Machine learning exp1 NBM

唐静雯 2015010846

1. Experiment design

The algorithm is below:
(1). Read the data in two categories : spam and ham according to the labels of the mails.
Split the data to two part: training and testing.
(2). V<-{vj} = {spam,ham}

pj <- $\frac{|vj\ \ mails|}{|mails|}$

Textj <- all the spam mails' text in the data set

P(wk|vj)<-$\frac{(nj\ +\ 1)}{(Nj\ +\ |Vocabulary|)}$

nj is the number of occurrence of word k in the vj training data set
Nj is the volume of the vj training data set
Vocabulary<- the whole set of words in the training data set.
(3). then result = argmax{vj}P(vj)$\prod P(ai|vj)$
ai is the ith word in the test mail.

2. Experiment result

The accuracy of the experiments are below:

Training	1st	2nd	3rd	4th	5th	average
5%	0.9175	0.9165	0.9140	0.9260	0.9195	0.9187
50%	0.9140	0.9045	0.9140	0.9115	0.9120	0.9112
100%	0.9690	0.9695	0.9725	0.9785	0.9745	0.9728

3. Experiment analysis

a. Issue 1

As it is shown in the table in the experiment result part, we can see that the accuracy of the naïve bayes machine is quite high. Also the accuracy will increase a lot if all the data set is used as the training data compared with the occasions when only part of the data set is used.

Training	1st	2nd	3rd	4th	5th	average
5%	0.9175	0.9165	0.9140	0.9260	0.9195	0.9187
50%	0.9140	0.9045	0.9140	0.9115	0.9120	0.9112
100%	0.9690	0.9695	0.9725	0.9785	0.9745	0.9728

As for the similar accuracies of the training set of 5% and 50% of the whole data set, I believe it indicates that the volume of the training data is not as critical as that the training data set have included the test data. On the other hand, the machine with the whole data set as the training data behaves much better perhaps because it includes the testing emails.

In summary, the volume of the training data is not that critical to the accuracy of the naïve bayes machine (at least at such training scale level). And of course the accuracy of the machine with whole data set as training set is the largest.

b. Issue 2

When the testing mail have words that are not included in the vocabulary warehouse, the posterior probability of the word will be zero which will make the probability we estimate zero. To solve this problem, we use m-estimation in which the posterior probability of a word is $\frac{n_c+mp}{n+m} = \frac{n_c+1}{n+|Vocabulary|}$. In the equation, n_c means the number of the word in the whole text data and n means the number of the mails in the training data, |Vocabulary| is the number of all the words we have posterior probabilities calculated. Under this assumption, even if the word has not shown in the training data, the probability is still positive which make sense.

c. Issue 3

Some words are common in a mail no matter what kind of mail it is like “com”. So I check it online and find a list of common words in mail and remove them from the vocabulary base.

Also due to the phone number is common but we have already remove it when dealing with the data in the first step together with some special symbols like the cut-off line.

As for the “edu” I think the training data set will automatically deal with this term with a high probability to be a ham mail(but there are mails sent by edu mailboxes which are spam mails like ‘HULT’ university aiming at specific goals). So I choose not to do anything with the domain names of the mails.

4. Experiment discussion

When implementing the machine, several **problems** come up like to read in the data correctly and split them into proper phrases in Chinese. Below are some of them:

a. # large data matrix

When using R to deal with the data in TermDocumentMtrix form, the tdm is so big due to the huge number of training data as to failing to transfer to matrix. As the ultimate goal is the counts of each word rather than the matrix itself, I split the tdm to proper size of parts and add the counts together to get the frequency of a word. The problem may be solve in a better way. However this way is sufficient in this case and I will use Python instead of R in next homework to avoid such problem.

b. # frequency selection

There are ways that use frequent words (like occur more than 5 times in the training data set as the vocabulary base) maybe to reduce the data scale. While it may make sense when it comes to the probability that some rarely used words will not have so much influence in the calculation, I still choose the whole word set as the vocabulary base because it performs better whose reason is discussed in detail in part c.

