Next Word Prediction

Data Science Course#10 Capstone Course Project

SH

Overview

This is a course project aiming at learning natural language processing and developing a shiny application for next word prediction. Dictionaries of unigrams, bigrams, trigrams, and quadgrams were pregenerated from text files "en_US.blogs.txt" and "en_US.news.txt". The Simple Stupid Backoff algorithm was used for the next word prediction. In the application, when you type in a phrase(s)/sentence(s), the application would give you up to five sugestions of the likely next word.

How The Model Works and What Affects Its Predictive Performance

The Simple Stupid Backoff algorithm is based on the fact that phrases have short-term memory, and thus, the likely next word can be predicted from only a few immediate previous words. In this application, the quadgram dictionary is first looked up for the word. If not found, the trigram dictionary is used, then the bigram dictionary. If still not found, five most-frequently-occurring words in the unigram dictionary are presented as the suggest word(s).

It is conceivable that the predictive accuracy of this model depends heavily on the size and quality of the ngram dictionaries. In other words, building a high quality and good size ngram dictionary is the key to its performance, which requires assembly of many (training) text files processed by using a good memory-ample computer. Similarity of the training text files and the input phrases is another key for a successful prediction. (Due to this consideration, file "en_US.twitter.txt" was not used for dictionary building.) Finally, speed is another key for a good application, which was mitigated via an alphabet indexing method for ngram lookup.(It's still somewhat slow, though.)

The Application

Type in a few words in the "Input Word(s)" box. Hit the "Go!" button. The suggested next word will appear in the "Suggested Next Word" box. Note that while the suggested words is ordered from the most- to the least-frequently-occurring word, the probability of the word appearing is not calculated (which is not required by this project).

Link to the application: https://sh2016.shinyapps.io/appnextwordprediction/

(Improvement ideas: Write more interface/code so that users can submit phrases to keep improving the ngram dictionaries and hence the prediction accuracy.)

R Code Examples in the App

```
library(tm); library(text2vec) # ...
# remove URLs and non-ascii characters and punctuations
removeURL <- function(x) {...}; removePunc <- function(x) {...}
# build unigram dictionary
tokens <- newsT$content %>% tolower() %>% word_tokenizer()
it <- itoken(tokens, ids=newsT$id)
vocab <- create_vocabulary(it, ngram=c(1L,1L)) %>% prune_vocabulary(term_count_min=2)
vectorizer <- vocab_vectorizer(vocab)
dtm <- tokens %>% itoken() %>% create_dtm(vectorizer)
# next word prediction
sentencePredict <- function(aString) { # ...
  predict.uni <- function() { ... }; predict.bi <- function(aWord) { ... }
  predict.tri <- function(twoWords) { ... };
  predict.quad <- function(threeWords) { # ...
    # call predict.tri(twoWords) ... if quadgram not found, and so on
    # ...
  } }
```