Let Me Finish Your Sentence For You

by: Mia

March 17, 2015

Milestone report

Ever wanted someone to finish your sentence for you? Who completes your thoughts and is always on the same page as you? Look no further than my app, who is probably way more reliable anyway.

Let's cover our bases first. I've created this milestone report to report on my progress in building a shiny app and predictive model that predicts the next word from user provided text input. For the predictive model, I've explored probabilistic models (i.e. N-grams) and rule-based models.

Agenda for this milestone

- Demonstrate that you've successfully loaded and manipulated data
- Perform exploratory analysis of the data set
- Report any interesting findings that you amassed so far
- Illustrate important summaries of the data set using tables and plots
- My goals for the eventual Shiny app and predictive algorithm

Data

The data provided contain samples of tweets, blogs and news. The initial dataset consists of data sources in four different languages: English, Finnish, Russian and German. Here we will consider the English resources.

A basic data table containing summary statistics including: line counts, word counts, and size

```
##         line counts word counts document size
## twitter     2360148    30373603     166816544
## news        1010242    34372530     205243643
## blogs        899288    37334147     208623081
```

Some initial thoughts

- twitter is short, as expected since a tweet can have a maximum of 140 characters. we see tons of informal characters and less grammar, which means more noise
- news is written with better grammar & structure, and topics are focused
- blogs do not have as much noise as twitter and cover more topics than the news

- blogs have the largest document size and word count, which will help build a better model for prediction

Since loading the 3 files (news, twitter, and blog) may be too large, the blog data will be the best bet for building a model. For the sake of size, I will use a subset the first 300,000 lines in the blog data set.

Exploratory analysis

I conducted an N-grams analysis, with Uni and Bigrams. First, I checked the frequencies of the most used words. I've focused on the tm (text mining) and RWeka libraries for the initial exploration.

Tokenization

```
stopifnot(require(tm,        quietly=TRUE))   # framework for text mining
## Warning: package 'NLP' was built under R version 3.1.2
stopifnot(require(stringi,   quietly=TRUE))   # framework for advanced string
manipulation
## Warning: package 'stringi' was built under R version 3.1.2
stopifnot(require(SnowballC, quietly=TRUE))   # framework for stemming
stopifnot(require(RWeka,     quietly=TRUE))   # framework for n-grams
## Warning: package 'RWeka' was built under R version 3.1.2
stopifnot(require(ggplot2,   quietly=TRUE))   # framework for plots
##
## Attaching package: 'ggplot2'
##
## The following object is masked from 'package:NLP':
##
##     annotate
library(png)
library(grid)
library(RCurl)
## Warning: package 'RCurl' was built under R version 3.1.2
## Loading required package: bitops
library(bitops)

## create a UnigramTokenizer (RWeka)
UnigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 1, max =
1))

## create a BigramTokenizer (RWeka)
BigramTokenizer <- function(x) NGramTokenizer(x, Weka_control(min = 2, max =
2))
```

Clean data

```
## load blog data
ensubset <-
VCorpus(DirSource(directory="~/Desktop/DataScienceGit/Capstone/final/en_US/su
bset", encoding="UTF-8"), readerControl=list(reader=readPlain))
## remove punctuation
```

```
ensubset <- tm_map(x=ensubset, FUN=removePunctuation)
## remove numbers
ensubset <- tm_map(x=ensubset, FUN=removeNumbers)
## stopwords
ensubset <- tm_map(x=ensubset, FUN=removeWords, words=stopwords(kind="en"))
## remove extra white spaces
ensubset <- tm_map(x=ensubset, FUN=stripWhitespace)
## switch to lowercase
ensubset <- tm_map(x=ensubset, FUN=tolower)
ensubset <- tm_map(x=ensubset, PlainTextDocument)
```

Create a TermDocumentMatrix

```
tdmUnigram <- TermDocumentMatrix(ensubset,
control=list(tokenizer=UnigramTokenizer))
tdmBigram <- TermDocumentMatrix(ensubset,
control=list(tokenizer=BigramTokenizer))
```

Stats collected

- how many words (stemmed at this point) do we have in the dictionary?

```
tdmUnigram$nrow
## [1] 112164
```

- how many different 2-words expressions do we have?

```
tdmBigram$nrow
## [1] 1220832
```

- which are the most frequent words? for sake of space, I'll only show the first 10

```
head(findFreqTerms(tdmUnigram, 160),10)
##  [1] "ability"    "able"       "absolutely" "accept"     "access"
##  [6] "according"  "account"    "across"     "act"        "action"
```

- which are the most used 2-words expressions? for sake of space, I'll only show the first 10

```
head(findFreqTerms(tdmBigram, 10),10)
##  [1] "a beautiful" "a big"       "a bit"       "a couple"    "a day"
##  [6] "a days"      "a friend"    "a girl"      "a good"      "a great"
```

Apply same methodology to stemmed words

```
ensubset <- tm_map(x=ensubset, FUN=stemDocument)
tdmUnigramStemmed <- TermDocumentMatrix(ensubset,
control=list(tokenizer=UnigramTokenizer))
tdmBigramStemmed <- TermDocumentMatrix(ensubset,
control=list(tokenizer=BigramTokenizer))
```

Same stats for stemmed words

- how many words (stemmed at this point) do we have in the dictionary?

```
tdmUnigramStemmed$nrow
## [1] 86742
```

- how many different 2-words expressions do we have?

```
tdmBigramStemmed$nrow
## [1] 1106577
```

- which are the most frequent words? for sake of space, I'll only show the first 10

```
head(findFreqTerms(tdmUnigramStemmed, 160),10)
##  [1] "100"     "2008"    "2009"    "2010"    "2011"    "2012"    "abil"
##  [8] "abl"     "absolut" "abus"
```

- which are the most used 2-words expressions? for sake of space, I'll only show the first 10

```
head(findFreqTerms(tdmBigramStemmed, 40),10)
##  [1] "1 cup"    "10 minut" "10 year"  "12 cup"   "14 cup"   "15 minut"
##  [7] "2 hour"   "2 week"   "2 year"   "20 minut"
```

Interesting findings so far

Since the stemmed vocabulary (size of Unigrams) is a sufficiently large set, we might not need to utilize the extent of the initial data set, when analyzing Unigrams. Regarding Bigrams, the difference between stemmed vs. non-stemmed sets of words was negligible. In regards to the very-high frequency Bigrams, they seem to be composed of very short simple words (which look the same stemmed).

Plots of frequencies, for Unigrams and Bigrams

```
unigrams <- inspect(tdmUnigram)
tblUnigrams <- table(unigrams)
hist (log(tblUnigrams), main="Histogram of Unigrams", breaks=50,
xlab="Unigrams")
```

```
bigrams <- inspect(tdmBigram)
tblBigrams <- table(bigrams)
hist (log(tblBigrams), main="Histogram of Bigrams", breaks=50,
xlab="Bigrams")
```

Next steps

Going forward, I will be basing my predictive algorithm and shiny app on an n-gram frequency approach. Also, these findings would encourage the use of stemming, since the matrix would be a lot smaller. Two hurdles for the project I'm experiencing so far: #1 the size of the prediction model, and #2 ensuring adequate phrase coverage. This strategy may evolve as my investigation continues.