c. **# structure of the training data**

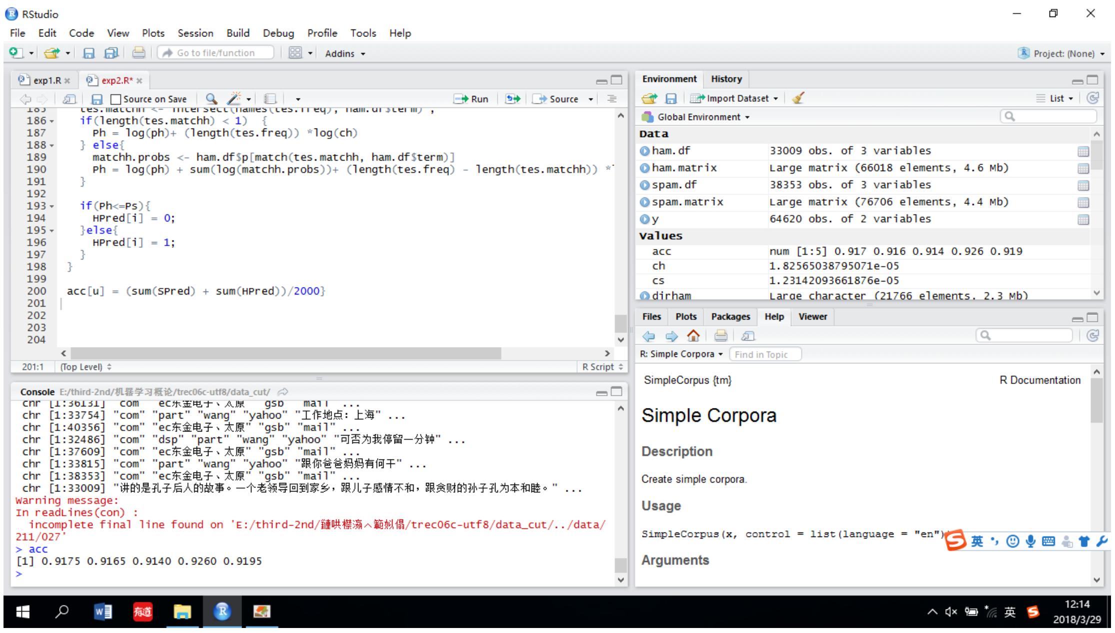
When I first try the frequent words as the vocabulary the result is very bad. Because usually the mode of the spam mails are more remarkable than that of ham mails. So the frequent words are more and the terms of posterior probability of an existing word are more than the terms of the ham probability. We know that the probability term of an existing word is usually bigger than that of a new word. And much more such terms will lead to much larger probability of the mail to be a spam one. So the prediction result intends to classify the mail as a spam one which is incorrect. So I change to the one-frequency demand and make sure that the number of spam training mails is the same as that of the ham ones.

d. # small p value comparison

Make log to the probabilities we calculated because they are too small to be compared.

5. Appendix:

A. Screenshot of the running results:



```r
    tes.matchh <- intersect(names(tes.freq), ham.df$term) ;
    if(length(tes.matchh) < 1)  {
      Ph = log(ph)+ (length(tes.freq)) *log(ch)
    } else{
      matchh.probs <- ham.df$p[match(tes.matchh, ham.df$term)]
      Ph = log(ph) + sum(log(matchh.probs))+ (length(tes.freq) - length(tes.matchh)) *
    }

    if(Ph<=Ps){
      HPred[i] = 0;
    }else{
      HPred[i] = 1;
    }
  }

 acc[u] = (sum(SPred) + sum(HPred))/2000}
```

Console E:/third-2nd/机器学习概论/trec06c-utf8/data_cut/

```
 chr [1:36131] "com" "ec东金电子、太原" "gsb" "mail" ...
 chr [1:33754] "com" "part" "wang" "yahoo" "工作地点：上海" ...
 chr [1:40356] "com" "ec东金电子、太原" "gsb" "mail" ...
 chr [1:32486] "com" "dsp" "part" "wang" "yahoo" "可否为我停留一分钟" ...
 chr [1:37609] "com" "ec东金电子、太原" "gsb" "mail" ...
 chr [1:33815] "com" "part" "wang" "yahoo" "跟你爸爸妈妈有何干" ...
 chr [1:38353] "com" "ec东金电子、太原" "gsb" "mail" ...
 chr [1:33009] "讲的是孔子后人的故事。一个老领导回到家乡，跟儿子感情不和，跟贪财的孙子孔为本和睦。" ...
Warning message:
In readLines(con) :
  incomplete final line found on 'E:/third-2nd/鏈哄櫒瀛︿範姒傝/trec06c-utf8/data_cut/../data/
211/027'
> acc
[1] 0.9175 0.9165 0.9140 0.9260 0.9195
>
```

```
Data
ham.df          33009 obs. of 3 variables
ham.matrix      Large matrix (66018 elements, 4.6 Mb)
spam.df         38353 obs. of 3 variables
spam.matrix     Large matrix (76706 elements, 4.4 Mb)
y               64620 obs. of 2 variables
Values
acc             num [1:5] 0.917 0.916 0.914 0.926 0.919
ch              1.82565038795071e-05
cs              1.23142093661876e-05
dirham          Large character (21766 elements, 2.3 Mb)
```

