token appear in both email groups, the appearances of each token count only once for each email received. I estimated these word counts based on some previous work done by Vasanth Elavarasan[1] and the book written by Jonathan A. Zdziarski[16, p. 65]. Due to the email group size for Zdziarski's word counts list is smaller than Elanvarasan's, I only took the ratio factor into account and reproduced the new word counts according to our dataset. For example, tokens such as The, Vehicle and Viagra are not included in Elavarasan's word list. In order to make use of Zdziarski's data, I simply multiply the ratio in his dataset by the total number in the new table. Here is how I get the new frequencies for these three tokens.(Table 2.1) Note the training dataset here has a total number of 432 spam and 2170 ham.(Table 2.2)

The	$96/224 \times 432 \approx 185$	$48/112 \times 2170 \approx 930$
Vehicle	$11/224 \times 432 \approx 21$	$3/112 \times 2170 \approx 58$
Viagra	$20/224 \times 432 \approx 39$	$1/112 \times 2170 \approx 19$

Table 2.1: Table of estimated data

Feature	Appearances in Spam	Appearances in Ham
A	165	1235
Advised	12	42
As	2	579
Chance	45	35
Clarins	1	6
Exercise	6	39
For	378	1829
Free	253	137
Fun	59	9
Girlfriend	26	8
Have	291	2008
Her	38	118
I	9	1435
Just	207	253
Much	126	270
Now	221	337
Paying	26	10
Receive	171	98
Regularly	9	87
Take	142	287
Tell	76	89
The	185	930
Time	212	446
To	389	1948
Too	56	141
Trial	26	13
Vehicle	21	58
Viagra	39	19
You	391	786
Your	332	450

Table 2.2: Table of training data

2.2 Filtering process

Firstly, we break the email we want to classify into a group of individual words $w_1, \ldots, w_n$, and denote this email as a capital letter E. The probability of receiving the email E is equal to the probability of receiving the list of words $w_1, \ldots, w_n$.

$$P(E) = P(w_1, \ldots, w_n) \tag{2.1}$$

But a strict application of the Bayes theorem requires a list of all the possible combinations of the probabilities of words in the email. In order to calculate the value of $P(w_1, \ldots, w_n)$, we need a huge training dataset if the email contains a big number of words(n is big). Even if we have a reasonable large dataset, it is still very unlikely that we can find all the possible combinations everytime we receive a new email. Therefore we need to make a big assumption that the words in the email are independent of each other and randomly displayed in the email[5]. In other words, we assume the probability of a chosen word being classified as a spammy word does not affect the probabilities of other words being spammy. The purpose of assumption is to simplify the computations of probabilities in the Bayes formula and reduce the size of the training dataset we need. Despite this assumption is clearly false in most real-world cases, naive Bayes often performs classification very well. The reason is correct probability estimation is not necessary for accurate prediction in practice, the key issue that we are looking for is only the order of the probabilities[8]. Therefore by this naive Bayes assumption, (2.1) becomes:

$$P(w_1, \ldots, w_n) = \prod_{i=1}^{n} P(w_i)$$

Next, let's denote the two classes of emails as spam(S) and ham(H). What we need to know next are the two probabilities $P(E|S)$ (probability of given a email from email class S that it is the email E) and $P(E|H)$ (probability of given a email from email class H that it is the email E), and we have to make a assumption of conditional independence[9]. The assumption here is that the appearance of a particular token w_i is statically independent of the appearance of any other tokens $w_j, j \neq i$, given that we have either a spam or a ham. Then we can express $P(E|S)$ and $P(E|H)$ as:

$$\begin{aligned} P(E|S) &= P(w_1, \ldots, w_n|S) \\ &= \prod_{i=1}^{n} P(w_i|S) \end{aligned}$$

and

$$\begin{aligned} P(E|H) &= P(w_1, \ldots, w_n|H) \\ &= \prod_{i=1}^{n} P(w_i|H) \end{aligned}$$

The reason that we set up a training dataset is to estimate how *spammy* each word is, where probabilities $P(w_i|S)$(probability of given a email from email class S which it contains the word w_i) and $P(w_i|H)$(probability of given a email from email class H which it contains the word w_i) are needed. They denote the conditional probability that a given email contains the word w_i under the assumption that this email is spam or ham respectively. We estimate these probabilities by calculating the frequencies of the words appear in either groups of emails from the training dataset. In the following formula, $P(w_i \cap S)$ is the probability that a given email is a spam email and contains the word w_i. Thus, by Bayes theorem:

$$P(w_i|S) = \frac{P(w_i \cap S)}{P(S)} \qquad and \qquad P(w_i|H) = \frac{P(w_i \cap H)}{P(H)}$$

For example,the word *Free* in our training dataset:

$$P(w_8 \cap S) = \frac{253}{432 + 2170}$$

and

$$P(S) = \frac{432}{432 + 2170}$$

thus, we get

$$P(w_8|S) = \frac{253}{432}$$

The following step is to compute the posterior probability of spam email given the overall probability of the sampling email by Bayes' rule, this is the crucial part of the entire classification.

$$\begin{aligned} P(S|E) &= \frac{P(E|S)P(S)}{P(E)} \\ &= \frac{P(S)\prod_{i=1}^{n} P(w_i|S)}{P(E)} \end{aligned} \tag{2.2}$$

and similarly,

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)} \tag{2.3}$$
$$= \frac{P(H)\prod_{i=1}^{n} P(w_i|H)}{P(E)}$$

Therefore we can classify the email by comparing the probabilities of $P(S|E)$(probability of a given email is classified as spam which belongs to the email class S) and $P(H|E)$(probability of a given email is classified as ham which belongs to the email class H). Firstly, we find the ratio of these two probabilities. In the equations below, the denominators $P(E)$ from (2.2) and (2.3) cancel out each other.

$$\frac{P(S|E)}{P(H|E)} = \frac{P(E|S)P(S)}{P(E|H)P(H)}$$
$$= \frac{P(S)\prod_{i=1}^{n} P(w_i|S)}{P(H)\prod_{i=1}^{n} P(w_i|H)}$$
$$= \frac{P(S)}{P(H)}\prod_{i=1}^{n}\frac{P(w_i|S)}{P(w_i|H)}$$

But there is a problem here, the products in the above equations can be extremely small values if we have a big amount of words w_i. To overcome this issue, we apply log to the probability ratio.

$$\log\frac{P(S|E)}{P(H|E)} = \log\left(\frac{P(S)}{P(H)}\prod_{i=1}^{n}\frac{P(w_i|S)}{P(w_i|H)}\right)$$
$$= \log\frac{P(S)}{P(H)} + \sum_{i=1}^{n}\log\frac{P(w_i|S)}{P(w_i|H)} \tag{2.4}$$

At this point, we can use equation (2.4) to calculate the log posterior probability when we receive a new email. If the result is greater than zero(which means $P(S|E) > P(H|E)$), we classify email E as spam. Similarly, we classify the email as ham if it is less than zero(which means $P(S|E) < P(H|E)$).

