Portfolio 2 - assignment

Christian Laursen (bfd962)

Mickey Johansson (mvx394)

Mushtaba Osmani (wgq323)

11010 characters - 4,59 normal pages

17/12 - 19

Introduction

In the following assignment we have conducted simple data mining on a dataset consisting of transcripts of danish news broadcasts divided into different text documents. The oldest broadcast being from 1939 and the newest from 2010. The transcripts have no punctuation and are extremely roughly split into different paragraphs, which are supposed to cover one story each, but they rarely do. Because of the limited amount of data, combined with no easy way to sort through the different stories of each transcript, we decided to focus on the broadcasts themselves, instead of trying to query for a specific topic. Our research question therefore consists of finding out if and how the broadcasts have changed over time, more specifically if there is a variation in usage of tokens throughout specific time-periods.

Pre-processing and Description of our Collection

First of all, a list of (Danish) stopwords has been used to exclude uninteresting words from the texts. Furthermore, all words were converted into lowercase. This was done in order to count all instances of words, for example. To make more relevant comparisons across the texts, which are divided into years spanning from 1939-2010, they were split into 4 different time periods. The older texts from 1939-1959, from 1960-1989 and the newer ones from 1990-1999 and 2000-2010. This opened up for the possibility of comparing how and if news stories have changed throughout the years. The collection consists of 25 different text documents with 227.872 characters in total. Meaning it is a small collection compared to other options.

If the documents were split up in stories by each paragraph/line break, we would have been better able to work with individual stories and gather additional and more precise data. For example, we would be able to work with story count, length of stories, how many stories or specific topics are mentioned etc. Afterwards, this could be compared with the different periods explained above. Moreover, we would have been able to implement n-grams for each story to find word combinations throughout the texts and give them more context to find out in what ways some topics were talked about.

However, since there are no ways to tell whenever a story ends or begins in the documents, we would have to split each story with a line break manually. With 227.872 characters, this would require a great amount of work and time. We decided that the outcome compared to the amount of effort, was not worth it which is why we did not perform this type of cleanup.

Code description of our pre-processing:

To start working with our dataset, we first had to open every text and save them in a list. With a loop we go through every file in the directory, if the file ends with .txt we open it and append the text to a list with all the characters made lowercase. This gives us a list with texts from all the broadcasts. We also created a list to contain the filename of each broadcast. From the filename we can extract the date of each broadcast, which we use to divide the texts into different time periods.

```
# Opening folder of files, and saving each text
directory = 'C:/Users/Christian/Dropbox/KommIT/5_semester/Open Data Science/danish_news_corpus'
files = []
dates = []
for file in os.listdir(directory):
    dates.append(file)
    if file.endswith(".txt"):
        filename = os.path.join(directory, file)
        with open(filename, encoding='utf-8') as handle:
            text = "\n".join([x.strip() for x in handle])
            files.append(text.lower())
```

To divide the texts into time periods, we used the filenames from the list we just created. By going through the list, we could open each file one by one and get the year of each file by slicing the filename to only have the 4 characters that stated the year it was from. Then, depending on the year, we added the text from the files into 4 lists each matching a specific period of time. We also created 4 strings to contain all the texts from each time period, which we use later for word clouds.

```
#Creating text lists based on different dates
#Also creating strings with all the texts from each period (used for wordclouds later)
list1 = []
text1 = ' '
list2 = []
text2= ' '
list3 = []
text3= ' '
list4 = []
text4= ' '

for i,file in enumerate(dates):
    with open((directory+"/"+dates[i]), encoding='utf-8') as handle:
        text = "\n".join([x.strip() for x in handle])
        if 1930 < int(dates[i][-12:-8]) <1960:
            list1.append(text.lower())
            text1 += text.lower() + ' '
        if 1959 < int(dates[i][-12:-8]) <1990:
            list2.append(text.lower())
            text2 += text.lower() + ' '
        if 1989 < int(dates[i][-12:-8]) <2000:
            list3.append(text.lower())
            text3 += text.lower() + ' '
        if 1999 < int(dates[i][-12:-8]) <2020:
            list4.append(text.lower())
            text4 += text.lower() + ' '
```