SimpleCorpus {tm} R Documentation

Simple Corpora

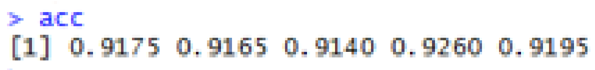
Description

Create simple corpora.

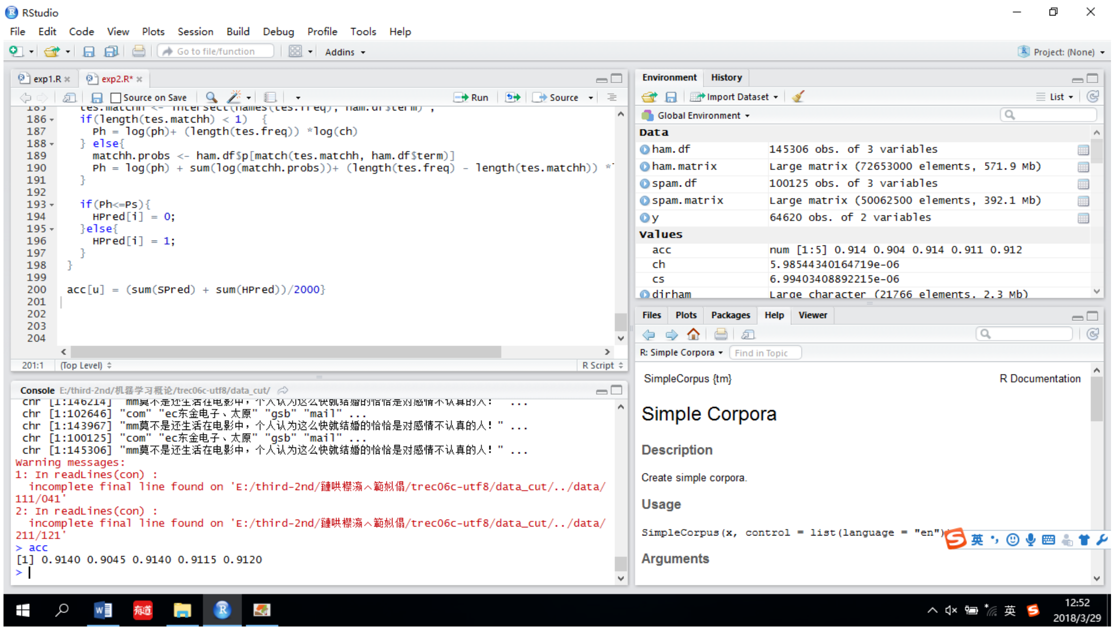
Usage

```
SimpleCorpus(x, control = list(language = "en"))
```

Arguments



```
> acc
[1] 0.9175 0.9165 0.9140 0.9260 0.9195
```



```
  if(length(tes.matchh) < 1)  {
    Ph = log(ph)+ (length(tes.freq)) *log(ch)
  } else{
    matchh.probs <- ham.df$p[match(tes.matchh, ham.df$term)]
    Ph = log(ph) + sum(log(matchh.probs))+ (length(tes.freq) - length(tes.matchh)) *
  }

  if(Ph<=Ps){
    HPred[i] = 0;
  }else{
    HPred[i] = 1;
  }
}

acc[u] = (sum(SPred) + sum(HPred))/2000}
```

```
 chr [1:102646] "com" "ec东金电子、太原" "gsb" "mail" ...
 chr [1:143967] "mm莫不是还生活在电影中，个人认为这么快就结婚的恰恰是对感情不认真的人！" ...
 chr [1:100125] "com" "ec东金电子、太原" "gsb" "mail" ...
 chr [1:145306] "mm莫不是还生活在电影中，个人认为这么快就结婚的恰恰是对感情不认真的人！" ...
Warning messages:
1: In readLines(con) :
  incomplete final line found on 'E:/third-2nd/鏈哄櫒瀛︿範姒傝/trec06c-utf8/data_cut/../data/
111/041'
2: In readLines(con) :
  incomplete final line found on 'E:/third-2nd/鏈哄櫒瀛︿範姒傝/trec06c-utf8/data_cut/../data/
211/121'
> acc
[1] 0.9140 0.9045 0.9140 0.9115 0.9120
> |
```

```
Data
ham.df        145306 obs. of 3 variables
ham.matrix    Large matrix (72653000 elements, 571.9 Mb)
spam.df       100125 obs. of 3 variables
spam.matrix   Large matrix (50062500 elements, 392.1 Mb)
y             64620 obs. of 2 variables
Values
acc           num [1:5] 0.914 0.904 0.914 0.911 0.912
ch            5.98544340164719e-06
cs            6.99403408892215e-06
dirham        Large character (21766 elements, 2.3 Mb)
```