2.3 Application

Once we have the naive Bayes classifier ready, we can investigate how spammy each word is based on the training dataset. Here is the new table with the columns $P(w_i|S)$ and $P(w_i|H)$ added on. (Table 2.3)

With a total number of 432 spam and 2170 ham. We can calculate the values of $P(S)$ and $P(H)$:

$$P(S) = \frac{432}{432 + 2170} = 0.1660261$$

$$P(H) = \frac{2170}{432 + 2170} = 0.8339739$$

For our sample emails, example of spam ES contains 14 different words $w_i, i = 1, \ldots, n$ and example of ham EH contains 15 different words $w_j, j = 1, \ldots, m$. We apply the naive Bayes classification as follow:

$$\begin{aligned}
\log \frac{P(S|ES)}{P(H|ES)} &= \log \frac{P(S)}{P(H)} + \sum_{i=1}^{n} \log \frac{P(w_i|S)}{P(w_i|H)} \\
&= \log \frac{0.1660261}{0.8339739} + \left[\log \frac{0.0601852}{0.0046083} + \cdots + \log \frac{0.0208333}{0.0059908}\right] \\
&= 14.8341602
\end{aligned}$$

$$\begin{aligned}
\log \frac{P(S|EH)}{P(H|EH)} &= \log \frac{P(S)}{P(H)} + \sum_{i=1}^{m} \log \frac{P(w_j|S)}{P(w_j|H)} \\
&= \log \frac{0.1660261}{0.8339739} + \left[\log \frac{0.8750000}{0.8428571} + \cdots + \log \frac{0.0601852}{0.0400922}\right] \\
&= -5.1473422
\end{aligned}$$

Therefore the value of $\log \frac{P(S|ES)}{P(H|ES)}$ is greater than zero, which indicates email ES is spam. And the value of $\log \frac{P(S|EH)}{P(H|EH)}$ is negative, it shows that email EH is classified as ham. Finally, we successfully classified two sample emails into the correct email groups by using naive Bayes classification.

Feature	In Spam	In Ham	$P(w_i\|S)$	$P(w_i\|H)$
A	165	1235	0.3819444	0.5691244
Advised	12	42	0.0277778	0.0193548
As	2	579	0.0046296	0.2668203
Chance	45	35	0.1041667	0.0161290
Clarins	1	6	0.0023148	0.0027650
Exercise	6	39	0.0138889	0.0179724
For	378	1829	0.8750000	0.8428571
Free	253	137	0.5856481	0.0631336
Fun	59	9	0.1365741	0.0041475
Girlfriend	26	8	0.0601852	0.0036866
Have	291	2008	0.6736111	0.9253456
Her	38	118	0.0879630	0.0543779
I	9	1435	0.0208333	0.6612903
Just	207	253	0.4791667	0.1165899
Much	126	270	0.2916667	0.1244240
Now	221	337	0.5115741	0.1552995
Paying	26	10	0.0601852	0.0046083
Receive	171	98	0.3958333	0.0451613
Regularly	9	87	0.0208333	0.0400922
Take	142	287	0.3287037	0.1322581
Tell	76	89	0.1759259	0.0410138
The	185	930	0.4282407	0.4285714
Time	212	446	0.4907407	0.2055300
To	389	1948	0.9004630	0.8976959
Too	56	141	0.1296296	0.0649770
Trial	26	13	0.0601852	0.0059908
Vehicle	21	58	0.0486111	0.0267281
Viagra	39	19	0.0902778	0.0087558
You	391	786	0.9050926	0.3622120
Your	332	450	0.7685185	0.2073733

Table 2.3: New table of training data

Chapter 3

Paul Graham's Method

3.1 Filtering process

Paul Graham has a slightly different way of implementing naive Bayes method on spam classifications.[2] His training dataset also contains two groups of emails, one for spam and the other for ham. The difference is that the size of his groups are almost the same, each has about 4000 emails in it. And not like the sample emails I created in the last chapter, which only have message bodies, he takes into account the header, javascript and embedded html in each email. For better filtering, he enlarges the domain of tokens by considering alphanumeric characters, dashes, apostrophes, and dollar signs as tokens, and all the others as token separators. Of course, more symbols will be discovered as useful indicators of spam in the future. But at that time, he ignores purely digits tokens, and html comments.

The filtering process begin with generating the same table as Table 2.2, which states the number of appearance of each token in both email groups. Again, the appearances of each token count only once for each email received. After that, he calculates the probability of each token that an email containing it is a spam[16, p. 69]. Now, we denote AS and AH as the token's frequency of appearances in spam and ham respectively. By using our training dataset, the probability $P(S|w_i)$ of a token will be computed as follow:

$$P(S|w_i) = \frac{AS/432}{AS/432 + AH/2170}$$

For example,the probability of the token *Exercise* will be:

$$P(S|w_6) = \frac{6/432}{6/432 + 39/2170} = 0.4359180$$

In this case, the word *Exercise* is considered as a hammy word where probability 0.5 is neutral.

Here, Graham discovered two ways to reduce false positives in the classification. One way is doubling the number of appearances in ham for each token in the email. By doing this, we can find out which token is rarely used in legitimate email and also consider some of the tokens as not appearing in ham at all. The other slight bias Graham used to fight against false positives is using the number of emails in each individual email group, rather than the total number of emails in both groups, as the divisor in the equation of calculating probabilities $P(S|w_i)$.

The modified version of Graham's method calculate the probability as follow:

$$P(S|w_i) = \frac{AS/432}{AS/432 + 2AH/2170}$$

Then, the probability of the token *Exercise* will become:

$$P(S|w_6) = \frac{6/432}{6/432 + 2 \times 39/2170} = 0.2787054$$

Clearly, the word *Exercise* is more hammy in this version as the probability is closer to 0. Which shows Graham is trying to enhance the effect of all the hammy words in the calculation of overall probability by doubling their appearances in ham. This slight bias will help reducing the probability that filters misjudge a ham as spam, which means minimize the number of false positives.

In Graham's classification, the filter will assign a probability to the token only if its appearance in total is more than five times. In maths, the threshold will be $AS + 2AH > 5$ for any token in the dataset.

In case of a token's appearance in both email groups is less than five times, this type of tokens will be considered as *unknown tokens*(or called hapaxes). Which means a word that never occurs in both email groups, and it will be assigned a neutral probability 0.5, also known as the hapaxial value.(In Graham's method, he use 0.4 since he thought unknown tokens are usually fairly innocent.) This action can effectively prevent dictionary attack which flood the emails with lots of random words in order to mislead filters. At this moment, we ignore these tokens in our calculation by assigning the neutral probability and they will become useful after more relative data has been collected. Then another problem arised, how about the probabilities of tokens that only appear in one of the email groups? These are called single-corpus tokens, we normally assign a probability of 0.99 for tokens that only found in spam and a probability of 0.01 for tokens only appear in ham.

Based on Paul Graham's method, our training data table will look like this.(Table 3.1)

Now we tokenize the new email we receive, same as what we did for the original naive Bayes classification. But not taking all the tokens into account, only pick up the most *interesting* 15 tokens. The further a token's

Feature	Appearances in Spam	Appearances in Ham	$P(S\|w_i)$
A	165	1235	0.2512473
Advised	12	42	0.4177898
As	2	579	0.0086009
Chance	45	35	0.7635468
Clarins	1	6	0.2950775
Exercise	6	39	0.2787054
For	378	1829	0.3417015
Free	253	137	0.8226372
Fun	59	9	0.9427419
Girlfriend	26	8	0.8908609
Have	291	2008	0.2668504
Her	38	118	0.4471509
I	9	1435	0.0155078
Just	207	253	0.6726596
Much	126	270	0.5396092
Now	221	337	0.6222218
Paying	26	10	0.8671995
Receive	171	98	0.8142107
Regularly	9	87	0.2062346
Take	142	287	0.5541010
Tell	76	89	0.6820062
The	185	930	0.3331618
Time	212	446	0.5441787
To	389	1948	0.3340176
Too	56	141	0.4993754
Trial	26	13	0.8339739
Vehicle	21	58	0.4762651
Viagra	39	19	0.8375393
You	391	786	0.5554363
Your	332	450	0.6494897

Table 3.1: Training data table with Paul Graham's method

probability is from the neutral probability 0.5, the more *interesting* it is. And Graham choosed the number 15 for a reason, he found out that a large percentage of spams in his dataset tend to contain around 15 spammy words in them. Therefore if you consider 30 tokens instead of 15, most likely you will include 15 innocent tokens in the calculation which are clearly noise added by the spammers. The number of spam in your inbox will increase as more innocent tokens are included in the calculation. But at the same time, you also need to have a sufficient amount of tokens in order to make the correct decision. It is possible for a legitimate email to contain the words like *Paying, Free,* and *Trail*, even they are considered as Top 5 spammy words in my training dataset. Here is a example for it:

> Hey,Leon. Tell you a great news, the new bar is giving out free trail of all the cocktails this week. And Gary is paying tonight since he is about to finish the course. So leave your wallet at home and join us for dinner at 7pm, don't be late!

