Camilla Ejby Pedersen & Sara Liv Broe

Text-mining - the guardian

Table of Contents

[Research question 2](#_Toc29205815)

[Task 1 and 2: 2](#_Toc29205816)

[Task 3: 4](#_Toc29205817)

[Task 4: 5](#_Toc29205818)

# Research question

Which topics was mostly used to discuss Donald Trump?

# Task 1 and 2:

First of all, we visited the site open-platform.theguardian.com/explore/ and found out what we wanted. On this site there was different parameters we could use such as date, tags and so on. By using the parameters, we got an instant look at which articles we would get. We could hereby quickly find the different filters that we needed in our code.

We searched for ‘Trump’ and saw that the relevant articles had the show-tags ‘usnews’ and ‘worldnews’. Then we made another search where we used the said show-tags. Furthermore, we chose to narrow the articles to a year. So, we asked for all articles related to ‘Trump’, ‘usnews’ and worldnews’ which were publicated between October the 30th 2018 and October the 30th 2019. Since Trump is a highly popular topic, we felt quite confident that we would get enough articles within the time frame.   
  
We searched on ‘content’ and got 8810 articles.

These articles got imported to Python. Since we narrowed it down to a year, we got 365 days worth of articles. The code we used imported the articles pr. day. Which means that we could have one article in day one, but 50 articles in day two.

The articles was saved as Json-files.

We then asked to get all of the Json files – however, we ran into a problem. It seems that there were some files (that the computer put in between the Json files) that couldn’t be read in Python. We therefore asked to get all the files, but, if Python couldn’t read the files, then we asked it to exclude these.

By now we had only limited the files through the site, mentioned earlier in this document. So the next step was to use code to limit the articles further. We used stopwords – we imported this part and therefore couldn’t be sure what exactly these stopwords were.

However, we found out, that we could simply just “print” the stopwords and then we could see which words we excluded.

Furthermore, we asked it to remove ‘s-endings, so words like “Trump “and “Trump’s” are the same words. We also made all words lowercase, which gives us troubles with a word like “us”. We don’t know if it’s “USA” or “me and you”. However, we felt that the benefits were bigger than the negatives. After that we also chose not to include comma and dots and that the words should be separated by white space.

After this we chose the 10 most common words for each article and to see these in a list. However, it took way to long, so to find out if our code was working or not and to get an indicator as to which words we would get, we chose to only take a small part of the articles and test our code on these.

We asked for the list in reverse, since we wanted the numbers that occurred most frequently on top of the list.

It turned out that our code was working, however, we ran into a problem, since we both got relevant words like ‘Trump’, but we also got words like ‘said’ and ‘would’.

This led us to limit the words even further. Also, we had only searched the articles for ‘Trump’ which means that we could end up with articles written about ‘Ivanka Trump’ or ‘Melanie Trump’.

In order to limit the “noise” furthermore, we chose to add our own stopwords. That way, we were able to see which words would come back and if we had words, that didn’t made sense, then they could easily be added to our list of words. As mentioned before, we could end up with words such as Ivanka Trump. Ivanka is not a part of our initially stopwords, but the word is irrelevant, and we could therefore manually exclude the word if this was part of the most popular words that we would extract from the documents.

The first ten words we got was:

A screenshot of a cell phone

Description automatically generated

We chose to exclude all of them, because, even though “Trump” is relevant and “house” and “us” could be relevant, then they aren’t relevant in regard to our research question.

So, we ran the code again and got this result:

A screenshot of a cell phone

Description automatically generated

We have both “-“ and “said” that runs twice, because we forgot to remove “said” and we removed ”\_” instead of “-“. However, it’s very easy to put the words into our code, so all we have to do is to insert these and run the code again. You can keep adding words to the list and keep running the code, so that you, in the end, end up with a meaningful result.

We chose to keep the word president, as it describes Trump.

Our code needs to run through each file, which means that it takes a long time to get through the over 8000 files that we have. Therefore, we chose to only demonstrate this and not to finish it and thereby we do not end up with only meaningful words.

**Characteristics of collection**

﻿Number of documents: 8810

Number of words before processing (number of words in total): 12074130

Number of words after processing: 6949785

Average word count per document before processing: 1370.5028376844496

Average word count per document after processing: 788.8518728717366

Average length of documents: 8685.226901248581

# Task 3:

Query: Donald Trump

We chose this query since it is close to our initial query and because although we are thinking of the same person, we might get different words do to the more specific query.

﻿Number of documents: 7406

Number of words before processing (number of words in total): 10249192

Number of words after processing: 6094812

Average word count per document before processing: 1383.9

Average word count per document after processing: 822.95

Average length of documents: 8800.19

Comparison between the two datasets:

|  |  |  |
| --- | --- | --- |
|  | Original (Trump) | Subset (Donald Trump) |
| Number of documents | 8810 | 7406 |
| Number of words before processing (number of words in total) | 12074130 | 10249192 |
| Number of words after processing | 6949785 | 6094812 |
| Average word count per document before processing | 1370.5028376844496 | 1383.9 |
| Average word count per document after processing | 788.8518728717366 | 822.95 |
| Average length of documents | 8685.226901248581 | 8800.19 |

What strikes us is, that our original query has more documents and more words, however the average word count is higher in the subset. So even though there are fewer articles regarding the query “Donald Trump”, then the journalist wrote longer articles.

The subset contains over 7000 articles and we would therefore conclude, that it contains enough articles for further research.

# Task 4:

The main topics surrounding our subset were:

A screenshot of a cell phone

Description automatically generated

Some of these makes sense and some we can’t use. However, when we played around with the values, we got topics that were very relevant, e.g. these topics that we made wordclouds with:

A picture containing bottle

Description automatically generated

A picture containing bottle

Description automatically generated

We aren’t able to answer our initial research question. However, even if we had words like: “orange”, “hateful”, “powerful”, “genius”, “dump”, “generous” etc., we wouldn’t be able to tell for sure that these words were used to describe Donald Trump. The words could be used on things he thought of others or it could be a description of his staff, family or voters. We would only be able to say how likely or unlikely it is that these words were used to describe Trump himself.