SimpleCorpus {tm} R Documentation

Simple Corpora

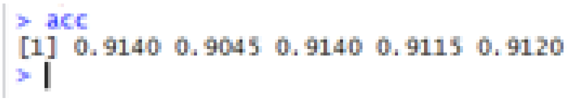
Description

Create simple corpora.

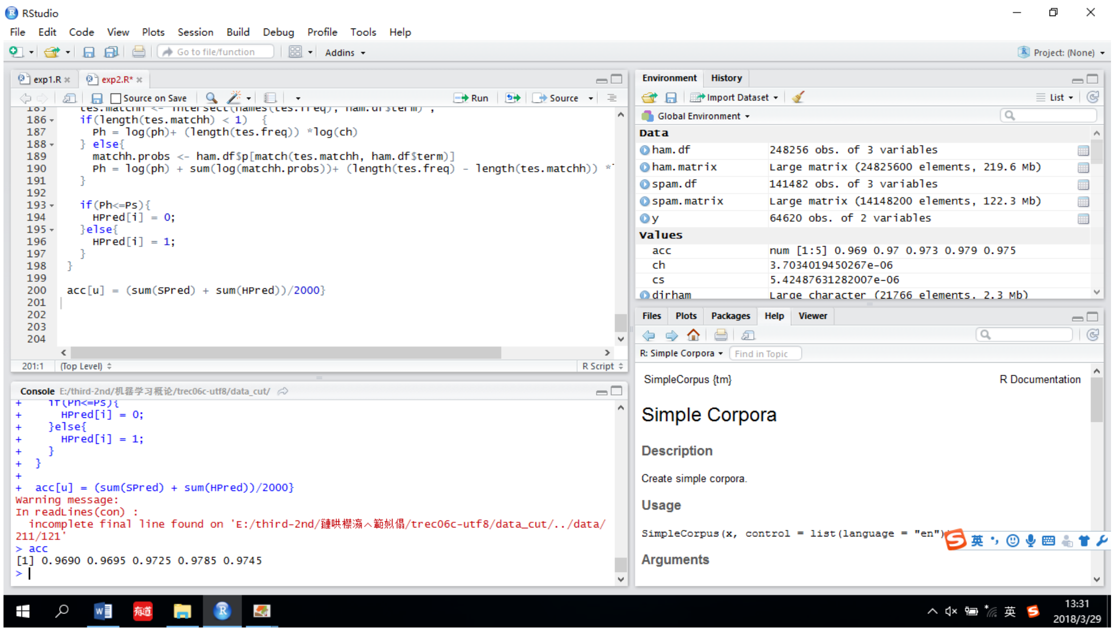
Usage

SimpleCorpus(x, control = list(language = "en"))

Arguments



```
> acc
[1] 0.9140 0.9045 0.9140 0.9115 0.9120
> |
```



```
  if(length(tes.matchh) < 1)  {
    Ph = log(ph)+ (length(tes.freq)) *log(ch)
  } else{
    matchh.probs <- ham.df$p[match(tes.matchh, ham.df$term)]
    Ph = log(ph) + sum(log(matchh.probs))+ (length(tes.freq) - length(tes.matchh)) *
  }

  if(Ph<=Ps){
    HPred[i] = 0;
  }else{
    HPred[i] = 1;
  }
}

acc[u] = (sum(SPred) + sum(HPred))/2000}
```

```
+     HPred[i] = 0;
+   }else{
+     HPred[i] = 1;
+   }
+ }
+
+ acc[u] = (sum(SPred) + sum(HPred))/2000}
Warning message:
In readLines(con) :
  incomplete final line found on 'E:/third-2nd/鏈哄櫒瀛︿範姒傝/trec06c-utf8/data_cut/../data/
211/121'
> acc
[1] 0.9690 0.9695 0.9725 0.9785 0.9745
> |
```

```
Data
ham.df        248256 obs. of 3 variables
ham.matrix    Large matrix (24825600 elements, 219.6 Mb)
spam.df       141482 obs. of 3 variables
spam.matrix   Large matrix (14148200 elements, 122.3 Mb)
y             64620 obs. of 2 variables
Values
acc           num [1:5] 0.969 0.97 0.973 0.979 0.975
ch            3.7034019450267e-06
cs            5.42487631282007e-06
dirham        Large character (21766 elements, 2.3 Mb)
```

