Classification of Documents using Text Mining Package “tm”

Pavel Brazdil
LIAAD - INESC Porto LA
FEP, Univ. of Porto

Aug. 2011

References

- I. Feinerer, K.Hornik, and D. Mayer, “Text mining infrastructure in R,” *Journal of Statistical Software,* vol. 25, pp. 1-54, 2008.
- M. Moens, “Information extraction: algorithms and prospects in a retrieval context,” *Computational Linguistics*, vol. 34, pp. 315-317, 2008.
- J. Cowie and W. Lehnert, “Information extraction,” Communications of the ACM, vol. 39, pp. 80-91, 1996.
- I. Feinerer. (2010, Jan.) Introduction to the *tm* Package – Text Mining in R. http://cran.r-project.org/web/packages/tm/vignettes/tm.pdf
- R. Feldman ans J. Sanger: *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*. Cambridge University Press, 2006.
- C. Silva and B. Ribeiro, “The importance of stop word removal on recall values in text categorization,” in *International Joint Conference on Neural Networks*, 2003.
- J. Rennie. (2008, 30th January). 20 Newsgroups data set. Available: http://people.csail.mit.edu/jrennie/20Newsgroups/
- L. Francis, Flynn Matt, Text Mining Handbook, Casualty Actuarian Society E-Furum, 2010.
- G. Bettina, K.Hornik, *Topic Models in R,* Wirtschaftsuniversitat Wien
- L. Torgo, A Linguagem R, Programação para a análise de dados, Escolar Editora, 2009.

References

- Raymond Mooney and U. Nahm, Text Mining with Information Extraction http://userweb.cs.utexas.edu/ml/papers/discotex-meml-03.pdf
- José Lino Uber, Descoberta de Conhecimento com o uso de text Mining aplicado ao SAC, 2004.

Acknowledgements:

The author wishes to thank to the following people for their contributions that helped to improve this material:
- students of M.ADSAD 08/09 and 09/10 and students of MAPi 2009/10;
- Jan Knotek, U.Masaryk, Czech R., visiting student in 2010 of U. Porto under Erasmus.

Overview

1. Introduction
2. Preprocessing document collection using ***tm***
    - 2.1 The dataset 20Newsgroups
    - 2.2 Creating a directory with a corpus for 2 subgroups
    - 2.3 Preprocessing operations
    - 2.4 Creating a Document-Term matrix
    - 2.5 Converting the Matrix into Data Frame
    - 2.6 Including class information
3. Identifying Informative Terms (columns)
4. Classification of documents
    - 4.1 Using a Decision Tree classifier
    - 4.2 Using a Neural Net classifier
    - 4.3 Using a k-NN classifier
    - 4.4 Using a SVM classifier
    - 4.5 Using a Naive Bayes
5. Classifiers comparison chart

1. Introduction

Package “tm” of R permits to process text documents in an effective manner

The work with this package can be initiated using a command

```
> library(tm)
```

It permits to:

- Create a corpus – a collection of text documents
- Provide various preprocessing operations
- Create a Document-Term matrix
- Inspect / manipulate the Document-Term matrix (e.g. convert into a data frame needed by classifiers)
- Train a classifier on pre-classified Document-Term data frame
- Apply the trained classifier on new text documents to obtain class predictions and evaluate performance

2 Classification of documents
2.1 The dataset 20Newsgroups

This data is available from
http://people.csail.mit.edu/jrennie/20Newsgroups/

There are two directores:

- 20news-bydate-train (for training a classifier)
- 20news-bydate-test (for applying a classifier / testing)

Each contains 20 directories, each containing the text documents belonging to one newsgroup.

The data (20news-bydate-tar.gz) can be copied to your PC.

Then extract files to your directories:
../20news-bydate-train and ../20news-bydate-test

After entering in R, use “Change Directory” to the above.

