Assignment 4: Text Analysis

Panagiotis Giagkoulas (s3423883)
Carlos Huerta (s3743071)
Saim Eser Comak (s3432548)
Emile Muller (s3787915)
Danny Rogaar (s2393344)
Daan Opheikens (s3038416)

Group 14

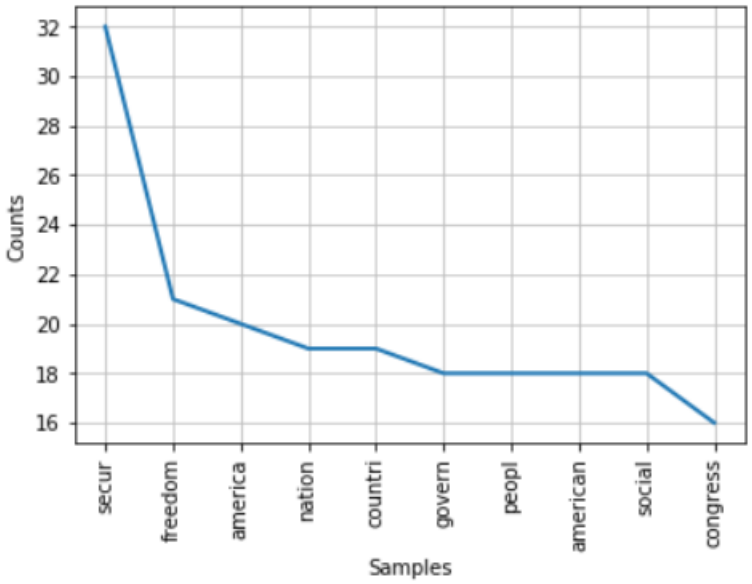
September 30, 2018

1: State of the corpus

1.1: Compute term frequencies for all the documents

For this exercise we will be using the *stateunion* corpus, which contains transcripts of the annual state of the union speeches by US presidents from 1945 until 2006. We will be using python nltk package to help us with this task. First of all we must "clean" our data to produce better results. We are going to use a nltk's word tokenizer along with a nltk's porter stemmer, we also used a simple filter on our stemmed dataset to remove characters such as ",", ".", "?", stop words and words below 4 characters long. After that we used nltk's freq dist library to compute word frequencies and generate a set of word counts in the document. That way we can plot a figure of the most common words on the dataset.

The code of this exercise can be found on the StateOfTheCorpus.ipynb on the github repository.
https://github.com/RUG-IDS/team-14
Example of most common stemmed words of 2005 G.W. Bush Speech.

