IS4240 Assignment 3

Prepared By Group 8:
Ning Wei Wei (A0084106M)
Prassanthi Muralidharan (A0123403Y)
Raghavendhra Balaraman(A0123443R)
Tran Minh Thy(A0074353H)
Zhang You (A0102049U)

We will build our classifier model through following steps. And the classification method we are using is Bayesian classifier.

1. Data collection

Firstly, for the collection of training data, as there are 20,000 documents in total where 1,000 for each category, we collect 300 documents from each category and combine them to get 6,000 documents in total as our training data. For example, the R code for collecting training data from category " sci.electronics":

```
library(tm)
sci.electronics <- Corpus( DirSource ("sci.electronics"))
sci.electronics.train<-sci.electronics[1:300]
```

With regards to the testing dataset, we choose 100 data from each category and get 2,000 data in total to be our testing data. For example, the R code for collecting testing data from category "sci.electronics":

```
sci.electronics.test <- sci.electronics[301:400]
```

2. Text pre-processing

Before we apply our learning algorithm to build our classifier model, we need our data in a "spreadsheet" form. Our textual training and test data are unstructured data and are not in the form of a spreadsheet. Hence, we need to convert our textual data to be in a spreadsheet form, which called Term-Document Matrix (TDM).

2.1 Text Transformation

2.1.1 Simple transforms

We want to replace "/", used sometimes to separate alternative words, with a space. In addition, we want to replace "@" and "|" with a space too. It helps us to avoid the two words being run into one string of characters.

Code:

```
toSpace <- content_transformer(function(x,pattern) gsub(pattern," ",x))
test_pre <- tm_map(test_pre,toSpace,"/|@|\\|")
```

2.1.2 Conversion to Lower Case

General character processing functions in R can be used to transform our corpus. A common requirement is to map our data to lower case.

Code: test_pre<-tm_map(test_pre, content_transformer(tolower))

2.1.3 Remove Numbers

Numbers in this case are not relevant to our analyses. This transform can remove numbers simply

Code: test_pre<-tm_map(test_pre, removeNumbers)

2.1.4 Remove punctuation

Punctuation can provide grammatical context which supports understanding.

Code: test_pre<-tm_map(test_pre, removePunctuation)

2.1.5 Remove English Stop Words

Stop words are common words found in a language. Words like *for, very, and, of ,are, etc… are common stop words.* We should remove from the data since they are not important.

Code: test_pre<-tm_map(test_pre, removeWords,stopwords('english'))

2.1.6 Strip Whitespace

We use strip whitespace to remove extra space where multiple whitespace characters are collapsed to a single blank.

Code: test_pre<-tm_map(test_pre, stripWhitespace);

2.1.7 Stemming

Stemming uses an algorithm that removes common word endings for English words such as “es”, “ed”, and “s”.

Code: test_pre <- tm_map(test_pre, stemDocument)

2.2 Building Feature Vector

2.2.1 Term Weighting -TF.IDF weighting

TF.IDF (Term Frequency Inverse Document Frequency) is an approach to reflect how important a word is. For example,we have 6000 documents in our corpus, the word “graphic” occurs 381 times, but in just 300 documents, hence, we cannot guarantee that term is important, only due to its frequency.

Code:

DocumentTermMatrix(train_corpus_pre,control=list(weighing=weightTfldf,minWordLength=2, minDocFreq=5))

2.2.2 Bi-gram

Text categorization techniques are keyword-based. So far, we only treat our training data as a bag of words (BOW) , that is, identify terms with all the words occurring in the document and perform categorizations based mainly on the presence or absence of keywords. However, we see that we can improve our categorization performance by automatically extracting and using phrases, especially two-word phrases, which means bigrams.

Code:

DocumentTermMatrix(train_corpus_pre,control=list(minWordLength=2, minDocFreq=5))

2.2.3 Noticement

Due to the time constraints, we cannot apply both bi-gram and term-weight to create our term document matrix. Hence, we decide to create a term matrix based on bi-gram method.

2.2.4 Create a document term matrix

A document term matrix is simply a matrix with documents as the rows and terms as the columns and a count of the frequency of words as the cells of the matrix. We call the matrix is the term document matrix (TDM).

Code: test_corpus <- DocumentTermMatrix(test_pre,control=list(minWordLength=2, minDocFreq=5))

3. Building Classifier Model - Bayesian Method

Amongst three algorithms we learned in class which are Bayesian, Logistic Regression and Support Vector Machines, we choose to apply Bayesian method to build our classifier model, which called Naive Bayes classifier since it is fast with competitive performance compared to the other state-of-art classifiers.

