Next Word Prediction

Riccardo Finotello

08/10/2020

Synopsis

The goal of the project is to build a webapp capable of predicting the next word given an incomplete sentence as input. Here we present the initial exploratory data analysis performed on the dataset provided by SwiftKey. We show how we build the training and the test folds and some basic reference plots showing how they are formed.

Corpora

After downloading the datasets we use a biased coin toss to select:

1. up to 5% of the entries in the news, blogs and Twitter corpora for the **training** set,
2. up to 1% of the entries in the news, blogs and Twitter corpora for the **test** set.

In total we select more than 5×10^5 sentences from the sets.

The training set is thus formed by:

Text	Types	Tokens	Sentences
blogs.txt	90699	2114459	103327
news.txt	69582	1193877	56506
twitter.txt	52825	726873	51379

While the test set is made by:

Text	Types	Tokens	Sentences
blogs.txt	36109	428679	20899
news.txt	37072	391008	18420
twitter.txt	22331	181121	12804

Tokens

Tokenisation is made possible through the library *quanteda*. In the process we remove punctuation, symbols and separators to prune the dataset from elements we are not interesting in learning. However we keep most social media related objects and numbers.

The training tokens have also been pruned of profanity and bad words. We use this list as source for the bad words appearing in the sentences (the list was formerly used by Google).

The training tokens are thus formed as:

file	tokens
blogs.txt	1856996
news.txt	1029209
twitter.txt	592754

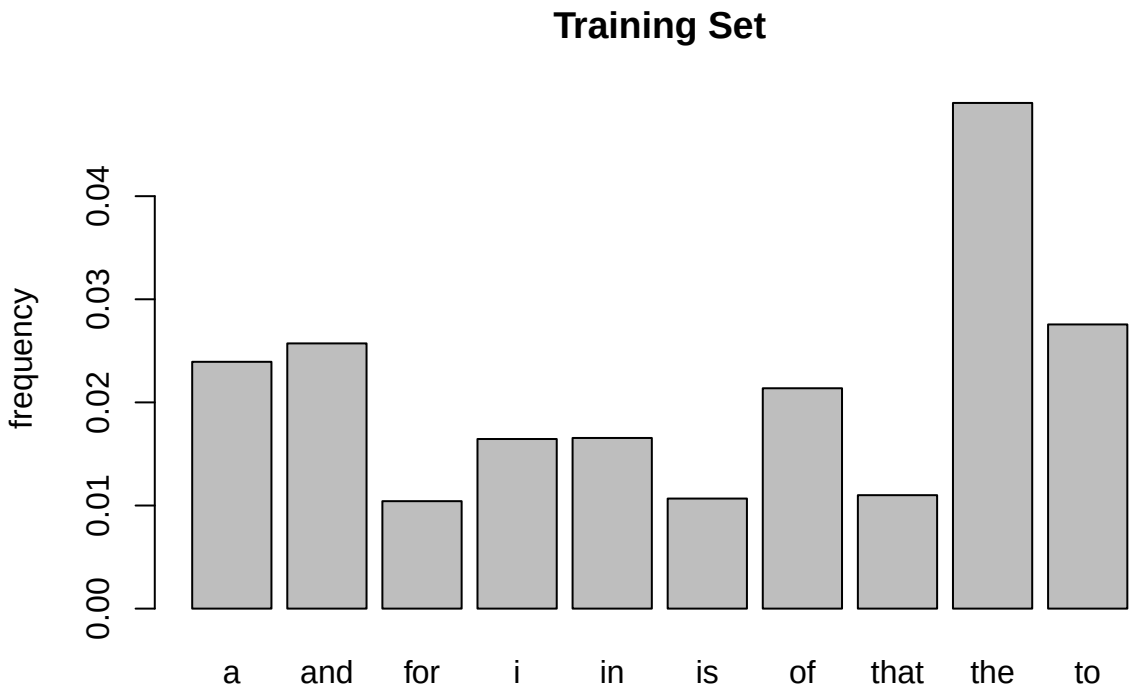
While the training set is made of:

file	tokens
blogs.txt	377257
news.txt	337274
twitter.txt	148733

Frequency Estimation

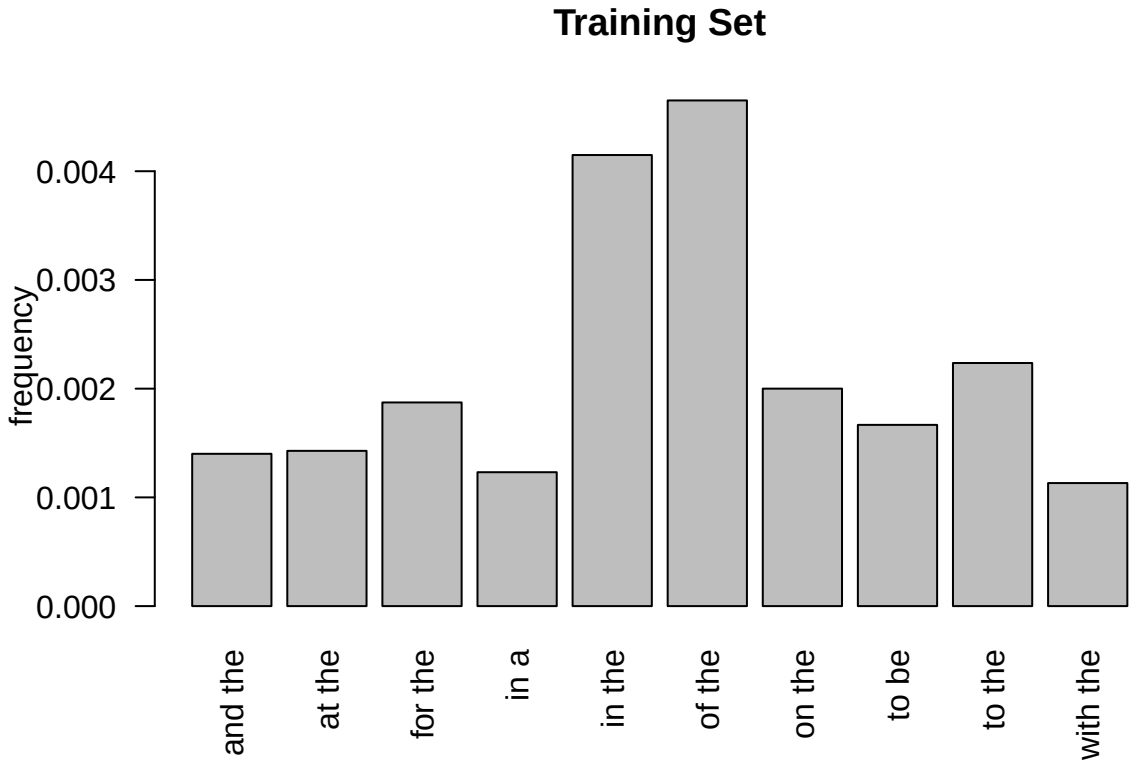
Before moving further with the analysis we show the distributions in frequency of words and basic n-grams (groups of words).

In fact the most used words in the **training** set are:



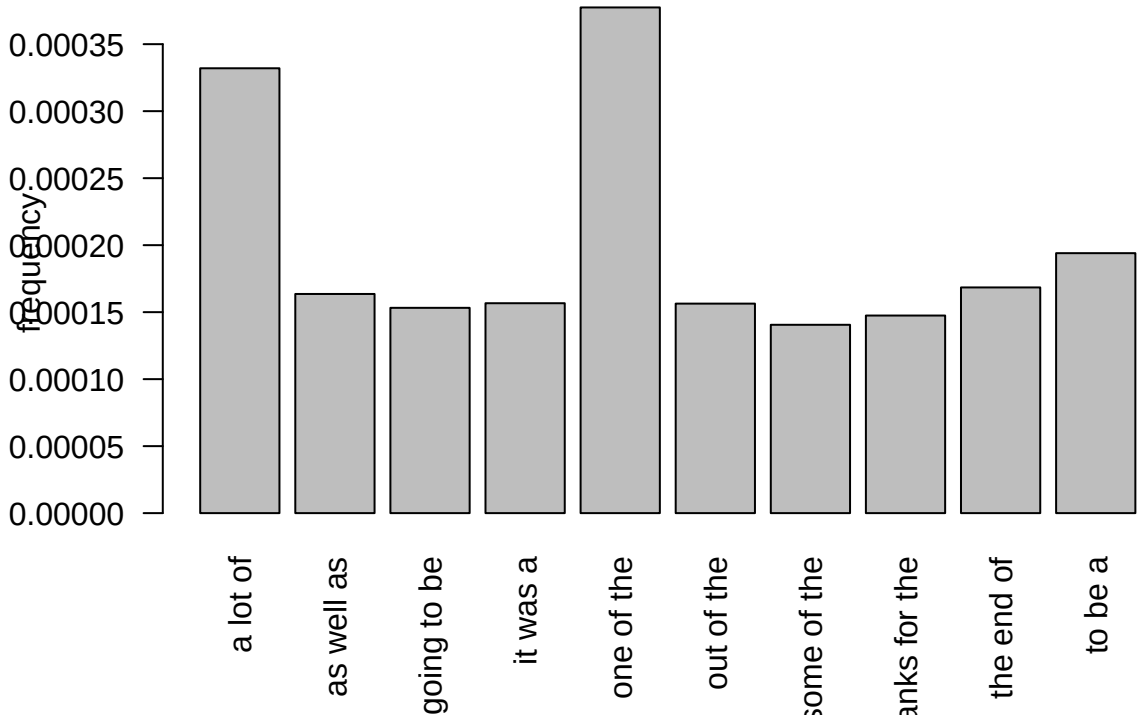
We can clearly see that the most frequent words are therefore stop words in English, as expected.