For sure, you do not want to classify a email from your close friend as spam. Otherwise, you will miss out a lovely evening with free dinner and cocktails provided. This can be one of the worst outcome when you get false positives.

After collecting the 15 most *interesting* tokens, we combine their probabilities $P(S|w_i)$ by Bayes' theorem to produce a overall probability $P(S|E)$ with the assumption that the tokens are all independent of each other.[16, p. 76] The combination works as follow, with $n = 15$ in this case:

$$P(S|E) = \frac{\prod_{i=1}^{n} P(S|w_i)}{\prod_{i=1}^{n} P(S|w_i) + \prod_{i=1}^{n} P(H|w_i)} \tag{3.1}$$

where

$$P(H|w_i) = 1 - P(S|w_i)$$

In Graham's algorithm, he classifies the email as spam if the overall probability $P(S|E)$ is higher than 0.9. The boundary seems to be quite high at this point, but the applications of this method in the next section will show that the overall probabilities will always end up somewhere near either 0 or 1.

3.2 Application

In the first section, we already have a table contains Graham's probabilities $P(S|w_i)$ of each token. Since our sample emails have only 15 words, I will

Feature	In Spam	In Ham	$P(S\|w_i)$	Interestingness
Paying	26	10	0.8671995	0.3671995
Viagra	39	19	0.8375393	0.3375393
Trial	26	13	0.8339739	0.3339739
Free	253	137	0.8226372	0.3226372
Receive	171	98	0.8142107	0.3142107
Chance	45	35	0.7635468	0.2635468
A	165	1235	0.2512473	0.2487527
Have	291	2008	0.2668504	0.2331496
To	389	1948	0.3340176	0.1659824
For	378	1829	0.3417015	0.1582985
Now	221	337	0.6222218	0.1222218
You	391	786	0.5554363	0.0554363
Much	126	270	0.5396092	0.0396092
Too	56	141	0.4993754	0.0006246

Table 3.2: Ordered training data table for sample spam

Feature	In Spam	In Ham	$P(S\|w_i)$	Interestingness
As	2	579	0.0086009	0.4913991
I	9	1435	0.0155078	0.4844922
Regularly	9	87	0.2062346	0.2937654
Have	291	2008	0.2668504	0.2331496
Exercise	6	39	0.2787054	0.2212946
Clarins	1	6	0.2950775	0.2049225
Just	207	253	0.6726596	0.1726596
The	185	930	0.3331618	0.1668382
To	389	1948	0.3340176	0.1659824
For	378	1829	0.3417015	0.1582985
Your	332	450	0.6494897	0.1494897
Advised	12	42	0.4177898	0.0822102
Take	142	287	0.5541010	0.0541010
Her	38	118	0.4471509	0.0528491
Time	212	446	0.5441787	0.0441787

Table 3.3: Ordered training data table for sample ham

include all the tokens in the calculation first and see how it works. But in order to attain the best information of the testing email, I decide to add another column into the table, which measures the *Interestingness* of each token. I rearrange the order of each token by its absolute value of $P(S|w_i) - 0.5$, which directly shows us how hammy or spammy each token is, and how much effect each token contribute to the calculation of the overall probability. Above is the ordered training data tables for the sample spam and ham: (Table 3.2 and Table 3.3)

Before we do any statistical combination of these token probabilities, let's have a brief look at the tables first. For the table of sample spam, five out of fourteen words are considered hammy words since their probabilities are less than 0.5. But at the same time, all of the top five most interesting words are very spammy with probabilities exceed 0.8. And plus the other four spammy words left in the table, we can simply make a decision for the classification already. Similarly, the sample ham contains only four spammy words in the entire message. And the top six most interesting words are also hammy words with token probabilities less than 0.3. Therefore it is very straight forward for a real person to classify the email with these tables, but computer needs a more logical way to do that.

This is where Graham's Bayesian combination formula(3.1) becomes useful, let $P(S|ES)$ and $P(S|EH)$ be the overall probabilities for the examples of spam and ham respectively:

$$\begin{aligned} P(S|ES) &= \frac{P(S|w_1) \times \cdots \times P(S|w_{14})}{P(S|w_1) \times \cdots \times P(S|w_{14}) + P(H|w_1) \times \cdots \times P(H|w_{14})} \\ &= \frac{0.8671995 \times \cdots \times 0.4993754}{0.8671995 \times \cdots \times 0.4993754 + 0.1328005 \times \cdots \times 0.5006246} \\ &= 0.9988236 \end{aligned}$$

$$\begin{aligned} P(S|EH) &= \frac{P(S|w_1) \times \cdots \times P(S|w_{15})}{P(S|w_1) \times \cdots \times P(S|w_{15}) + P(H|w_1) \times \cdots \times P(H|w_{15})} \\ &= \frac{0.0086009 \times \cdots \times 0.5441787}{0.0086009 \times \cdots \times 0.5441787 + 0.9913991 \times \cdots \times 0.4558213} \\ &= 8.913810 \times 10^{-7} \end{aligned}$$

As I mentioned in the end of last section, the overall probabilities do end up very close to 0 or 1 in our examples. Therefore we correctly classified the two sample emails with Graham's method, but we used every single token in our examples instead of a proportion of the tokens. What if we only consider the top 5 most *interesting* tokens in this relatively small email samples? Let's calculate the new overall probabilities $P(S|ES^*)$ and $P(S|EH^*)$ with just five tokens included in each calculation.

$$
\begin{aligned}
P(S|ES^*) &= \frac{P(S|w_1) \times \cdots \times P(S|w_5)}{P(S|w_1) \times \cdots \times P(S|w_5) + P(H|w_1) \times \cdots \times P(H|w_5)} \\
&= \frac{0.8671995 \times \cdots \times 0.8142107}{0.8671995 \times \cdots \times 0.8142107 + 0.1328005 \times \cdots \times 0.1857893} \\
&= 0.9997092
\end{aligned}
$$

$$
\begin{aligned}
P(S|EH^*) &= \frac{P(S|w_1) \times \cdots \times P(S|w_5)}{P(S|w_1) \times \cdots \times P(S|w_5) + P(H|w_1) \times \cdots \times P(H|w_5)} \\
&= \frac{0.0086009 \times \cdots \times 0.2787054}{0.0086009 \times \cdots \times 0.2787054 + 0.9913991 \times \cdots \times 0.7212946} \\
&= 4.993545 \times 10^{-6}
\end{aligned}
$$

Both results are slightly bigger than the previous ones, which means the messages are considered more spammy in the reduced classification method. What caused these small changes in the final results? Remember that we calculate the *interestingness* of each token by computer the absolut value of $P(S|w_i) - 0.5$, this time we only want the value of $P(S|w_i) - 0.5$. Exclude the top five most interesting tokens, we calculate the sum of $P(S|w_i) - 0.5$ from the remaining tokens:

In the sample spam:

$$
\begin{aligned}
\sum_{i=1}^{n} [P(S|w_i) - 0.5] &= 0.2635468 + \cdots + (-0.0006246) \\
&= -0.3259937
\end{aligned}
$$

In the sample ham:

$$
\begin{aligned}
\sum_{i=1}^{n} [P(S|w_i) - 0.5] &= (-0.2049225) + \cdots + 0.0441787 \\
&= -0.4106717
\end{aligned}
$$

If the value of $P(S|w_i) - 0.5$ is negative, it means the word w_i is considered to be a hammy word since its probability is less than 0.5. Thus these two negative results from the above equations show us that the remaining tokens in both emails contribute a hammy effect to the overall probabilities on the whole. This is why the reduced method gave out higher overall probabilities by eliminating them from the calculations. But this small change does not really affect the classification, we can still make the right decision with the reduced method after all.