Code: naive_classifier = naiveBayes(dtm.sci.rel, as.factor(class))

4. Classification, Experimental Results

Attributes	**Accuracy %**		
	Iteration 1	**Iteration 2**	**Iteration 3**
alt.atheism	0	0	1
comp.graphics	1	0	6
comp.os.ms-windows.misc	9	18	REMOVED
comp.sys.ibm.pc.hardware	4	34	REMOVED
comp.sys.mac.hardware	89	REMOVED	REMOVED
comp.windows.x	0	0	42
misc.forsale	33	87	REMOVED
rec.autos	3	4	32
rec.motorcycles	0	1	53
rec.sport.baseball	48	78	REMOVED
rec.sport.hockey	7	8	72

sci.crypt	2	1	1
sci.electronics	1	0	75
sci.med	0	0	1
sci.space	0	0	0
soc.religion.christian	5	7	7
talk.politics.guns	0	0	1
talk.politics.mideast	7	6	7
talk.politics.misc	1	0	0
talk.religion.misc	0	0	0

After the first run using our model, we could see the accuracy of each category from confusion matrix and found that the it was not accurate to classify the documents where the testing documents are classified into wrong categories. And almost documents are classified into the category “comp.sys.mac.hardware”, it may because the category “comp.sys.mac.hardware” contains most words which are commonly used by all other categories. Therefore, we should drop this category in order to reduce and misleading results and test the performance of our model.

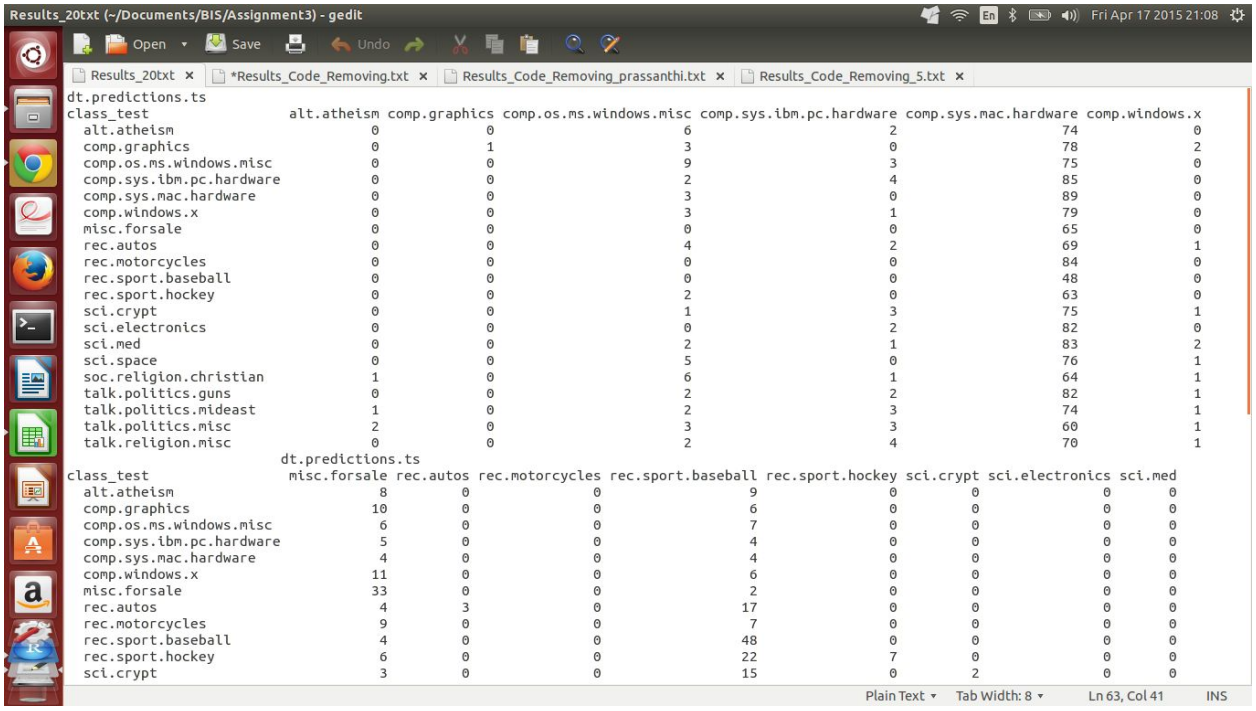
After dropping the category “comp.sys.mac.hardware”, the results showed that the accuracy is improved. However, there are still some dominant categories. Again, we remove them recursively.
(The confusion matrix for iteration 1 and 3 are shown in the appendix.)

5. Limitations

Due to the time constraints, we cannot split our data into two parts to become training data and testing data. We decide to choose the 30% of data to be our training data and 10% of the rest to be our testing data. We understand that the evaluation above is highly dependent on how the training and testing data is sampled. We also understand that if we use different set of training or testing data, it could result in different accuracy. We list it as our limitations in this report, however, our model uses bigger training data set than testing data set, which helps our model tend to perform better.

Appendix:

1. Confusion Matrix for iteration 1



2. Confusion Matrix for iteration 3