To remove stopwords from our texts, we opened a text-file containing danish stopwords and saved it as a list. Then we created a function that takes a list of words, loops through every word and adds each word to a new list if the word is not one of the words in our list of stopwords. We use this function later when analysing our texts.

```
# Open a file with danish stopwords, and saving it in a list
f = open("stopord.txt", encoding="UTF-8")
stopord = f.read()
stopord = stopord.split("\n")
f.close()

#Get words that are not stop-words:
def remove(list_of_words):
    iwords = []
    for word in list_of_words:
        if word not in stopord:
            iwords.append(word)
    return iwords
```

Now we have 5 lists, one list with all the broadcast texts and 4 lists with broadcast texts from specific time periods, and also a function to remove stopwords, meaning we could start analysing our data.

Model and visualization of our topic

The lack of specific story grouping, meant that we resorted to more general research methods, but specified these by dividing the texts into periods.

Code description of methods used for model and visualization:
We created a few functions to show different information about the broadcasts of each period. One function is to simply display the total amount of characters in a list of texts by taking the sum of the length of each text.

```
#Total number of characters:
def chars(list_of_texts):
    length = []
    for text in list_of_texts:
        length.append(len(text))
    print("Total amount of characters:",sum(length))
```

Another function is used to count the average length of paragraphs in a list of texts. We do this by dividing the total length of texts in the list with the total amount of paragraphs in the texts.

```
# Average paragraph length:
def paralength(list_of_texts):
    listlength = 0
    amount_of_p = 0
    for d,text in enumerate(list_of_texts):
        paragraphs = text.split("\n")
        listlength += len(text)
        for paragraph in paragraphs:
            amount_of_p += 1
    aplength.append(listlength/amount_of_p)
    print("Average paragraph length in this list of texts:",listlength/amount_of_p)
```

Another function that we made is used to count the amount of paragraphs in a list of texts and the average number of paragraphs in each text of that list. The paragraphs are counted by counting the amount of ‘\n’ characters in the texts. The average number of paragraphs is

calculated by dividing the total amount of paragraphs in the list with the number of texts in that list.

```
# Counting the amount of paragraphs
def paragraph(list_of_texts):
    chars = []
    for text in list_of_texts:
        for char in text:
            chars.append(char)
    print("Paragraphs in this list of texts:",chars.count("\n"))
    anpt.append((chars.count("\n")/len(list_of_texts)))
    print("Average number of paragraphs pr. text:",chars.count("\n")/len(list_of_texts))
```

We also created a function to get a list with every single word from a list of texts. We do this by looping through each text in the list, split the text into words and append each word into a new list.

```
#Total words:
def wordlist(list_of_texts):
    total_words = []
    for text in list_of_texts:
        for word in text.split():
            total_words.append(word)
    return total_words
```

We use this function with other functions we have created that takes a list of words as an argument. For instance, the *remove-function* mentioned earlier, that we use to remove stopwords. We also use it with another function to count the number of unique words. By using the *wordlist-function* as argument in the *unique-function* we can transform the list of words into a set, which removes every duplicate word, and then back into a list. The length of this list is then the amount of unique words.

```
#Unique words:
def unique(list_of_words):
    uwords = list(set(list_of_words))
    print("Unique words: ",len(uwords))
```

We also use it to count the average word length by dividing the total length of the words with the total amount of words.

```
#Average word length:
def averagelength(list_of_words):
    total_length = 0
    for word in list_of_words:
        total_length += len(word)
    return total_length/len(list_of_words)
```

Furthermore, a function was created to display the frequency of the most common words in a list of texts, both with and without stopwords. The function also creates a simple graph of the 30 most frequent words without stopwords. The function uses the *FreqDist-function* of the *nltk.probability module* to count the frequencies of the wordlists created by the *wordlist-function* mentioned above. The graph is made by using *.plot* from the *matplotlib.pyplot module.*

```
# Function showing freqency:
def freq(list_of_texts):
    print("Frequency with stopwords:")
    fdist = FreqDist(wordlist(list_of_texts))
    print(fdist.most_common(30),"\n")
    print("Frequency without stopwords:")
    fdist = FreqDist(remove(wordlist(list_of_texts)))
    print(fdist.most_common(30),"\n")
    fdist.plot(10,cumulative=False)
    plt.show()
```