3.3 Tim Peter's criticism

Despite Graham's method works fine on our sample emails, there is still room for improvements. Tim Peter[7] claimed that Graham mistakenly assumed $P(S) = P(H) = 0.5$ in everyone's dataset, which means the probability of a email being a spam is equal to the probability of being a ham in the dataset. It is in fact true in Graham's dataset where he has the same number of messages in each group of emails. But this is unlikely to be true for everyone since the amounts of spam and ham people receive in their datasets will not be exactly the same, normally we receive more ham than spam in the real world. Therefore, the way Tim Peter applies naive Bayes classification is the following:

Firstly, let E be the email we want to classify. And by Bayes theorem, we can get:

$$\begin{aligned} P(S|E) &= \frac{P(E|S)P(S)}{P(E)} \\ &= \frac{P(E|S)P(S)}{P(S)P(E|S) + P(H)P(E|H)} \end{aligned}$$

With the assumption of all the tokens are independent from each other, we combine the token probabilities as below:

$$P(S|E) = \frac{P(S)\prod_{i=1}^{n} P(w_i|S)}{P(S)\prod_{i=1}^{n} P(w_i|S) + P(H)\prod_{i=1}^{n} P(w_i|H)} \tag{3.2}$$

But from Graham's method, we only know the values of $P(S|w_i)$. In order to make use of these probabilities, we transform the equation into the following by applying Bayes theorem in it one more time:

$$P(w_i|S) = \frac{P(S|w_i)P(w_i)}{P(S)}$$

Then equation (3.2) becomes:

$$P(S|E) = \frac{P(S)\prod_{i=1}^{n} (P(S|w_i)P(w_i)/P(S))}{P(S)\prod_{i=1}^{n} (P(S|w_i)P(w_i)/P(S)) + P(H)\prod_{i=1}^{n} (P(H|w_i)P(w_i)/P(H))}$$

All the $P(w_i)$ cancel out each other in the numerator and denominator, after reducing some of the $P(S)$ in the equation gives us:

$$P(S|E) = \frac{P(S)^{1-n} \prod_{i=1}^{n} P(S|w_i)}{P(S)^{1-n} \prod_{i=1}^{n} P(S|w_i) + P(H)^{1-n} \prod_{i=1}^{n} P(H|w_i)} \tag{3.3}$$

This is obviously different from Graham's formula (3.1), his formula does not include the terms $P(S)^{1-n}$ and $P(H)^{1-n}$. The reason is that $P(S) = P(H)$ in Graham's method, they will be canceled out in the equation. This is clearly not appropriate for my dataset with 432 spam and 2170 ham, where $P(S) = 0.1660261$ and $P(H) = 0.8339739$ in my case. But Peter's method is actually equivalent to method we used in the second chapter, where previously $P(S|E)$ is calculated as follow:

$$\begin{aligned} P(S|E) &= \frac{P(E|S)P(S)}{P(E)} \\ &= \frac{P(S) \prod_{i=1}^{n} P(w_i|S)}{P(E)} \end{aligned}$$

This equation is identical to our formula (3.2), with a substitution of the denominator $P(E)$:

$$\begin{aligned} P(E) &= P(S)P(E|S) + P(H)P(E|H) \\ &= P(S) \prod_{i=1}^{n} P(w_i|S) + P(H) \prod_{i=1}^{n} P(w_i|H) \end{aligned}$$

After applying the Bayes theorem to replace $P(w_i|S)$ with a function of $P(S|w_i)$:

$$P(w_i|S) = \frac{P(S|w_i)P(w_i)}{P(S)}$$

Then we will end up with Peter's formula (3.3) in the following way:

$$\begin{aligned} P(S|E) &= \frac{P(S) \prod_{i=1}^{n} P(w_i|S)}{P(S) \prod_{i=1}^{n} P(w_i|S) + P(H) \prod_{i=1}^{n} P(w_i|H)} \\ &= \frac{P(S)^{1-n} \prod_{i=1}^{n} P(S|w_i)}{P(S)^{1-n} \prod_{i=1}^{n} P(S|w_i) + P(H)^{1-n} \prod_{i=1}^{n} P(H|w_i)} \end{aligned}$$

Therefore we can transform the formula (3.3) into equation (2.4) by the following steps:

$$\frac{P(S|E)}{P(H|E)} = \frac{P(S)^{1-n}\prod_{i=1}^{n} P(S|w_i)}{P(H)^{1-n}\prod_{i=1}^{n} P(H|w_i)}$$

Then substitute $P(S|w_i) = \frac{P(w_i|S)P(S)}{P(w_i)}$ into the above equation, after cancel out $P(w_i)$ in the numerator and denominator, we will get:

$$\frac{P(S|E)}{P(H|E)} = \frac{P(S)\prod_{i=1}^{n} P(w_i|S)}{P(H)\prod_{i=1}^{n} P(w_i|H)}$$

In the last step, by applying log to this probability ratio, we will get exactly the same expression as equation (2.4).

Now we can classify the sample emails again with Tim Peter's amendment added into the equation, and at the same time, stop doubling the numbers of appearances in ham for each token. In order to reduce the time for calculation, we only consider the top 5 most interesting tokens in the sample emails:

For the sample spam:

$$\begin{aligned} P(S)^{1-n}\prod_{i=1}^{n} P(S|w_i) &= 0.1660261^{-4} \times (0.9288772 \times \cdots \times 0.8975922) \\ &= 821.2162372 \end{aligned}$$

and

$$\begin{aligned} P(H)^{1-n}\prod_{i=1}^{n} P(H|w_i) &= 0.8339739^{-4} \times (0.0711228 \times \cdots \times 0.1024078) \\ &= 1.1727064 \times 10^{-5} \end{aligned}$$

then we get

$$\begin{aligned} P(S|ES) &= \frac{P(S)^{1-n}\prod_{i=1}^{n} P(S|w_i)}{P(S)^{1-n}\prod_{i=1}^{n} P(S|w_i) + P(H)^{1-n}\prod_{i=1}^{n} P(H|w_i)} \\ &= 0.99999999 \end{aligned}$$

For the sample ham:

Feature	In Spam	In Ham	$P(S\|w_i)$	Interestingness
As	2	579	0.0170552	0.4829448
I	9	1435	0.0305419	0.4694581
Just	207	253	0.8042995	0.3042995
Your	332	450	0.7875038	0.2875038
Take	142	287	0.7130824	0.2130824

Table 3.4: Reduced training data table for sample ham

$$\begin{aligned} P(S)^{1-n}\prod_{i=1}^{n} P(S|w_i) &= 0.1660261^{-4} \times (0.0170552 \times \cdots \times 0.7130824) \\ &= 0.3096396 \end{aligned}$$

and

$$\begin{aligned} P(H)^{1-n}\prod_{i=1}^{n} P(H|w_i) &= 0.8339739^{-4} \times (0.9829448 \times \cdots \times 0.2869176) \\ &= 0.0235044 \end{aligned}$$

then we get

$$\begin{aligned} P(S|EH) &= \frac{P(S)^{1-n}\prod_{i=1}^{n} P(S|w_i)}{P(S)^{1-n}\prod_{i=1}^{n} P(S|w_i) + P(H)^{1-n}\prod_{i=1}^{n} P(H|w_i)} \\ &= 0.9294468 \end{aligned}$$

Unfortunately, Tim Peter's new formula classifies our sample ham as spam with overall probability 0.9294468. From Table 3.4, we can see there are three spammy tokens among the top 5 most *interesting* tokens, where the top 5 tokens used to be all hammy ones with the doubling effect. Just out of curiosity, I tried the process again with all tokens included. The answer still remains the same, the value turns out to be extremely close to 1(precisely $P(S|EH) = 0.9999999947$).

