Capstone Project - Exploratory Data Analysis

Pedro A. Alonso Baigorri

4 de julio de 2018

Objective

The objective of this project is to apply data science in the area of natural language processing to create a prototype of a Text Predictor application similar to what the mobile text editors are using when they suggest some text to introduce based on previous words.

To create this application we are going to use a dataset ("Corpora") that includes texts collected from twitter, blogs and news data sources.

In this document I will report the result of the exploratory data analysis of the dataset and the planned strategies to develop the text predictor.

Opening the dataset

As I mentioned before the dataset is composed by 3 different text files:

- en_US.twitter.txt
- en_US.blogs.txt
- en_US.news.txt

I have built some R code to read these files and load them into R objects in a efficent way, saving the objects locally so, the next time I need to load the objects I can avoid to download and read the text files again.

Dataset Basic Summary

Once I have loaded the dataset I can get some basic features such as:

- the number of lines:

```
summary(dataset)
```

```
##         Length  Class  Mode
## twitter 2360148 -none- character
## blogs    899288 -none- character
## news      77259 -none- character
```

- the number of words:

```
library(ngram)
wordcount(dataset$twitter)
```

```
## [1] 30373543
```

```
wordcount(dataset$blogs)
```

```
## [1] 37334131
```

```
wordcount(dataset$news)
```

```
## [1] 2643969
```

Cleaning the dataset

To manipulate the dataset I'm going to use the library "corpus" which has several functions for NLP such as the obtention of datagrams.

when loading the dataset into corpus objected by default the library is doing some cleaning and processo operations, such us manage the case issued and other possibilities. The by default options are the following:

```
library(corpus)
corpusList = list("twitter", "blogs", "news")

corpusList$twitter <- as_corpus_text(dataset$twitter)
corpusList$blogs <- as_corpus_text(dataset$blogs)
corpusList$news <- as_corpus_text(dataset$news)

text_filter(corpusList$news)
```

```
## Text filter with the following options:
##
##     map_case: TRUE
##     map_quote: TRUE
##     remove_ignorable: TRUE
##     combine: NULL
##     stemmer: NULL
##     stem_dropped: FALSE
##     stem_except: NULL
##     drop_letter: FALSE
##     drop_number: FALSE
##     drop_punct: FALSE
##     drop_symbol: FALSE
##     drop: NULL
##     drop_except: NULL
##     connector: _
##     sent_crlf: FALSE
##     sent_suppress:  chr [1:155] "A." "A.D." "a.m." "A.M." "A.S." "AA." ...
```

To decide if other type of processing is required, I will obtain the 3-grams of one of the dataset to see the current results:

```
term_stats(corpusList$news, ngrams = 2:3)
```

```
##    term    count support
## 1  of the  13993   12025
## 2  in the  13423   11594
## 3  , "     12055   11014
## 4  . "     15033    9852
## 5  , and    8610    7818
## 6  , the    8121    7350
## 7  said .   6933    6807
## 8  . the    6654    6026
## 9  to the   6384    5929
## 10 â €     10348    5514
## 11 on the   5498    5097
## 12 for the  5349    5038
## 13 , but    5120    4890
## 14 €       8007    4705
```

```
## 15 , a      4597    4291
## 16 at the   4488    4203
## 17 in a     3989    3809
## 18 and the  3990    3804
## 19 to be    3547    3311
## 20 €  s    4487    3284
## <U+22EE>(3075571 rows total)
```

As can be observed, there area some problems due to punctuations and symbols that can be a problem for the predictor. So I will remove all these components in all the datasets:

```
text_filter(corpusList$twitter)$drop_symbol = TRUE
text_filter(corpusList$twitter)$drop_number = TRUE
text_filter(corpusList$twitter)$drop_punct = TRUE

text_filter(corpusList$blogs)$drop_symbol = TRUE
text_filter(corpusList$blogs)$drop_number = TRUE
text_filter(corpusList$blogs)$drop_punct = TRUE

text_filter(corpusList$news)$drop_symbol = TRUE
text_filter(corpusList$news)$drop_number = TRUE
text_filter(corpusList$news)$drop_punct = TRUE
```

