Capstone report 1

Paul van der Kooy

11/14/2016

Synopsis

This report describes the approach and activities for the Data Sciences specialization Capstone project. The objective of the project is to develop a text prediction application like those used on mobile telephones and app's like WhatsApp. This report covers the initial data analysis and high-level plans for the application. It covers:

1 Loading the data
2 Capture and report key statistics of the data sets
3 Take a workable sample of the data
4 Clean and preprocess the data
5 Apply profanity filter
6 Tokenization
7 Frequency analysis and 2 and 3-grams
8 Findings & Conclusions
9 Way forwards

Exploratory Analysis

Initial exploratory analysis was done by reading a lot about NLP from different sources on the Internet. Familiarized myself with some on the R packages providing NLP functionality `NLP, OpenNLP, RWeka, tm, tokenizers`. Next the text for the 3 input datasets were read to investigate its basic properties and format. Results of this analysis can be found in the table below. Due to the significant size of the input data the analysis was made on a sample size of 5% of the 3 input files.

Sample characteristics:

statistic	Twitter	News	Blogs	unit
Original size	167	206	210	Mb
Max. record length	140	1105	2784	characters
Records in sample	118007	50512	44964	records

Pre-processing

For cleansing and tokenization of the text the `tm` package in R was utilized plus regular expressions using the `gsub` function. To make the texts ready for natural language processing the following steps were executed to clean up the data:

- Conversion to lowercase
- Profanity removal
- Break records/lines into seperate sentences
- Remove redundant whitespace and tabs
- Remove numbers
- Remove special characters and punctuations

For the profinity filter a "bad word" list is used from the following source: https://raw.githubusercontent.com/LDNOOBW/List-of-Dirty-Naughty-Obscene-and-Otherwise-Bad-Words/master/en

To tokenize the dataset the `termFreq` function was used to determine word counts and frequencies.

More statistics after the preprocessing of the text:

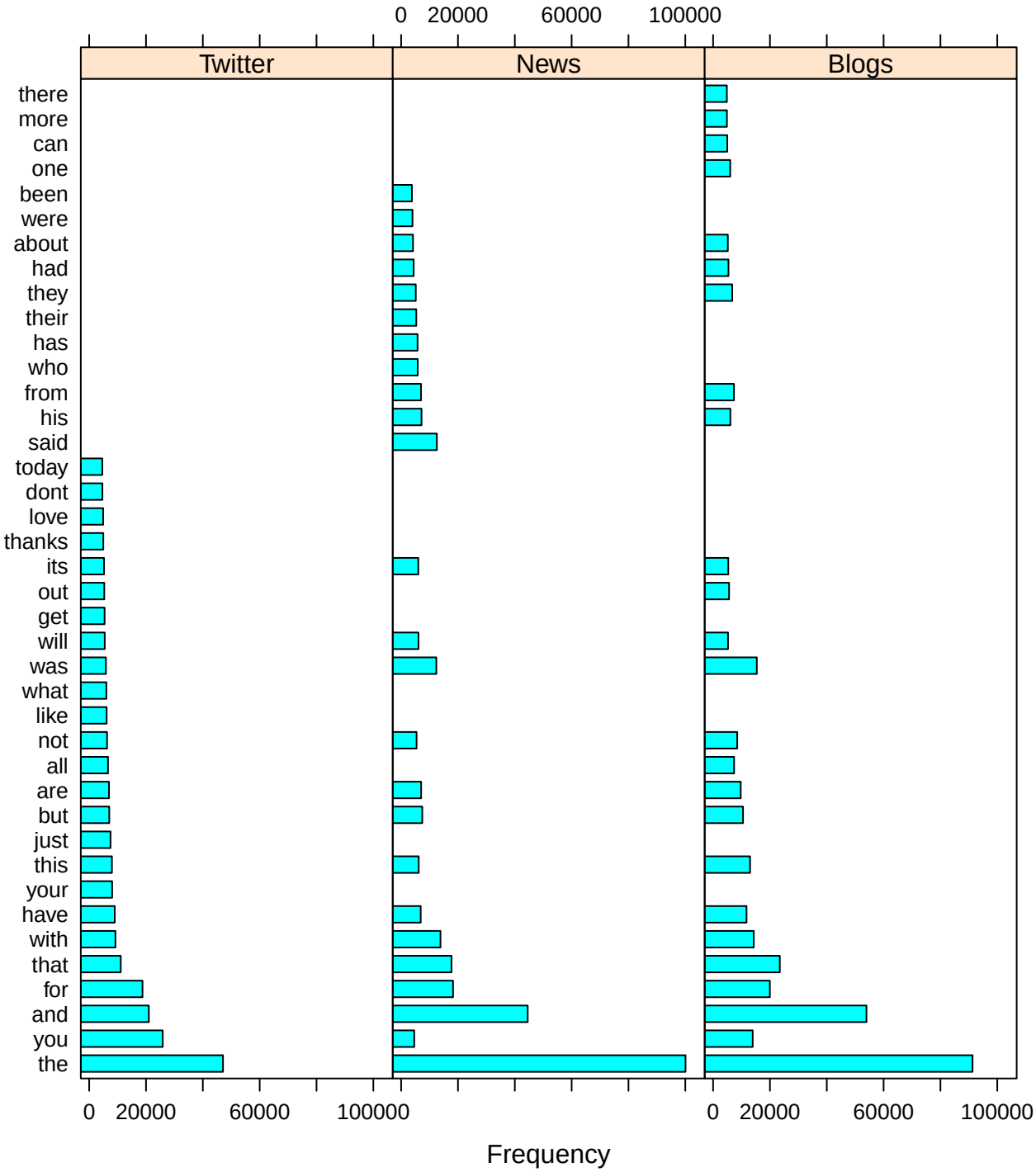
statistic	Twitter	News	Blogs	unit
Max. record length	139	1079	2718	characters
Lines in sample	118007	50512	44964	lines
Total words	1125408	1367324	1447389	words
Unique words	9443	14871	14604	words