If we compare the ratio of each token's number of appearance in both email groups(Table 3.5), we will discover that the spammy tokens appear a lot more often in spam than the hammy ones. Therefore by doubling the number of appearance in ham, Graham makes every single token become more hammy in the classification by twisting its ratio of appearance in ham. And the good thing is that you will get less false positives, but the bad thing will be spam which contain a few relatively hammy tokens might be able to get through the filter as false negatives.

Feature	In Spam	In Ham
As	0.0046296	0.2668203
I	0.0208333	0.6612903
Just	0.4791667	0.1165899
Your	0.7685185	0.2073733
Take	0.3287037	0.1322581

Table 3.5: Ratio of appearance for sample ham

As a conclusion, Graham tries to minimize the false positives by lower the boundary of spam classificaion. I hope there is other method to improve the filters' performance without boosting the number of spam we receive in the email inbox.

Chapter 4

Unknown Tokens

4.1 Modified sample emails

In order to investigate the consequence of containing an unknown token in the email, I add one extra word into each of the sample emails we had earlier. Word *great* is added into the sample spam and word *do* is added into the sample ham, now the messages become:

- Example for spam with one extra word:

  Paying too much for VIAGRA?

  Now,you have a ***great*** chance to receive a FREE TRIAL!

- Example for ham with one extra word:

  For the Clarins, just take your time. As i have advised her to ***do*** exercise regularly.

And of course, these two extra words are not included in my training dataset. The filters will therefore consider them as *unknown tokens* in the classification later.

4.2 Problem

These tokens will cause a failure to the entire naive Bayes filtering process. The key factor here is that the overall probability of a email is a product of all the individual token probabilities, one of them becomes 0 will result in the overall probability equal to 0. Let's review the previous two naive Bayes filtering methods one by one to give us a better understanding of what has gone wrong.

In the naive Bayes classification, we calculate token probabilities as follow, where AS and AH are the number of appearances of each token in spam and ham email groups respectively:

$$P(w_i|S) = \frac{P(w_i \cap S)}{P(S)} = \frac{AS}{432}$$

and

$$P(w_i|H) = \frac{P(w_i \cap H)}{P(H)} = \frac{AH}{2170}$$

An unknown token will have probabilities $P(w_i|S) = 0$ and $P(w_i|H) = 0$, since the numerators in both equations will be 0 with no relevant data available.

And our overall probability of the email is calculated by combining the token probabilities in the following way:

$$\log \frac{P(S|E)}{P(H|E)} = \log \frac{P(S)}{P(H)} + \sum_{i=1}^{n} \log \frac{P(w_i|S)}{P(w_i|H)}$$

Even though the log domain prevents the overall probability equal to 0 by using addition to produce the final result. But still, the whole equation is not defined due to one of the denominators $P(w_i|H) = 0$. Therefore this type of filtering method fails to classify messages containing unknown tokens.

In the other hand, Graham's method use a different route in measuring token probabilities:

$$P(S|w_i) = \frac{AS/432}{AS/432 + AH/2170}$$

Because an unknown token has never appear in the dataset before, we will have $AS = AH = 0$. The method collapses again since the denominator equals to 0.

4.3 Solution

Firstly, I want to introduce Graham's solution to deal with the unknown tokens(the method we briefly described in the last chapter). His method is the simplest one among all, which just attaches a probability of 0.4 to any unknown tokens. From his observations, he believes that spammy words are usually too familiar and unknown tokens should be relatively innocent. This neutral probability(or called hapaxial value) can protect filters from dictionary attacks which contain a lot random words that normally don't appear in our datasets. By assigning a fairly neutral probability to the unknown tokens, they won't be able to appear in the top 15 most *interesting*

tokens. Therefore the classification will only take them into account until more data has been collected.

But it is not advisable to handle this deadly issue with assigning a fixed token probability all the time. We need a more precise way to measure the token probabilities of rare words, which should include such words that have appeared in our dataset less than 5 times(where Graham consider them as unknown tokens).

The second solution will be Gary Robinson's smoothing method[10, 11], it is one of the best solutions to solve the problem. The formula is based on a assumption that the classification of a e-mail containing word w_i is a binomial random variable with a beta distribution prior. Same as other binomial models, we have n trials in total. Each trial represents the classification result of a new email containing a certain word w_i, the trial is considered as a *success* if the email is classified as spam and a *fail* if the email is classified as ham. Under the big assumption that all tokens are independent to each other, this model clearly meets all the criterias for being a binomial experiment. And now we can assume a beta distribution $Beta(\alpha, \beta)$ for the prior, which helps us to predict the probability of a extra token being a spammy word as follow:

$$P^*(S|w_i) = \frac{\alpha + AS}{\alpha + \beta + (AS + AH)}$$

where AS and AH are the numbers of appearances of word w_i in spam and ham respectively.

At this point, Robinson tries to connect this formula to the probabilities we have calculated by Graham's formula for each token. The substitution he used is:

$$AS = (AS + AH) \times P(S|w_i)$$

But this equation will not work for my dataset. Because the sizes of spam emails and ham emails in my dataset are not equal, which will break the equation above. Therefore in my case, I think it will be more suitable to remain AS as before in the formula.

Then, Robinson introduced two parameters to predict the probability of an unknown token:

C : the confidence level of our general knowledge

G : predicted probability of a unknown token being in a spam, based on our general knowledge of the word

Their values will be adjusted in practice, aim for high accuracy in the classification and low false positives. Normally, we set a starting value of 1 for C and a neutral probability of 0.5 as a starting point for G. Now, we use the following substitutions to get our new formula:

$$\alpha + \beta = C \qquad and \qquad \alpha = C \times G$$

and we get:

$$P^*(S|w_i) = \frac{C \times G + AS}{C + (AS + AH)}$$

In the case of unknown tokens, since $AS = AH = 0$, the token probability will be exactly 0.5, based on our general knowledge.

Generally, the formula weakens the token's confidence if it has only a few appearances in either spam and ham email groups. And the best thing is it lets us consider both our general knowledge and the limited data we have got from unknown tokens in the prediction of their probabilities. Thus in the next section, we will replace all the probabilities $P(S|w_i)$ with $P^*(S|w_i)$ so that we can get a more reliable classification result without any extreme values from the unknown tokens.

4.4 Application

For Graham's solution, we don't really need to do anything. Basically, the slightly hammy probability 0.4 he assigned to unknown tokens will not change the value of overall probability too much. So our classificaion results will be almost the same until more data has been collected, just ignore the unknown tokens for now.

But Robinson's solution completely changed the way of measuring our token probabilities, we need to calculate another probability table with Robinson's formula:

$$P^*(S|w_i) = \frac{1 \times 0.5 + AS}{1 + (AS + AH)}$$

Table 4.1 in the next page is the new table for our dataset, include the unknown tokens.

And it will be better to use Tim Peter's formula (3.3) to calculate the overall probabilities this time, since we have more ham than spam in the dataset. Another purpose is to see whether Peter's formula still classifies the sample ham as spam with these new token probabilities, it is bad to have a false positive at all time. First, we use all the tokens in our sample emails to calculate our overall probabilities.

Classification of the sample spam:

$$\begin{aligned} P(S)^{1-n} \prod_{i=1}^{n} P(S|w_i) &= 0.1660261^{-14} \times (0.1181299 \times \cdots \times 0.5000000) \\ &= 15557.318093 \end{aligned}$$

Feature	Appearances in Spam	Appearances in Ham	$P^*(S\|w_i)$
A	165	1235	0.1181299
Advised	12	42	0.2272727
As	2	579	0.0042955
Chance	45	35	0.5617284
Clarins	1	6	0.1875000
Exercise	6	39	0.1413043
For	378	1829	0.1714221
Free	253	137	0.6483376
Fun	59	9	0.8623188
Girlfriend	26	8	0.7571429
Have	291	2008	0.1267391
Her	38	118	0.2452229
I	9	1435	0.0065744
Just	207	253	0.4501085
Much	126	270	0.3186398
Now	221	337	0.3962433
Paying	26	10	0.7162162
Receive	171	98	0.6351852
Regularly	9	87	0.0979381
Take	142	287	0.3313953
Tell	76	89	0.4608434
The	185	930	0.1662186
Time	212	446	0.3224583
To	389	1948	0.1665954
Too	56	141	0.2853535
Trial	26	13	0.6625000
Vehicle	21	58	0.2687500
Viagra	39	19	0.6694915
You	391	786	0.3323430
Your	332	450	0.4246488
Great	0	0	0.5000000
Do	0	0	0.5000000

Table 4.1: Training data table with Gary Robinson's method

and

$$P(H)^{1-n}\prod_{i=1}^{n}P(H|w_i) = 0.8339739^{-14}\times(0.8818701\times\cdots\times 0.5000000)$$
$$= 0.001179893$$

then we get

$$P(S|ES) = \frac{P(S)^{1-n}\prod_{i=1}^{n}P(S|w_i)}{P(S)^{1-n}\prod_{i=1}^{n}P(S|w_i)+P(H)^{1-n}\prod_{i=1}^{n}P(H|w_i)}$$
$$= 0.999999924$$

Classification of the sample ham:

$$P(S)^{1-n}\prod_{i=1}^{n}P(S|w_i) = 0.1660261^{-15}\times(0.2272727\times\cdots\times 0.5000000)$$
$$= 0.01249972$$

and

$$P(H)^{1-n}\prod_{i=1}^{n}P(H|w_i) = 0.8339739^{-15}\times(0.7727273\times\cdots\times 0.5000000)$$
$$= 0.1992480$$

then we get

$$P(S|EH) = \frac{P(S)^{1-n}\prod_{i=1}^{n}P(S|w_i)}{P(S)^{1-n}\prod_{i=1}^{n}P(S|w_i)+P(H)^{1-n}\prod_{i=1}^{n}P(H|w_i)}$$
$$= 0.059031178$$

Finally, we get the correct classification results in Peter's formula by using Robinson's method to calculate our token probabilities. In particular, the table below shows us the enormous difference in token probabilities from the sample ham between Robinson's formula and Graham's formula(Table 4.2). Apparently all the tokens become more hammy under Robinson's formula, and not a single token considered as a spammy word from this new formula. By considering cases where not much data has been collected for tokens, Robinson's formula improves the accuracy for many neutral-looking

Feature	In Spam	In Ham	$P^*(S\|w_i)$	$P(S\|w_i)$
Advised	12	42	0.2272727	0.5893536
As	2	579	0.0042955	0.0170552
Clarins	1	6	0.1875000	0.4556909
Exercise	6	39	0.1413043	0.4359180
For	378	1829	0.1714221	0.5093555
Have	291	2008	0.1267391	0.4212816
Her	38	118	0.2452229	0.6179742
I	9	1435	0.0065744	0.0305419
Just	207	253	0.4501085	0.8042995
Regularly	9	87	0.0979381	0.3419477
Take	142	287	0.3313953	0.7130824
The	185	930	0.1662186	0.4998070
Time	212	446	0.3224583	0.7048131
To	389	1948	0.1665954	0.5007694
Your	332	450	0.4246488	0.7875038
Do	0	0	0.5000000	0.4000000

Table 4.2: Comparison table of token probabilities in sample ham

Feature	In Spam	In Ham	$P^*(S\|w_i)$	Interestingness
A	165	1235	0.1181299	0.3818701
Have	291	2008	0.1267391	0.3732609
To	389	1948	0.1665954	0.3334046
For	378	1829	0.1714221	0.3285779
Paying	26	10	0.7162162	0.2162162

Table 4.3: New ordered training data table for sample spam

emails. It successfully classified our sample ham as a legitimate email this time, which reduced the number of false positives.

Next, we consider only the top 5 most *interesting* tokens in the calculation. Let's have a table of these tokens for each sample email first. (Table 4.3 and Table 4.4)

Surprisingly, the top 4 most *interesting* tokens in the table for sample spam are all hammy words based on the token probabilities. This makes it difficult for the formula to classify the message correctly with this misleading information.

The new overall probabilities of the sample emails are calculated as follow:

Classification of the sample spam:

Feature	In Spam	In Ham	$P^*(S\|w_i)$	Interestingness
As	2	579	0.0042955	0.4957045
I	9	1435	0.0065744	0.4934256
Regularly	9	87	0.0979381	0.4020619
Have	291	2008	0.1267391	0.3732609
Exercise	6	39	0.1413043	0.3586957

Table 4.4: New ordered training data table for sample ham

$$P(S)^{1-n}\prod_{i=1}^{n}P(S|w_i) = 0.1660261^{-4}\times(0.1181299\times\cdots\times 0.7162162)$$
$$= 0.4030312$$

and

$$P(H)^{1-n}\prod_{i=1}^{n}P(H|w_i) = 0.8339739^{-4}\times(0.8818701\times\cdots\times 0.2837838)$$
$$= 0.3119720$$

then we get

$$P(S|ES) = \frac{P(S)^{1-n}\prod_{i=1}^{n}P(S|w_i)}{P(S)^{1-n}\prod_{i=1}^{n}P(S|w_i)+P(H)^{1-n}\prod_{i=1}^{n}P(H|w_i)}$$
$$= 0.5636775$$

Classification of the sample ham:

$$P(S)^{1-n}\prod_{i=1}^{n}P(S|w_i) = 0.1660261^{-4}\times(0.0042955\times\cdots\times 0.1413043)$$
$$= 0.0000652$$

and

$$P(H)^{1-n}\prod_{i=1}^{n}P(H|w_i) = 0.8339739^{-4}\times(0.9957045\times\cdots\times 0.8586957)$$
$$= 1.3831702$$

then we get

$$
\begin{aligned}
P(S|EH) &= \frac{P(S)^{1-n}\prod_{i=1}^{n} P(S|w_i)}{P(S)^{1-n}\prod_{i=1}^{n} P(S|w_i) + P(H)^{1-n}\prod_{i=1}^{n} P(H|w_i)} \\
&= 0.000047129
\end{aligned}
$$

With a slightly spammy result from the classification of the sample spam, we can still identify the message as a spam. The reason will be that the number of ham in our dataset is roughly five times the amount of spam we have got. This factor reduces the effect of hammy words in the classification and give more attention to the spammy ones. We also classify the sample ham as a legitimate email with no doubt. In conclusion, Robinson's formula solves the problem of unknown tokens smoothly and generates a more accurate classification for emails containing rare words.