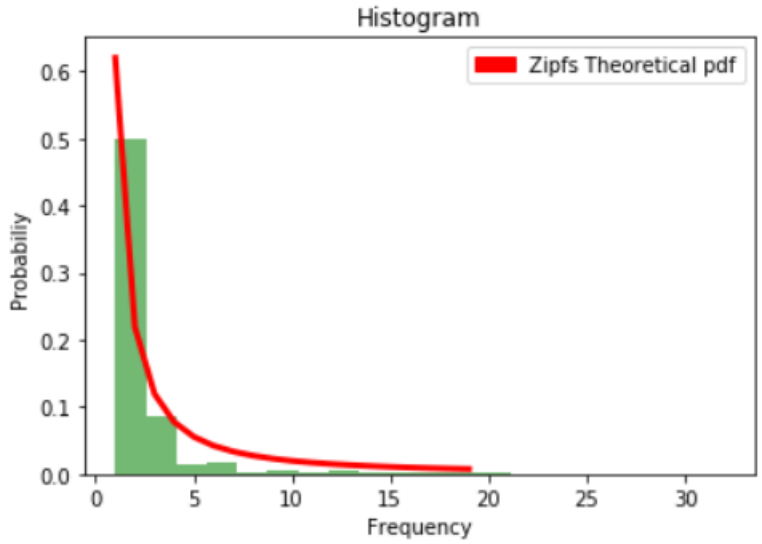


1.2: Zipf's Law Verification

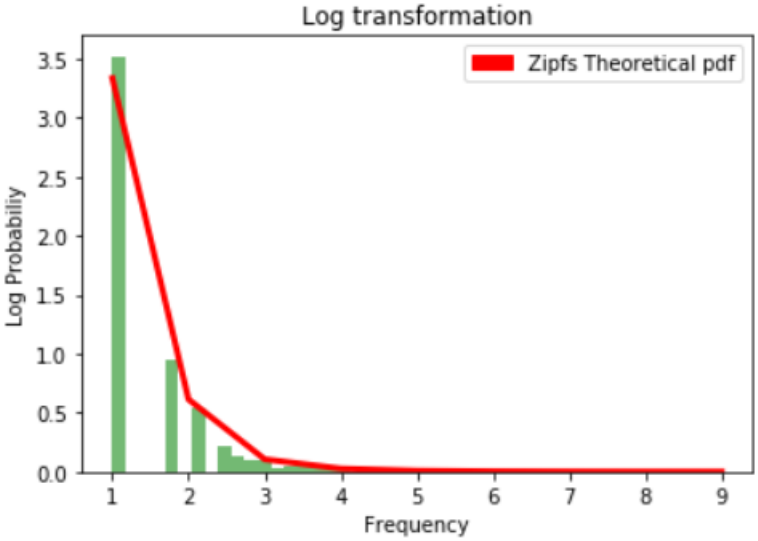
Zipf's law verification was done using the same code as above, but by also applying a log transformation to the results obtained on the exercise. To do this we needed to extract the frequencies to an array form and then use numpy's *log* function to get the log of the probabilities. Afterwards we used scipy's special function to compute a Zipf's theoretical

distribution and fit a line across the histogram of the frequencies and the log of the frequencies. We will continue using G.W. Bush 2006 Speech as an example.

The code of this exercise can be found on the StateOfTheCorpus.ipynb on the github repository.
https://github.com/RUG-IDS/team-14



As we can observe the distribution of the words are very skewed towards the range from [1 - 5] occurrences, the Zipf's theoretical distribution does indeed fit very well our data.



After applying a log transformation to the distribution we can observe that this frequencies smoothens to a less right skewed pattern, the fit Log of the Zipf's distribution fits our data in an even better way.

1.3: Data Clean-up

Our data clean-up process was made using both a porter stemmer and a lemmatization method, both methods have the removal of stop words form the English language using nltk's library.

1.4: Collocations

Collocations are sequence of words that tends to occur more frequently than the other possible sequences. To extract collocations many approaches can be used. In our report we will use n-gram grouping functions of nltk library to reveal top 10 high frequency trigrams which are three word sequences inside a text. First, the text within state union corpus is collected in a variable and concatenated so that we have one large sentence.Stemming step was skipped as

we were interested in finding collocations in their true form and type. After tagging tokens, we filtered out 'NNS' and 'NNPS' types. We then filtered out stop words and punctuation. Running TrigramCollocationFinder() function on our preprocessed data we collected the output. The initial results can be seen in table 1

Collocations			Frequency
fiscal, JJ	year, NN	1947,CD	68
dollars,NNS	fiscal,JJ	year,NN	35
million, CD	dollars,NNS	fiscal,JJ	28
fiscal,JJ	year,NN	1946,CD	21
current,JJ	fiscal,JJ	year,NN	19
million,CD	new, JJ	jobs, NNS	18
next, JJ	fiscal,JJ	year,NN	17
health,NN	care,NN	system,NN	16
weapons,NNS	mass,VBP	destruction,NN	15
must, MD	work,VB	together,RB	15

Table 1: Initial trigram results

Initial results returned invaluable collocations which led us to remove words such as "fiscal", "1947","1946",etc. After removing these words, we obtained the following results with sliding window parameter set to 3.

Collocations			Frequency
million, CD	new, JJ	jobs, NNS	18
health,NN	care,NN	system,NN	16
weapons,NNS	mass,VBP	destruction,NN	15
must, MD	work,VB	together,RB	15
recommended,VBD	1945,CD	legislation,NN	12
gross,JJ	national, JJ	product, NN	11
lower, JJR	interest,NN	rates,NNS	10
campaign,NN	finance,NN	reform,NN	10
distinguished,VBD	members,NNS	citizen,NN	9
must, MD	work,VB	sure,RB	8

Table 2: Filtered trigram results

Now much more meaningful collocations were listed. Next we tweaked the sliding window parameter for TrigramCollocationFinder() function. As the sliding window parameter increased (6,7,8..etc), the algorithm was able to find other sequences with frequencies as high as 56. However the collocations obtained from higher sliding window parameter didn't yield meaningful sequences therefore it is better to keep this parameter low, such as 3 or 4.

