Data Science Capstone: Milestone Report

Introduction

For this project, we will use natural language processing tools to examine a Corpus from the Capstone Dataset, available at *https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/Coursera-SwiftKey.zip*, in order to build a predictive text tool. The corpus that we will use will be built off of three data sources, showing data from blogs, news stories, and tweets. This report will describe some preliminary analysis on the dataset, and will outline a strategy for building a predictive text algorithm.

Summary Statistics

Here is some high level information for each file:

1. **en_US.blogs.txt**: 210MB, 900 thousand lines of text, 37 million words
2. **en_US.news.txt**: 206MB, 2.4 million lines of text, 34 million words
3. **en_US.twitter.txt**: 167MB, 2.4 million lines of text, 30 million words

Sampling and preprocessing

For our preliminary analysis, we will randomly sample 50,000 lines of text from each document. We will convert all text to lower case, and remove all punctuation and numbers, using the built in flunctionality from the `tm` package.

Word frequencies

Here we see the top twelve most frequently occurring words in the blogs sample, along with their frequency:

```
tf_blogs[1:12]
```

```
##         the         and        that         for         you
## 0.063799462 0.037867266 0.015842460 0.012554005 0.009943767 0.00986
##         was        this        have         but         are
## 0.009698997 0.008388460 0.007650964 0.006756662 0.006654037 0.00608
```

Here are the top twelve most frequently occurring words in the news sample, along with their frequency:

```
tf_news[1:12]
```

```
##         the         and         for        that        with
## 0.069461851 0.032710602 0.012942788 0.012477841 0.009351187 0.00872
##        said         his        from        have         are
## 0.007968462 0.005986759 0.005542258 0.005293125 0.005106843 0.00482
```

```
tf_twitter[1:12]
```

```
##         the         you         and         for        that
## 0.041347451 0.022070168 0.019435745 0.017688267 0.009855426 0.00801
##        your        have         are        this        just
## 0.007636965 0.007401473 0.006976708 0.006617969 0.005684807 0.00547
```

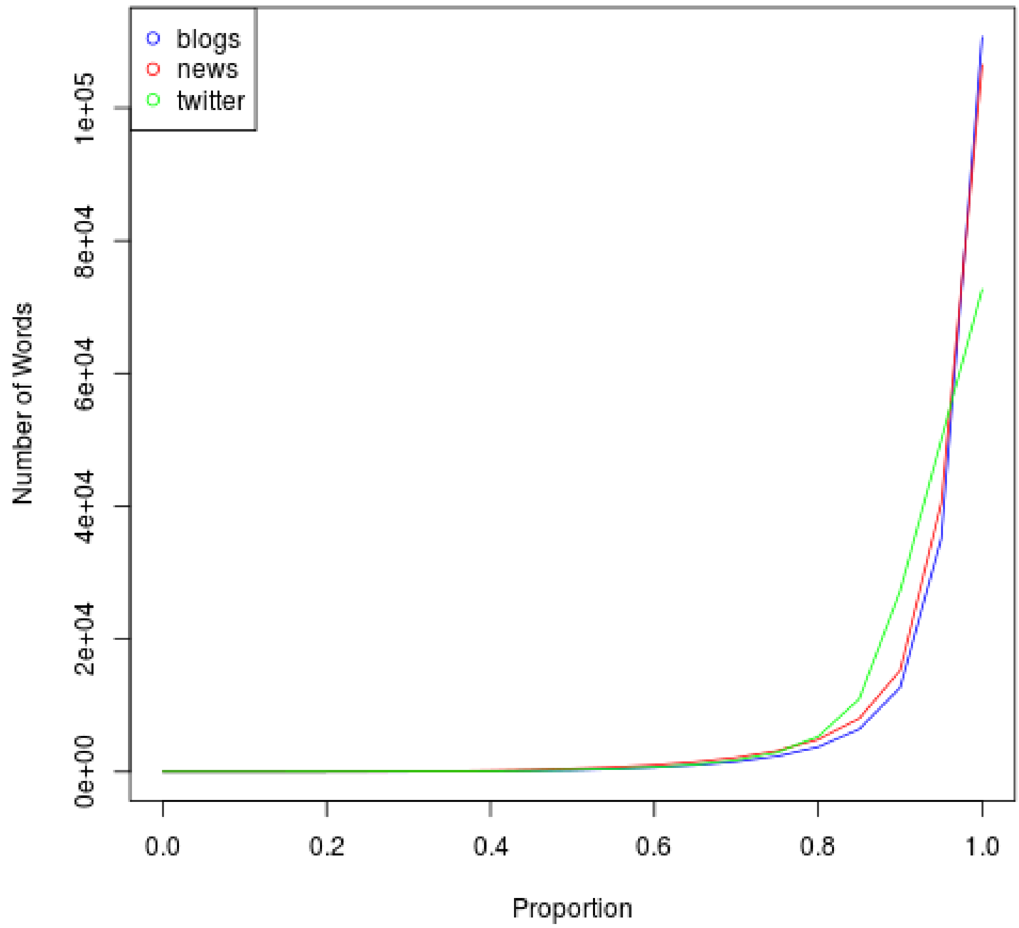
One might wonder how many distinct words must be included to account for a certain proportion of the total words in each data sample. For example, we see that 267 distinct words account for 50% of the words in the blogs sample, while we must include 12,670 words to account for 90% of the words in the same sample:

```
words_cutoff(tf_blogs,0.5)
```

```
## [1] 267
```

```
words_cutoff(tf_blogs,0.9)
```

```
## [1] 12670
```



