**An information retrieval and natural language processing study into British party politics during a time of nation crisis**

# Introduction

This report was commissioned by the Conservative Party with the aim of identifying political strategies employed by the main opposition party, Labour (the **Opposition**). This is a particularly relevant endeavour in the current time period due to the presence of unprecedented political and social conditions that emerged with the outbreak of the coronavirus. Traditionally, the relationship between the Government and the Opposition is characterised by strong antagonism. The Opposition generally attempts to take any opportunity to reprimand the Government. The coronavirus outbreak raises the question on how the Opposition will behave in that scenario and whether it will retain a confrontational attitude or whether it will attempt to strike a more re-conciliatory approach to show unity during a time of crisis. It will also more generally analyse what political topics the Opposition focuses their attention on. This report will use data processing, natural language processing and textual analysis tools in order to investigate British newspaper content, social media and website content. This will be used to derive the Opposition’s political strategy employed during the crisis.

# Methodology

## Data sources

NewsAPI

The first and main source to gather information is content from British newspapers. Newspapers actively document the relationship between the Government and the Opposition. This report will utilise *NewsAPI* (<https://newsapi.org/>) to collect articles as this provides an efficient way to extract relevant articles via personalised queries from a large database. Given the backwards time limitation of the free version of *NewsAPI*, this report will analyse one month’s worth of articles, namely those published between 18 March and 18 April. This should provide a genuine insight into political behaviour during the coronavirus outbreak given that official lockdown in the UK commenced on 23 March.[1](#_bookmark0) While it would be appropriate to consider articles from all major British newspapers, another API limitation restricts article retrieval to the *BBC* and the *Independent*. While a different API end point could be used to get articles from more newspapers,[2](#_bookmark1) that one also comes with the disadvantage that it only collects what it considers “top” articles. However, for the purpose of this report, it is more appropriate to collect all articles within the timeframe albeit only from two newspapers. This should still provide sufficient insight as we can assume that many newspapers publish similar content. In order to make a retrieve all articles relevant to Opposition comments or publications, queries to the NewsAPI shall be made with regard to the following keywords: “Labour”, “Opposition”, “Shadow”, “Corbyn”, “Starmer”. “Labour”, “Opposition” and “Shadow” are all related words that are commonly used to denote the Opposition in newspapers. “Corbyn” and “Starmer” refer to the party’s leader names during the specified one month period. The API queries will be divided into one that collects all articles containing one of the aforementioned keywords excluding coronavirus and one containing all four including coronavirus. This is to verify whether there is any significant difference in attitude surrounding the dominant coronavirus topic and other topics. A further subdivision will be applied with regard to the timeframe as Labour’s leadership changed during the collection period. This will help to identify if there are any changes in tactics during that period. A total of 670 articles shall be analysed.

Social media and official website

A second data source will be used in order to complement the data from newspapers. As newspaper articles are written by journalists and not by Opposition members directly, it is possible that some meaning will be lost or changed. Therefore, it is deemed as appropriate to also analyse a data source

1 BBC (2020)

1. An alternative end point would be the “Top headlines” endpoint. This report uses the “Everything” endpoint.

with comments or announcements that are published directly by Opposition members. For that purpose, this report shall also gather all publications made on the official Labour Twitter pages of Jeremy Corbyn and Keir Starmer as well as announcements on the official Labour website during the same time period (18 March until 18 April). As sometimes both Corbyn or Starmer would retweet posts from Labour’s twitter account, all retweets from Labour were excluded in order to avoid duplication.

This data source will only be used to complement the *NewsAPI* and thus will not be used at each stage of the analysis. The short timeframe imposed by the *NewsAPI* limitation means that only a relatively small data set of social media tweets and website publications are available. This is the reason why only limited analysis will be conducted on this set. It still provides a meaningful complement to the analysis.

## Natural language processing and textual analysis

Natural language processing (NLP) will be applied to all articles that are retrieved from all data sources. This is done with the purpose of cleaning and structuring the data sets. Part of this process includes producing a wordcloud which visually displays the frequency of words found in the dataset. This word cloud will reveal the main topics that the Opposition talks about. In addition, textual analysis will be performed on the cleaned and structured dataset. This will involve the creation of two word lists containing a lexicon with “confrontational” words and a lexicon with “cooperative” words. These will be generated by using the Python Natural Language Tool Kit (NLTK) and stemming. The data sets will be checked against the two lexica in order to identify the Opposition’s attitude based on textual analysis.

# Process

## API call and data collection

NewsAPI

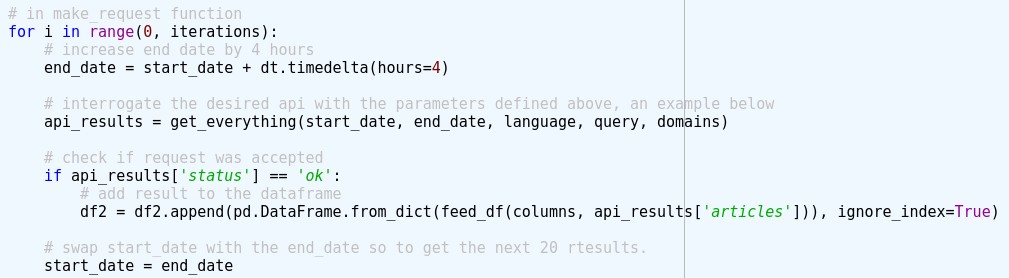
After requesting multiple API keys from *NewsAPI*, the first step is to check whether the API end point is working as expected through the use of *Postman API*. Using the keywords and time frame specified in the methodology section, the *Postman API* indicates that there are 670 total responses for all four sets of keyword queries (these will be explained further below). This indicates that the *NewsAPI* end point is working.

The “Get Everything” *NewsAPI* endpoint enables the collection of all articles from the *BBC* and the *Independent* within a one month time period. Snippet 1 shows the parameters that are set: the timeframe, the language, the keywords, the newspapers and the data columns that we are interested in. All collected data will be stored in a dataframe. The API request further specifies a loop function in order to maximise the number of articles that can be collected. Without the loop function, one manual request would need to made per 20 articles, as the API only returns 20 articles per request. The loop function is shown in Snippet

2. It starts collecting 20 articles starting from the start date plus four hours. It appends the collected articles to a dataframe. Then it switches the start time with the end time and repeats the same process 100 times (as set by the iterations parameter in Snippet 1) or until the daily API limitation has been reached.



*Snippet 1*

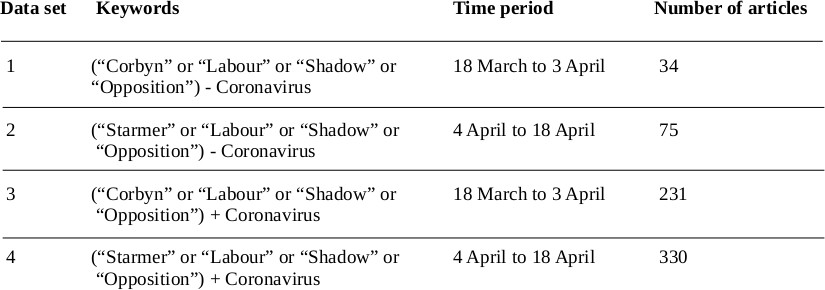


*Snippet 2*

As specified in the methodology section, two separate queries are made: one that excludes coronavirus as a keyword and one that has to include the keyword. Furthermore, given that the Labour leadership changed in the middle of our data collection, queries are sub-divided. One is done for the first time period from 18 March to 3 April with the name “Corbyn” and a second one for the time period from 4 April to 18 April

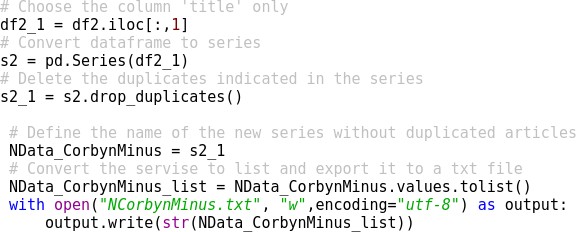
with “Starmer”. The API calls with the specified keyword searches generate four data sets. The four data sets are summarised in table 1.

*Snippet 4*



*Table 1*

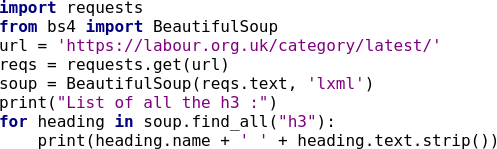
This means that overall 670 articles were collected, which is in line with expectations set through the *Postman API* query. However, note that articles excluding coronavirus as a keyword are much less numerous. This makes intuitive sense as the coronavirus dominates the news these days. All collected data, including the article publication date and the article title, now rests in a dataframe. In fact, only the titles are needed for further processing and they need to be exported to *.txt* files to enable natural language processing. This process is shown in snippet 4.



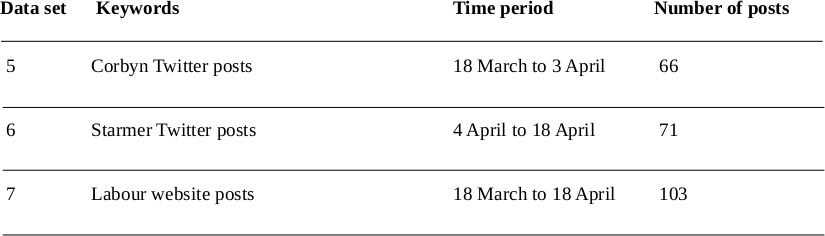
Social media and official website

Restrictions on the Twitter API make it difficult to efficiently retrieve information beyond 7 days. In order to retrieve full post content from a specific account, upgraded API access is required. In order to cover a one month period, it thus revealed more straightforward to simply copy and paste one month’s worth of past posts from Corbyn and Starmer’s Twitter accounts and then submit them to the same NPL and textual analysis procedure as the other data sets.

Regarding the posts on the official Labour website, it might have been efficient to use web parsing in Python. However, the web request as shown in snippet 5 returned the “403 Forbidden” error. This indicates that the webserver understood the request but did not allow it.[3](#_bookmark2) In light of this limitation, the collection of posts from the Labour website was thus ultimately also done by using simple copy and paste functions. Table 2 summarises the data sets generated in this manner.