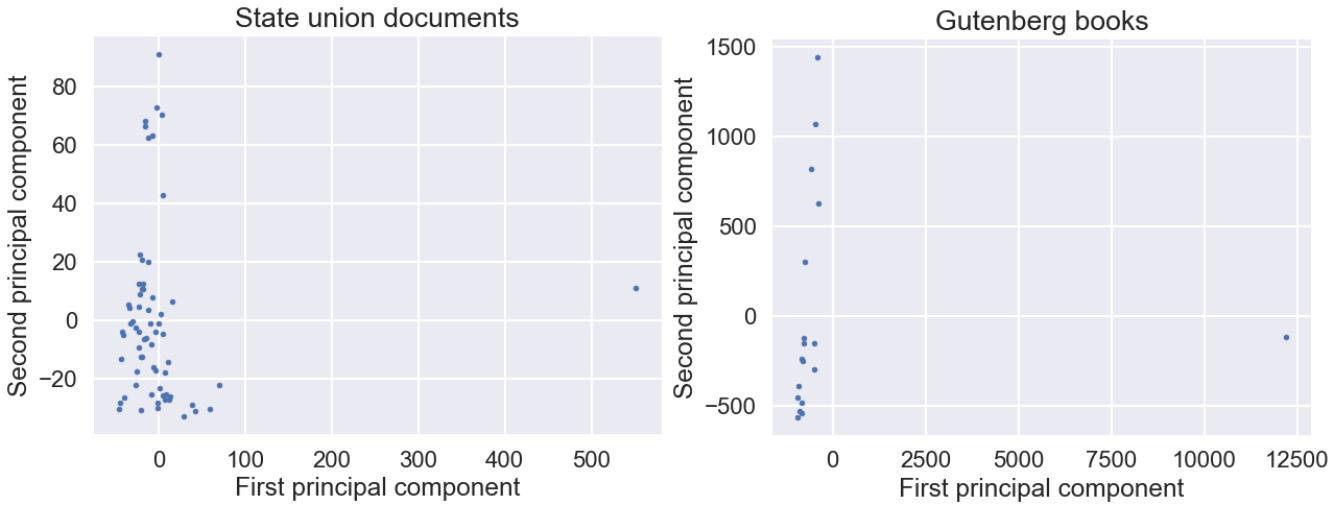
1.5: Document analysis

In order to compare documents int he datasets, we gather all the words that compose them. The word count is seen as a vector representing a specific document. PCA is applied to the word vectors such that a linear combinations of counts for a subset of words form the principal components that are to explain the variation between documents. For a PCA resulting in 2 dimensions, the results are shown in figure 1a and 1b.
In the mentioned figures, we see a small and large cluster of similar texts in the state_union dataset as well as an outlier, far removed from the other texts based on the first principal component. A similar outlier is present in the gutenberg set. For gutenberg, however, all other texts are quite spread out on the second component and a cluster is only identified with small scores of the second component.

2: Your own search engine

2.1: TF.IDF representation of the corpus

TF.IDF stands for *Term Frequency-Inverse Document Frequency*. This statistic reflects the significance of a word in a specific document within a corpus. This statistic takes into account the term frequency (TF) of a word in a



(a) State union documents.

(b) Gutenberg books.

Figure 1: Gutenberg and state union texts in their corresponding two principal components.

document, which represents how much it is used, and the inverse document frequency (IDF) of a word in a corpus, which represents how much it is used in documents other than the one being examined. TF.IDF is the dot product of the two aforementioned statistics. Its value allows us to estimate how important a word is to a document in a corpus, or in other words how often it appears in it and not in the other documents of the corpus.
In this case we make use of the TF.IDF as a representation of the words and later make use of this representation to search the corpus and given a sentence, locate the document it possibly originated from.

2.2: Top 10 representative words

The State of the Union speeches given yearly by the president of the United States can differ a lot from person to person and is of course subjectively to time. We tried to find proof for that by looking at the 10 most representative words of three random speeches. For this, we used TF.DIF again and selected three random State of the Union speeches to compare to each other.

Nixon 1970	**Nixon 1974**	**Reagan 1987**
New	Years	America
America	1974	Congress
People	Year	People
Seventies	America's	Let
World	America	100th
Shall	Peace	Constitution
Congress	New	Soviet
Clean	Energy	World
Peace	Congress	Freedom
Time	People	Sun

As can be seen from the table above, the results were that a lot of the same words are spoken in the speeches, despite president or time. For instance, these three speeches all talk about America and Congress. However, some differences are notable; in Reagan's speech of 1987, which was near the end of the Cold-War, the Soviet Union was being named way more often than in the previous speeches. Also, the word 'Seventies' is logically more prevalent in speeches of that time period. Reagan also spoke about the 100th congress in his speech, which is of course a special occasion. Despite this, we need to research further if the difference between most representative words is very notable or if they are roughly the same.

2.3: where-was-that

In order to relate corpus documents with an arbitrary string, we represent the string as well as the documents by their TF-IDF vectors. The text string is simultaneously processed so that its vector has scores for all of the words

in the corpus. This enables calculation of the cosine similarity between the string representation and the document representations. Such a similarity measure increases as words and documents share and lack the same words. The top 10 cosine similarities then find the most similar documents in the given corpus. The similarities are shown and it is left up to the user to define what makes a clear hit. The top result can readily be printed, however.