# Homework, The Last

*Aaron Politsky*

*December 3, 2015*

## Part 1

After we load up the emails and create a Corpus, let's create the document term matrix.

```
corpus = tm_map(corpus, content_transformer(removeNumbers))
corpus = tm_map(corpus, content_transformer(removePunctuation))
corpus = tm_map(corpus, content_transformer(tolower))
corpus = tm_map(corpus, content_transformer(removeWords), stopwords("english"))
corpus = tm_map(corpus, content_transformer(stripWhitespace))

dtm = DocumentTermMatrix(corpus)
```

We'll use a sparsity value of .99 at first

```
# next we remove infrequent words
# we keep columns that are less than 0.99 percent sparse
# this is the parameter that you will need to tune in the homework
sparse_dtm = removeSparseTerms(x=dtm, sparse = 0.99)
sparse_dtm
```

```
## <<DocumentTermMatrix (documents: 5728, terms: 1636)>>
## Non-/sparse entries: 331057/9039951
## Sparsity           : 96%
## Maximal term length: 16
## Weighting          : term frequency (tf)
```

```
# convert all elements to binary
# The occurrence of the word fantastic tells us a lot
# The fact that it occurs 5 times may not tell us much more
sparse_dtm = weightBin(sparse_dtm)

# split into train and test using sampling_vector
df = as.data.frame(as.matrix(sparse_dtm))
df_train = df[sampling_vector,]
df_test = df[-sampling_vector,]
spam_train = emails$spam[sampling_vector]
spam_test = emails$spam[-sampling_vector]

library(e1071)
```

### Naive Bayes

Let's train our classification model:

```
### your code for classification goes below
nb_model = naiveBayes(df_train, spam_train)

if (file.exists("nb_train_predictions.RData")) {
  load("nb_train_predictions.RData")
} else {
  nb_train_predictions = predict(nb_model, df_train)
  save(nb_train_predictions, file = "nb_train_predictions.RData")
}
mean(nb_train_predictions == spam_train)
```

```
## [1] 0.9024443
```

```
table(actual = spam_train, predictions = nb_train_predictions)
```

```
##       predictions
## actual  ham spam
##   ham  3071  431
##   spam   16 1064
```

```
# compute test error
if (file.exists("nb_test_predictions.RData")) {
  load("nb_test_predictions.RData")
} else {
  nb_test_predictions = predict(nb_model, df_test)
  save(nb_test_predictions, file = "nb_test_predictions.RData")
}
```

Our test accuracy is:

```
mean(nb_test_predictions == spam_test)
```

```
## [1] 0.8970332
```

```
table(actual = spam_test, predictions = nb_test_predictions)
```

```
##       predictions
## actual ham spam
##   ham  747  111
##   spam   7  281
```

## Stemming

Let's try Stemming

```
# also try stemming as a preprocessing step
stemmed = tm_map(corpus, stemDocument, language = "english")
stemmed.dtm = DocumentTermMatrix(stemmed)
stemmed.dtm = removeSparseTerms(x=stemmed.dtm, sparse = 0.99)
```

```
stemmed.dtm = weightBin(stemmed.dtm)
stemmed_df = as.data.frame(as.matrix(stemmed.dtm))
stemmed_df_train = stemmed_df[sampling_vector,]
stemmed_df_test = stemmed_df[-sampling_vector,]

# train model on stemmed corpora
model_stem = naiveBayes(stemmed_df_train, spam_train)
if (file.exists("nb_test_predictions_stem.RData")) {
  load("nb_test_predictions_stem.RData")
} else {
  nb_test_predictions_stem = predict(model_stem, stemmed_df_test)
  save(nb_test_predictions_stem, file = "nb_test_predictions_stem.RData")
}
```

Our test accuracy using stemming is:

```
mean(nb_test_predictions_stem == spam_test)
```

```
## [1] 0.9144852
```

```
table(actual = spam_test, predictions = nb_test_predictions_stem)
```

```
##       predictions
## actual ham spam
##   ham  769   89
##   spam   9  279
```

## Different values of Sparsity

Let's try a range of sparsity values:

```
# try different values of sparsity
stemmed.dtm = DocumentTermMatrix(stemmed)
sparsity <- seq(0.9, 0.99, by = .02)
dtms.by.sparsity <- lapply(sparsity, function(sp) {
  as.data.frame(as.matrix(weightBin(removeSparseTerms(x=stemmed.dtm, sparse = sp))))
})

if (file.exists("nb.test.predictions.by.sparsity.Rda")) {
  load("nb.test.predictions.by.sparsity.Rda")
} else {
  nb.test.predictions.by.sparsity <-
    lapply(dtms.by.sparsity, function(dtm) {
      df.train = dtm[sampling_vector,]
      df.test  = dtm[-sampling_vector,]
      nb_model = naiveBayes(df.train, spam_train)
      nb_test_predictions = predict(nb_model, df.test)
    })
  save(nb.test.predictions.by.sparsity, file = "nb.test.predictions.by.sparsity.Rda")
}
names(nb.test.predictions.by.sparsity) <- sparsity
```

Our test accuracy as a function of sparsity is:

```
sapply(nb.test.predictions.by.sparsity, function(nb_test_predictions)
  mean(nb_test_predictions == spam_test)
)
```

```
##       0.9      0.92      0.94      0.96      0.98
## 0.8952880 0.8926702 0.8952880 0.9162304 0.9354276
```