Bigram and trigram analysis

We will now perform some basic bigram and trigram analysis using the DocumentTermMatrix functionality in the `tm` packages.

Here we see the most common bigrams (pairs of words) for each dataset, along with the number of occurrences for each bigram in the sample:

```
head(sort(blogs_dtm2_sums,decreasing = T))
```

```
##  of the  in the  to the  on the   to be and the
##   10233    8467    4841    4050    3637    3154
```

```
head(sort(news_dtm2_sums,decreasing = T))
```

```
##  of the  in the  to the  on the for the  at the 
##    9174    8761    4211    3526    3368    2842
```

```
head(sort(twitter_dtm2_sums,decreasing = T))
```

```
##  in the for the  of the  on the   to be  to the 
##    1608    1541    1161     966     960     935
```

Here we see the most common trigrams (triplets of words) for each dataset, along with the number of occurrences for each trigram in the sample:

```
head(sort(blogs_dtm3_sums,decreasing = T))
```

```
##   a lot of one of the the end of out of the    to be a as well as
##        718        710        370        362        357        356
```

```
head(sort(news_dtm3_sums,decreasing = T))
```

```
##  one of the    a lot of  as well as part of the  the end of     to
##         658         515         295         293         286
```

```
head(sort(twitter_dtm3_sums,decreasing = T))
```

```
##     thanks for the        going to be looking forward to 
##                172                161                146 
##     for the follow       cant wait to            to be a 
##                138                129                113
```

Next steps

In our next step, will use the n-gram analysis data to create a predictive model for user entered text. For example, we could use our bigram data to predict the most likely bigram the starts with the previous word entered. Likewise, we could use our trigram data to predict the most likely trigram that starts with the previous two words entered. The best approach will make use of the unigram, bigram, and trigram data, and will require that we predict n-grams that do not appear in our dataset. One possible strategies would involve linear interpolation on our unigram, bigram, and trigram data. Another possible strategy would involve the use of discounting methods to assign probabilities to n-grams that do not appear in our dataset.

We will also want to make use of larger samples, or even the entire data set to make better predictions. In order to do this, we will need to optimize some of our data processing code, and we will also need to allow our code to run for longer periods of time.

Eventually, we will want to make use of even more sophisticated techniques to improve upon our n-gram models, but it will be important to keep things simple, so that our code runs in a reasonable amount of time for the end user.