Plots and advanced features of the dataset (n-grams)

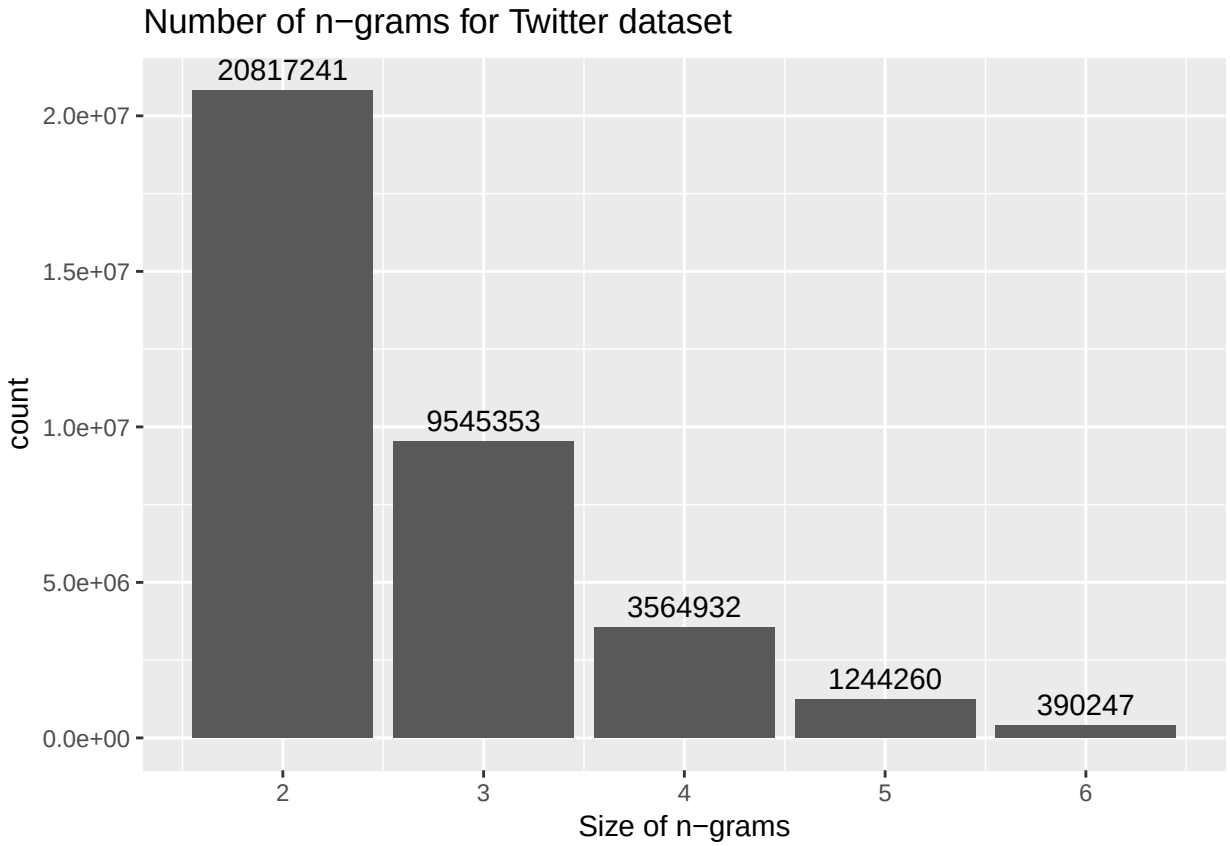
To create the text predictor is required to see what type of n-grams can be obtained from the dataset.