Also, to further visualize the results, a word cloud of the most frequent words in a text was formed, without all the stop words in our list of Danish stop words. This was done by creating a function using the *matplotlib* and *Wordcloud* modules. Since we saved all the broadcast texts in different strings for each time period earlier, we could use this function on those strings, to get a word cloud for each time period. We also added all the texts into a single string, so that we could create a word cloud from words of all the texts.

```
# Putting all text from each document into a single string
alltext = ' '
for text in files:
    alltext += text+' '

# Function to create wordclouds
def wc(text):
    wordcloud = WordCloud(width = 600, height = 600,
                    background_color ='white',
                    stopwords = stopord,
                    min_font_size = 10).generate(text)

    plt.figure(figsize = (8, 8), facecolor = None)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.tight_layout(pad = 0)
    plt.show()
```

We put together a function that would run most of the different functions we had created at once, since we would want to run the functions for each of our time periods.

```
# Function printing out information
def printInfo(a_list):
    print("Number of texts:",len(a_list))
    statistics(a_list)
    chars(a_list)
    freq(a_list)
    average(a_list)
    unique(wordlist(a_list))
    paragraph(a_list)
    paralength(a_list)
```

```
print("\nAll the texts:")
printInfo(files)
wc(alltext)
print("\nTexts from 1930-1959:")
printInfo(list1)
wc(text1)
print("\nTexts from 1960-1989:")
printInfo(list2)
wc(text2)
print("\nTexts from 1990-1999:")
printInfo(list3)
wc(text3)
print("\nTexts from 2000-2010:")
printInfo(list4)
wc(text4)
```

This gives us a fairly expansive breakdown with info, graphs and word clouds for all the texts at once, and also for each time period. It also showed us some numbers which were interesting, which we decided to put together in a dataframe. We did this by creating lists to contain the data. We added a line of code in some of our previous mentioned functions, so that when the information was printed to the screen, it was also appended to a list at the same time. Then when we had data from all time periods in our lists, we could add these lists to a dataframe.

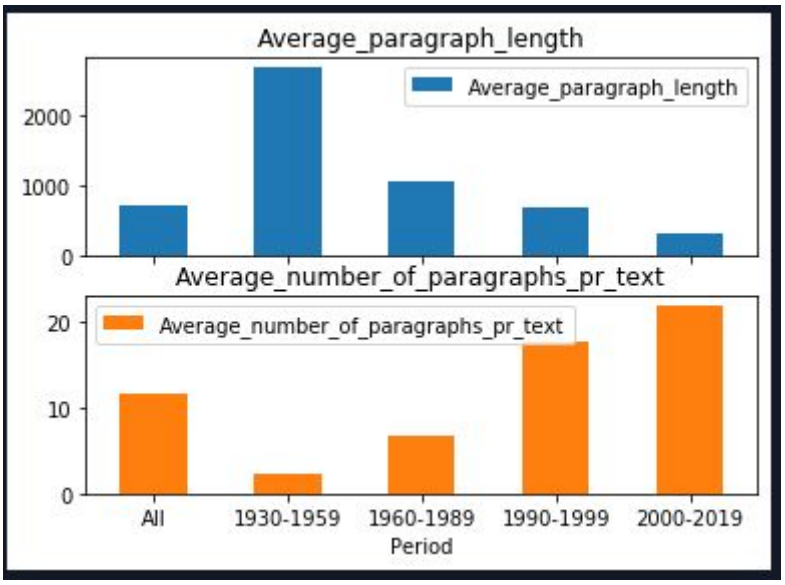
```
#Dataframe to contain data of each time period
period = []
awlength = []
aplength = []
atlength = []
anpt = []
```

```
#Creating a dataframe with data from each time period
df = pd.DataFrame()
df['Period'] = period
df['Average_word_length(without_Stopwords'] = awlength
df['Average_paragraph_length'] = aplength
df['Average_text_length'] =atlength
df['Average_number_of_paragraphs_pr_text']=anpt
print(df.head())
```

The dataframe provide an easy overview of some of the numbers about each time period. We also used the numbers in the dataframe to create a bar chart.