20Newsgroups

Subgroup "comp"
- comp.graphics
- comp.os.ms-windows.misc
- comp.sys.ibm.pc.hardware
- comp.sys.mac.hardware
- comp.windows.x

Subgroup "misc"
- misc.forsale

Subgroup "rec"
- rec.autos
- rec.motorcycles
- rec.sport.baseball
- rec.sport.hockey

Subgroup "sci"
- sci.crypt
- sci.electronics <- chosen here
- sci.med
- sci.space

Subgroup "talk.politics"
- talk.politics.guns
- talk.politics.mideast
- talk.politics.misc

Subgroup "religion"
- talk.religion.misc <- chosen here
- alt.atheism
- soc.religion.christian

2.2 Creating a Corpus

This involves :

- Invoking *"Change directory"* in R to 20news-bydate-train
- Selecting one of the newsgroups (e.g.sci.electronics)
- Loading a package "*tm.plugin.mail*"
- Invoking the instruction Corpus():

```
sci.electr.train <- Corpus( DirSource ("sci.electronics"),
     readerControl=list(reader=readMail, language="en_US" ) )
```

If we type :

```
> length(sci.electr.train)
[1] 591
```

Similarly, we obtain documents from another class (e.g. talk.religion.misc):
talk.religion.train (377 documents)

and also obtain the test data:

sci.electr.test (393 documents)
talk.religion.test (251 documents)

Note: The instruction *getReaders()* shows the instructions for reading in information.

Example of one document

sci.electr.train[[1]]

```
In article <00969FBA.E640FF10@AESOP.RUTGERS.EDU>
      mcdonald@AESOP.RUTGERS.EDU writes:
>[...]
>There are a variety of water-proof housings I could use but the real meat
>of the problem is the electronics...hence this posting.  What kind of
>transmission would be reliable underwater, in murky or even night-time
>conditions?  I'm not sure if sound is feasible given the distortion under-
>water...obviously direction would have to be accurate but range could be
>relatively short (I imagine 2 or 3  hundred yards would be more than enough)
>
>Jim McDonald
…
```


2.3 Preprocessing

The Objective of Preprocessing:

Documents are normally represented using words, terms or concepts.

Considering all possible words as potential indicators of a class can create problems in training a given classifier.

It is desirable to avoid building a classifier using dependencies based on too few cases (spurious regularities).

The aim of preprocessing is to help to do this.

The function tm_map (available in "*tm*") can be used to carry out various preprocessing steps.

The operation is applied to the whole corpus (there is not need to program this using a loop).

Preprocessing using tm_map

The format of this function is as follows:
tm_map(*Corpus*, *Function*)

The second argument *Function* determines what is to be done:

PlainTextDocument – removes XML from the document,
removeWords, stopwords(language='english')
– removes stopwords for the language specified
stripWhitespace – removes extra spaces,
tolower – transforms all upper case letters to lower case,
removePunctuation – removes punctuation symbols,
removeNumbers – removes numbers,

Example of use:

```
> sci.electr.train <- tm_map(sci.electr.train, tolower)
```

etc.
This can be repeated for the other 3 collections of documents

Merging document collections

Instead of repeating this for all 4 documents collections
we can merge the four document collections and
perform the preprocessing on the resulting large collection only once.

This can be done using the function c():

```
> docs <- c(sci.electr.train, talk.religion.train, sci.electr.test, talk.religion.test)
> length(docs)
[1] 1612
```

Merging document collections

We need to remember the indices of each document sub-collection
to be able to separate the document collections later.

sci.electr.train	– documents 1 .. 591
talk.religion.train	– documents 592 .. 968 (377 docs)
sci.electr.test	– documents 969 .. 1361 (393 docs)
talk.religion.test	– documents 1362 .. 1612 (251 docs)

One single collection is important for the next step (document-term matrix).
We will use variables l1 etc. for this.

```
> l1 <- length(sci.electr.train)  (591 docs)
> l2 <- length(talk.religion.train)  (377 docs)
> l3 <- length(sci.electr.test)  (393 docs)
> l4 <- length(talk.religion.test)  (251 docs)
```


Preprocessing the entire document collection

```
> docs.p <- docs
> docs.p <- tm_map(docs.p, PlainTextDocument)
> docs.p <- tm_map(docs.p, removeWords, stopwords(language="english"))
> docs.p <- tm_map(docs.p, stripWhitespace)
> docs.p <- tm_map(docs.p, tolower)
> docs.p <- tm_map(docs.p, removePunctuation)
> docs.p <- tm_map(docs.p, removeNumbers)
```

Note:
The stopwords can be inspected using:

```
> stopwords(language="english")
[1] "a"       "about"    "above"    "across"    "after"    "again"     "against"  "all"
[9] "almost"   "alone"    "along"    "already"   "also"     "although"  "always"   "am"
…
[481]  "youngest"  "your"    "you're"   "yours"    "yourself"   "yourselves"  "you've"   "z"
```

Result of preprocessing of one document

Original document:

> docs[[1]]   (=sci.electr.train[[1]])

In article <00969FBA.E640FF10@AESOP.RUTGERS.EDU>
mcdonald@AESOP.RUTGERS.EDU writes:

dropped

>[...]
>There are a variety of water-proof housings I could use but the real meat
>of the problem is the electronics...hence this posting. What kind of
>transmission would be reliable underwater, in murky or even night-time
…

undesirable

Pre-processed document:

> docs.p[[1]]

in article fbaeffaesoprutgersedu mcdonaldaesoprutgersedu writes
there variety waterproof housings i real meat
electronicshence posting what
transmission reliable underwater murky nighttime
conditions i sound feasible distortion
waterobviously direction accurate range
relatively short i imagine   hundred yards

2.4 Creating Document-Term Matrix (DTM)

Existing classifiers that exploit *propositional representation*,
(such as decision trees, kNN, NaiveBayes, SVM etc.)

require that data be represented in the form of a table, where:
each row contains one case (here a document),
each column represents a particular atribute / feature (here a word).

The function *DocumentTermMatrix(..)* can be used to create such a table.

The format of this function is:

DocumentTermMatrix(<DocCollection>, control=list(<Options>))

Simple Example:

> DocumentTermMatrix(docs.p)

Creating Document-Term Matrix (DTM)

Simple Example:

```
> DocumentTermMatrix(docs.p)
A document-term matrix (1612 documents, 21906 terms)

Non-/sparse entries: 122787/35189685
Sparsity           : 100%
Maximal term length: 135
Weighting          : term frequency (tf)
```

problematic

Options of *DTM*

Most important options of DTM:

weighting=TfIdf	weighting is Tf-Idf
minWordLength=*WL*	the minimum word length is *WL*
minDocFreq=*ND*	each word must appear at least ND times in one of the documents

Other options of DTM

These are not really needed, if preprocessing has been carried out:

stemming = TRUE	stemming is applied
stopwords=TRUE	stopwords are eliminated
removeNumbers=True	numers are eliminated

Generating DTM with different options

```
> dtm.mx <- DocumentTermMatrix( docs.p,
   control=list(minWordLength=3, minDocFreq=2))
> dtm.mx
A document-term matrix (1612 documents, 2959 terms)
Non-/sparse entries: 7849/4762059
Sparsity          : 100%
Maximal term length: 26
Weighting          : term frequency (tf)
```

better

```
> dtm.mx.tfidf <- DocumentTermMatrix( docs.p, control=list(weighting=weightTfIdf,
      minWordLength=2, minDocFreq=2))
> dtm.mx.tfidf
Similar results to above
```


Inspecting the DTM

Function *dim(DTM)* permits to obtain the dimensions of the *DTM* matrix.
Ex.

```
> dim(dtm.mx)
[1] 1612 2959
```

Inspecting some of the column names:
(ex. 10 columns / words starting with column / word 101)

```
> colnames(dtm.mx) [101:110]
[1] "angels"    "angra"    "animals"  "anneser"  "anode"    "anointed"
[7] "anonymous" "another"  "answer"   "answerfax"
```

Inspecting the DTM

Inspecting a part of the DTM matrix:
(ex. the first 10 documents and 20 columns)

```
> inspect( dtm.mx )[1:10,101:106]
Docs     adultery advance advanced advantages advent advertise
   [1,]        0       0        0          0      0         0
   [2,]        0       0        0          0      0         0
   [3,]        0       0        0          0      0         0
   [4,]        0       0        0          0      0         0
   [5,]        0       0        0          0      0         0
   [6,]        0       0        0          0      0         0
   [7,]        0       0        0          0      0         0
   [8,]        0       0        0          0      0         0
   [9,]        0       0        0          0      0         0
  [10,]        0       0        0          0      0         0
```

As we can see, the matrix is very sparse. By chance all values are 0s.

Note: The DTM is not and ordinary matrix, as it exploits object-oriented representation (includes meta-data).

The function inspect(..) converts this into an ordinary matrix which can be inspected.

Finding Frequent Terms

The function *findFreqTerms(DTM, N)* permits to find all the terms that appear *at least N times* in one of the documents.

Ex.

```
> freqterms100 <- findFreqTerms( dtm.mx, 100)
> freqterms100
[1] "wire"    "elohim"  "god"     "jehovah" "lord"
```

From talk.religion

```
> freqterms40 <- findFreqTerms(dtm.mx, 40)
> freqterms40
 [1] "cable"    "circuit"  "ground"   "neutral"  "outlets"  "subject"  "wire"     "wiring"
 [9] "judas"    "ra"       "christ"   "elohim"   "father"   "god"      "gods"     "jehovah"
[17] "jesus"    "lord"     "mcconkie" "ps"       "son"      "unto"
```

```
> table(as.matrix(dtm.mx)[,"cable"])
   0    5    6    8   43
1608    1    1    1    1
```

“cable” appears in 1 document 43 times

Removing Sparse Terms

The function *removeSparseTerms(DTM, S)* permits to eliminate all the terms that have relatively few values in a column.

The level of sparseness is controled by a parameter S.
All terms that contain more than 1-S values equal to 0 are dropped.

Suppose S=0.99, then 1-S represents 1%.
As our matrix has 1612 rows, it is required that 1%=16 values are non-empty.

Ex.

```
> dtm.mx.aux <- removeSparseTerms( dtm.mx, 0.95)
> dim(dtm.mx.aux)
[1] 1612 10
```

Only 10 tems were kept

```
> dtm.mx.aux <- removeSparseTerms( dtm.mx, 0.99)
> dim(dtm.mx.aux)
[1] 1612 181
```

This function is quite useful, leading to improved results in classification.

An alternative way is to select *informative terms* (see later), although this method requires usage of a separate program.

2.5 Converting DTM into a Data Frame

Existing classifiers in R require that data be represented as a data frame (particular representation of tables).

So, we need to convert the matrix into a data frame:

```
> dtm <- as.data.frame(inspect( dtm.mx ))
> rownames(dtm)<- 1:nrow(dtm.mx)
> dtm$wire[180:195]
 [1]  0  6  0  8 108  0  0  0  0  0  0  0  0  0  0  0
> dtm$god[180:195]
 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
> dtm$wire[(592+141):(592+160)]
 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
> dtm$god[(592+141):(592+160)]
 [1] 0  0  7 10  0  9  0  0  9  0  2 36  0  0  0  0  0  2  3  0
```

sci.electr portion

talk.religion portion

2.5 Converting DTM into a Data Frame

Repeating this for the tfidf version:

```
> dtm.tfidf <- as.data.frame(inspect( dtm.mx.tfidf ))
> rownames(dtm.tfidf)<- 1:nrow(dtm.mx.tfidf)

> round(dtm.tfidf$wire[180:195],2)
 [1]  0  6  0  8 108  0  0  0  0  0  0  0  0  0  0  0
> round(dtm$god[180:195] ,2)
 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

The numbers should be different

2.5 Converting DTM into a Data Frame

```
> table(dtm$wire)
   0    2   3   4   6   8  10 108
1591   10   3   3   1   2   1   1

> table(dtm$god)
   0    2   3   4   5   6   7   8   9  10  11  12  13  14  15  19  21  32  36 116
1488   37  24  12   4   7   7   6   6   3   2   1   2   2   3   2   3   1   1   1
```

Word "god" appears 3 times in 24 documents

2.6 Appending class information

This includes two steps:
- Generate a vector with class information,
- Append the vector as the last column to the data frame.

Step 1. Generate a vector with class values (e.g. “sci”, “rel”)
We know that (see slide 6):

sci.electr.train – 591 docs
sci.electr.test – 393 docs
talk.religion.train – 377 docs
talk.religion.test - 251 docs

So:

```
> class <- c(rep(“sci”,591), rep(“rel”,377), rep(“sci”,393), rep(“rel”,251))
```

Step2. Append the class vector as the last column to the data frame

```
> dtm <- cbind( dtm, class)
> last.col <- length(dtm)
[1] 2960   (the number of columns has increased by 1)
```


3. Identifying Informative Terms (Columns)

Preparing the Data for Training a Classifier and Testing :

Generate the training data with the appropriate lines of *dtm*

```
> dtm.tr <- dtm[1:(l1+l2), 1:last.col]
> dim(dtm.tr)
[1]  968 2960
```

Generate the test data with the appropriate lines of *dtm*

```
> dtm.ts <- dtm[(l1+l2+1):(l1+l2+l3+l4),1:(last.col-1)]
> dim(dtm.ts)
[1]  644 2959
```

Identifying Informative Terms (Columns)

```
info.terms <- vector()

find.info.terms <- function(dtm.tr,min.info)
ix.class <- ncol(dtm.tr)
default.info<-info(table(dtm.tr[,ix.class]))
cat("default.info: ", default.info, "\n")

n.atr <- ncol(dtm.tr)-1
n.info.terms <- 0
info.term.ixs <- vector()
col.names <- names(dtm.tr)

# continue
```

Index class (assuming that is the last column)

Initial information of class

Identifying Informative Terms (Columns)

```
for (atri in 1:n.atr) {
 if (sum(dtm.tr[,atri])>0)
  { # begin if
  no.dif.atr.val<-length(table(dtm.tr[,atri]))
  atr.class.table<-table(dtm.tr[,atri],dtm.tr[,ix.class])
  n.rows<-nrow(dtm.tr)
  atr.info <- 0
  for (atr.val in 1: no.dif.atr.val)
    { # begin for
    atr.peso <- sum( atr.class.table[atr.val,]) / n.rows
    atr.info1 <- atr.peso * info(atr.class.table[atr.val,])
    atr.info<-atr.info + atr.info1 }
    info.gain <- default.info - atr.info
    if (info.gain > min.info)
      { info.term.ixs[n.info.terms] <- atri
        n.info.terms <- n.info.terms+1 }
    } #end for
} # end if
```

Process all attributes

If sum of values of ***atri*** is not 0, continue

If information gain > threshold

Identifying Informative Terms (Columns)

```
cat("\n", "Vão ser mantidos ", n.info.terms, " atributos: ", "\n")
cat( col.names[info.term.ixs[1:10]], " etc. ")
cat( col.names[info.term.ixs[n.info.terms-1]],"\n")
cat("Vão ser eliminados ", n.atr-n.info.terms, " atributos", "\n")
return(col.names[info.term.ixs])                    # Corrected 29 June 2011
} # end function
```

Function "info" calculates information relative to a list specifying a distribution:

```
Info <- function(x){
inf <- 0
sumx <- sum(x)
for (i in x) {
  pi <- i/sumx
  infi <- (pi)*log2(pi)
  if (is.na(infi)) infi <- 0
  inf <- inf - infi }
  return(inf)
}
```


Identifying Informative Terms (Columns)

```
> info.terms <- find.info.terms(dtm.tr,0.005)
default.info:  0.964452
 Vão ser mantidos  170  atributos:
accept according agree amp and article articles audio basis belief  etc. you
Vão ser eliminados  2789  atributos
```

4. Classification of Documents
4.1 Preparatory Steps for Usage of Decision Tree

We note that some reserved words of R, such as *break* etc. cause a problem for *rpart.*
We get messages such as: *Error in eval(expr, envir, enclos) : no loop to break from ..*
To overcome this, words need to be substituted by other terms (e.g. *break.t*)