SimpleCorpus {tm} R Documentation

Simple Corpora

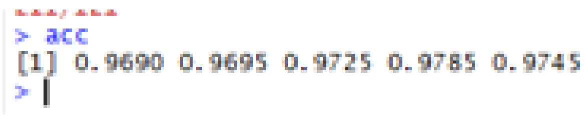
Description

Create simple corpora.

Usage

SimpleCorpus(x, control = list(language = "en"))

Arguments



```
> acc
[1] 0.9690 0.9695 0.9725 0.9785 0.9745
> |
```

B. Code:

```
# 清空变量，加载程序包
rm(list = ls())
setwd("E:/third-2nd/机器学习概论/trec06c-utf8/data_cut");

y = read.table("E:/third-2nd/机器学习概论/trec06c-utf8/label/index");
```

```
yspam = y$V2[y$V1  == "spam"]
yham = y$V2[y$V1  == "ham"]
dirspam = paste("E:/third-2nd/机器学习概论/trec06c-utf8/data_cut/",yspam,sep="")
dirham = paste("E:/third-2nd/机器学习概论/trec06c-utf8/data_cut/",yham,sep="")
if(require(tm) == FALSE) {
  install.packages("tm")
  library(tm)
}
if(require(ggplot2) == FALSE) {
  install.packages("ggplot2")
  library(ggplot2)
}



# 首先处理垃圾邮件
# 定义函数，用于读取邮件正文内容，以第一个空行作为开始标志
#mixseg <- worker();
get.msg <- function(path)
{
  con <- file(path, open = "rt", encoding = "UTF-8")
  text <- readLines(con)

  #msg = segment(text, mixseg);
  msg <- tryCatch(text[seq(which(text == "")[1] + 1, length(text), 1)], error = function(e)
e)
  close(con)
  return(paste(msg, collapse = "\n"))
}


# 构建一个文本资料库
get.tdm <- function(doc.vec)
{

    #mixseg <- worker();
    #doc.vec = segment(doc.vec, mixseg);
    doc.corpus <- Corpus(VectorSource(doc.vec))
    control <- list(stopwords = TRUE,
                    removePunctuation = TRUE,
                    removeNumbers = TRUE,
                    minDocFreq = 2)
  doc.dtm <- TermDocumentMatrix(doc.corpus, control)
  return(doc.dtm)
```

```
}

# 保存邮件内容
#mail.path2 <- dir(mail.path)
#mail.docs<-dir(mail.path2[1])
#spam.docs <- spam.docs[which(spam.docs != "cmds")]

spam <- sapply(dirspam,
                   function(p) get.msg(p))
#mixseg <- worker();
#spam = segment(spam, mixseg);
ham <- sapply(dirham,
              function(p) get.msg(p))
spam.tdm <- get.tdm(spam)
ham.tdm <- get.tdm(ham)

strainvol = 42854*0.5;
trainspam.tdm = spam.tdm[,sample(1:42854,strainvol)];
sms_freq_words = findFreqTerms(trainspam.tdm,1)
str(sms_freq_words)#ham.tdm <- get.tdm(ham)
spamfreq = trainspam.tdm[sms_freq_words,]# 用 TDM 构建一套垃圾邮件训练数据
#spam.matrix = apply(spam.tdm,1,as.vector)
spam.counts = rep(0,length(sms_freq_words))
for (i in 1:200){
  spam.matrix <- as.matrix(spamfreq[,(100*(i-1)+1):(100*i)])
  spam.counts <- spam.counts + rowSums(spam.matrix);
}
#spam.matrix <- as.matrix(spamfreq)
#spam.counts <- rowSums(spam.matrix)
spam.df  <-  data.frame(cbind(names(spam.counts),  as.numeric(spam.counts)),
stringsAsFactors = FALSE)
names(spam.df) <- c("term", "frequency")
spam.df$frequency <- as.numeric(spam.df$frequency)
pspam <- (spam.counts + 1)/(42854 + length(sms_freq_words));
#sapply(1:nrow(spam.matrix),
                          #function(i)
                          #{
                            #length(which(spam.matrix[i,  ]  >  0))  /
ncol(spam.matrix)
                          #})
#spam.density <- spam.df$frequency / sum(spam.df$frequency)

# 统计整个语料库中每个词项的频次
spam.df <- transform(spam.df,