	Period	Average_word_length(without_Stopwords	Average_paragraph_length	Average_text_length	Average_number_of_paragraphs_pr_text
0	All	7.765575	725.707006	9114.880000	11.560000
1	1930-1959	7.977583	2684.550000	8948.500000	2.333333
2	1960-1989	7.858547	1066.419355	8264.750000	6.750000
3	1990-1999	7.729294	698.734043	13136.200000	17.800000
4	2000-2019	7.408068	307.115942	7063.666667	22.000000

```
# Bar charts showing average paragraph length and average number of paragraphs pr. text of each time period
ax = df.plot.bar(x='Period',y=['Average_paragraph_length','Average_number_of_paragraphs_pr_text'], rot=0, subplots = True)
```



Bar graph with average paragraph length and number of paragraphs per text

Description of the results and concluding words

In summary, in an attempt to answer how broadcasts may have changed over time, we divided texts from Danish news broadcasts into different time periods and visualized different aspects of the usage of tokens. To begin with, to visualize the frequency of the words in the texts, one could look at the word clouds (compare appendix 5, 8, 11 & 14) and line graphs (appendix). The line graphs highlight the 10 most frequent words in each time period, excluding Danish stopwords. The most interesting changes the graphs show are that words in the initial time periods were dominated by foreign countries and to a lesser extent words related to war such as “styrker” (forces). The time period of the 90s includes more weather-related words such as “regn” (rain) and in the 2000s the most frequent words were also dominated by weather-related words, including “grader” (degrees). We find this change in word use especially interesting as it indicates a clear shift in how the news broadcasts talk about the news. There are a couple plausible reasons as to why there is this change. Firstly, the dominance of war would have certainly plagued the news in the 1930-1959 period, which in turn leads to news broadcasts reporting on wars in foreign countries. We argue that the Suez war and the Hungarian revolution is the dominating aspect of the news from 1930-1959 as most frequent words were Egypt, France, Israel & Hungary. Secondly, a variation in words used in recent times also indicates that news broadcasts are now covering more topics and as a result the frequency of words are also more varied. This aspect of the topics being more varied, will be explored further when we describe our bar chart.

When comparing the word clouds, one notices that they look fairly similar, the largest (most frequent) words being words such as ‘siger’, ‘hundrede’, ‘år’ and ‘dag’. However, there are exceptions, such as only the word cloud for the oldest time period. Here the biggest words are related to different countries such as: ‘ungarn’, ‘franske’, ‘egypten’ and ‘egyptiske’.

Moreover, in the bar graph above, one can see that the average amount of paragraphs in the texts increase with time, meaning that the list with the oldest texts, has the

fewest paragraphs, and the list with the newest texts, has the most paragraphs. Meanwhile, the list with the oldest texts has the longest paragraphs, and the paragraphs are shortest in the list with the newest texts. This could indicate that in the more recent broadcasts additional topics were covered, but in less detail, while the old broadcasts had fewer topics but were covered in more detail.

Finally, we also have results showing the number of unique words used in the texts, the average word length of texts and the average text length of each time period. These results were collected as part of our research process, but the results did not end up providing any meaningful patterns in relation to our examination of the dataset.

Appendix

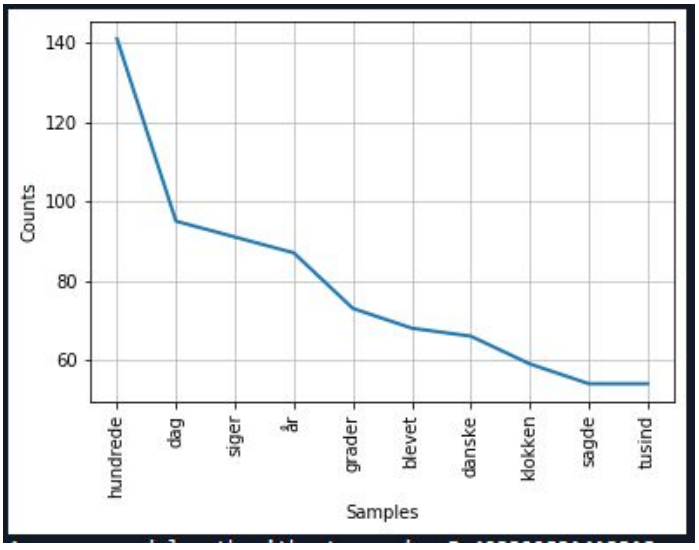
All texts:

Appendix 1:

```
Frequency without stopwords:
[('hundrede', 141), ('dag', 95), ('siger', 91), ('år', 87), ('grader', 73), ('blevet', 68), ('danske', 66),
('klokken', 59), ('sagde', 54), ('tusind', 54), ('egyptiske', 51), ('sidste', 49), ('øh', 49), ('kroner', 48),
('nitten', 46), ('vind', 46), ('israel', 45), ('første', 44), ('danmark', 44), ('franske', 44), ('mener', 44),
('israelske', 41), ('landet', 41), ('går', 41), ('regn', 38), ('morgen', 37), ('regering', 36), ('procent', 35),
('egypten', 34), ('nat', 34)]
```

Appendix 2:

Appendix 3:



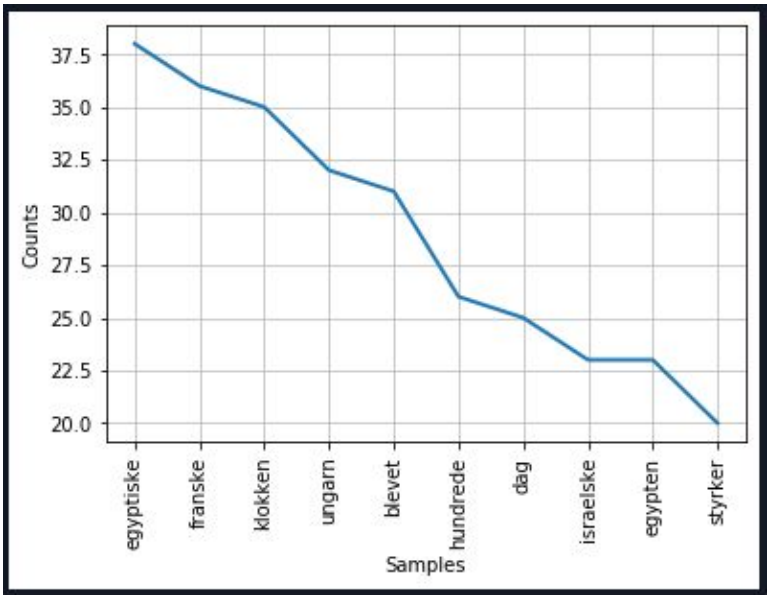
1930-1959:

Appendix 4:

```
Frequency without stopwords:
[('egyptiske', 38), ('franske', 36), ('klokken', 35), ('ungarn', 32), ('blevet', 31), ('hundrede', 26), ('dag', 25),
('israelske', 23), ('egypten', 23), ('styrker', 20), ('engelske', 19), ('israel', 19), ('eden', 19), ('russiske', 18),
('engelsk-franske', 16), ('tropper', 15), ('england', 15), ('radio', 15), ('tredive', 14), ('ungarnske', 14),
('udenrigsminister', 14), ('morges', 14), ('budapest', 14), ('aften', 13), ('danmark', 13), ('suezkanalen', 13),
('frankrig', 13), ('nitten', 13), ('tyske', 12), ('går', 12)]
```

Appendix 5:

Appendix 6:



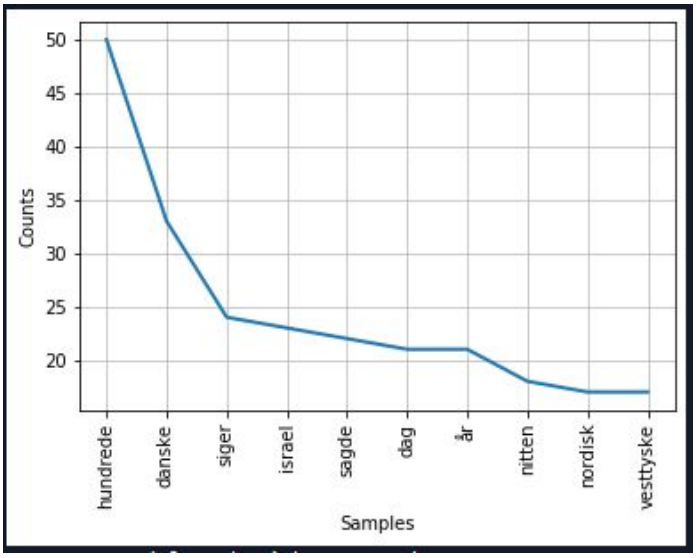
1960-1989

Appendix 7:

```
Frequency without stopwords:
[('hundrede', 50), ('danske', 33), ('siger', 24), ('israel', 23), ('sagde', 22), ('dag', 21), ('år', 21), ('nitten',
18), ('nordisk', 17), ('vesttyske', 17), ('nat', 16), ('kroner', 16), ('første', 15), ('sidste', 15), ('regering',
15), ('tusind', 15), ('nasser', 14), ('morgen', 14), ('hele', 14), ('klokken', 14), ('radioavisen', 13), ('præsident',
13), ('procent', 13), ('egyptiske', 12), ('israelske', 12), ('amerikanske', 12), ('landet', 12), ('vejr', 12),
('medlemmer', 11), ('blevet', 11)]
```

Appendix 8:

Appendix 9:



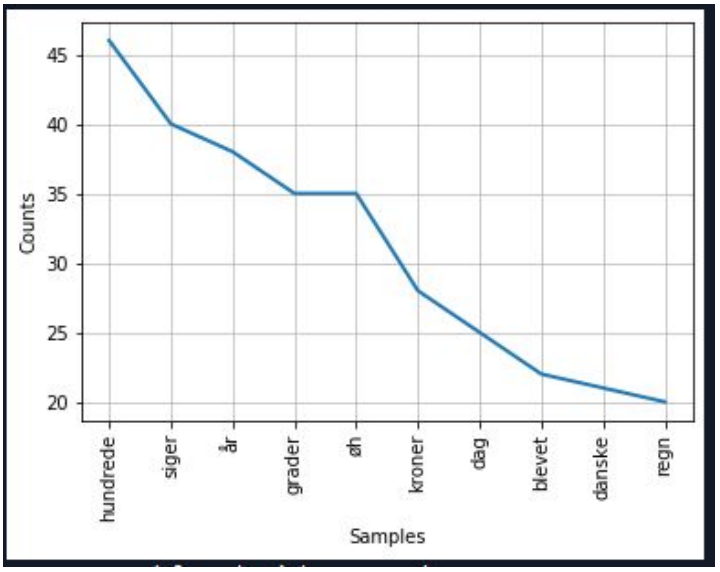
1990-2000:

Appendix 10:

```
Frequency without stopwords:
[('hundrede', 46), ('siger', 40), ('år', 38), ('grader', 35), ('øh', 35), ('kroner', 28), ('dag', 25), ('blevet', 22),
('danske', 21), ('regn', 20), ('vest', 20), ('sidste', 20), ('mener', 20), ('byger', 19), ('danmark', 18), ('henrik',
18), ('millioner', 18), ('sol', 17), ('tusind', 15), ('syd', 15), ('vind', 15), ('går', 15), ('første', 14),
('morgen', 13), ('hård', 13), ('procent', 13), ('eu', 13), ('natten', 12), ('dagen', 12), ('skriver', 12)]
```

Appendix 11:

Appendix 12:



2000-2019:

Appendix 13:

```
Frequency without stopwords:
[('grader', 28), ('dag', 24), ('siger', 22), ('år', 20), ('hundrede', 19), ('vind', 18), ('sagde', 18), ('mener', 16),
('viser', 14), ('tusind', 14), ('irak', 13), ('regn', 13), ('sol', 12), ('hård', 11), ('byger', 10), ('fortalte', 10),
('dansk', 10), ('regeringen', 10), ('let', 10), ('frisk', 10), ('egne', 10), ('fortæller', 9), ('sagen', 9), ('stod',
9), ('skyet', 9), ('går', 8), ('landet', 8), ('indslaget', 8), ('usa', 8), ('procent', 8)]
```

Appendix 14:

Appendix 15: