LING 410X: Language as Data

Semester: Spring '18

Instructor: Sowmya Vajjala

Iowa State University, USA

1 March 2018

Outline

- Sentiment classification with tm - review of last class
- Other things you can do with tm library and document term matrix
- Practice exercises
- Reminder: Assignment 4 due on 10th (I will be traveling after 8th evening for 2-3 days and may not be accessible immediately).

Review of last class

Steps in text classification

- Step 1: We have a classification problem, and a dataset containing examples for that
  movie review sentiment classification: about 250 examples each for positive and negative reviews.

Steps in text classification

- Step 1: We have a classification problem, and a dataset containing examples for that
  movie review sentiment classification: about 250 examples each for positive and negative reviews.
- Step 2: Read that data into R
- Step 3: Create a document term matrix (DTM), Also, keep track of what each document's category is

Steps in text classification

- Step 1: We have a classification problem, and a dataset containing examples for that
  movie review sentiment classification: about 250 examples each for positive and negative reviews.
- Step 2: Read that data into R
- Step 3: Create a document term matrix (DTM), Also, keep track of what each document's category is
- Step 4: Use an existing classification algorithm - give DTM as input. (training)
- Step 5: Use the classifier (output of Step 4) on new texts, to check how it is doing (testing)

Steps in text classification

- Step 1: We have a classification problem, and a dataset containing examples for that
  movie review sentiment classification: about 250 examples each for positive and negative reviews.
- Step 2: Read that data into R
- Step 3: Create a document term matrix (DTM), Also, keep track of what each document's category is
- Step 4: Use an existing classification algorithm - give DTM as input. (training)
- Step 5: Use the classifier (output of Step 4) on new texts, to check how it is doing (testing)
- Step 6: Repeat these two steps until you are happy, changing different settings. Once you are convinced, you can stop, and start actually using it.

Steps 2: Reading the data

Because of the way that data is organized, I have to call cleanCorpus function for each folder, and combine them.

```
#Create training corpus
corpus_train_pos <- cleanCorpus("data/train/pos")
corpus_train_neg <- cleanCorpus("data/train/neg")
corpus_train <- c(corpus_train_pos,corpus_train_neg)
training_size <- length(corpus_train_pos) + length(corpus_train_neg)

#Create testing corpus
corpus_test_pos <- cleanCorpus("data/test/pos")
corpus_test_neg <- cleanCorpus("data/test/neg")
corpus_test <- c(corpus_test_pos,corpus_test_neg)

#Combine both:
fullcorpus <- c(corpus_train,corpus_test)
```

Step 3: Document Term Matrix

```
dtm_together <- DocumentTermMatrix(fullcorpus)
dtm_together_df <- as.data.frame.matrix(dtm_together)
labels <-  c(rep("positive",length(corpus_train_pos)),
             rep("negative",length(corpus_train_neg)),
             rep("positive",length(corpus_test_pos)),
             rep("negative",length(corpus_test_neg)))
dtm_together_df <- cbind(dtm_together_df,labels)
```

Note: In actual real-world situations, you make your DocumentTermMatrix with only training data, not full corpus! In real world, you will also typically have more data.

Step 4: Building the classifier

```
max_col <- ncol(dtm_together_df)
train <- dtm_together_df[1:training_size,1:max_col-1]
#last column is the category
class_train <- dtm_together_df[1:training_size,max_col]

test <- dtm_together_df[training_size:length(fullcorpus),1:max_col-1]
class_test <- dtm_together_df[training_size:length(fullcorpus),max_col]

model.svm <- svm(train, class_train)
```

Step 5(a): Checking if the classifier is good

- predict with the svm function as in the textbook.

```
final.result <- predict(model.svm, test)

#compare predictions with actual values:
predictions <- cbind(as.data.frame(final.result), class_test)
predictions

#evalute how good this classifier is:
table(final.result, class_test)
```

Step 5(b): Trying to improve by Removing sparse terms

```
#original dtm
dtm_together <- DocumentTermMatrix(fullcorpus)

#dtm after removing sparse terms
dtm_together_2 <- removeSparseTerms(DocumentTermMatrix(fullcorpus), sparse=0.7)
#this removes 70% of the sparse terms. So, number of columns in dtm is also
#drastically reduced!

#inspect() function in tm is useful to see the differences:
inspect(dtm_together)
inspect(dtm_together_2)
```

Step 5(b): Trying to improve by Increasing training data

- I have a larger version of the data set with more examples (around 1000 examples per category)
- We can repeat this process, just changing the corpus folders in the beginning.
- The actual dataset has around 12000 examples per category.

Other uses with TM

Some useful functions in tm to explore the data-1

```
#Inspecting corpus:
inspect((corpus_train[16]))
#Just to know what is in the document.

#remove your own stopwords.
docs <- tm_map(docs, removeWords, c("word1", "word2"))
#word1, word2 will be removed from your corpus

#Get the dimensions (rows,columns) of a document term matrix:
dim(dtm_together)
```

Some useful functions in tm to explore the data-2

```
inspect(dtm_together[1:5, 1000:1005])
#Inspect a document term matrix called dtm,
#giving the dimensions we want to see.

#Removing sparse terms:
dtm <- removeSparseTerms(dtm, 0.9)
inspect(dtm_together)
```

some new R functions

```
m <- matrix(sample.int(100),5,5)
rowSums(m)
colSums(m)
rowMeans(m)
colMeans(m)
```

Some useful functions in tm to explore the data-3

```
#Getting word frequencies for entire collection (not one document)
freq <- colSums(as.matrix(dtm_together))
freq <- sort(colSums(as.matrix(dtm_together)), decreasing=TRUE)

#10 most frequent words in the corpus
freq[1:10]

#Most frequent words in a range in the corpus
findFreqTerms(dtm_together, lowfreq=100, highfreq=500)

#Finding correlations between words
findAssocs(dtm_together, "good", corlimit = 0.3)
```

Practice exercises

Working with text classification

- install the following packages: tm, SnowballC, e1071 in RStudio
- Download dataset.zip, largedataset.zip from Canvas, and the ClassificationWithTM.R file.
- Create a folder with today's date in Downloads and copy all these into that.
- Unzip both the zip files.

Exercise-1

Follow ClassificationWithTM.R file line by line, make changes where necessary, and do the following:

Get a document term matrix, perhaps only for training data (i.e., only the train folder), and look for size of the the matrix, frequent words, correlated words etc.

Exercise-2

In ClassificationWithTM.R , explore various sparsity thresholds, read tm documentation for other options, or add more examples to training data (largerdataset zip file). Check which of those changes you make gives you the best predictions for the test data.

note: Best prediction is that where diagonals in the last table() output have maximum counts.

Post on Forum

Post about what you learnt today in the discussion forum for today's date.

Next Week

- General review
- Final project descriptions and discussion
- Time to work on A4 in the class
- Mid term feedback
- To do for you: Take a look at uploaded project descriptions, think about your ideas, whether you want to work with someone else (2 people max, per team) etc.