At the same time we can also take a look at the occurrences of common bigrams in English:



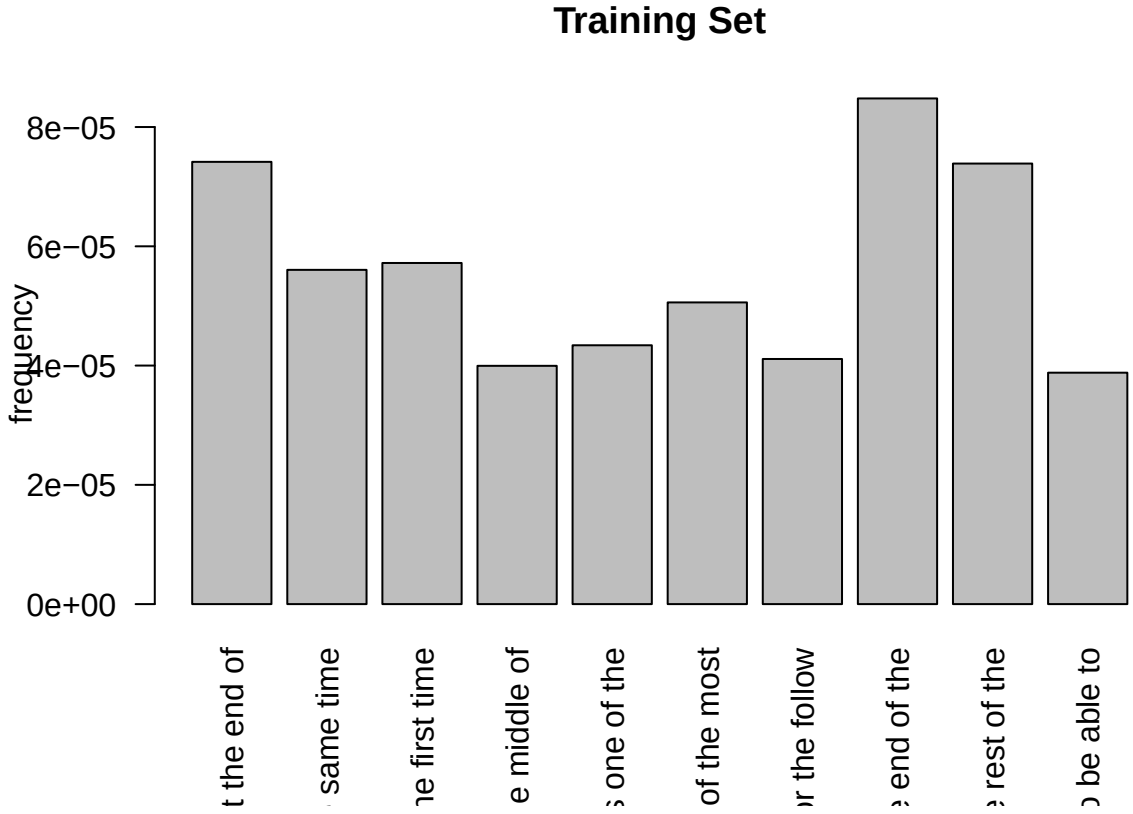
Which again shows that conjunctions and articles are definitely more common than other words. We will therefore need to be extremely careful when predicting these words: they must be precise in order to build the right sentence.

We can then do the same for groups of three words in English:



We finally start to recognise full constructions in English. The web app will therefore need to capture these features.

Groups of four words are finally showing a hint of more complex sentences:



Future Directions

From what we have shown, it seems that the main challenge will be the estimation of the probability that a random word follows a given group of 4, 3, or 2 other words. We will most probably implement a version of the Kneser-Ney smoothing algorithm, since a simple estimation of the frequency might result to be too inaccurate. In general it seems that the choice of the training set as presented here might be a good starting point for the analysis.