Chapter 5

Bayesian Noise Reduction

5.1 Introduction

This chapter we are going to look at one of the major improvements in Bayesian spam filtering, particularly useful in making the filtering process more effective and gain more confidence in the classification. Jonathan A. Zdziarski first discussed about the Bayesian noise reduction algorithm in 2004[15], and also include this idea in his book later[16, p. 231].

The *noise* we are talking about here is defined as words inconsistent with the disposition of the pattern they belong to and can possibly lead to a misclassification of the message if not removed. Generally, the noise we receive can be divided into two different groups. One group is the noise in daily chatting, it can be one of user's close friends discuss about girls or any spammy topics in a email which should indeed be classified as ham. The other group will be the noise in spam, which normally is a bunch of random words with innocent token probabilities. They are added by the spammers in an attempt to flood filters with misleading information. But by applying the BNR algorithm before the classification, our filters will get a clearer display of the messages' original disposition and have a higher confidence level in decision making.

5.2 Zdziarski's BNR algorithm

The Bayesian noise reduction algorithm consists of three stages: pattern contexts learning, identifying *interesting* contexts and detecting anomalies in the contexts. In stage one, we sort out all the tokens into a set of patterns which contains three consecutive tokens each(the size of pattern may vary, normally a window size of 3 is used in the algorithm). In stage two, value of the contexts will be measured and only the most *interesting* contexts will be considered into the next stage. In the final stage, we detect the statistical anomalies in the contexts we get from the previous stage and eliminate them

Token	$P(S\|w_i)$	Band
Advised	0.59	0.60
Chance	0.87	0.85
I	0.03	0.05
Regularly	0.34	0.35

Table 5.1: Pattern examples

from the classification.

Before we can analyse the contexts, each token value will be assigned to a band with width 0.05. We try to reduce the number of patterns need to be identified in our dataset by doing this. For instance, some of the tokens from our dataset will fall into the bands shown at Table 5.1.

With a window size of 3, the following patterns will be created from the tokens above:

0.60_0.85_0.05 0.85_0.05_0.35

In practice, we will need to create such patterns for all the words in the email we receive. A series of these patterns is called *artificial contexts*, and they will be treated as individual tokens in the calculation later. Then we calculate a probability for each pattern by using Graham's method, but we do not bias the number of appearances in ham for the BNR algorithm. Thus the probability of pattern is calculated as follow:

$$P(S|C_i) = \frac{AS/432}{AS/432 + AH/2170} \tag{5.1}$$

where in the equation above, AS is the number of appearance of the context C_i in spam and similarly AH represents the number of appearance in ham. We will also apply Robinson's formula in calculating unknown tokens' probabilities, in this case, both of our unknown tokens will have a token probability of 0.5 and fall into the band 0.50.

Once we have all the context values, the next step is to find the most *interesting* contexts. This time we require not only the context's value need to be at the two ends(close to 0 or 1), but also contains at least one contradictory token. In order to be a *interesting* context in most Bayesian filters, its value need to be in the range of either 0.00 to 0.25 or 0.75 to 1.00. And the context must include at least one token with a relatively big difference between its probability and the overall probability of the pattern, normally a threshold of 0.30 will be suitable for the difference.

For the two patterns we created above, the pattern probabilities calculated by equation (5.1) is the following:

0.60_0.85_0.05 [0.0260111] 0.85_0.05_0.35 [0.0956177]

Feature	$P(S\|w_i)$	Bands	$P(S\|C_i)$
Paying	0.9288772	0.95	0.7462422
Too	0.6661112	0.65	0.4255745
Much	0.7009691	0.70	0.9556228
For	0.5093555	0.50	0.9134995
Viagra	0.9115879	0.90	0.8967813
Now	0.7671230	0.75	0.8354241
You	0.7141871	0.70	0.6012374
Have	0.4212816	0.40	0.0125417
A	0.4015949	0.40	0.5648126
Great	0.5000000	0.50	0.3582559
Chance	0.8659218	0.85	0.9633257
To	0.5007694	0.50	0.2894563
Receive	0.8975922	0.90	0.8691328
A	0.4015949	0.40	0.9991318
Free	0.9026889	0.90	None
Trial	0.9094719	0.90	None

Table 5.2: Context values in sample spam

Both of the context values lie in the range of 0.00 to 0.25, which means once we see patterns display in the same way as them, we can assume they are jointly innocent no matter what words are actually included. And the spammy tokens inside of these contexts are contradictory tokens if the differences between their individual token probabilities and the overall probabilities are bigger than 0.30. Therefore token values 0.60 and 0.85 are considered as contradictory tokens in the first context, and token value 0.85 is the only contradictory token in the second context. The last step of the BNR algorithm is to eliminate these contradictory tokens from the classifications, which leaves us only the useful data to evaluate the email.

5.3 Application

Before we apply the BNR algorithm on our sample emails, there is one very important feature of the algorithm we need to keep in mind, which is that it sorts out the emails without any background information of the emails' dispositions.

Firstly, we will tidy up the contents we have in our sample spam. Different from the previous methods, this time we need to display all the tokens in exactly the same order as they appear in the actual email. And we can have the same token appear in our table as many times as it does in the email, Table 5.2 shows us a table of context values in the sample spam, with all the tokens divided into bands with a width of 0.05.

Based on this table, we identify all the *interesting* patterns and find out the contradictory tokens in them. For the sample spam, we decide to eliminate the following tokens:

For Have A Great To A

0.50 0.40 0.40 0.50 0.50 0.40

Thus, only the following tokens will be considered in the classification:

Paying Too Much Viagra Now You Chance Receive Free Trial

0.95 0.65 0.70 0.90 0.75 0.70 0.85 0.90 0.90 0.90

As you can see, the *noise* we removed from the email are the tokens with neutral probabilities. Now we can classify the sample spam easily with purely spammy tokens within the email. By using Tim Peter's formula (3.3), we get a overall probability of 1.0000000 for the sample spam(the value is not actually equal to 1, but extremely close to). This will be a ideal case for applying the BNR algorithm to prevent innocent word attacks from the spammers.

Next, we will apply the BNR algorithm in the sample ham, which is more important since reducing the number of false positives is crucial for all types of filters. Similarly, we will create a table of context values for the sample ham(Table 5.3).

The table of context values is quite different this time, half of the pattern probabilities are less than 0.6. Therefore the dispositions of our patterns from this sample email are relatively hammy, which is consistent to the character of a ham. And also we have less *interesting* patterns but one more contradictory tokens in the sample ham, the eliminations is as follow:

Clarins Just As I Have Advised Her

0.45 0.80 0.00 0.05 0.40 0.60 0.60

And the remaining tokens is the following:

For The Take Your Time To Do Exercise Regularly

0.50 0.50 0.70 0.80 0.70 0.50 0.50 0.45 0.35

Even three relatively spammy tokens have been removed from the classification, the overall probability of the sample ham still indicates the email as a spam with a very spammy value(0.9999997) under Peter's method. In this case, I decide to enlarge the window size of patterns. Instead of a window

Feature	$P(S\|w_i)$	Bands	$P(S\|C_i)$
For	0.5093555	0.50	0.3699153
The	0.4998070	0.50	0.2298456
Clarins	0.4556909	0.45	0.8231455
Just	0.8042995	0.80	0.6542389
Take	0.7130824	0.70	0.7543587
Your	0.7875038	0.80	0.9723396
Time	0.7048131	0.70	0.9128645
As	0.0170552	0.00	0.9988764
I	0.0305419	0.05	0.3207863
Have	0.4212816	0.40	0.7324567
Advised	0.5893536	0.60	0.2378975
Her	0.6179742	0.60	0.5156789
To	0.5007694	0.50	0.5982764
Do	0.5000000	0.50	0.2180439
Exercise	0.4359180	0.45	None
Regularly	0.3419477	0.35	None

Table 5.3: Context values in sample ham

Feature	$P(S\|w_i)$	Bands	$P(S\|C_i)$
For	0.5093555	0.50	0.8965467
The	0.4998070	0.50	0.8145563
Clarins	0.4556909	0.45	0.9543357
Just	0.8042995	0.80	0.7043126
Take	0.7130824	0.70	0.2398641
Your	0.7875038	0.80	0.1974217
Time	0.7048131	0.70	0.1746881
As	0.0170552	0.00	0.7169663
I	0.0305419	0.05	0.0965754
Have	0.4212816	0.40	0.5524785
Advised	0.5893536	0.60	0.6519348
Her	0.6179742	0.60	0.4145956
To	0.5007694	0.50	None
Do	0.5000000	0.50	None
Exercise	0.4359180	0.45	None
Regularly	0.3419477	0.35	None

Table 5.4: New context values in sample ham

size of 3, we will create a new table of context values for the sample ham with a window size of 5(Table 5.4).

Surprisingly, with the same number of *interesting* patterns(7 patterns, same as the previous case with a window size of 3), we have 10 contradictory tokens under the new criteria for patterns.

Contradictory tokens:

For The Clarins Take Your Time Have Advised Her To

0.50 0.50 0.45 0.70 0.80 0.70 0.40 0.60 0.60 0.50

And we have the following tokens left for classification:

Just As I Do Exercise Regularly

0.80 0.00 0.05 0.50 0.45 0.35

With only one spammy token in our classification, our overall probability for the sample ham decreases to 0.7426067 by using Peter's formula. Basically, the BNR algorithm will be able to perceive more inconsistencies by expanding the window size of patterns and improve the accuracy of classification.

Since our filter still classify the sample ham as a spam, I tried the BNR algorithm again with a window size of 6 for the patterns. And two more tokens are eliminated as contradictory tokens, which are the following:

Just Do

0.80 0.50

These are the top 2 most spammy words in the remaining text we got previously. Finally we have all hammy tokens in the classification, and the result is very encouraging this time, a overall probability of 0.0270686 for the sample ham. That means we managed to reduce the amount of false positives with bigger window size for the patterns, in a exchange of slightly higher false negative rates. This side effect should be acceptable for most Bayesian filters, because reducing the number of false positives is always their number one priority.

Chapter 6

Conclusion

The aim of this project is to demonstrate the high performance of a simple statistical filtering method: naive Bayes classification.

The project starts with a introduction of the basic naive Bayes classification in Chapter 2, working along with our two sample emails as training data. The filtering process was quite straightforward, and by applying the log into the equation overcomes the problem of extremely small value from the products. And the application in the end showed that the naive Bayes classification successfully classified the two sample emails into the correct email groups.

In Chapter 3, we introduced Paul Graham's way of implementing the naive Bayes classification on spam filtering. In Graham's method, he decides to calculate the probabilities $P(S|w_i)$ and $P(H|w_i)$ for each token instead of probabilities $P(w_i|S)$ and $P(w_i|H)$ in the previous chapter. And he doubles the number of appearance in ham for each token in order to reduce false positives in the classification. Another important feature of his method is that he only consider the top 15 most interesting tokens in the classification. As the sample emails we had only have 15 tokens in each, I applied the method on the sample emails again with only the top 5 most interesting tokens were considered. Not surprisingly, the results of the classification were still correct. But Graham's method is created based on a dataset with equal amount of spam and ham, which is clearly not suitable for general cases. Tim Peter's adjustment for the method sorted out this problem, but unfortunately, his method failed to classify our sample ham correctly in the application later.

Chapter 4 discussed the problem of containing unknown tokens in our classification and the sample emails were modified for further testing. Two solutions were introduced in this chapter, where Graham's solution is simply assign a neutral probability of 0.4 to all the unknown tokens and Gary Robinson's smoothing method completely changed the way of calculating the token values. By considering cases where not much data has been collected

for tokens, Robinson's formula improves the accuracy for many neutral-looking emails. It successfully classified our sample ham as a legitimate email in the application later, which reduced the number of false positives.

The last chapter showed us one of the major improvements in Bayesian spam filtering: Bayesian noise reduction. The BNR algorithm was demonstrated in three stages, pattern contexts learning, identifying interesting contexts and detecting anomalies in the contexts. And we managed to classify our sample emails more effectively in the application section.

But still, there are many other ways of implementing the naive Bayes classification haven't been discussed in this project. Such as Multi-variate Bernoulli naive Bayes classifier[14, 6], Multinomial naive Bayes classifier[4] and Poisson naive Bayes classifier[13]. All of them have great improvements based on the basic naive Bayes classification and works particularly well under different circumstances. Spam filtering is a race between spammers and spam filtering programmers, study and research in this field will never stop as long as the spam still appears in our email inbox.

Acknowledgement

I want to give many thanks to my project supervisor, Matthias Troffaes. For his superb guidance throughout my project and lots of decent suggestions during the final stage of this project.

Bibliography

[1] Vasanth Elavarasan and Mohamed Gouda. Email filters that use spammy words only, May 2006.

[2] Paul Graham. A plan for spam. In *Reprinted in Paul Graham, Hackers and Painters, Big Ideas from the Computer Age, O Really, 2004*, 2002.

[3] Konstantinos V. Chandrinos George Paliouras Ion Androutsopoulos, John Koutsias and Constantine D. Spyropoulos. An evaluation of naive bayesian anti-spam filtering. In *Workshop on Machine Learning in the New Information Age*, pages 9–17, Jun 2000.

[4] J. Teevan J. Rennie, L. Shih and D. Karger. Tackling the poor assumptions of naive bayes text classifiers. In *ICML*, 2003.

[5] David D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 4–15. Springer Verlag, Heidelberg, DE, 1998.

[6] A. Mccallum and K. Nigam. A comparison of event models for naive bayes text classification, 1998.

[7] Tim Peter, August 2002. http://mail.python.org/pipermail/python-dev/2002-August/028216.html.

[8] Prabhakar Raghavan and Christopher Manning. Text classification: The naive bayes algorithm, 2003.

[9] Peter E. Hart Richard O. Duda and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, November 2000.

[10] Gary Robinson. A statistical approach to the spam problem. *Linux J.*, 2003(107), March 2003.

[11] Gary Robinson. Spam detection, January 2006.

[12] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk e-mail. In *Learning for Text*

Categorization: Papers from the 1998 Workshop, Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05.

[13] Hee C. Seo Sang B. Kim and Hae C. Rim. Poisson naive bayes for text classification with feature weighting. In *Proceedings of the sixth international workshop on Information retrieval with Asian languages*, pages 33–40, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[14] Ion Androutsopoulos Vangelis Metsis and Georgios Paliouras. Spam filtering with naive bayes - which naive bayes?, 2006.

[15] Jonathan A. Zdziarski. Bayesian noise reduction:contextual symmetry logic utilizing pattern consistency analysis, December 2004.

[16] Jonathan A. Zdziarski. *Ending spam:Bayesian content filtering and the art of statistical language classification.* No Starch Press, San Francisco, 2005.