```
rename.terms.in.list <- function(list) {
      for (i in 1:length(list)) {
            cat("replaced", list[i], "at", i, "with", paste(list[i],".t", sep=""), "\n")
            list[i]<- paste(list[i],".t", sep="")
      } #end for i
      return(list)
}
> info.terms <- rename.terms.in.list(info.terms)
replaced break at 130 with break.t
replaced else ..
```


Preparatory Steps

We have to modify also the data frames:

```
rename.terms.in.dtm <- function(dtm) {
for (i in 1:length(dtm)) {
      cat("replaced", names(dtm)[i], "at", i, "with", paste(names(dtm)[i],".t", sep=""), "\n")
      names(dtm)[i] <- paste(names(dtm)[i],".t", sep="")
} #end for i
return(dtm)
}

> dtm.tr<- rename.terms.in.dtm(dtm.tr)
replaced break at  688 with break.t
replaced else at 1826 with else.t
…
> dtm.ts<- rename.terms.in.dtm(dtm.ts)
…
```

Classification of Docs using a Decision Tree

```
> names.tr <-paste(info.terms, collapse='+')
> names.tr
"abortion.t+abraham.t+absolute.t+accept.t+according.t+accusing.t+act.t+action.t+..
> clas.formula <- as.formula( paste('class.t', names.tr, sep='~') )  # Modified 29 June 2011
> clas.formula
“class ~ abortion.t+abraham.t+absolute.t+accept.t+according.t+accusing.t+act.t+ ..
> rpart(clas.formula, dtm.tr)
```


Inspecting the Decision Tree

```
> dt <- rpart(clas.formula, dtm.tr)
> dt
n= 968
node), split, n, loss, yval, (yprob)
      * denotes terminal node
1) root 968 377 sci (0.38946281 0.61053719)
    2) god.t>=1 73   1 rel (0.98630137 0.01369863) *
    3) god.t< 1 895 305 sci (0.34078212 0.65921788)
      6) jesus.t>=1 28   0 rel (1.00000000 0.00000000) *
      7) jesus.t< 1 867 277 sci (0.31949250 0.68050750)
       14) objective.t>=1 21   0 rel (1.00000000 0.00000000) *
       15) objective.t< 1 846 256 sci (0.30260047 0.69739953)
         30) koresh.t>=1 15   0 rel (1.00000000 0.00000000) *
         31) koresh.t< 1 831 241 sci (0.29001203 0.70998797)
           62) and.t>=1 21   3 rel (0.85714286 0.14285714) *
           63) and.t< 1 810 223 sci (0.27530864 0.72469136)
            126) christian.t>=1 16   2 rel (0.87500000 0.12500000) *
            127) christian.t< 1 794 209 sci (0.26322418 0.73677582)
              254) evidence.t>=1 9   0 rel (1.00000000 0.00000000) *
              255) evidence.t< 1 785 200 sci (0.25477707 0.74522293)
                510) bull.t>=1 8   0 rel (1.00000000 0.00000000) *
                511) bull.t< 1 777 192 sci (0.24710425 0.75289575)
                 …
```

Evaluating the Decision Tree

Decision Tree　　Test data

```
> preds.dt <- predict(dt, dtm.ts, type="class")
```

Construct a confusion matrix:

```
> conf.mx.dt <- table(class.ts, preds.dt)
> conf.mx.dt
        dt.preds
class.ts rel sci
     rel  135 116
     sci   15 378
> error.rate.dt <- (sum(conf.mx.dt) – sum(diag(conf.mx.dt))) / sum(conf.mx.dt)
> error.rate.dt
[1] 0.2034161  (20.3 %)
```


Evaluating the Classifer

```
> tp <- conf.mx[1,1]    (true positives)
> fp <- conf.mx[2,1]    (false positives)
> tn <- conf.mx[2,2]    (true negatives)
> fn <- conf.mx[1,2]    (false negatives)
```

	+^	-^
+	TP	FN
-	FP	TN

```
> error.rate <- (fp + fn) / (tp + tn + fp + fn)
> recall = tp / (tp + fn)
> precision = tp /  (tp + fp)
```

	+^	-^
+	TP	FN
-	FP	TN

```
> f1 = 2 * precision * recall / (precision + recall)
```

Normally it is necessary to calculate these measures for both classes.

The measures can be combined using either a *micro-average* or *macro-average.*

$$\text{MacroF1} = \frac{2 * (Pi + Pj)/2 * (Ri + Ri)/2}{(Pi + Pj)/2 + (Ri + Ri)/2}$$

4.2 Classification of Docs using a Neural Net

```
> library(nnet)
> nnet.classifier <- nnet(clas.formula, data=dtm.tr, size=2, rang=0.1,
  decay=5e-4, maxit=200)

> preds.nn <- predict(nnet.classifier, dtm.ts, type="class")

> conf.mx.nn <- table(class.ts, preds.nn)
> conf.mx.nn
        preds.nn
class.ts rel sci
     rel 184  67
     sci  24 369
> error.rate.nn <- (sum(conf.mx.nn) - sum(diag(conf.mx.nn))) / sum(conf.mx.nn)
> error.rate.nn
[1] 0.1413043  (14.1%)
```


4.3 Classification of Docs using a k-NN Classifier

```
> library(class)
> preds.knn <- knn(dtm.tr[, info.terms], dtm.ts[, info.terms], class.tr, k=1)

> conf.mx.knn <- table(class.ts, preds.knn)
> conf.mx.knn
        preds.knn
class.ts rel sci
     rel 134 117
     sci  16 377
> error.rate.knn <- (sum(conf.mx.knn) - sum(diag(conf.mx.knn))) / sum(conf.mx.knn)
> error.rate.knn
[1] 0.2065217  (20.7 %)
```

Notes:
Usage of *tfidf* coding could improve this result
Usage of k=10 or 40 led to worse results.

4.4 Classification of Docs using a SVM Classifier

```
> library(e1071)
> svm.classifer <- svm(clas.formula, dtm.tr)
> preds.svm <- predict(svm.classifier, dtm.ts)
> conf.mx.svm <- table(class.ts, preds.svm)
> conf.mx.svm
 preds.svm
class.ts rel sci
    rel 136 115
    sci  27 366
> error.rate.svm <- (sum(conf.mx.svm) - sum(diag(conf.mx.svm))) / sum(conf.mx.svm)
> error.rate.svm
[1] 0.2204969  (22.0 %)
```

Note: Search for optimal svm parameter settings would most likely improve the result.
Some options:
kernel = “linear”
cost = c(1, 10, 50)

4.5 Classification of Docs using a Naive Bayes

```
> library(RWeka)
> NB<-make_Weka_classifier("weka/classifiers/bayes/NaiveBayes")
> nb.classifier<-NB(clas.formula, dtm.tr)
> preds.nb<-predict(nb.classifier, dtm.ts)

> conf.mx.nb<-table(class.ts, preds.nb)
> conf.mx.nb
         preds.nb
class.ts rel sci
      rel 153  98
      sci  28 365

> error.rate.nb <- (sum(conf.mx.nb) - sum(diag(conf.mx.nb))) / sum(conf.mx.nb)
> error.rate.nb
[1] 0.1956522  (19.6 %)
```

Note: Naive Bayes classifier from software Weka was used
(Naive Bayes from library e1071 led to errors).

5 Comparison of Classification Results

	Error Rate	Macro F1	Prec1	Rec1	F1-1	Prec2	Rec2	F1-2
DTree	0.203	0.789	0.900	0.538	0.673	0.765	0.962	0.852
NN	**0.141**	**0.850**	0.887	0.733	0.802	0.846	0.939	0.890
k-NN	0.207	0.785	0.893	0.534	0.668	0.763	0.959	0.850
SVM	0.220	0.766	0.834	0.542	0.657	0.761	0.931	0.838
SVM-tun	**0.161**	**0.828**	0.827	0.741	0.781	0.845	0.901	0.872
NB	0.196	0.792	0.845	0.610	0.708	0.788	0.929	0.853

Note: Stemming of documents does not provide better results in this case.
Note2: SVM is SVM with default parameters, SVM-tun is SVM with kernel="linear", cost=50