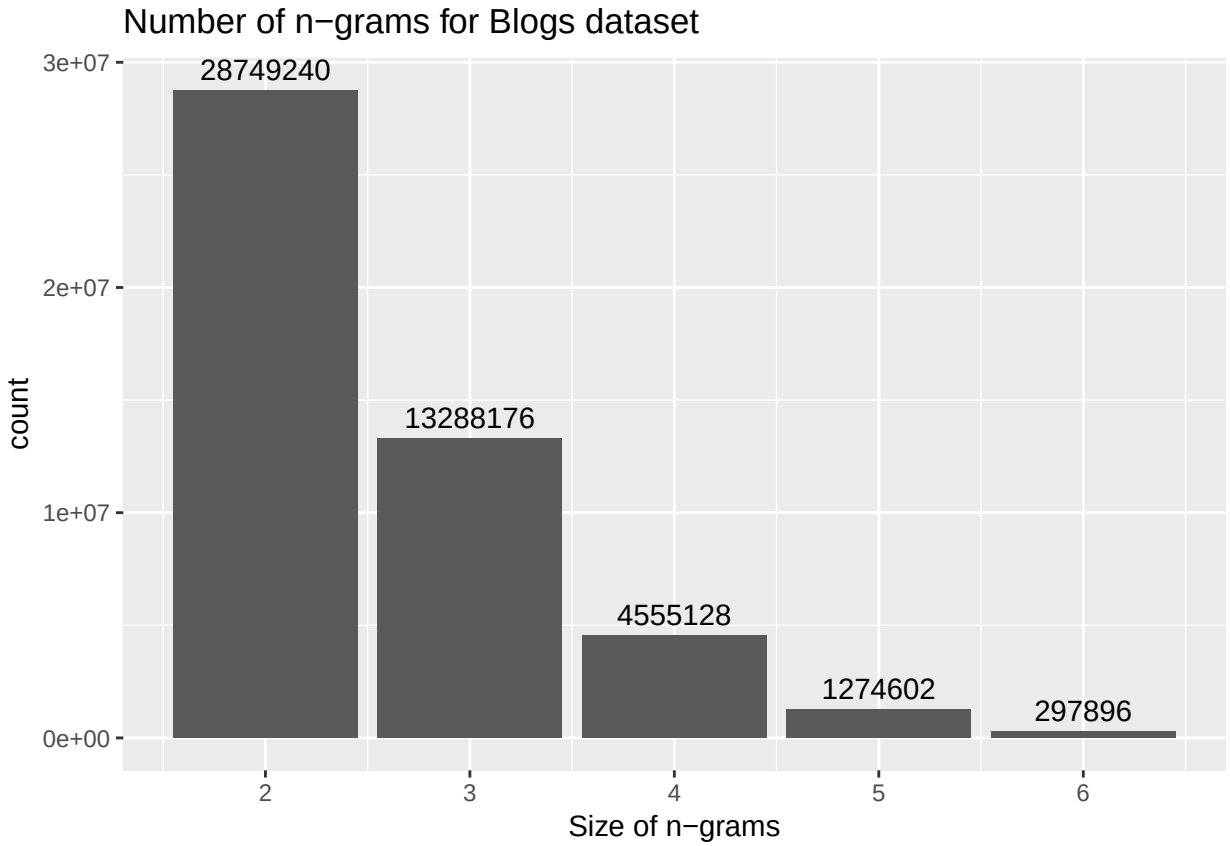
To do this, I'm obtaining the ngrams from 2 to 8 with a minimum of 2 concurrences and I will plot the number of occurrences per type of ngram.

