

Classification of Documents using Text Mining Package “tm”

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

Escola de verão
Aspectos de processamento da LN
F. Letras, UP, 4th June 2009

<http://www.liaad.up.pt>

Overview

1. Introduction
2. Preprocessing document collection in “tm”
 - 2.1 The dataset 20Newsgroups
 - 2.2 Creating a directory with a corpus for 2 subgroups
 - 2.3 Preprocessing operations
 - 2.4 Creating Document-Term matrix
 - 2.5 Converting the Matrix into Data Frame
 - 2.6 Including class information
3. Classification of documents
 - 3.1 Using a kNN classifier
 - 3.2 Using a Decision Tree classifier
 - 3.3 Using a Neural Net

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

3

2 Classification of documents

2.1 The dataset 20Newsgroups

This data is available from

<http://people.csail.mit.edu/jrennie/20Newsgroups/>

There are two directories:

- 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 can be copied to a PC you are working on (it is more convenient)

4

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

5

2.2 Creating a Corpus

This involves :

- Selecting one of the newsgroups (e.g.sci.electronics)
- Invoking “*Change directory*” in R to 20news-bydate-train
- Invoking the instruction Corpus():

```
sci.electr.train <- Corpus( DirSource (“sci.electronics”),  
  readerControl=list(reader=readNewsgroup, language=“en_US” ) )
```

If we type :

```
> sci.electr.train or length(sci.electr.train)  
the system responds “A corpus with 591 documents”
```

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

```
talk.religion.train (377 documents)
```

Similarly, we can obtain the test data:

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

6

Example of one document

`sci.electr.train[[1]]`

An object of class "NewsgroupDocument"

```
[1] "In article <00969FBA.E640FF10@AESOP.RUTGERS.EDU>  
    mcdonald@AESOP.RUTGERS.EDU writes:"  
[2] ">[...]"  
[3] ">There are a variety of water-proof housings I could use but the real meat"  
[4] ">of the problem is the electronics...hence this posting. What kind of"  
[5] ">transmission would be reliable underwater, in murky or even night-time"  
[6] ">conditions? I'm not sure if sound is feasible given the distortion under-"  
[7] ">water...obviously direction would have to be accurate but range could be"  
[8] ">relatively short (I imagine 2 or 3 hundred yards would be more than enough)"  
[9] ">"  
[10] ">Jim McDonald"  
...  
[35] " ET \"Tesla was 100 years ahead of his time. Perhaps now his time comes.\""  
[36] "-----"
```

7

Metadata accompanying the document

Slot "Newsgroup":	character(0)
Slot "URI":	file("sci.electronics/52434", encoding = "UTF-8")
Slot "Cached":	[1] TRUE
Slot " Author ":	[1] " et@teal.csn.org (Eric H. Taylor) "
Slot "DateTimeStamp":	character(0)
Slot "Description":	[1] ""
Slot "ID":	[1] "1"
Slot "Origin":	character(0)
Slot " Heading ":	[1] " Re: HELP_WITH_TRACKING_DEVICE "
Slot "Language":	[1] "en_US"
Slot "LocalMetaData":	list()

See Annex 1 for more details on metadata (slots) etc.

8

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 **tmMap** (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).

9

Preprocessing using tmMap

The format of this function is as follows:

tmMap(*Corpus*, *Function*)

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

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

Example of use:

```
> sci.electr.train <- tmMap(sci.electr.train, asPlain)
> sci.electr.train <- tmMap (sci.electr.train, removeSignature)
etc.
```

This can be repeated for the other 3 collections of documents

10

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()`:

```
> sci.rel.tr.ts <- c(sci.electr.train, talk.religion.train, sci.electr.test, talk.religion.test)
> sci.rel.tr.ts
```

A text document collection with 1612 text documents.