*Snippet 5*



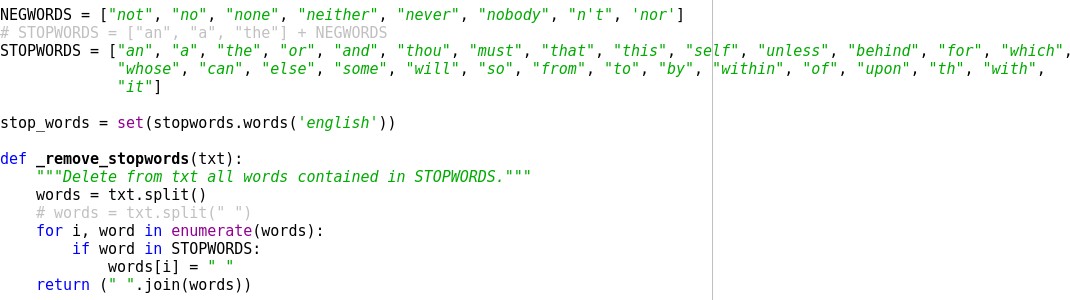
*Table 2*

1. HTTP-Statuses (2020)

## Pre-processing and natural language processing

Cleaning data

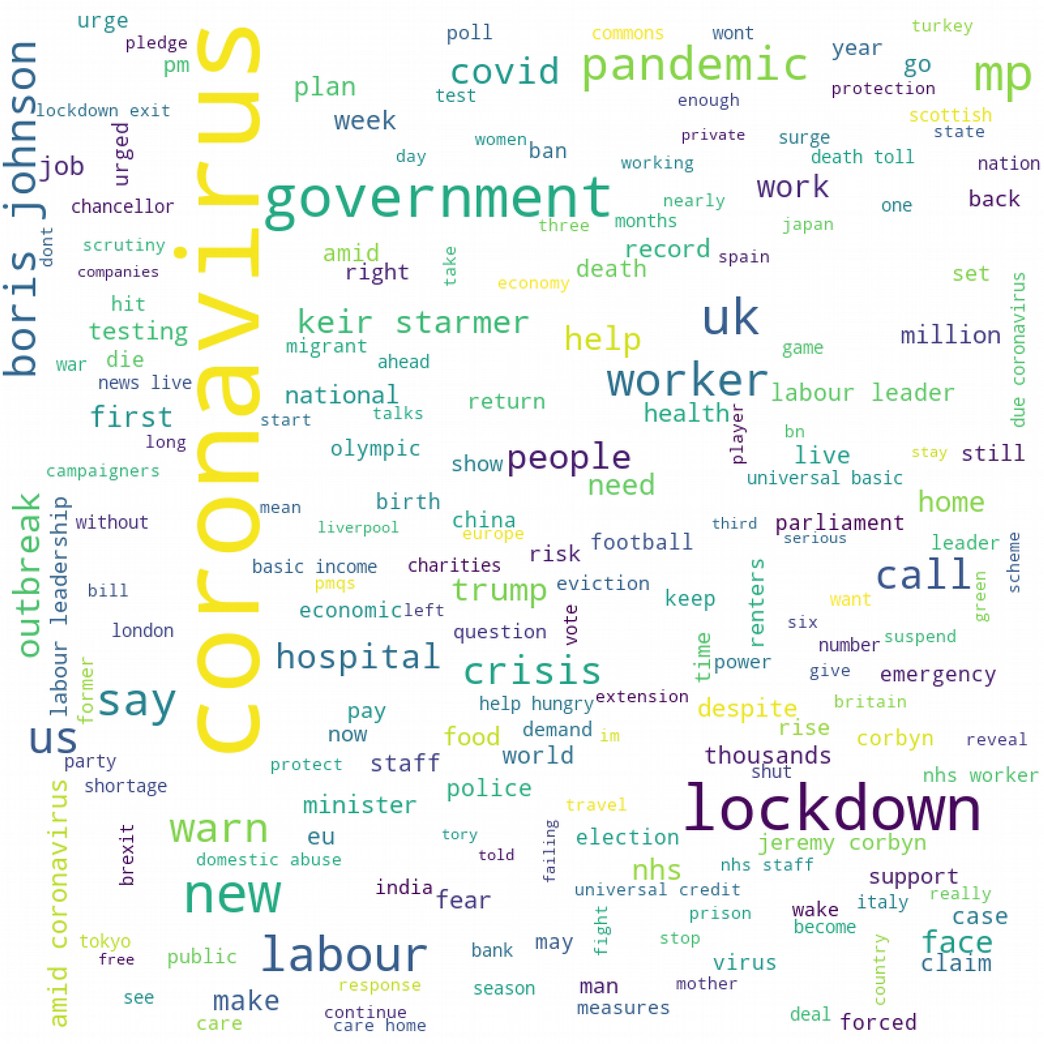
Now that all the text data from the article titles are saved in *.txt* files, more preparation or “pre- processing” is needed to enable natural language processing and textual analysis. First, negative and stop words need to be removed because the presence of those words risks distorting textual analysis. This refers to words such as “and”, “or”, “not”, “unless”, etc. This process is depicted in snippet 6. In order to further standardise the data more cleaning is done with regard to removing punctuation, white spaces and numbers as well as uncapitalising capitalised letters.



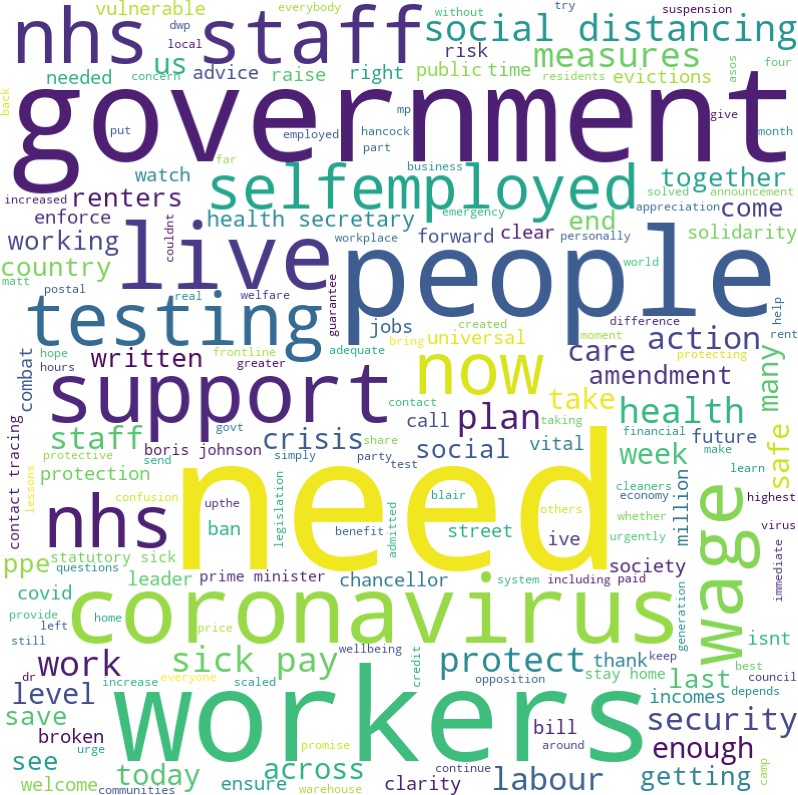
*Snippet 6*

Wordclouds

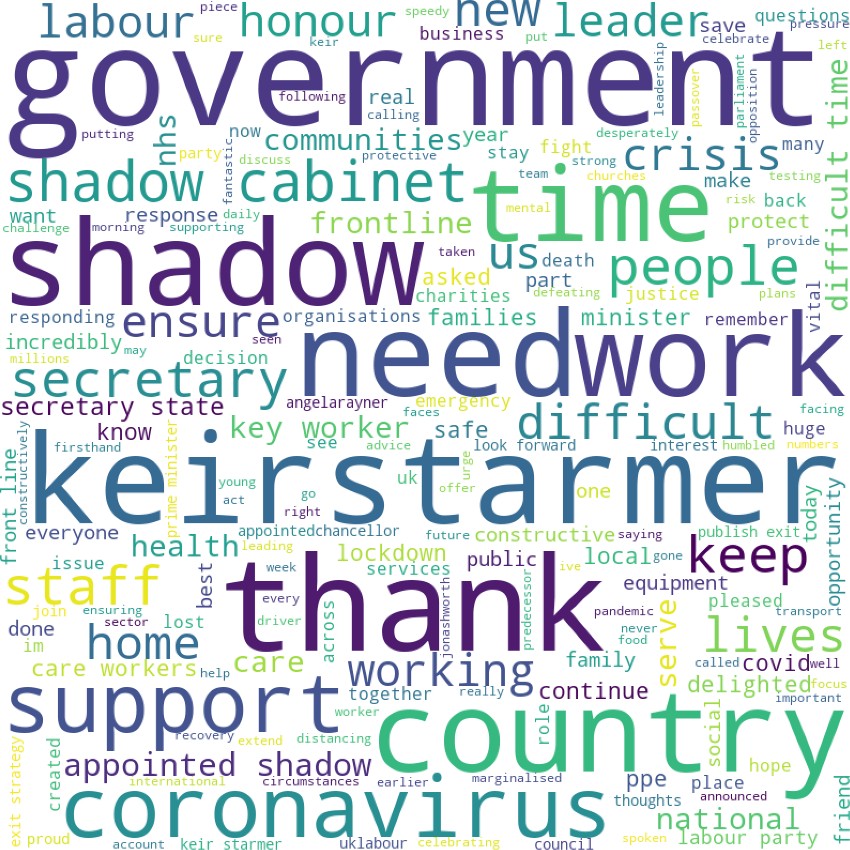
The cleaned data sets can now be used to generate wordclouds. A wordcloud creates a visual depiction of the frequency with which words are used in the dataset. A larger font indicates a higher frequency. This will be particularly useful to identify the Opposition’s focus areas. The wordcloud is generated with the code depicted in snippet 6. This code is repeated for each data set. However, for simplicity’ sake data sets 1 and 2 as well as data sets 3 and 4 will be combined at this stage and two word clouds are generated: one for data sets 1 and 2 (figure 1), and one for data sets 3 and 4 (figure 2). In addition, a wordcloud is generated for each Opposition leader’s Twitter posts (figure 3 and 4). An analysis of the wordclouds will be provided in the next section.

*Figure 1 – non-coronavirus related articles Figure 2 – coronavirus related articles*



*Figure 3 – Corbyn’s Twitter posts*



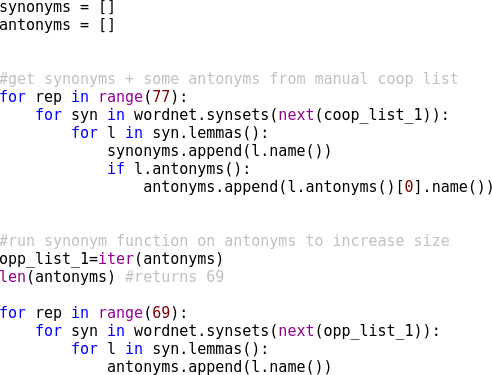
*Figure 4 – Starmer’s Twitter posts*

### Textual analysis

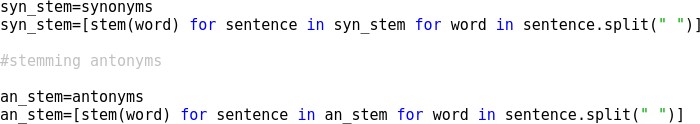
Lexica

The next step in the analysis will be a textual analysis of the words used in the data sets. This step will be useful to identify the attitude that the Opposition is adopting. To do textual analysis, the data sets need to be checked against a lexicon, i.e. a list of words that are indicative of a specific attitude. While there are several well-developed public lists for specific textual analysis, such as for e.g. to evaluate positive or negative comments, there is no such list available that indicates a “cooperative” or a “confrontational” attitude. For that reason, two individual lexica will be generated for this report by making use of the NLTK module in Python.

Any lexicon requires initial manual input. Therefore around 20 words are manually input into the first lexicon which contains words that indicate a cooperative attitude. Examples for words in this list include “cooperate”, “support” or “unite”. Moving forward, all initial words are passed through a synonym and an antonym function using NLTK. Synonyms for cooperative words are appended to the lexicon for cooperative words and antonyms for cooperative words are appended to the lexicon with confrontational words (see snippet 7). This process is then repeated: all newly appended antonyms are run through the synonyms function in order to maximise the number of words. Next, a stemming function is applied in order to avoid issues in textual analysis with words not having the “appropriate” form. For example, a lexicon might contain the word “united” but the text that is being analysed may contain the word “unity”. This should be a match but it would not be in this case. Stemming solves this issue by reducing words to their root. For example, “united” would become “unit”. All lists in the lexicon are thus “stemmed” (see snippet 8). Finally, both lexica are converted into sets in order to remove duplicates. This process yields a lexicon with 283 cooperative words and a lexicon with 523 “confrontational” words.



*Snippet 7*

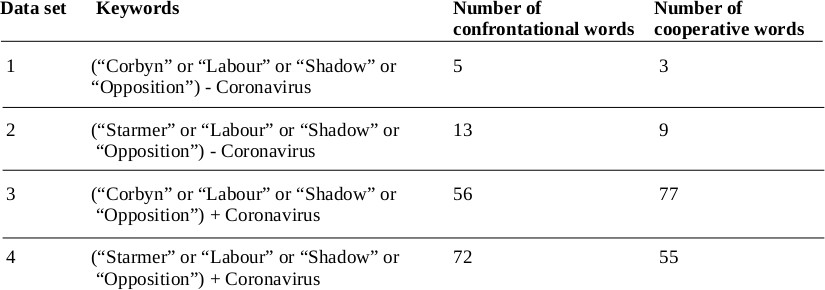


*Snippet 8*

Now that appropriate lexica have been generated, all data sets will be checked against them. This process will verify how many words from each lexica occur in the data sets and with which frequency. It will then list the number of matches in each lexicon.

Textual analysis in articles data sets

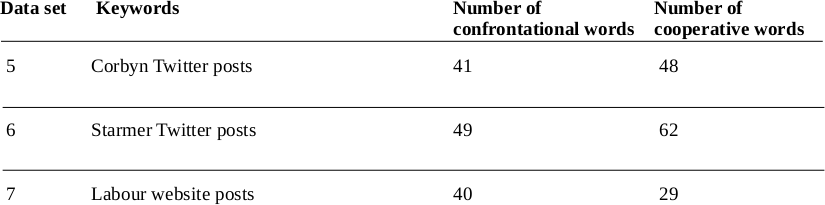
The results of running each article data set through the lexica are listed in table 3 below.



*Table 3*

Textual analysis in social media and website data sets

The results of running each the social media and the website data set through the lexica are listed in table 4 below.



*Table 4*

# Analysis and discussion

## Opposition topics

Certain deductions can be made based on the generated wordclouds in the previous section.

Figure 1 indicates the frequency of words in the non-corona data set (data sets 1 and 2). The relatively smaller data set that was retrieved from *NewsAPI* in that category already showcases that non- coronavirus related topics remain on the Opposition’s agenda but to a much lower degree than coronavirus related ones. Prominent words (excluding “Labour” or “Starmer”,etc.) that appear in the wordcloud include “antisemitism”, “leadership”, “Syria”, “lockdown”, “food” and “death”. The last three words show that despite the exclusion of the word “coronavirus”, coronavirus-related topics still appear in this dataset. It shows that many ancillary topics around coronavirus remain at the forefront. Furthermore, there are a few other topics that dominated the Opposition’s voice, namely the Labour leadership contest and election as indicated by the world “leadership”, “Corbyn” and “Starmer”, and potentially the Syrian civil war or the pending court case against the Syrian government, as indicated by the words “Syria” and “torture”. The presence of Syria as a discussion topic is relatively surprising especially given that it may not be the first concern of citizens at this point in time. It may, however, indicate that the Opposition is attempting to capitalise on non-coronavirus related topics.

Figure 2 depicts the frequency of words in the corona-related data set (data sets 3 and 4). Besides generic words surrounding coronavirus such as “lockdown”, “coronavirus”, “pandemic” or “outbreak”, high frequency terms include “hospital”, “crisis”, “staff”, “police”, “renters”, “China” and “Trump”. This provides strong grounds to believe that there is a clear focus around public sector resources during the crisis such as indicated by the first four high frequency terms. It is unsurprising that strong focus centres around those topics as these are always areas of high concern and relevance among the electorate. Figure 3 confirms this with emphasis around words such as “NHS”, “staff” and “workers”. Other concerns may include “renters” which may be a reference to the vulnerability of that age group

during the crisis. Some international affairs also figure high in the list, as indicated by “Trump” and “China”, which may also not come as a surprise due to the controversies emerging in both these countries during the crisis.

Both leader’s Twitter account, as depicted in figures 3 and 4, also show the dominant coronavirus theme. An emphasis around work related issues also appears with indication such as “selfemployed”, “wage”, “working” or “key worker”. This is also a typical Labour topic and makes particular sense in the current situation as the coronavirus is having a negative impact on job markets.[4](#_bookmark3)

## Opposition attitude

Indications on the Opposition’s attitude can be derived from the textual analysis as depicted in tables 3 and 4. Data sets 1 and 2 show a slight preference towards confrontational words but the overall words are very low given the low number of articles in that category in the first place. More interestingly, data sets 3 and 4, which include coronavirus related topics, show a difference in attitude between both leaders of the Opposition. During Jeremy Corbyn’s period as party leader (data sets 1 and 3), a slight tendency towards cooperative words can be found. This is echoed by Corbyn’s Twitter account. The opposite is true during Keir Starmer’s tenure (data set 6) with a slight tendency towards confrontational words. Interestingly, Starmer’s social media shows a slight tendency towards cooperative words. Announcements on Labour’s website (data set 7) show a preference towards confrontational words.

Nonetheless, all the results in this category are not very conclusive as there is no particularly strong support for either stance. There are, however, indications that the new Opposition leader may be looking for more confrontation than the outgoing one. This could also make sense as the leaving leader was well aware of his exit and might therefore have placed less emphasis on confrontation. The new leader on the other hand might be interested in starting his time in office with more confrontation in order to give the appearance of a “strong” Opposition. The results might, however, also simply indicate that there is a very mixed response from the Opposition. They might criticise certain policies but they might also try to appeal to national unity during the crisis.

1. Resolution Foundation (2020)

# Limitations and qualifications

This report should be read and interpreted while bearing in mind certain limitations and qualifications.

## Data sets limitations

Despite overcoming many API limitations, some have remained. Unfortunately, using the free version of *NewsAPI* meant that only two newspapers were used to generate the data sets. This might have created a bias in favour of what those particular newspapers reported. This might, however, be mitigated by the fact that both newspapers tend to be situated on the political centre rather than on one end of the political spectrum.[5](#_bookmark4)

In addition, the use of newspaper articles may be considered to be an imperfect data source. As newspapers report what politicians say or do, there is always an element of subjectivity in it. To mitigate this, this report also used a secondary data source with direct input from Opposition politicians to complement the findings from the *NewsAPI*. The ease of collection and large data sets of the *NewsAPI*, however, also add strength to the use of newspaper articles.

Finally, the selection of our data set was made on the basis of a number of keywords. These keywords may occasionally retrieve an irrelevant article and as well as occasionally omit a relevant one. However, manual inspection of the retrieved data set revealed that most articles can be said to be relevant for the purpose of this investigation.

## Imperfect NLP and textual analysis

There may also be imperfections surrounding the NLP and textual analysis. Typical data cleaning methods are necessary to enable textual analysis. However, some of these techniques such as, for example, the removal of negative words may actually alter the meaning of the text in some instances. Removing the word “not” in front of “good” reverses the meaning of the expression. Subsequent textual analysis may thus record certain words in the wrong category, i.e. in our case interpret a word as cooperative that may actually be confrontational. There is also a general risk that textual analysis isolates words in their meaning and this does not catch them in the way they were intended in the original article or social media post.

Furthermore, the creation of lexica also has some subjective elements to it. It may also be less developed than some well-established lexica that have been tested and amended by thousands of users. This might imply that certain relevant words were omitted and that perhaps less relevant words were included. Nonetheless, this report tried to use a novel approach to a political problem and was thus unable to use an

1. YouGov (2017)

established lexicon. The subjectivity has also been arguably minimised through the use of the NLTK Python module and the double use of the synonyms and antonyms function.

# Conclusion

The purpose of this report was to investigate the political strategies employed the Opposition in the current political climate dominated by the outbreak of the coronavirus. More precisely, this report aimed at identifying the key areas and topics raised by the Opposition as well as identifying to what extent it adopted a cooperative or confrontational attitude. Employing natural language processing and textual analysis on data retrieved from the NewsAPI, social media and the official Labour website, certain results emerge. The analysis shows that coronavirus related topics is undisputedly the dominant discussion topic for the Opposition. Ancillary topics such as public sector resources, the NHS, the police, employment during the crisis also receive significant attention. Not many non-coronavirus related topics remain at the forefront with an exception to the Labour leadership contest and the political situation in Syria. Textual analysis results give mixed evidence as to whether the Opposition is obviously being more cooperative or confrontational. There is, however, a slight indication that the new Labour leadership under Starmer may be the start of a slightly more confrontational approach. The results discussed in this report will prove useful to the formulation of political strategies of the Conservative Party in response to the Opposition’s moves.

# Sources

BBC News. 2020. *UK Lockdown Extended For 'At Least' Three Weeks*. [online] Available at:

<https://[www.bbc.com/news/uk-52313715>](http://www.bbc.com/news/uk-52313715) [Accessed 24 April 2020].

Httpstatuses.com. 2020. *HTTP Status Codes — Httpstatuses.Com*. [online] Available at:

<https://httpstatuses.com/> [Accessed 24 April 2020].

Resolutionfoundation.org. 2020. *The Economic Effects Of Coronavirus In The UK • Resolution Foundation*. [online] Available at: <https://[www.resolutionfoundation.org/publications/the-economic-](http://www.resolutionfoundation.org/publications/the-economic-) effects-of-coronavirus-in-the-uk/> [Accessed 24 April 2020].

Yougov.co.uk. 2020. *How Left Or Right-Wing Are The UK’S Newspapers? | Yougov*. [online] Available at: <https://yougov.co.uk/topics/politics/articles-reports/2017/03/07/how-left-or-right-wing-are-uks- newspapers> [Accessed 23 April 2020].

# Appendix – Code

##API call

#API call for Corbyn data sets

import pandas as pd import datetime as dt import requests as request

def make\_request(request\_params, request\_uri): # Api Key

apikey = "d1ae8a54ce6a4385a95dd45b842bbad9"

# base url

base\_url = ["ht](http://newsapi.org/v2/)t[p://newsapi.org/v2/](http://newsapi.org/v2/)"

# headers

headers = {"Content-Type": "Application/JSON", "Authorization": apikey}

r = request.get(base\_url + request\_uri, headers=headers, params=request\_params) return r.json()

def get\_everything(from\_date, to\_date, language, query, domains):

return make\_request({'language': language, 'from': from\_date, 'to': to\_date, 'q': query, 'domains': domains}, 'everything')

def feed\_df(columns, results):

feed = {column: [] for column in columns} # insert values in dictionary

for result in results:

for column in columns:

value = result[column].strip('\n') if result[column] is not None else None feed[column].append(value)

return feed

#%% Data retrieval Corbyn +/- Coronoavirus iterations = 100

# from when to start the interrogation start\_date = dt.datetime(2020, 3, 18, 0)

# which language the results should be in language = 'en'

# what are you looking for, accepts logical expressions

query = '(Corbyn OR Labour OR Opposition OR Shadow)+Coronavirus' #source

domains = 'bbc.co.uk, independent.co.uk', # dataframe columns

columns = ['publishedAt', 'title']

# set pandas dataframe columns

df1 = pd.DataFrame(data={'publishedAt': [], 'title': []})

# Interrogate iterations times the api

# default result set is 20 to increase it uncomment ad adjust the value (max is 100) # request\_params["pageSize"] = 100

# in make\_request function for i in range(0, iterations):

# increase end date by 2 hours

end\_date = start\_date + dt.timedelta(hours=4)

# interrogate the desired api with the parameters defined above, an example below api\_results = get\_everything(start\_date, end\_date, language, query, domains)

# check if request was accepted if api\_results['status'] == 'ok':

# add result to the dataframe

df1 = df1.append(pd.DataFrame.from\_dict(feed\_df(columns, api\_results['articles'])), ignore\_index=True)

# swap start\_date with the end\_date so to get the next 20 rtesults start\_date = end\_date

# Choose the column 'title' only df1\_1 = df1.iloc[:,1]

# Convert dataframe to series s1 = pd.Series(df1\_1)

# Delete the duplicates indicated in the series s1\_1 = s1.drop\_duplicates()

# Define the name of the new series without duplicated articles NData\_CorbynPlus = s1\_1

# Convert the servise to list and export it to a txt file NData\_CorbynPlus\_list = NData\_CorbynPlus.values.tolist() with open("NCorbynPlus.txt", "w",encoding="utf-8") as output:

output.write(str(NData\_CorbynPlus\_list))

# from when to start the interrogation start\_date = dt.datetime(2020, 3, 18, 0)

# Keywords that are selected for the research

query = '(Corbyn OR Labour OR Opposition OR Shadow)-Coronavirus'

# set pandas dataframe columns

df2 = pd.DataFrame(data={'publishedAt': [], 'title': []})

# in make\_request function for i in range(0, iterations):

# increase end date by 4 hours

end\_date = start\_date + dt.timedelta(hours=4)

# interrogate the desired api with the parameters defined above, an example below api\_results = get\_everything(start\_date, end\_date, language, query, domains)

# check if request was accepted if api\_results['status'] == 'ok':

# add result to the dataframe

df2 = df2.append(pd.DataFrame.from\_dict(feed\_df(columns, api\_results['articles'])), ignore\_index=True)

# swap start\_date with the end\_date so to get the next 20 rtesults. start\_date = end\_date

# Choose the column 'title' only df2\_1 = df2.iloc[:,1]

# Convert dataframe to series s2 = pd.Series(df2\_1)

# Delete the duplicates indicated in the series s2\_1 = s2.drop\_duplicates()

# Define the name of the new series without duplicated articles NData\_CorbynMinus = s2\_1

# Convert the servise to list and export it to a txt file NData\_CorbynMinus\_list = NData\_CorbynMinus.values.tolist() with open("NCorbynMinus.txt", "w",encoding="utf-8") as output:

output.write(str(NData\_CorbynMinus\_list))

#API call of Starmer data sets import pandas as pd

import datetime as dt import requests as request

def make\_request(request\_params, request\_uri): # Api Key

apikey = "49ddb1daf8f94c1a8dcb43784cf1c423"

# base url

base\_url = ["ht](http://newsapi.org/v2/)t[p://newsapi.org/v2/](http://newsapi.org/v2/)"

# headers

headers = {"Content-Type": "Application/JSON", "Authorization": apikey}

r = request.get(base\_url + request\_uri, headers=headers, params=request\_params) return r.json()

def get\_everything(from\_date, to\_date, language, query, domains):

return make\_request({'language': language, 'from': from\_date, 'to': to\_date, 'q': query, 'domains': domains}, 'everything')

def feed\_df(columns, results):

feed = {column: [] for column in columns} # insert values in dictionary

for result in results:

for column in columns:

value = result[column].strip('\n') if result[column] is not None else None feed[column].append(value)

return feed

#%% Data retrieval Starmer +/- Coronoavirus # Interrogate iterations times the api

# default result set is 20 to increase it uncomment ad adjust the value (max is 100) # request\_params["pageSize"] = 100

iterations = 100

# from when to start the interrogation start\_date = dt.datetime(2020, 4, 4, 0)

# which language the results should be in language = 'en'

# Keywords that are selected for the research

query = '(Starmer OR Labour OR Opposition OR Shadow)+Coronavirus' #source

domains = 'bbc.co.uk, independent.co.uk', # dataframe columns

columns = ['publishedAt', 'title']

# set pandas dataframe columns

df1 = pd.DataFrame(data={'publishedAt': [], 'title': []})

# in make\_request function for i in range(0, iterations):

# increase end date by 4 hours

end\_date = start\_date + dt.timedelta(hours=4)

# interrogate the desired api with the parameters defined above, an example below api\_results = get\_everything(start\_date, end\_date, language, query, domains)

# check if request was accepted if api\_results['status'] == 'ok':

# add result to the dataframe

df1 = df1.append(pd.DataFrame.from\_dict(feed\_df(columns, api\_results['articles'])), ignore\_index=True)

# swap start\_date with the end\_date so to get the next 20 rtesults start\_date = end\_date

# Choose the column 'title' only df1\_1 = df1.iloc[:,1]

# Convert dataframe to series s1 = pd.Series(df1\_1)

# Delete the duplicates indicated in the series s1\_1 = s1.drop\_duplicates()

# Define the name of the new series without duplicated articles NData\_StarmerPlus = s1\_1

# Convert the servise to list and export it to a txt file NData\_StarmerPlus\_list = NData\_StarmerPlus.values.tolist() with open("NStarmerPlus.txt", "w",encoding="utf-8") as output:

output.write(str(NData\_StarmerPlus\_list))

# from when to start the interrogation start\_date = dt.datetime(2020, 4, 4, 0)

# Keywords that are selected for the research

query = '(Starmer OR Labour OR Opposition OR Shadow)-Coronavirus'

# set pandas dataframe columns

df2 = pd.DataFrame(data={'publishedAt': [], 'title': []})

# in make\_request function for i in range(0, iterations):

# increase end date by 4 hours

end\_date = start\_date + dt.timedelta(hours=4)

# interrogate the desired api with the parameters defined above, an example below api\_results = get\_everything(start\_date, end\_date, language, query, domains)

# check if request was accepted if api\_results['status'] == 'ok':

# add result to the dataframe

df2 = df2.append(pd.DataFrame.from\_dict(feed\_df(columns, api\_results['articles'])), ignore\_index=True)

# swap start\_date with the end\_date so to get the next 20 rtesults. start\_date = end\_date

# Choose the column 'title' only df2\_1 = df2.iloc[:,1]

# Convert dataframe to series s2 = pd.Series(df2\_1)

# Delete the duplicates indicated in the series s2\_1 = s2.drop\_duplicates()

# Define the name of the new series without duplicated articles NData\_StarmerMinus = s2\_1

# Convert the servise to list and export it to a txt file NData\_StarmerMinus\_list = NData\_StarmerMinus.values.tolist() with open("NStarmerMinus.txt", "w",encoding="utf-8") as output:

output.write(str(NData\_StarmerMinus\_list))

# attempted web parsing import requests

from bs4 import BeautifulSoup

url = 'https://[www.labour.org.uk/categories/latest//'](http://www.labour.org.uk/categories/latest//%27) reqs = requests.get(url)

soup = BeautifulSoup(reqs.text, 'lxml') print("List of all the h1, h2, h3 :")

for heading in soup.find\_all(["h1", "h2", "h3"]): print(heading.name + ' ' + heading.text.strip())

##NLP and textual Analysis of data sets import re

import nltk

from collections import Counter

from wordcloud import WordCloud # using python 3.7 import matplotlib

matplotlib.use("TkAgg")

from matplotlib import pyplot as plt from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

#cleaning datasets

NEGWORDS = ["not", "no", "none", "neither", "never", "nobody", "n't", 'nor'] # STOPWORDS = ["an", "a", "the"] + NEGWORDS

STOPWORDS = ["an", "a", "the", "or", "and", "thou", "must", "that", "this", "self", "unless", "behind", "for", "which",

"whose", "can", "else", "some", "will", "so", "from", "to", "by", "within", "of", "upon", "th",

"with",

"it"]

stop\_words = set(stopwords.words('english'))

def \_remove\_stopwords(txt):

"""Delete from txt all words contained in STOPWORDS.""" words = txt.split()

# words = txt.split(" ")

for i, word in enumerate(words): if word in STOPWORDS:

words[i] = " " return (" ".join(words))

with open('NCorbynPlus.txt', 'r',encoding='utf-8') as labour\_read: # read(n) method will put n characters into a string labour\_string = labour\_read.read()

labour\_split = str.split(labour\_string, sep=',') doc\_out = []

for k in labour\_split: cleantextprep = str(k)

# Regex cleaning

expression = "[^a-zA-Z ]" # keep only letters, numbers and whitespace cleantextCAP = re.sub(expression, '', cleantextprep) # apply regex cleantext = cleantextCAP.lower() # lower case

cleantext = \_remove\_stopwords(cleantext) bound = ''.join(cleantext) doc\_out.append(bound) # a list of sentences

# print clean text for line in doc\_out:

print(line)

##Verify data sets against lexicons

# Confrontational lexicon ndct = ''

with open('bl\_opp.csv', 'r', encoding='utf-8', errors='ignore') as infile: for line in infile:

ndct = ndct + line

# create a list of confrontational words ndct = ndct.split('\n')

ndct = [entry for entry in ndct] len(ndct)

# Cooperative lexicon pdct = ''

with open('bl\_coop.csv', 'r', encoding='utf-8', errors='ignore') as infile: for line in infile:

pdct = pdct + line

pdct = pdct.split('\n')

pdct = [entry for entry in pdct] len(pdct)

# Count words being collected in the lexicon def decompose\_word(doc):

txt = []

for word in doc: txt.extend(word.split())

return txt

def wordcount(words, dct): counting = Counter(words) count = []

for key, value in counting.items(): if key in dct:

count.append([key, value]) return count

# decompose a list of sentences into words by self-defined function tokens = decompose\_word(doc\_out)

# decompose a list of sentences into words from NLTK module tokens\_nltk = nltk.word\_tokenize(str(doc\_out))

# generate wordcloud

comment\_words = ' ' for token in tokens:

comment\_words = comment\_words + token + ' '

wordcloud = WordCloud(width = 800, height = 800, background\_color ='white',

min\_font\_size = 10).generate(comment\_words)

plt.figure(figsize=(8, 8), facecolor=None) plt.imshow(wordcloud)

plt.axis("off") plt.tight\_layout(pad=0)

plt.savefig("NCorbynPlus.png",format='png',dpi=200) plt.show()

# Number of words in article nwords = len(tokens)

nwc = wordcount(tokens, ndct) # wordcount(text,lexicon)

# [['die', 3], ['famine', 1], ['lies', 2], ['foe', 1], ['cruel', 1], ['gaudy', 1], ['waste', 2], ['pity', 1], ['besiege', 1],

['tattered', 1], ['weed', 1], ['sunken', 1], ['shame', 3], ['excuse', 1], ['cold', 1], ['beguile', 1], ['wrinkles', 1],

['dies', 1], ['abuse', 1], ['deceive', 1], ['hideous', 1], ['sap', 1], ['frost', 1], ['prisoner', 1], ['bereft', 1],

['ragged', 1], ['forbidden', 1], ['death', 1], ['burning', 1], ['weary', 1], ['feeble', 1], ['sadly', 1], ['annoy', 1],

['offend', 1], ['chide', 1], ['wilt', 2], ['fear', 1], ['wail', 1], ['weep', 1], ['deny', 1], ['hate', 2], ['conspire', 1]]

pwc = wordcount(tokens, pdct)

# [['tender', 2], ['bright', 1], ['abundance', 1], ['sweet', 5], ['fresh', 2], ['spring', 1], ['proud', 1], ['worth',

1], ['beauty', 7], ['treasure', 3], ['praise', 2], ['fair', 3], ['proving', 1], ['warm', 1], ['fond', 1], ['lovely', 2],

['golden', 2], ['loveliness', 1], ['free', 1], ['beauteous', 2], ['great', 1], ['gentle', 2], ['work', 1], ['fairly', 1],

['excel', 1], ['leads', 1], ['willing', 1], ['happier', 2], ['gracious', 2], ['homage', 1], ['majesty', 1],

['heavenly', 1], ['strong', 1], ['adore', 1], ['like', 2], ['joy', 2], ['gladly', 1], ['pleasure', 1], ['sweetly', 1],

['happy', 1], ['pleasing', 1], ['well', 1], ['enjoys', 1], ['love', 4], ['beloved', 1]]

# Total number of confrontational/cooperative words ntot, ptot = 0, 0

for i in range(len(nwc)): ntot += nwc[i][1]

for i in range(len(pwc)): ptot += pwc[i][1]

# Print results print('Cooperative words:') for i in range(len(pwc)):

print(str(pwc[i][0]) + ': ' + str(pwc[i][1]))

print('Total number of cooperative words: ' + str(ptot)) print('\n')

print('Percentage of cooperative words: ' + str(round(ptot / nwords, 4))) print('\n')

print('Confrontational words:') for i in range(len(nwc)):

print(str(nwc[i][0]) + ': ' + str(nwc[i][1]))

print('Total number of confrontational words: ' + str(ntot)) print('\n')

print('Percentage of confrontational words: ' + str(round(ntot / nwords, 4)))

#Textual Analysis Corbyn – Coronavirus data set import re

import nltk

from collections import Counter

from wordcloud import WordCloud # using python 3.7 import matplotlib

matplotlib.use("TkAgg")

from matplotlib import pyplot as plt from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

#clean data set

NEGWORDS = ["not", "no", "none", "neither", "never", "nobody", "n't", 'nor'] # STOPWORDS = ["an", "a", "the"] + NEGWORDS

STOPWORDS = ["an", "a", "the", "or", "and", "thou", "must", "that", "this", "self", "unless", "behind", "for", "which",

"whose", "can", "else", "some", "will", "so", "from", "to", "by", "within", "of", "upon", "th",

"with",

"it"]

stop\_words = set(stopwords.words('english'))

def \_remove\_stopwords(txt):

"""Delete from txt all words contained in STOPWORDS.""" words = txt.split()

# words = txt.split(" ")

for i, word in enumerate(words): if word in STOPWORDS:

words[i] = " " return (" ".join(words))

with open('NCorbynMinus.txt', 'r',encoding='utf-8') as labour\_read: # read(n) method will put n characters into a string

labour\_string = labour\_read.read() labour\_split = str.split(labour\_string, sep=',')

doc\_out = []

for k in labour\_split: cleantextprep = str(k)

# Regex cleaning

expression = "[^a-zA-Z ]" # keep only letters, numbers and whitespace cleantextCAP = re.sub(expression, '', cleantextprep) # apply regex cleantext = cleantextCAP.lower() # lower case

cleantext = \_remove\_stopwords(cleantext) bound = ''.join(cleantext) doc\_out.append(bound) # a list of sentences

# print clean text for line in doc\_out:

print(line)

### Read in BL lexicon # confrontational lexicon ndct = ''

with open('bl\_opp.csv', 'r', encoding='utf-8', errors='ignore') as infile: for line in infile:

ndct = ndct + line

# create a list of confrontational words ndct = ndct.split('\n')

ndct = [entry for entry in ndct] len(ndct)

# cooperative lexicon pdct = ''

with open('bl\_coop.csv', 'r', encoding='utf-8', errors='ignore') as infile: for line in infile:

pdct = pdct + line

pdct = pdct.split('\n')

pdct = [entry for entry in pdct] len(pdct)

# Count words being collected in the lexicon

def decompose\_word(doc): txt = []

for word in doc: txt.extend(word.split())

return txt

def wordcount(words, dct): counting = Counter(words) count = []

for key, value in counting.items(): if key in dct:

count.append([key, value]) return count

# decompose a list of sentences into words by self-defined function tokens = decompose\_word(doc\_out)

# decompose a list of sentences into words from NLTK module tokens\_nltk = nltk.word\_tokenize(str(doc\_out))

# generate wordcloud comment\_words = ' ' for token in tokens:

comment\_words = comment\_words + token + ' '

wordcloud = WordCloud(width = 800, height = 800, background\_color ='white',

min\_font\_size = 10).generate(comment\_words)

plt.figure(figsize=(8, 8), facecolor=None) plt.imshow(wordcloud)

plt.axis("off") plt.tight\_layout(pad=0)

plt.savefig("NCorbynMinus.png",format='png',dpi=200) plt.show()

# Number of words in article nwords = len(tokens)

nwc = wordcount(tokens, ndct) # wordcount(text,lexicon)

# [['die', 3], ['famine', 1], ['lies', 2], ['foe', 1], ['cruel', 1], ['gaudy', 1], ['waste', 2], ['pity', 1], ['besiege', 1],

['tattered', 1], ['weed', 1], ['sunken', 1], ['shame', 3], ['excuse', 1], ['cold', 1], ['beguile', 1], ['wrinkles', 1],

['dies', 1], ['abuse', 1], ['deceive', 1], ['hideous', 1], ['sap', 1], ['frost', 1], ['prisoner', 1], ['bereft', 1],

['ragged', 1], ['forbidden', 1], ['death', 1], ['burning', 1], ['weary', 1], ['feeble', 1], ['sadly', 1], ['annoy', 1],

['offend', 1], ['chide', 1], ['wilt', 2], ['fear', 1], ['wail', 1], ['weep', 1], ['deny', 1], ['hate', 2], ['conspire', 1]]

pwc = wordcount(tokens, pdct)

# [['tender', 2], ['bright', 1], ['abundance', 1], ['sweet', 5], ['fresh', 2], ['spring', 1], ['proud', 1], ['worth',

1], ['beauty', 7], ['treasure', 3], ['praise', 2], ['fair', 3], ['proving', 1], ['warm', 1], ['fond', 1], ['lovely', 2],

['golden', 2], ['loveliness', 1], ['free', 1], ['beauteous', 2], ['great', 1], ['gentle', 2], ['work', 1], ['fairly', 1],

['excel', 1], ['leads', 1], ['willing', 1], ['happier', 2], ['gracious', 2], ['homage', 1], ['majesty', 1],

['heavenly', 1], ['strong', 1], ['adore', 1], ['like', 2], ['joy', 2], ['gladly', 1], ['pleasure', 1], ['sweetly', 1],

['happy', 1], ['pleasing', 1], ['well', 1], ['enjoys', 1], ['love', 4], ['beloved', 1]]

# Total number of cooperative/confrontational words ntot, ptot = 0, 0

for i in range(len(nwc)): ntot += nwc[i][1]

for i in range(len(pwc)): ptot += pwc[i][1]

# Print results print('cooperative words:') for i in range(len(pwc)):

print(str(pwc[i][0]) + ': ' + str(pwc[i][1])) print('Total number of cooperative words: ' + str(ptot)) print('\n')

print('Percentage of cooperative words: ' + str(round(ptot / nwords, 4))) print('\n')

print('confrontational words:') for i in range(len(nwc)):

print(str(nwc[i][0]) + ': ' + str(nwc[i][1]))

print('Total number of confrontational words: ' + str(ntot)) print('\n')

print('Percentage of confrontational words: ' + str(round(ntot / nwords, 4)))

#%% Textual Analysis Starmer + Coronoavirus import re

import nltk

from collections import Counter

from wordcloud import WordCloud # using python 3.7 import matplotlib

matplotlib.use("TkAgg")

from matplotlib import pyplot as plt from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

NEGWORDS = ["not", "no", "none", "neither", "never", "nobody", "n't", 'nor'] # STOPWORDS = ["an", "a", "the"] + NEGWORDS

STOPWORDS = ["an", "a", "the", "or", "and", "thou", "must", "that", "this", "self", "unless", "behind", "for", "which",

"whose", "can", "else", "some", "will", "so", "from", "to", "by", "within", "of", "upon", "th",

"with",

"it"]

stop\_words = set(stopwords.words('english'))

def \_remove\_stopwords(txt):

"""Delete from txt all words contained in STOPWORDS.""" words = txt.split()

# words = txt.split(" ")

for i, word in enumerate(words): if word in STOPWORDS:

words[i] = " " return (" ".join(words))

with open('NStarmerPlus.txt', 'r',encoding='utf-8') as labour\_read: # read(n) method will put n characters into a string labour\_string = labour\_read.read()

labour\_split = str.split(labour\_string, sep=',') doc\_out = []

for k in labour\_split: cleantextprep = str(k)

# Regex cleaning

expression = "[^a-zA-Z ]" # keep only letters, numbers and whitespace cleantextCAP = re.sub(expression, '', cleantextprep) # apply regex cleantext = cleantextCAP.lower() # lower case

cleantext = \_remove\_stopwords(cleantext) bound = ''.join(cleantext) doc\_out.append(bound) # a list of sentences

# print clean text for line in doc\_out:

print(line)

### Read in BL lexicon

# confrontational lexicon ndct = ''

with open('bl\_opp.csv', 'r', encoding='utf-8', errors='ignore') as infile: for line in infile:

ndct = ndct + line

# create a list of confrontational words ndct = ndct.split('\n')

ndct = [entry for entry in ndct] len(ndct)

# cooperative lexicon pdct = ''

with open('bl\_coop.csv', 'r', encoding='utf-8', errors='ignore') as infile: for line in infile:

pdct = pdct + line

pdct = pdct.split('\n')

pdct = [entry for entry in pdct] len(pdct)

# Count words being collected in the lexicon def decompose\_word(doc):

txt = []

for word in doc: txt.extend(word.split())

return txt

def wordcount(words, dct): counting = Counter(words) count = []

for key, value in counting.items(): if key in dct:

count.append([key, value]) return count

# decompose a list of sentences into words by self-defined function tokens = decompose\_word(doc\_out)

# decompose a list of sentences into words from NLTK module tokens\_nltk = nltk.word\_tokenize(str(doc\_out))

# generate wordcloud comment\_words = ' ' for token in tokens:

comment\_words = comment\_words + token + ' '

wordcloud = WordCloud(width = 800, height = 800, background\_color ='white',

min\_font\_size = 10).generate(comment\_words)

plt.figure(figsize=(8, 8), facecolor=None) plt.imshow(wordcloud)

plt.axis("off") plt.tight\_layout(pad=0)

plt.savefig("NStarmerPlus.png",format='png',dpi=200) plt.show()

# Number of words in article nwords = len(tokens)

nwc = wordcount(tokens, ndct) # wordcount(text,lexicon)

# [['die', 3], ['famine', 1], ['lies', 2], ['foe', 1], ['cruel', 1], ['gaudy', 1], ['waste', 2], ['pity', 1], ['besiege', 1],

['tattered', 1], ['weed', 1], ['sunken', 1], ['shame', 3], ['excuse', 1], ['cold', 1], ['beguile', 1], ['wrinkles', 1],

['dies', 1], ['abuse', 1], ['deceive', 1], ['hideous', 1], ['sap', 1], ['frost', 1], ['prisoner', 1], ['bereft', 1],

['ragged', 1], ['forbidden', 1], ['death', 1], ['burning', 1], ['weary', 1], ['feeble', 1], ['sadly', 1], ['annoy', 1],

['offend', 1], ['chide', 1], ['wilt', 2], ['fear', 1], ['wail', 1], ['weep', 1], ['deny', 1], ['hate', 2], ['conspire', 1]]

pwc = wordcount(tokens, pdct)

# [['tender', 2], ['bright', 1], ['abundance', 1], ['sweet', 5], ['fresh', 2], ['spring', 1], ['proud', 1], ['worth',

1], ['beauty', 7], ['treasure', 3], ['praise', 2], ['fair', 3], ['proving', 1], ['warm', 1], ['fond', 1], ['lovely', 2],

['golden', 2], ['loveliness', 1], ['free', 1], ['beauteous', 2], ['great', 1], ['gentle', 2], ['work', 1], ['fairly', 1],

['excel', 1], ['leads', 1], ['willing', 1], ['happier', 2], ['gracious', 2], ['homage', 1], ['majesty', 1],

['heavenly', 1], ['strong', 1], ['adore', 1], ['like', 2], ['joy', 2], ['gladly', 1], ['pleasure', 1], ['sweetly', 1],

['happy', 1], ['pleasing', 1], ['well', 1], ['enjoys', 1], ['love', 4], ['beloved', 1]]

# Total number of cooperative/confrontational words ntot, ptot = 0, 0

for i in range(len(nwc)): ntot += nwc[i][1]

for i in range(len(pwc)): ptot += pwc[i][1]

# Print results print('cooperative words:') for i in range(len(pwc)):

print(str(pwc[i][0]) + ': ' + str(pwc[i][1])) print('Total number of cooperative words: ' + str(ptot)) print('\n')

print('Percentage of cooperative words: ' + str(round(ptot / nwords, 4))) print('\n')

print('confrontational words:') for i in range(len(nwc)):

print(str(nwc[i][0]) + ': ' + str(nwc[i][1]))

print('Total number of confrontational words: ' + str(ntot)) print('\n')

print('Percentage of confrontational words: ' + str(round(ntot / nwords, 4)))

#%% Textual Analysis Starmer - Coronoavirus import re

import nltk

from collections import Counter

from wordcloud import WordCloud # using python 3.7 import matplotlib

matplotlib.use("TkAgg")

from matplotlib import pyplot as plt from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

NEGWORDS = ["not", "no", "none", "neither", "never", "nobody", "n't", 'nor'] # STOPWORDS = ["an", "a", "the"] + NEGWORDS

STOPWORDS = ["an", "a", "the", "or", "and", "thou", "must", "that", "this", "self", "unless", "behind", "for", "which",

"whose", "can", "else", "some", "will", "so", "from", "to", "by", "within", "of", "upon", "th",

"with",

"it"]

stop\_words = set(stopwords.words('english'))

def \_remove\_stopwords(txt):

"""Delete from txt all words contained in STOPWORDS.""" words = txt.split()

# words = txt.split(" ")

for i, word in enumerate(words): if word in STOPWORDS:

words[i] = " " return (" ".join(words))

with open('NStarmerMinus.txt', 'r',encoding='utf-8') as labour\_read: # read(n) method will put n characters into a string

labour\_string = labour\_read.read()

labour\_split = str.split(labour\_string, sep=',') doc\_out = []

for k in labour\_split: cleantextprep = str(k)

# Regex cleaning

expression = "[^a-zA-Z ]" # keep only letters, numbers and whitespace cleantextCAP = re.sub(expression, '', cleantextprep) # apply regex cleantext = cleantextCAP.lower() # lower case

cleantext = \_remove\_stopwords(cleantext) bound = ''.join(cleantext) doc\_out.append(bound) # a list of sentences

# print clean text for line in doc\_out:

print(line)

### Read in BL lexicon # confrontational lexicon ndct = ''

with open('bl\_opp.csv', 'r', encoding='utf-8', errors='ignore') as infile: for line in infile:

ndct = ndct + line

# create a list of confrontational words ndct = ndct.split('\n')

ndct = [entry for entry in ndct] len(ndct)

# cooperative lexicon pdct = ''

with open('bl\_coop.csv', 'r', encoding='utf-8', errors='ignore') as infile: for line in infile:

pdct = pdct + line

pdct = pdct.split('\n')

pdct = [entry for entry in pdct] len(pdct)

# Count words being collected in the lexicon

def decompose\_word(doc): txt = []

for word in doc: txt.extend(word.split())

return txt

def wordcount(words, dct): counting = Counter(words) count = []

for key, value in counting.items(): if key in dct:

count.append([key, value]) return count

# decompose a list of sentences into words by self-defined function tokens = decompose\_word(doc\_out)

# decompose a list of sentences into words from NLTK module tokens\_nltk = nltk.word\_tokenize(str(doc\_out))

# generate wordcloud comment\_words = ' ' for token in tokens:

comment\_words = comment\_words + token + ' '

wordcloud = WordCloud(width = 800, height = 800, background\_color ='white',

min\_font\_size = 10).generate(comment\_words)

plt.figure(figsize=(8, 8), facecolor=None) plt.imshow(wordcloud)

plt.axis("off") plt.tight\_layout(pad=0)

plt.savefig("NStarmerMinus.png",format='png',dpi=200) plt.show()

# Number of words in article nwords = len(tokens)

nwc = wordcount(tokens, ndct) # wordcount(text,lexicon)

# [['die', 3], ['famine', 1], ['lies', 2], ['foe', 1], ['cruel', 1], ['gaudy', 1], ['waste', 2], ['pity', 1], ['besiege', 1],

['tattered', 1], ['weed', 1], ['sunken', 1], ['shame', 3], ['excuse', 1], ['cold', 1], ['beguile', 1], ['wrinkles', 1],

['dies', 1], ['abuse', 1], ['deceive', 1], ['hideous', 1], ['sap', 1], ['frost', 1], ['prisoner', 1], ['bereft', 1],

['ragged', 1], ['forbidden', 1], ['death', 1], ['burning', 1], ['weary', 1], ['feeble', 1], ['sadly', 1], ['annoy', 1],

['offend', 1], ['chide', 1], ['wilt', 2], ['fear', 1], ['wail', 1], ['weep', 1], ['deny', 1], ['hate', 2], ['conspire', 1]]

pwc = wordcount(tokens, pdct)

# [['tender', 2], ['bright', 1], ['abundance', 1], ['sweet', 5], ['fresh', 2], ['spring', 1], ['proud', 1], ['worth',

1], ['beauty', 7], ['treasure', 3], ['praise', 2], ['fair', 3], ['proving', 1], ['warm', 1], ['fond', 1], ['lovely', 2],

['golden', 2], ['loveliness', 1], ['free', 1], ['beauteous', 2], ['great', 1], ['gentle', 2], ['work', 1], ['fairly', 1],

['excel', 1], ['leads', 1], ['willing', 1], ['happier', 2], ['gracious', 2], ['homage', 1], ['majesty', 1],

['heavenly', 1], ['strong', 1], ['adore', 1], ['like', 2], ['joy', 2], ['gladly', 1], ['pleasure', 1], ['sweetly', 1],

['happy', 1], ['pleasing', 1], ['well', 1], ['enjoys', 1], ['love', 4], ['beloved', 1]]

# Total number of cooperative/confrontational words ntot, ptot = 0, 0

for i in range(len(nwc)): ntot += nwc[i][1]

for i in range(len(pwc)): ptot += pwc[i][1]

# Print results print('cooperative words:') for i in range(len(pwc)):

print(str(pwc[i][0]) + ': ' + str(pwc[i][1])) print('Total number of cooperative words: ' + str(ptot)) print('\n')

print('Percentage of cooperative words: ' + str(round(ptot / nwords, 4))) print('\n')

print('confrontational words:') for i in range(len(nwc)):

print(str(nwc[i][0]) + ': ' + str(nwc[i][1]))

print('Total number of confrontational words: ' + str(ntot)) print('\n')

print('Percentage of confrontational words: ' + str(round(ntot / nwords, 4))) #%% Textual Analysis Corbyn & Starmer + Coronoavirus

import re import nltk

from collections import Counter

from wordcloud import WordCloud # using python 3.7 import matplotlib

matplotlib.use("TkAgg")

from matplotlib import pyplot as plt from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

NEGWORDS = ["not", "no", "none", "neither", "never", "nobody", "n't", 'nor'] # STOPWORDS = ["an", "a", "the"] + NEGWORDS

STOPWORDS = ["an", "a", "the", "or", "and", "thou", "must", "that", "this", "self", "unless", "behind", "for", "which",

"whose", "can", "else", "some", "will", "so", "from", "to", "by", "within", "of", "upon", "th",

"with",

"it"]

stop\_words = set(stopwords.words('english'))

def \_remove\_stopwords(txt):

"""Delete from txt all words contained in STOPWORDS.""" words = txt.split()

# words = txt.split(" ")

for i, word in enumerate(words): if word in STOPWORDS:

words[i] = " " return (" ".join(words))

with open('CSPlus.txt', 'r',encoding='utf-8') as labour\_read: # read(n) method will put n characters into a string labour\_string = labour\_read.read()

labour\_split = str.split(labour\_string, sep=',') doc\_out = []

for k in labour\_split: cleantextprep = str(k)

# Regex cleaning

expression = "[^a-zA-Z ]" # keep only letters, numbers and whitespace cleantextCAP = re.sub(expression, '', cleantextprep) # apply regex cleantext = cleantextCAP.lower() # lower case

cleantext = \_remove\_stopwords(cleantext) bound = ''.join(cleantext) doc\_out.append(bound) # a list of sentences

# print clean text for line in doc\_out:

print(line)

## Create lexica

import pandas as pd import nltk nltk.download('wordnet') nltk.download('punkt')

from nltk.corpus import wordnet

from nltk.stem import WordNetLemmatizer from stemming.porter2 import stem wordnet\_lemmatizer = WordNetLemmatizer()

coopdf=pd.Series.from\_csv(

'/home/dubious/Documents/University\_19-20/Financial\_Information/bl\_cooperate.csv',index\_col=Fals e)

coopdf.unique() coop\_list=list(coopdf) coop\_list\_1= iter(coop\_list) len(coop\_list) #returns 77

synonyms = [] antonyms = []

#get synonyms + some antonyms from manual coop list for rep in range(77):

for syn in wordnet.synsets(next(coop\_list\_1)): for l in syn.lemmas():

synonyms.append(l.name()) if l.antonyms():

antonyms.append(l.antonyms()[0].name())

#run synonym function on antonyms to increase size opp\_list\_1=iter(antonyms)

len(antonyms) #returns 69

for rep in range(69):

for syn in wordnet.synsets(next(opp\_list\_1)): for l in syn.lemmas():

antonyms.append(l.name())

#repeat same process for my manual oppose list

coopdf2=pd.Series.from\_csv('/home/dubious/Documents/University\_19-20/Financial\_Information/ bl\_oppose.csv',index\_col=False)

coopdf2.unique() coop\_list2=list(coopdf2) coop\_list\_2= iter(coop\_list2) len(coop\_list2) #returns 117

for rep in range(117): # reverse antonyms/synonym allocation for syn in wordnet.synsets(next(coop\_list\_2)):

for l in syn.lemmas(): antonyms.append(l.name()) if l.antonyms():

synonyms.append(l.antonyms()[0].name())

#stemming synonyms syn\_stem=synonyms

syn\_stem=[stem(word) for sentence in syn\_stem for word in sentence.split(" ")] #stemming antonyms

an\_stem=antonyms

an\_stem=[stem(word) for sentence in an\_stem for word in sentence.split(" ")]

#eliminate duplicates by converting to set and back to list synonyms\_set=set(syn\_stem) antonyms\_set=set(an\_stem)

len(synonyms\_set) # return total sample size: 342 len(antonyms\_set) # return total sample size: 610

#finally export s\_list=list(synonyms\_set) a\_list=list(antonyms\_set) s\_df=pd.DataFrame(data=s\_list) a\_df=pd.DataFrame(data=a\_list)

s\_df.to\_csv('/home/dubious/Documents/University\_19-20/Financial\_Information/bl\_coop.csv') a\_df.to\_csv('/home/dubious/Documents/University\_19-20/Financial\_Information/bl\_opp.csv')

## Read in BL lexicon

# confrontational lexicon ndct = ''

with open('bl\_opp.csv', 'r', encoding='utf-8', errors='ignore') as infile: for line in infile:

ndct = ndct + line

# create a list of confrontational words ndct = ndct.split('\n')

ndct = [entry for entry in ndct] len(ndct)

# cooperative lexicon pdct = ''

with open('bl\_coop.csv', 'r', encoding='utf-8', errors='ignore') as infile: for line in infile:

pdct = pdct + line

pdct = pdct.split('\n')

pdct = [entry for entry in pdct] len(pdct)

# Count words being collected in the lexicon def decompose\_word(doc):

txt = []

for word in doc: txt.extend(word.split())

return txt

def wordcount(words, dct): counting = Counter(words) count = []

for key, value in counting.items(): if key in dct:

count.append([key, value]) return count

# decompose a list of sentences into words by self-defined function tokens = decompose\_word(doc\_out)

# decompose a list of sentences into words from NLTK module

tokens\_nltk = nltk.word\_tokenize(str(doc\_out))

# generate wordcloud comment\_words = ' ' for token in tokens:

comment\_words = comment\_words + token + ' '

wordcloud = WordCloud(width = 800, height = 800, background\_color ='white',

min\_font\_size = 10).generate(comment\_words)

plt.figure(figsize=(8, 8), facecolor=None) plt.imshow(wordcloud)

plt.axis("off") plt.tight\_layout(pad=0)

plt.savefig("Corbyn&StarmerPlus.png",format='png',dpi=200) plt.show()

# Number of words in article nwords = len(tokens)

nwc = wordcount(tokens, ndct) # wordcount(text,lexicon)

# [['die', 3], ['famine', 1], ['lies', 2], ['foe', 1], ['cruel', 1], ['gaudy', 1], ['waste', 2], ['pity', 1], ['besiege', 1],

['tattered', 1], ['weed', 1], ['sunken', 1], ['shame', 3], ['excuse', 1], ['cold', 1], ['beguile', 1], ['wrinkles', 1],

['dies', 1], ['abuse', 1], ['deceive', 1], ['hideous', 1], ['sap', 1], ['frost', 1], ['prisoner', 1], ['bereft', 1],

['ragged', 1], ['forbidden', 1], ['death', 1], ['burning', 1], ['weary', 1], ['feeble', 1], ['sadly', 1], ['annoy', 1],

['offend', 1], ['chide', 1], ['wilt', 2], ['fear', 1], ['wail', 1], ['weep', 1], ['deny', 1], ['hate', 2], ['conspire', 1]]

pwc = wordcount(tokens, pdct)

# [['tender', 2], ['bright', 1], ['abundance', 1], ['sweet', 5], ['fresh', 2], ['spring', 1], ['proud', 1], ['worth',

1], ['beauty', 7], ['treasure', 3], ['praise', 2], ['fair', 3], ['proving', 1], ['warm', 1], ['fond', 1], ['lovely', 2],

['golden', 2], ['loveliness', 1], ['free', 1], ['beauteous', 2], ['great', 1], ['gentle', 2], ['work', 1], ['fairly', 1],

['excel', 1], ['leads', 1], ['willing', 1], ['happier', 2], ['gracious', 2], ['homage', 1], ['majesty', 1],

['heavenly', 1], ['strong', 1], ['adore', 1], ['like', 2], ['joy', 2], ['gladly', 1], ['pleasure', 1], ['sweetly', 1],

['happy', 1], ['pleasing', 1], ['well', 1], ['enjoys', 1], ['love', 4], ['beloved', 1]]

# Total number of cooperative/confrontational words ntot, ptot = 0, 0

for i in range(len(nwc)): ntot += nwc[i][1]

for i in range(len(pwc)): ptot += pwc[i][1]

# Print results print('cooperative words:') for i in range(len(pwc)):

print(str(pwc[i][0]) + ': ' + str(pwc[i][1])) print('Total number of cooperative words: ' + str(ptot)) print('\n')

print('Percentage of cooperative words: ' + str(round(ptot / nwords, 4))) print('\n')

print('confrontational words:') for i in range(len(nwc)):

print(str(nwc[i][0]) + ': ' + str(nwc[i][1]))

print('Total number of confrontational words: ' + str(ntot)) print('\n')

print('Percentage of confrontational words: ' + str(round(ntot / nwords, 4))) #%% Textual Analysis Corbyn & Starmer - Coronoavirus

import re import nltk

from collections import Counter

from wordcloud import WordCloud # using python 3.7 import matplotlib

matplotlib.use("TkAgg")

from matplotlib import pyplot as plt from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

NEGWORDS = ["not", "no", "none", "neither", "never", "nobody", "n't", 'nor'] # STOPWORDS = ["an", "a", "the"] + NEGWORDS

STOPWORDS = ["an", "a", "the", "or", "and", "thou", "must", "that", "this", "self", "unless", "behind", "for", "which",

"whose", "can", "else", "some", "will", "so", "from", "to", "by", "within", "of", "upon", "th",

"with",

"it"]

stop\_words = set(stopwords.words('english'))

def \_remove\_stopwords(txt):

"""Delete from txt all words contained in STOPWORDS.""" words = txt.split()

# words = txt.split(" ")

for i, word in enumerate(words): if word in STOPWORDS:

words[i] = " "

return (" ".join(words))

with open('CSMinus.txt', 'r',encoding='utf-8') as labour\_read: # read(n) method will put n characters into a string labour\_string = labour\_read.read()

labour\_split = str.split(labour\_string, sep=',') doc\_out = []

for k in labour\_split: cleantextprep = str(k)

# Regex cleaning

expression = "[^a-zA-Z ]" # keep only letters, numbers and whitespace cleantextCAP = re.sub(expression, '', cleantextprep) # apply regex cleantext = cleantextCAP.lower() # lower case

cleantext = \_remove\_stopwords(cleantext) bound = ''.join(cleantext) doc\_out.append(bound) # a list of sentences

# print clean text for line in doc\_out:

print(line)

### Read in BL lexicon # confrontational lexicon ndct = ''

with open('bl\_opp.csv', 'r', encoding='utf-8', errors='ignore') as infile: for line in infile:

ndct = ndct + line

# create a list of confrontational words ndct = ndct.split('\n')

ndct = [entry for entry in ndct] len(ndct)

# cooperative lexicon pdct = ''

with open('bl\_coop.csv', 'r', encoding='utf-8', errors='ignore') as infile: for line in infile:

pdct = pdct + line

pdct = pdct.split('\n')

pdct = [entry for entry in pdct]

len(pdct)

# Count words being collected in the lexicon def decompose\_word(doc):

txt = []

for word in doc: txt.extend(word.split())

return txt

def wordcount(words, dct): counting = Counter(words) count = []

for key, value in counting.items(): if key in dct:

count.append([key, value]) return count

# decompose a list of sentences into words by self-defined function tokens = decompose\_word(doc\_out)

# decompose a list of sentences into words from NLTK module tokens\_nltk = nltk.word\_tokenize(str(doc\_out))

# generate wordcloud comment\_words = ' ' for token in tokens:

comment\_words = comment\_words + token + ' '

wordcloud = WordCloud(width = 800, height = 800, background\_color ='white',

min\_font\_size = 10).generate(comment\_words)

plt.figure(figsize=(8, 8), facecolor=None) plt.imshow(wordcloud)

plt.axis("off") plt.tight\_layout(pad=0)

plt.savefig("CorbynStarmerMinus.png",format='png',dpi=200) plt.show()

# Number of words in article nwords = len(tokens)

nwc = wordcount(tokens, ndct) # wordcount(text,lexicon)

# [['die', 3], ['famine', 1], ['lies', 2], ['foe', 1], ['cruel', 1], ['gaudy', 1], ['waste', 2], ['pity', 1], ['besiege', 1],

['tattered', 1], ['weed', 1], ['sunken', 1], ['shame', 3], ['excuse', 1], ['cold', 1], ['beguile', 1], ['wrinkles', 1],

['dies', 1], ['abuse', 1], ['deceive', 1], ['hideous', 1], ['sap', 1], ['frost', 1], ['prisoner', 1], ['bereft', 1],

['ragged', 1], ['forbidden', 1], ['death', 1], ['burning', 1], ['weary', 1], ['feeble', 1], ['sadly', 1], ['annoy', 1],

['offend', 1], ['chide', 1], ['wilt', 2], ['fear', 1], ['wail', 1], ['weep', 1], ['deny', 1], ['hate', 2], ['conspire', 1]]

pwc = wordcount(tokens, pdct)

# [['tender', 2], ['bright', 1], ['abundance', 1], ['sweet', 5], ['fresh', 2], ['spring', 1], ['proud', 1], ['worth',

1], ['beauty', 7], ['treasure', 3], ['praise', 2], ['fair', 3], ['proving', 1], ['warm', 1], ['fond', 1], ['lovely', 2],

['golden', 2], ['loveliness', 1], ['free', 1], ['beauteous', 2], ['great', 1], ['gentle', 2], ['work', 1], ['fairly', 1],

['excel', 1], ['leads', 1], ['willing', 1], ['happier', 2], ['gracious', 2], ['homage', 1], ['majesty', 1],

['heavenly', 1], ['strong', 1], ['adore', 1], ['like', 2], ['joy', 2], ['gladly', 1], ['pleasure', 1], ['sweetly', 1],

['happy', 1], ['pleasing', 1], ['well', 1], ['enjoys', 1], ['love', 4], ['beloved', 1]]

# Total number of cooperative/confrontational words ntot, ptot = 0, 0

for i in range(len(nwc)): ntot += nwc[i][1]

for i in range(len(pwc)): ptot += pwc[i][1]

# Print results print('cooperative words:') for i in range(len(pwc)):

print(str(pwc[i][0]) + ': ' + str(pwc[i][1])) print('Total number of cooperative words: ' + str(ptot)) print('\n')

print('Percentage of cooperative words: ' + str(round(ptot / nwords, 4))) print('\n')

print('confrontational words:') for i in range(len(nwc)):

print(str(nwc[i][0]) + ': ' + str(nwc[i][1]))

print('Total number of confrontational words: ' + str(ntot)) print('\n')

print('Percentage of confrontational words: ' + str(round(ntot / nwords, 4)))

#%% Textual Analysis- Corbyn Twitter import re

import nltk

from collections import Counter

from wordcloud import WordCloud # using python 3.7 import matplotlib

matplotlib.use("TkAgg")

from matplotlib import pyplot as plt from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

NEGWORDS = ["not", "no", "none", "neither", "never", "nobody", "n't", 'nor'] # STOPWORDS = ["an", "a", "the"] + NEGWORDS

STOPWORDS = ["an", "a", "the", "or", "and", "thou", "must", "that", "this", "self", "unless", "behind", "for", "which",

"whose", "can", "else", "some", "will", "so", "from", "to", "by", "within", "of", "upon", "th",

"with",

"it"]

stop\_words = set(stopwords.words('english'))

def \_remove\_stopwords(txt):

"""Delete from txt all words contained in STOPWORDS.""" words = txt.split()

# words = txt.split(" ")

for i, word in enumerate(words): if word in STOPWORDS:

words[i] = " " return (" ".join(words))

with open('StarmerTwitter.txt', 'r',encoding='utf-8') as labour\_read: # read(n) method will put n characters into a string labour\_string = labour\_read.read()

labour\_split = str.split(labour\_string, sep=',') doc\_out = []

for k in labour\_split: cleantextprep = str(k)

# Regex cleaning

expression = "[^a-zA-Z ]" # keep only letters, numbers and whitespace cleantextCAP = re.sub(expression, '', cleantextprep) # apply regex cleantext = cleantextCAP.lower() # lower case

cleantext = \_remove\_stopwords(cleantext) bound = ''.join(cleantext) doc\_out.append(bound) # a list of sentences

# print clean text for line in doc\_out:

print(line)

### Read in BL lexicon # confrontational lexicon ndct = ''

with open('bl\_opp.csv', 'r', encoding='utf-8', errors='ignore') as infile: for line in infile:

ndct = ndct + line

# create a list of confrontational words ndct = ndct.split('\n')

ndct = [entry for entry in ndct] len(ndct)

# cooperative lexicon pdct = ''

with open('bl\_coop.csv', 'r', encoding='utf-8', errors='ignore') as infile: for line in infile:

pdct = pdct + line

pdct = pdct.split('\n')

pdct = [entry for entry in pdct] len(pdct)

# Count words being collected in the lexicon def decompose\_word(doc):

txt = []

for word in doc: txt.extend(word.split())

return txt

def wordcount(words, dct): counting = Counter(words) count = []

for key, value in counting.items(): if key in dct:

count.append([key, value]) return count

# decompose a list of sentences into words by self-defined function tokens = decompose\_word(doc\_out)

# decompose a list of sentences into words from NLTK module

tokens\_nltk = nltk.word\_tokenize(str(doc\_out))

# generate wordcloud comment\_words = ' ' for token in tokens:

comment\_words = comment\_words + token + ' '

wordcloud = WordCloud(width = 800, height = 800, background\_color ='white',

min\_font\_size = 10).generate(comment\_words)

plt.figure(figsize=(8, 8), facecolor=None) plt.imshow(wordcloud)

plt.axis("off") plt.tight\_layout(pad=0)

plt.savefig("StarmerTwitter.png",format='png',dpi=200) plt.show()

# Number of words in article nwords = len(tokens)

nwc = wordcount(tokens, ndct) # wordcount(text,lexicon)

# [['die', 3], ['famine', 1], ['lies', 2], ['foe', 1], ['cruel', 1], ['gaudy', 1], ['waste', 2], ['pity', 1], ['besiege', 1],

['tattered', 1], ['weed', 1], ['sunken', 1], ['shame', 3], ['excuse', 1], ['cold', 1], ['beguile', 1], ['wrinkles', 1],

['dies', 1], ['abuse', 1], ['deceive', 1], ['hideous', 1], ['sap', 1], ['frost', 1], ['prisoner', 1], ['bereft', 1],

['ragged', 1], ['forbidden', 1], ['death', 1], ['burning', 1], ['weary', 1], ['feeble', 1], ['sadly', 1], ['annoy', 1],

['offend', 1], ['chide', 1], ['wilt', 2], ['fear', 1], ['wail', 1], ['weep', 1], ['deny', 1], ['hate', 2], ['conspire', 1]]

pwc = wordcount(tokens, pdct)

# [['tender', 2], ['bright', 1], ['abundance', 1], ['sweet', 5], ['fresh', 2], ['spring', 1], ['proud', 1], ['worth',

1], ['beauty', 7], ['treasure', 3], ['praise', 2], ['fair', 3], ['proving', 1], ['warm', 1], ['fond', 1], ['lovely', 2],

['golden', 2], ['loveliness', 1], ['free', 1], ['beauteous', 2], ['great', 1], ['gentle', 2], ['work', 1], ['fairly', 1],

['excel', 1], ['leads', 1], ['willing', 1], ['happier', 2], ['gracious', 2], ['homage', 1], ['majesty', 1],

['heavenly', 1], ['strong', 1], ['adore', 1], ['like', 2], ['joy', 2], ['gladly', 1], ['pleasure', 1], ['sweetly', 1],

['happy', 1], ['pleasing', 1], ['well', 1], ['enjoys', 1], ['love', 4], ['beloved', 1]]

# Total number of cooperative/confrontational words ntot, ptot = 0, 0

for i in range(len(nwc)): ntot += nwc[i][1]

for i in range(len(pwc)): ptot += pwc[i][1]

# Print results print('cooperative words:') for i in range(len(pwc)):

print(str(pwc[i][0]) + ': ' + str(pwc[i][1])) print('Total number of cooperative words: ' + str(ptot)) print('\n')

print('Percentage of cooperative words: ' + str(round(ptot / nwords, 4))) print('\n')

print('confrontational words:') for i in range(len(nwc)):

print(str(nwc[i][0]) + ': ' + str(nwc[i][1]))

print('Total number of confrontational words: ' + str(ntot)) print('\n')

print('Percentage of confrontational words: ' + str(round(ntot / nwords, 4))) #%% Textual Analysis- Starmer Twitter

import re import nltk

from collections import Counter

from wordcloud import WordCloud # using python 3.7 import matplotlib

matplotlib.use("TkAgg")

from matplotlib import pyplot as plt from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

NEGWORDS = ["not", "no", "none", "neither", "never", "nobody", "n't", 'nor'] # STOPWORDS = ["an", "a", "the"] + NEGWORDS

STOPWORDS = ["an", "a", "the", "or", "and", "thou", "must", "that", "this", "self", "unless", "behind", "for", "which",

"whose", "can", "else", "some", "will", "so", "from", "to", "by", "within", "of", "upon", "th",

"with",

"it"]

stop\_words = set(stopwords.words('english'))

def \_remove\_stopwords(txt):

"""Delete from txt all words contained in STOPWORDS.""" words = txt.split()

# words = txt.split(" ")

for i, word in enumerate(words): if word in STOPWORDS:

words[i] = " "

return (" ".join(words))

with open('StarmerTwitter.txt', 'r',encoding='utf-8') as labour\_read: # read(n) method will put n characters into a string labour\_string = labour\_read.read()

labour\_split = str.split(labour\_string, sep=',') doc\_out = []

for k in labour\_split: cleantextprep = str(k)

# Regex cleaning

expression = "[^a-zA-Z ]" # keep only letters, numbers and whitespace cleantextCAP = re.sub(expression, '', cleantextprep) # apply regex cleantext = cleantextCAP.lower() # lower case

cleantext = \_remove\_stopwords(cleantext) bound = ''.join(cleantext) doc\_out.append(bound) # a list of sentences

# print clean text for line in doc\_out:

print(line)

### Read in BL lexicon # confrontational lexicon ndct = ''

with open('bl\_opp.csv', 'r', encoding='utf-8', errors='ignore') as infile: for line in infile:

ndct = ndct + line

# create a list of confrontational words ndct = ndct.split('\n')

ndct = [entry for entry in ndct] len(ndct)

# cooperative lexicon pdct = ''

with open('bl\_coop.csv', 'r', encoding='utf-8', errors='ignore') as infile: for line in infile:

pdct = pdct + line

pdct = pdct.split('\n')

pdct = [entry for entry in pdct]

len(pdct)

# Count words being collected in the lexicon def decompose\_word(doc):

txt = []

for word in doc: txt.extend(word.split())

return txt

def wordcount(words, dct): counting = Counter(words) count = []

for key, value in counting.items(): if key in dct:

count.append([key, value]) return count

# decompose a list of sentences into words by self-defined function tokens = decompose\_word(doc\_out)

# decompose a list of sentences into words from NLTK module tokens\_nltk = nltk.word\_tokenize(str(doc\_out))

# generate wordcloud comment\_words = ' ' for token in tokens:

comment\_words = comment\_words + token + ' '

wordcloud = WordCloud(width = 800, height = 800, background\_color ='white',

min\_font\_size = 10).generate(comment\_words)

plt.figure(figsize=(8, 8), facecolor=None) plt.imshow(wordcloud)

plt.axis("off") plt.tight\_layout(pad=0)

plt.savefig("StarmerTwitter.png",format='png',dpi=200) plt.show()

# Number of words in article nwords = len(tokens)

nwc = wordcount(tokens, ndct) # wordcount(text,lexicon)

# [['die', 3], ['famine', 1], ['lies', 2], ['foe', 1], ['cruel', 1], ['gaudy', 1], ['waste', 2], ['pity', 1], ['besiege', 1],

['tattered', 1], ['weed', 1], ['sunken', 1], ['shame', 3], ['excuse', 1], ['cold', 1], ['beguile', 1], ['wrinkles', 1],

['dies', 1], ['abuse', 1], ['deceive', 1], ['hideous', 1], ['sap', 1], ['frost', 1], ['prisoner', 1], ['bereft', 1],

['ragged', 1], ['forbidden', 1], ['death', 1], ['burning', 1], ['weary', 1], ['feeble', 1], ['sadly', 1], ['annoy', 1],

['offend', 1], ['chide', 1], ['wilt', 2], ['fear', 1], ['wail', 1], ['weep', 1], ['deny', 1], ['hate', 2], ['conspire', 1]]

pwc = wordcount(tokens, pdct)

# [['tender', 2], ['bright', 1], ['abundance', 1], ['sweet', 5], ['fresh', 2], ['spring', 1], ['proud', 1], ['worth',

1], ['beauty', 7], ['treasure', 3], ['praise', 2], ['fair', 3], ['proving', 1], ['warm', 1], ['fond', 1], ['lovely', 2],

['golden', 2], ['loveliness', 1], ['free', 1], ['beauteous', 2], ['great', 1], ['gentle', 2], ['work', 1], ['fairly', 1],

['excel', 1], ['leads', 1], ['willing', 1], ['happier', 2], ['gracious', 2], ['homage', 1], ['majesty', 1],

['heavenly', 1], ['strong', 1], ['adore', 1], ['like', 2], ['joy', 2], ['gladly', 1], ['pleasure', 1], ['sweetly', 1],

['happy', 1], ['pleasing', 1], ['well', 1], ['enjoys', 1], ['love', 4], ['beloved', 1]]

# Total number of cooperative/confrontational words ntot, ptot = 0, 0

for i in range(len(nwc)): ntot += nwc[i][1]

for i in range(len(pwc)): ptot += pwc[i][1]

# Print results print('cooperative words:') for i in range(len(pwc)):

print(str(pwc[i][0]) + ': ' + str(pwc[i][1])) print('Total number of cooperative words: ' + str(ptot)) print('\n')

print('Percentage of cooperative words: ' + str(round(ptot / nwords, 4))) print('\n')

print('confrontational words:') for i in range(len(nwc)):

print(str(nwc[i][0]) + ': ' + str(nwc[i][1]))

print('Total number of confrontational words: ' + str(ntot)) print('\n')

print('Percentage of confrontational words: ' + str(round(ntot / nwords, 4))) #%% Textual Analysis- Corbyn & Starmer Twitter

import re import nltk

from collections import Counter

from wordcloud import WordCloud # using python 3.7 import matplotlib

matplotlib.use("TkAgg")

from matplotlib import pyplot as plt from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

NEGWORDS = ["not", "no", "none", "neither", "never", "nobody", "n't", 'nor'] # STOPWORDS = ["an", "a", "the"] + NEGWORDS

STOPWORDS = ["an", "a", "the", "or", "and", "thou", "must", "that", "this", "self", "unless", "behind", "for", "which",

"whose", "can", "else", "some", "will", "so", "from", "to", "by", "within", "of", "upon", "th",

"with",

"it"]

stop\_words = set(stopwords.words('english'))

def \_remove\_stopwords(txt):

"""Delete from txt all words contained in STOPWORDS.""" words = txt.split()

# words = txt.split(" ")

for i, word in enumerate(words): if word in STOPWORDS:

words[i] = " " return (" ".join(words))

with open('CSTwitter.txt', 'r',encoding='utf-8') as labour\_read: # read(n) method will put n characters into a string labour\_string = labour\_read.read()

labour\_split = str.split(labour\_string, sep=',') doc\_out = []

for k in labour\_split: cleantextprep = str(k)

# Regex cleaning

expression = "[^a-zA-Z ]" # keep only letters, numbers and whitespace cleantextCAP = re.sub(expression, '', cleantextprep) # apply regex cleantext = cleantextCAP.lower() # lower case

cleantext = \_remove\_stopwords(cleantext) bound = ''.join(cleantext) doc\_out.append(bound) # a list of sentences

# print clean text

for line in doc\_out: print(line)

### Read in BL lexicon # confrontational lexicon ndct = ''

with open('bl\_opp.csv', 'r', encoding='utf-8', errors='ignore') as infile: for line in infile:

ndct = ndct + line

# create a list of confrontational words ndct = ndct.split('\n')

ndct = [entry for entry in ndct] len(ndct)

# cooperative lexicon pdct = ''

with open('bl\_coop.csv', 'r', encoding='utf-8', errors='ignore') as infile: for line in infile:

pdct = pdct + line

pdct = pdct.split('\n')

pdct = [entry for entry in pdct] len(pdct)

# Count words being collected in the lexicon def decompose\_word(doc):

txt = []

for word in doc: txt.extend(word.split())

return txt

def wordcount(words, dct): counting = Counter(words) count = []

for key, value in counting.items(): if key in dct:

count.append([key, value]) return count

# decompose a list of sentences into words by self-defined function tokens = decompose\_word(doc\_out)

# decompose a list of sentences into words from NLTK module tokens\_nltk = nltk.word\_tokenize(str(doc\_out))

# generate wordcloud comment\_words = ' ' for token in tokens:

comment\_words = comment\_words + token + ' '

wordcloud = WordCloud(width = 800, height = 800, background\_color ='white',

min\_font\_size = 10).generate(comment\_words)

plt.figure(figsize=(8, 8), facecolor=None) plt.imshow(wordcloud)

plt.axis("off") plt.tight\_layout(pad=0)

plt.savefig("Corbyn&StarmerTwitter.png",format='png',dpi=200) plt.show()

# Number of words in article nwords = len(tokens)

nwc = wordcount(tokens, ndct) # wordcount(text,lexicon)

# [['die', 3], ['famine', 1], ['lies', 2], ['foe', 1], ['cruel', 1], ['gaudy', 1], ['waste', 2], ['pity', 1], ['besiege', 1],

['tattered', 1], ['weed', 1], ['sunken', 1], ['shame', 3], ['excuse', 1], ['cold', 1], ['beguile', 1], ['wrinkles', 1],

['dies', 1], ['abuse', 1], ['deceive', 1], ['hideous', 1], ['sap', 1], ['frost', 1], ['prisoner', 1], ['bereft', 1],

['ragged', 1], ['forbidden', 1], ['death', 1], ['burning', 1], ['weary', 1], ['feeble', 1], ['sadly', 1], ['annoy', 1],

['offend', 1], ['chide', 1], ['wilt', 2], ['fear', 1], ['wail', 1], ['weep', 1], ['deny', 1], ['hate', 2], ['conspire', 1]]

pwc = wordcount(tokens, pdct)

# [['tender', 2], ['bright', 1], ['abundance', 1], ['sweet', 5], ['fresh', 2], ['spring', 1], ['proud', 1], ['worth',

1], ['beauty', 7], ['treasure', 3], ['praise', 2], ['fair', 3], ['proving', 1], ['warm', 1], ['fond', 1], ['lovely', 2],

['golden', 2], ['loveliness', 1], ['free', 1], ['beauteous', 2], ['great', 1], ['gentle', 2], ['work', 1], ['fairly', 1],

['excel', 1], ['leads', 1], ['willing', 1], ['happier', 2], ['gracious', 2], ['homage', 1], ['majesty', 1],

['heavenly', 1], ['strong', 1], ['adore', 1], ['like', 2], ['joy', 2], ['gladly', 1], ['pleasure', 1], ['sweetly', 1],

['happy', 1], ['pleasing', 1], ['well', 1], ['enjoys', 1], ['love', 4], ['beloved', 1]]

# Total number of cooperative/confrontational words ntot, ptot = 0, 0

for i in range(len(nwc)): ntot += nwc[i][1]

for i in range(len(pwc)): ptot += pwc[i][1]

# Print results print('cooperative words:') for i in range(len(pwc)):

print(str(pwc[i][0]) + ': ' + str(pwc[i][1])) print('Total number of cooperative words: ' + str(ptot)) print('\n')

print('Percentage of cooperative words: ' + str(round(ptot / nwords, 4))) print('\n')

print('confrontational words:') for i in range(len(nwc)):

print(str(nwc[i][0]) + ': ' + str(nwc[i][1]))

print('Total number of confrontational words: ' + str(ntot)) print('\n')

print('Percentage of confrontational words: ' + str(round(ntot / nwords, 4)))

#%% Textual Analysis- Labour website import re

import nltk

from collections import Counter

from wordcloud import WordCloud # using python 3.7 import matplotlib

matplotlib.use("TkAgg")

from matplotlib import pyplot as plt from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

NEGWORDS = ["not", "no", "none", "neither", "never", "nobody", "n't", 'nor'] # STOPWORDS = ["an", "a", "the"] + NEGWORDS

STOPWORDS = ["an", "a", "the", "or", "and", "thou", "must", "that", "this", "self", "unless", "behind", "for", "which",

"whose", "can", "else", "some", "will", "so", "from", "to", "by", "within", "of", "upon", "th",

"with",

"it"]

stop\_words = set(stopwords.words('english'))

def \_remove\_stopwords(txt):

"""Delete from txt all words contained in STOPWORDS.""" words = txt.split()

# words = txt.split(" ")

for i, word in enumerate(words): if word in STOPWORDS:

words[i] = " "

return (" ".join(words))

with open('LabourWebsite.txt', 'r',encoding='utf-8') as labour\_read: # read(n) method will put n characters into a string labour\_string = labour\_read.read()

labour\_split = str.split(labour\_string, sep=',') doc\_out = []

for k in labour\_split: cleantextprep = str(k)

# Regex cleaning

expression = "[^a-zA-Z ]" # keep only letters, numbers and whitespace cleantextCAP = re.sub(expression, '', cleantextprep) # apply regex cleantext = cleantextCAP.lower() # lower case

cleantext = \_remove\_stopwords(cleantext) bound = ''.join(cleantext) doc\_out.append(bound) # a list of sentences

# print clean text for line in doc\_out:

print(line)

### Read in BL lexicon # confrontational lexicon ndct = ''

with open('bl\_opp.csv', 'r', encoding='utf-8', errors='ignore') as infile: for line in infile:

ndct = ndct + line

# create a list of confrontational words ndct = ndct.split('\n')

ndct = [entry for entry in ndct] len(ndct)

# cooperative lexicon pdct = ''

with open('bl\_coop.csv', 'r', encoding='utf-8', errors='ignore') as infile: for line in infile:

pdct = pdct + line

pdct = pdct.split('\n')

pdct = [entry for entry in pdct]

len(pdct)

# Count words being collected in the lexicon def decompose\_word(doc):

txt = []

for word in doc: txt.extend(word.split())

return txt

def wordcount(words, dct): counting = Counter(words) count = []

for key, value in counting.items(): if key in dct:

count.append([key, value]) return count

# decompose a list of sentences into words by self-defined function tokens = decompose\_word(doc\_out)

# decompose a list of sentences into words from NLTK module tokens\_nltk = nltk.word\_tokenize(str(doc\_out))

# generate wordcloud comment\_words = ' ' for token in tokens:

comment\_words = comment\_words + token + ' '

wordcloud = WordCloud(width = 800, height = 800, background\_color ='white',

min\_font\_size = 10).generate(comment\_words)

plt.figure(figsize=(8, 8), facecolor=None) plt.imshow(wordcloud)

plt.axis("off") plt.tight\_layout(pad=0)

plt.savefig("LabourWebsite.png",format='png',dpi=200) plt.show()

# Number of words in article nwords = len(tokens)

nwc = wordcount(tokens, ndct) # wordcount(text,lexicon)

# [['die', 3], ['famine', 1], ['lies', 2], ['foe', 1], ['cruel', 1], ['gaudy', 1], ['waste', 2], ['pity', 1], ['besiege', 1],

['tattered', 1], ['weed', 1], ['sunken', 1], ['shame', 3], ['excuse', 1], ['cold', 1], ['beguile', 1], ['wrinkles', 1],

['dies', 1], ['abuse', 1], ['deceive', 1], ['hideous', 1], ['sap', 1], ['frost', 1], ['prisoner', 1], ['bereft', 1],

['ragged', 1], ['forbidden', 1], ['death', 1], ['burning', 1], ['weary', 1], ['feeble', 1], ['sadly', 1], ['annoy', 1],

['offend', 1], ['chide', 1], ['wilt', 2], ['fear', 1], ['wail', 1], ['weep', 1], ['deny', 1], ['hate', 2], ['conspire', 1]]

pwc = wordcount(tokens, pdct)

# [['tender', 2], ['bright', 1], ['abundance', 1], ['sweet', 5], ['fresh', 2], ['spring', 1], ['proud', 1], ['worth',

1], ['beauty', 7], ['treasure', 3], ['praise', 2], ['fair', 3], ['proving', 1], ['warm', 1], ['fond', 1], ['lovely', 2],

['golden', 2], ['loveliness', 1], ['free', 1], ['beauteous', 2], ['great', 1], ['gentle', 2], ['work', 1], ['fairly', 1],

['excel', 1], ['leads', 1], ['willing', 1], ['happier', 2], ['gracious', 2], ['homage', 1], ['majesty', 1],

['heavenly', 1], ['strong', 1], ['adore', 1], ['like', 2], ['joy', 2], ['gladly', 1], ['pleasure', 1], ['sweetly', 1],

['happy', 1], ['pleasing', 1], ['well', 1], ['enjoys', 1], ['love', 4], ['beloved', 1]]

# Total number of cooperative/confrontational words ntot, ptot = 0, 0

for i in range(len(nwc)): ntot += nwc[i][1]

for i in range(len(pwc)): ptot += pwc[i][1]

# Print results print('cooperative words:') for i in range(len(pwc)):

print(str(pwc[i][0]) + ': ' + str(pwc[i][1])) print('Total number of cooperative words: ' + str(ptot)) print('\n')

print('Percentage of cooperative words: ' + str(round(ptot / nwords, 4))) print('\n')

print('confrontational words:') for i in range(len(nwc)):

print(str(nwc[i][0]) + ': ' + str(nwc[i][1]))

print('Total number of confrontational words: ' + str(ntot)) print('\n')

print('Percentage of confrontational words: ' + str(round(ntot / nwords, 4)))