Text mining

Now the basic text is cleaned and tokenized we can extract information out of the text samples. First is the top 25 word count visualized, followed by the n-grams of our sample data.

Word frequency

Figure 1: Top 25 frequent words in text samples.



N-grams

To be able to predict next words it is important to understand how words in sentences are sequenced. To visualize this we create n-grams showing the frequency of word sequences of 2 and 3 words long.

Figure 2: Top 25 frequent 2−grams in text samples.

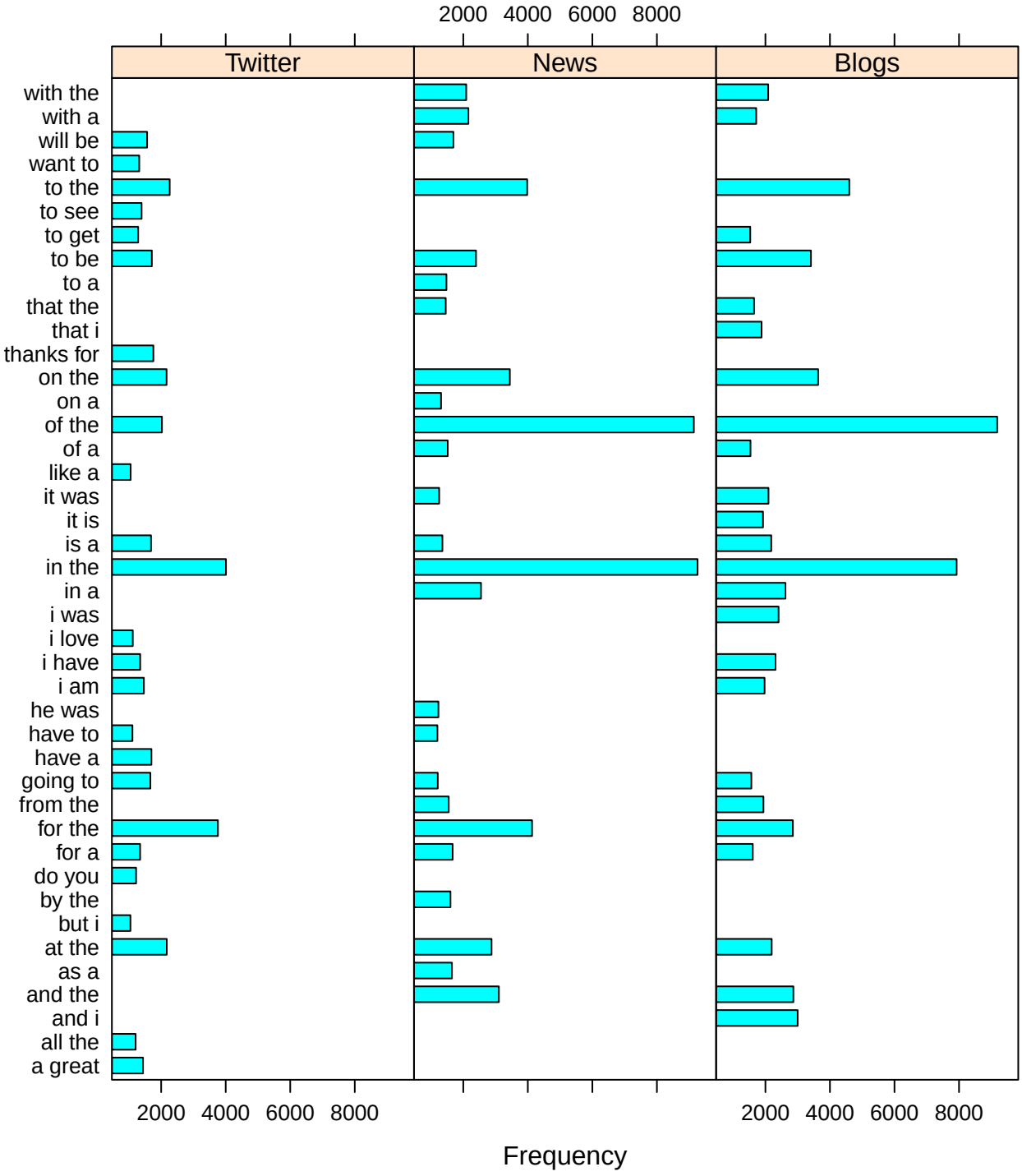
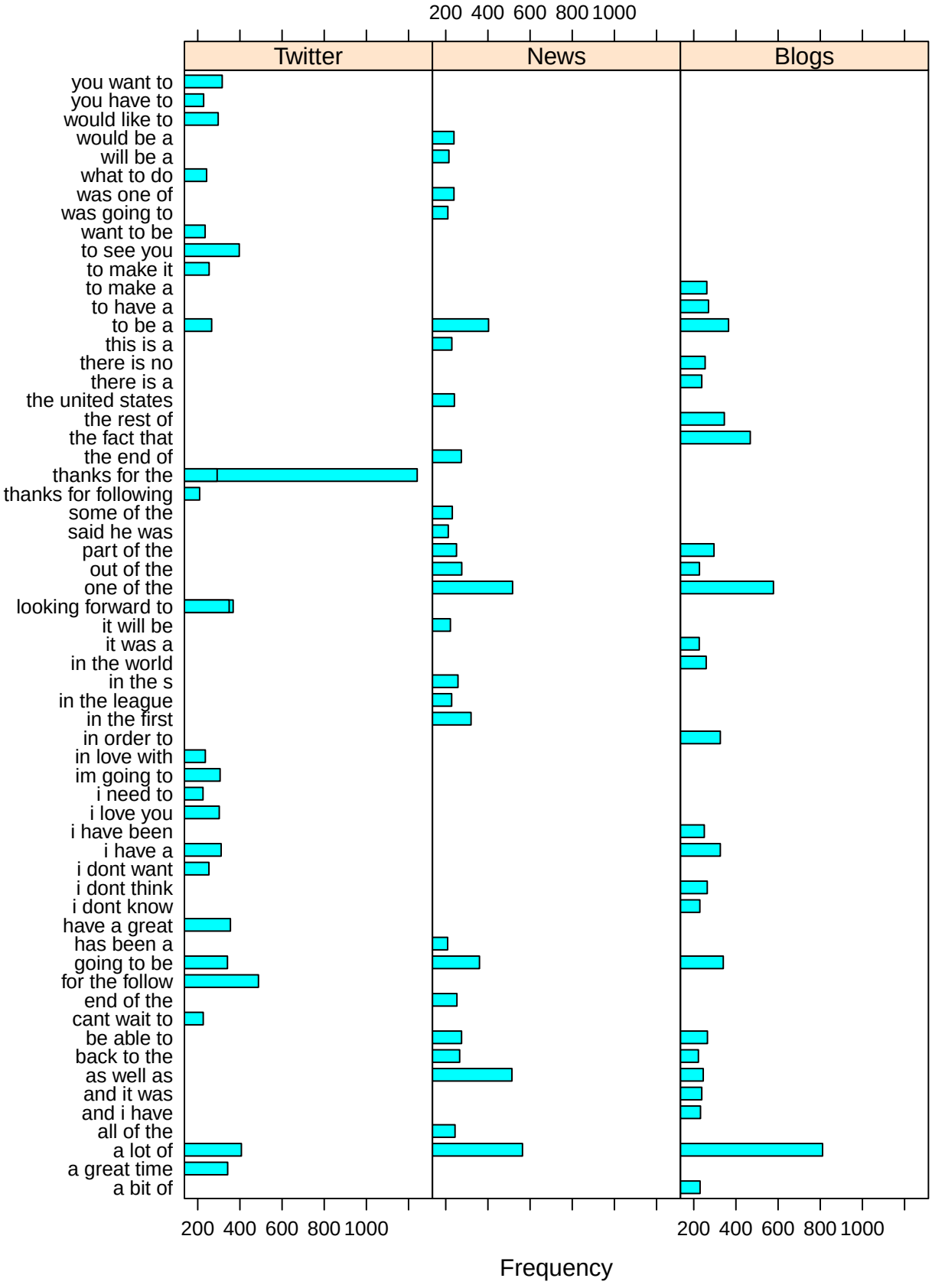