                                    #density = spam.density,
                                    p = pspam)

htrainvol = 21766;
trainham.tdm = ham.tdm[,sample(1:21766,htrainvol)];
sms_freq_wordsh = findFreqTerms(trainham.tdm,1)
str(sms_freq_wordsh)#ham.tdm <- get.tdm(ham)
hamfreq = trainham.tdm[sms_freq_wordsh,]# 用 TDM 构建一套垃圾邮件训练数据
#spam.matrix = apply(spam.tdm,1,as.vector)
ham.counts = rep(0,length(sms_freq_wordsh))
for (i in 1:200){
  ham.matrix <- as.matrix(hamfreq[,(100*(i-1)+1):(100*i)])
  ham.counts <- ham.counts + rowSums(ham.matrix);
}
pham = rep(0,length(sms_freq_wordsh));
ham.df    <-    data.frame(cbind(names(ham.counts),          as.numeric(ham.counts)),
stringsAsFactors = FALSE)
names(ham.df) <- c("term", "frequency")
ham.df$frequency <- as.numeric(ham.df$frequency)
pham <- (ham.counts + 1)/(21766 + length(sms_freq_wordsh));
ham.df <- transform(ham.df,
                                    #density = spam.density,
                                    p = pham)

ps = 42854/64620;
ph = 21766/64620;
  SPred = rep(0,1000);
  HPred = rep(0,1000);
  cs = 1/(42854 + length(sms_freq_words)) ;
  ch = 1/(21766 + length(sms_freq_wordsh)) ;

acc = rep(0,5);
for (u in 1:5){
  for ( i in 1:1000){
tes <- sapply(dirspam[sample(1:42854,1)],
                      function(p) get.msg(p))
tes.tdm = get.tdm(tes);
  tes.freq <- rowSums(as.matrix(tes.tdm))
  tes.match <- intersect(names(tes.freq), spam.df$term) ;

  if(length(tes.match) < 1)    {
     Ps = log(ps)+ (length(tes.freq)) *log(cs)
  } else{
     match.probs <- spam.df$p[match(tes.match, spam.df$term)]
     Ps = log(ps) + sum(log(match.probs))+ (length(tes.freq) - length(tes.match)) *log(cs)
  }

  tes.matchh <- intersect(names(tes.freq), ham.df$term) ;
  if(length(tes.matchh) < 1)    {
     Ph = log(ph)+ (length(tes.freq)) *log(ch)
  } else{
     matchh.probs <- ham.df$p[match(tes.matchh, ham.df$term)]
     Ph  =  log(ph)  +  sum(log(matchh.probs))+ (length(tes.freq)  -  length(tes.matchh))
*log(ch)
  }
  if(Ph<=Ps){
     SPred[i] = 1;
  }else{
     SPred[i] = 0;
  }
  }


  for ( i in 1:1000){
     tes <- sapply(dirham[sample(1:21766,1)],
                       function(p) get.msg(p))
     tes.tdm = get.tdm(tes);
     tes.freq <- rowSums(as.matrix(tes.tdm))
     tes.match <- intersect(names(tes.freq), spam.df$term) ;

     if(length(tes.match) < 1)    {
     Ps = log(ps)+ (length(tes.freq)) *log(cs)
     } else{
     match.probs <- spam.df$p[match(tes.match, spam.df$term)]
     Ps = log(ps) + sum(log(match.probs))+ (length(tes.freq) - length(tes.match)) *log(cs)
     }

     tes.matchh <- intersect(names(tes.freq), ham.df$term) ;
     if(length(tes.matchh) < 1)    {
     Ph = log(ph)+ (length(tes.freq)) *log(ch)
     } else{
     matchh.probs <- ham.df$p[match(tes.matchh, ham.df$term)]
     Ph  =  log(ph)  +  sum(log(matchh.probs))+ (length(tes.freq)  -  length(tes.matchh))
*log(ch)
     }

     if(Ph<=Ps){
     HPred[i]  = 0;
    }else{
        HPred[i]  = 1;
    }
}

acc[u] = (sum(SPred) + sum(HPred))/2000}
```