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)

11

Preprocessing the entire document collection

```
> sci.rel.tr.ts.p <- tmMap(sci.rel.tr.ts, asPlain)
> sci.rel.tr.ts.p <- tmMap(sci.rel.tr.ts.p, removeSignature)
> sci.rel.tr.ts.p <- tmMap(sci.rel.tr.ts.p, removeWords,
  stopwords(language="english"))
> sci.rel.tr.ts.p <- tmMap(sci.rel.tr.ts.p, stripWhitespace)
> sci.rel.tr.ts.p <- tmMap(sci.rel.tr.ts.p, tmTolower)
> sci.rel.tr.ts.p <- tmMap(sci.rel.tr.ts.p, removePunctuation)
> sci.rel.tr.ts.p <- tmMap(sci.rel.tr.ts.p, removeNumbers)
```

12

Result of preprocessing of one document

Original document:

```
> sci.rel.tr.ts[[1]] (=sci.electr.train[[1]])
```

An object of class "NewsgroupDocument"

```
[1] "In article <00969FBA.E640FF10@AESOP.RUTGERS.EDU>  
mcdonald@AESOP.RUTGERS.EDU writes:"
```

```
[2] ">[...]"
```

```
[3] ">There are a variety of water-proof housings I could use but the real meat"
```

```
[4] ">of the problem is the electronics, hence this posting. What kind of"
```

```
[5] ">transmission would be reliable underwater, in murky or even night-time"
```

```
[6] ">conditions? I'm not sure if sound is feasible given the distortion under-"
```

Pre-processed document:

```
> sci.rel.tr.ts.p[[1]]
```

```
[1] in article fbaeffaesoprutgersedu mcdonaldaesoprutgersedu writes
```

```
[2]
```

```
[3] there variety waterproof housings real meat
```

```
[4] of electronicsence posting
```

```
[5] transmission reliable underwater murky nighttime
```

```
[6] conditions sound feasible distortion under
```

undesirable

13

2.4 Creating Document-Term Matrix (DTM)

Existing classifiers that exploit *propositional representation*,
(such as 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 *attribute / 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(sci.rel.tr.ts.p)
```

Note:

This command is available only in R Version 2.9.1

In R Version 2.8.1 the function available is `TermDocMatrix(<DocCollection>)`

14

Creating Document-Term Matrix (DTM)

Simple Example:

```
> DocumentTermMatrix(sci.rel.tr.ts.p)
```

A document-term matrix (1612 documents, 21967 terms)

Non-/sparse entries: 121.191/35.289.613

Sparsity : 100%

Maximal term length: 135

Weighting : term frequency (tf)

problematic



15

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 in ND docs

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

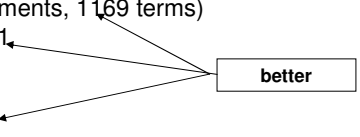
Improved example:

```
> dtm.mx.sci.rel <- DocumentTermMatrix( sci.rel.tr.ts.p,  
  control=list(weighting=weightTfIdf, minWordLength=2, minDocFreq=5))
```

16

Generating DTM with different options

```
> dtm.mx.sci.rel <- DocumentTermMatrix( sci.rel.tr.ts.p,  
  control=list(minWordLength=2, minDocFreq=5))  
> dtm.mx.sci.rel  
A document-term matrix (1612 documents, 1169 terms)  
Non-/sparse entries: 2.237/1.882.191  
Sparsity      : 100%  
Maximal term length: 26  
Weighting      : term frequency (tf)
```



```
> dtm.mx.sci.rel.tfidf <- DocumentTermMatrix( sci.rel.tr.ts.p,  
  control=list(weighing=weightTfIdf, minWordLength=2, minDocFreq=5))  
> dtm.mx.sci.rel.tfidf  
A document-term matrix (1612 documents, 1169 terms)  
Non-/sparse entries: 2.237/1.882.191  
Sparsity      : 100%  
Maximal term length: 26  
Weighting      : term frequency - inverse document frequency (tf-idf)
```

17

Inspecting the DTM

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

```
> dim(dtm.mx.sci.rel)  
1612 1169
```

Inspecting some of the column names:

(ex. 10 columns / words starting with column / word 101)

```
> colnames(dtm.mx.sci.rel) [101:110]  
[1] "blinker" "blood" "blue" "board" "boards" "body" "bonding" "book" "books"  
[10] "born"
```

18

Inspecting the DTM

Inspecting a part of the DTM matrix:

(ex. the first 10 documents and 20 columns)

```
> inspect( dtm.mx.sci.rel )[1:10,101:110]
```

Docs	blinker	blood	blue	board	boards	body	bonding	book	books	born
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0

As we can see, the matrix is very sparse. By chance there are no values other than 0s.

Note: The DTM is not an 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, ND)` permits to find all the terms that appear in at least `ND` documents.

Ex.

```
> freqterms100 <- findFreqTerms( dtm.mx.sci.rel, 100)
```

```
> freqterms100
```

```
[1] "wire" "elohim" "god" "jehovah" "lord"
```

From talk.religion

```
> freqterms40 <- findFreqTerms(dtm.mx.sci.rel, 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"
```

```
> freqterms10 <- findFreqTerms(dtm.mx.sci.rel, 10)
```

```
> length(freqterms10)
```

```
[1] 311
```

2.5 Converting TDM into a Data Frame

Existing classifiers in R (such as kNN, NaiveBayes, SVM etc.) require that data be represented in the form of a **data frame** (particular representation of tables).

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

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

```

> dtm.sci.rel$wire[180:200]
[1] 0 6 0 8 108 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
> dtm.sci.rel$god[180:200]
[1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

> dtm.sci.rel$wire[(592+141):(592+160)]
[1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
> dtm.sci.rel$god[(592+141):(592+160)]
[1] 0 0 0 10 0 5 0 9 0 0 35 0 0 0 0 0 0 0 0 0 0

```

21

Converting TDM into a Data Frame

We repeat this also for the tf.idf version:

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

```
> dtm.sci.rel.tfidf <- as.data.frame(inspect(dtm.mx.sci.rel.tfidf))
```

```
> round(dtm.sci.rel.tfidf$wire[180:200],1) ← sci.electr portion  
[1] 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0  
  
> round(dtm.sci.rel.tfidf$god[180:200],1) ← talk.religion portion  
[1] 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0  
  
> round(dtm.sci.rel.tfidf $wire[(592+141):(592+160)],1) ← sci.electr portion  
[1] 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0  
  
> round(dtm.sci.rel.tfidf$god[(592+141):(592+160)],1) ← talk.religion portion  
0.0 0.0 0.0 0.0 5.1 0.0 5.1 0.0 0.0 5.1 0.0 0.0 5.1 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
```

22

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

So:

```
> class <- c(rep("sci",591), rep("rel",377), rep("sci",393), rep("rel",251))
> class.tr <- c(rep("sci",591), rep("rel",377))
> class.ts <- c(rep("sci",393), rep("rel",251))
```

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

```
> dtm.sci.rel <- cbind( dtm.sci.rel, class)
> ncol(dtm.sci.rel)
[1] 1170 (the number of columns has increased by 1)
```

23

3 Classification of Documents

3.1 Using a KNN classifier

Preparing the Data

The classifier kNN in R requires that
the training data and test data have the same size.

So:

sci.electr.train – 591 docs	: Use the first 393 docs
talk.religion.train – 377 docs	: Use the first 251 docs
sci.electr.test – 393 docs	: Use all
talk.religion.test - 251 docs	: Use all

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

```
> m1 <- min(l1,l3) (393 docs)
> m2 <- min(l2,l4) (251 docs)
> m3 <- min(l1,l3) (393 docs)
> m4 <- min(l2,l4) (251 docs)
```

24

Preparing the training data

Generating the training data:

Calculate the last column of data (excluding the class):

```
> last.col <- ncol( dtm.sci.rel )-1
```

Generate the first part of the training data with lines of *sci.electr.train*

```
> sci.rel.tr <- dtm.sci.rel[1: m1, 1:last.col]
```

Generate a vector with the class values:

```
> class.tr <- dtm.sci.rel[1: m1, last.col+1]
```

Extend the training data by adding lines from *talk.religion.train*

```
> sci.rel.k.tr[(m1+1):(m1+m2),] <- dtm.sci.rel[(l1+1):(l1+m2),1:last.col]
```

```
> nrow(sci.rel.k.tr)
```

```
644
```

Add class values at the end of the class vector:

```
> class.k.tr[(m1+1):(m1+m2)] <- dtm.sci.rel[(l1+1):(l1+m2), last.col+1]
```

```
> length(class.tr)
```

```
644
```

25

Preparing the test data

Generate the test data using
the appropriate lines of *sci.electr.test* and *talk.religion.test*

```
> sci.rel.k.ts <- dtm.sci.rel[(l1+l2+1):(l1+l2+m3+m4),1:last.col]
```

```
> nrow(sci.rel.k.ts)
```

```
644
```

```
> class.k.ts <- dtm.sci.rel[(l1+l2+1):(l1+l2+m3+m4),last.col+1]
```

26

3.2 Classification of Docs using a Decision Tree

Here we will use the same training / test data as in
> library(rpart)

Here we will use just 20 frequent terms as attributes

```
> freqterms40
[1] "cable" "circuit" "ground" "neutral" "outlets" "subject" "wire"
[8] "wiring" "judas" "ra" "christ" "elohim" "father" "god"
[15] "gods" "jehovah" "jesus" "lord" "mcconkie" "ps" "son"
[22] "unto"

> dt <- rpart(class ~ cable + circuit + ground + neutral + outlets + subject + wire +
  wiring + judas + ra + christ + elohim + father + god + gods + jehovah + jesus +
  lord + mcconkie + ps + son + unto, sci.rel.tr)
```

27

Inspecting the Decision Tree

```
> dt
n= 968
node), split, n, loss, yval, (yprob)
* denotes terminal node
1) root 968 377 sci (0.3894628 0.6105372)
2) god>=2.5 26 0 rel (1.0000000 0.0000000) *
3) god< 2.5 942 351 sci (0.3726115 0.6273885)
6) jesus>=2.5 16 0 rel (1.0000000 0.0000000) *
7) jesus< 2.5 926 335 sci (0.3617711 0.6382289) *
```

28

Evaluating the Decision Tree



```
> dt.predictions.ts <- predict(dt, sci.rel.ts, type="class")
> table(class.ts, dt.predictions.ts)

dt.predictions.ts
class.ts rel sci
rel 22 229
sci 0 393
```

29

3.3 Classification of Documents using a Neural Net

Acknowledgements:

Rui Pedrosa, M.Sc. Student, M. ADSAD, 2009

```
> nnet.classifier <- nnet(class ~., data=sci.rel.tr, size=2, rang=0.1,
  decay=5e-4, maxit=200)
```

```
> predictions <- predict(nnet.classifier, sci.rel.ts, type="class")
```

The errors reported on a similar task were quite good - about 17%

30

5. Calculating the evaluation measures

The basis for all calculations is the **confusion matrix**, such as:

```
> conf.mx <- table(class.ts, predictions.ts)
```

```
> conf.mx
class.ts rel sci
rel 201 50
sci 3 390
```

```
> error.rate <- (sum(conf.mx) - diag(conf.mx)) / sum(conf.mx)
```

```
> 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)
```

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

31

Evaluation measures

Recall = $TP / (TP + FN) * 100\%$

	+	-
+	TP	FN
-	FP	TN

Precision = $TP / (TP + FP) * 100\%$

	+	-
+	TP	FN
-	FP	TN

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

32