Figure 3: Top 25 frequent 3−grams in text samples.



N-gram statistics

statistic	Twitter	News	Blogs	unit
Number of 2-grams	38707	65264	69327	occurences
Number of 3-grams	46779	84912	97591	occurences

Findings & Conclusions

Based in the exploratory analysis and initial mining of the text the following observations were made:

1. Text from natural sources need to be cleansed to allow for good analysis or as source for test prediction. This leads often to compromises between quality and the time, effort and resources you invest in it. Examples are:
   i) Removing punctuations corrupts words like: don't, let's, it's, you're, etc. or web addresses like wwwgooglecom
   j) Converting to lowercase removes the capital first character of names. Examples: Monday, November, Denmark, etc.
   k) Stemming is required to deal with derivations in words (e.g: Walk, Walking, Walked), but often leads to unwanted side effects (e.g: **argued** becomes **argu**). To do a good job stemming is complex
   l) In profanity filtering the removal is often escaped by tricks as glueing words together, or making small but immaterial spelling mistakes (e.g: Fcku). Also here a good profanity filter is complex
2. Finding the right percentage of the data to be representative for the language is not easy. I ended up with taking a 5% sample, which was manageable for computing, but whether this is sufficient to represent the language is at this stage hard to judge. (Sampling with other sample sizes was beyond the scope of this exercise). Looking at the overlap in findings between the 3 test data sets I assume my sample size was sufficiently large.
3. Analysing the word frequencies and the n-gram distributions I observe that the counts (unique numbers) are very high and likely unpractical for a fast algorithm or application. Also in the low frequency tails a lot of nonsense and unlikely words are observed. Hence for a practical application a significant part of the low frequency tails have to be removed.
4. My *News* sample contains 1367324 words (= 100%) and 14871 unique words. With as little as 400 of the most frequent words 50.3% of the sample text is covered. With 6000 words 90.3% of the sample text is covered. These figures give a clear guideline on where to cut of the word list for an effective text prediction algorithm. 2.69% of the unique words to cover 50% and 40.3% to cover 90% of the text. For Twitter text with a higher word repeatability this ratio is even better.
5. Strange or foreign words and misspellings are used less frequent and fall automatically in the low frequency tail of the analysis results.
6. Right-sizing the analysis results for text prediction is possible by:
   m) Making other samples and compare results
   n) Making the sample bigger until the results stabilise
7. I did not encounter any computing or memory resourcing problems. This might be due to the choice I made to process the data per file and not as combined corpus. Although the code is a bit longer due to some replication it allowed for a hassle free and quick analysis.

Way forwards (Modeling)

The first step in the modelling is combine the results of the 3 test datasets and cut the less frequent works and n-grams from the analysis results to get an efficient but still effective text prediction application.
A basic model for test prediction is to convert occurrence frequencies into a chances. New words or parts of them are compared to the frequency tables and the word with the highest likelihood is predicted. Higher order n-grams take priority over lower order n-grams. So the priority is take the 3-grams prediction, thereafter the 2-grams and finally the 1-grams prediction.
For unseen word sequences, words or parts of words the model should in my opinion not predict but instead add new options to the dictionary to allow the prediction model to learn from your own writing style or vocabulary.
Testing of the model can be achieved by monitoring the matches (= successful predictions) against failed predictions (= unseen words + wrong predictions).