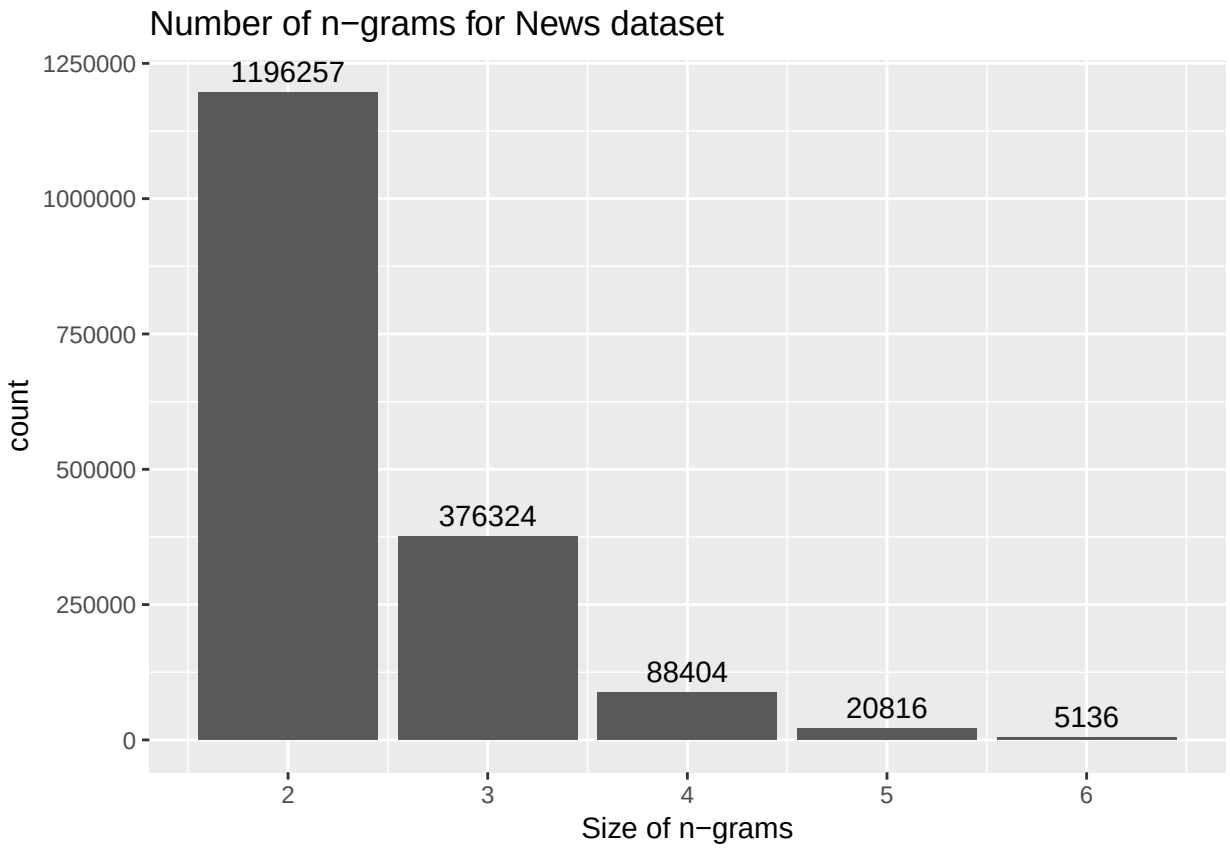
First of all I'll take a sample of 50% of the corpus to save time and memory. And then I'm plotting the number of occurrences of the different sizes of n-grams.

```
ngramsList <- list("twitter", "blogs", "news")

ngramsList$twitter <- term_stats(corpusList$twitter, ngrams = 2:6, min_count = 2, types = TRUE)
ngramsList$blogs <- term_stats(corpusList$blogs, ngrams = 2:6, min_count = 2, types = TRUE)
ngramsList$news <- term_stats(corpusList$news, ngrams = 2:6, min_count = 2, types = TRUE)
```


Number of n−grams for Twitter dataset
20817241
9545353
3564932
1244260
390247
2.0e+07
1.5e+07
1.0e+07
5.0e+06
0.0e+00
count
2
3
4
5
6
Size of n−grams


Number of n−grams for Blogs dataset
3e+07
2e+07
1e+07
0e+00
count
28749240
13288176
4555128
1274602
297896
2
3
4
5
6
Size of n−grams



According to these results I will select for the text predictor only the n-grams from 2 to 5, since the rest are less representative.

Strategies to build the text predictor

Based on previous results my strategy to build the text predictor can be summarized with the following scheme:

I will create several modules to get a list of potential words with probability based on counting the occurrences of different potential n-grams, where n can be from 5 to 2.

Then I will apply a weight to the different outputs, 0.8 for the higher ngram, and 0.1 for the lower, and then I will obtain the final probability for each word. Then the word with the highest probability will be selected. I will also use a combination of a sample of the three corpus.

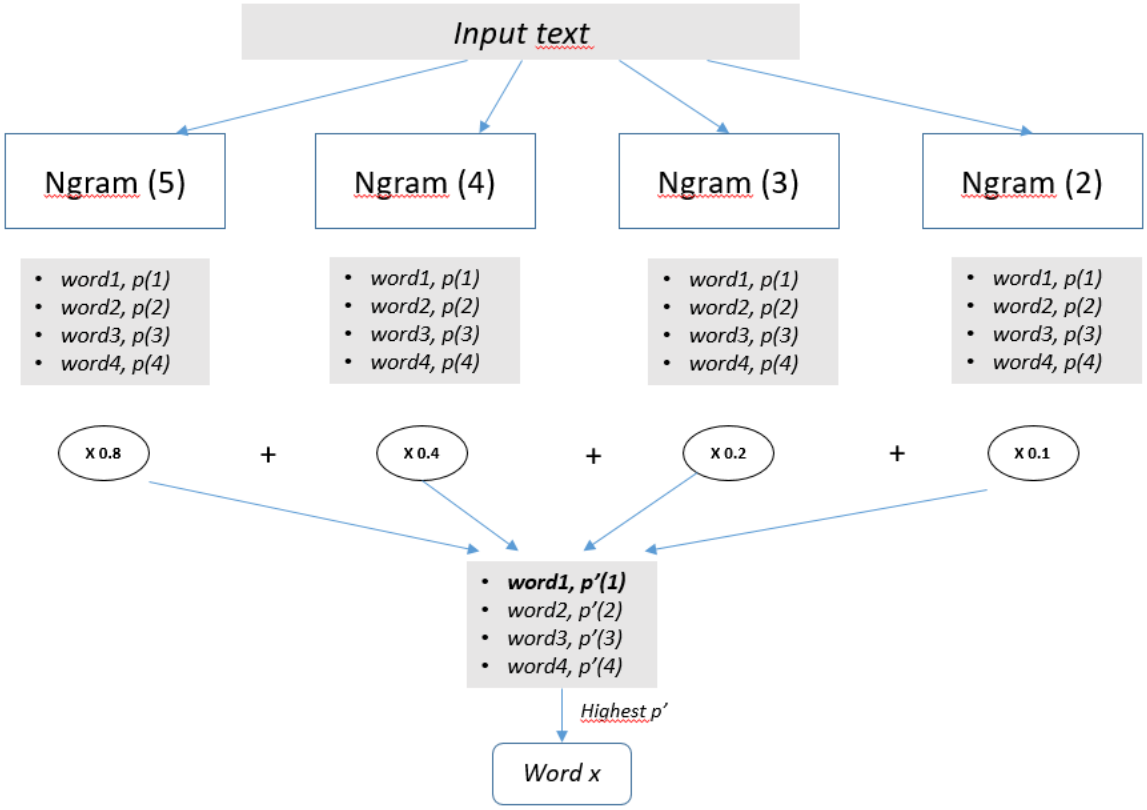

Input text
Ngram (5)
Ngram (4)
Ngram (3)
Ngram (2)
word1, p(1)
word2, p(2)
word3, p(3)
word4, p(4)
X 0.8
+
X 0.4
+
X 0.2
+
X 0.1
word1, p'(1)
word2, p'(2)
word3, p'(3)
word4, p'(4)
Highest p'
Word x

Figure 1: