Mapping COVID-19 sentiment and public debate on Reddit

Oliver Carter
C1835576

Supervisor: Steven Schockaert

Moderator: Amir Javed

Content

1. Introduction

Social media played a vital role during the pandemic; Providing millions across the U K with a means to reconnect communities, who had found themselves indefinitely disconnected from o ne another due to the introduction of new, drastic legislation aimed at preventing the spread of this novel coronavirus. This has been proved to be true, as Rajan (2020) from the BBC reveals that adults spent nearly 6 and a half hours a day watching TV and online video during the first lockdown. This much greater utilization of online services to reconnect with the world could serve to be an invaluable source of information for opinion mining.

This project is aimed at examining to what extent Reddit can be used as a useful tool for measuring public debate and sentiment towards COVID-19 and the restrictions that were introduced to slow down the spread of the virus and protect lives. This study will focus more so on the posts and comments from the entire year of year 2020 to focus on the beginning and the first peak of the pandemic.

Popularity in machine learning has arguably grown exponentially over the last decade, with further advancements in computational processing power potential leading to a demand for new ways to further the field of natural language processing. Machine learning tools which have grown from this rise will be used to analyze Reddit users' discourse across the subreddit dedicated to spreading news, advice, and media following the spread of the virus in the UK.

2. Acknowledgements

I would like to take the time to thank my supervisor my supervisor for being incredibly patient with me and for supporting this project all of the way. Our meetings gave me invaluable insight and advice to help keep my project going in the right direction.

3. Related Studies

A study done by Chen et al. (2020) is very closely related to my chosen area of study. Their study aimed to understand Twitter users' discourse and psychological reactions to COVID-19. They use sentiment analysis to derive emotions from within topics extracted from 1,963,285 tweets that contained hashtags containing chosen key-words.

The quality and quantity of their research is something I aspire to reach in future projects.

4. Table of figures

5. Background

This chapter will cover context to the site that data will be collected from, and the software that will be used to: collect, analyze, and extrapolate the data. Beginning with the online messaging and posting forum, Reddit, leading further onto the natural language processing tools used to overcome the tasks involved in ensuring thorough analysis of the data from my chosen source. These topics involve how I came to choose Reddit, and the applications used to extract the data from there, the theory behind sentiment analysis, the miscellaneous python libraries used, how I aim to use opinion mining to derive public debate, why I have chosen to implement a term selection algorithm to support my analysis and the implementation behind that, and finally, what topic modelling is and why it was important for me to uncover hidden topics of conversation and discourse within the text.

5.1. Reddit

Reddit is an online messaging forum that is made up of millions of subreddits which cover just about every single topic one could imagine. A subreddit is a specific online community which posts are associated and related to the topic which it adheres to. Subreddits are supervised by moderators who have the power to dictate what sort of content is allowed to be posted on the subreddit. Reddit in its entirety is presided by Reddit admins who have the authority to strip moderators of their title and remove entire subreddits. A subreddit can feature any number of submissions which can be posted by a user to discuss a topic of interest to the subreddit. This submission can then be commented upon by anyone viewing the subreddit. Submissions are the topics of conversion on a subreddit, whilst the comments are the debate around them.

Widman (2021) perfectly describes the jargon and design of Reddit and breaks down how to use all of Reddit's features in-depth, while mentioning that Reddit's most engaging feature is the ability to be incognito or 'socially low-key as you desire'.

I chose Reddit above all because I believed it to be the perfect challenge for myself. I believe there to be huge potential for to derisive public debate and opinion mine on Reddit.

5.1.1. r/coronavirusUK

The CoronavirusUK subreddit was first created on the 11th of February 2020 for UK residents to discuss their thoughts and opinions towards the surging pandemic, eleven days after the first two cases of coronavirus in the United Kingdom were confirmed by the BBC (2020). It claims to spread news, advice, and media following the UK's spread of the virus. The website subredditstats contains accurate statistics pertaining to every single subreddit on Reddit.

5.1.2. Python Reddit API Wrapper

Python Reddit API Wrapper (PRAW) simplifies access to Python's API. PRAW enables easy access every submission and comment from any subreddit on Reddit. This means that I can search any number of subreddits and return a list of the newest, most popular, or hottest submissions through a series of simple python commands. However it is limited somewhat because there is no feature to search for submissions or comments by date.

5.1.3. Subreddit-comments-dl

Subbreddit-comments-dl is a python application that enables the downloading of any number of submissions from a subreddit. Created by data engineer Simone Guardati, the python application downloads any number of submissions from a given subreddit within a specified date range. One of the main reasons why this application is perfect for my project is that it not only downloads submissions, but their comments too.

5.2. Sentiment Analysis

Sentiment Analysis is a machine-learning, natural language processing task that once solved has the power to identify and extract opinions from any given length of text. Also known as Opinion mining, its objective is to evaluate the attitude and emotion of the text's writer. Pandey (2018) claims that sentiment analysis is so important for many reasons such as its usefulness for practitioners and researchers in fields such as sociology, marketing, advertising, psychology and much more. All of which are heavily dependent on data from human-computer interaction. Pandey (2018) continues by stating that blogging content from social media sites such as Twitter and Facebook pose serious challenges, not only for the sheer volume of data involved, but also the variety of languages in which users use to express sentiment.

5.2.1. VADER

VADER, which stands for Valence Aware Dictionary for Sentiment Reasoning, sentiment analysis model to accurately measure the polarity and strength of emotion. VADER is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. Sentiment analysis, however, is actually a very hard task for computers to perform. Computers are not all that comfortable in comprehending figurative speech and understanding emotions through text is not always that straightforward; Sometimes people can be misled and 100% accuracy from computers is not feasible. Another problem which is not glaringly obvious for humans is that text can often contain multiple sentiments at once, such as the sentence "The best I can say about the movie is that it was interesting". The word 'interesting' does not straightforwardly convey positive sentiment which can confuse the algorithm. Beri (2020) goes into depth about why sentiment analysis is so important, why it's so hard to perform, and how to make your own implementation of VADER sentiment analysis in python. VADER relies on a dictionary that maps lexical features from text into sentiment scores. A compound score is derived from the percentage of positive, neutral, and negative sentiment within the text. Punctuation, capitalization, degree modifiers, conjunctions and preceding tri-grams all impact the sentiment. VADER was perfect for this task as Pandey (2018) points out that it works exceedingly well on social media type text and emojis too. The results of using VADER are quite remarkable and highlight the benefits that can be attained through the usage of VADER with regards to social media type text.

5.3. Term/Feature Selection

Feature or term selection will always play a key role in the world of machine learning. Identifying the most relevant terms in a document collection is a fairly common task. It's an incredibly important problem within machine learning in which one must select the most salient features in which to build a model. One such example of a solution is the chi-square test.

5.3.1. Chi-Square Test for Feature Selection

The chi-square test solves the term selection problem by testing the relationships of features between documents. Gajawada (2019) not only helps to explain the theory and mathematical formula behind the chi-square test but also shows how to implement a working example in Python also. The chi-square test is heavily used in statistics to test the independence of two values. For the purpose of my project, the chi-square test is going to be used to find terms which are highly salient in positive or negative comments, not both. This will be compared to terms found at sentiments which are above and below the interquartile range the compound scores, meaning these terms will be associated with extreme positive and negative sentiment.

5.4. Topic Modelling

Topic modeling is an unsupervised technique to find a set of topics discussed within documents. It's a generative statistical model that allows set of observations to be explained by unobserved groups to explain why some parts of the data are similar. Kapadia (2019) describes topic models as statistical language models used for uncovering hidden structure in a collection of texts. It's two main tasks are Dimensionality Reduction and Unsupervised Learning. Latent Dirichlet Allocation is just one of many existing algorithms used to perform topic modelling.

5.4.1. Latent Dirichlet Allocation

In Latent Dirichlet Allocation (LDA), each document may be viewed as a mixture of various topics where each document is considered to have a set of topics that are assigned to it via LDA. It is a generative probabilistic model that makes the assumption that every topic is a combination over a set of words, and each document is a combination of over a set of topic probabilities. Essentially, each document is assumed to be characterized by a particular set of topics, with each topic assumed to be characterized by a particular set of words.

5.4.2. Gensim

Gensim is a framework for topic modelling in Python. Gensim allows me to implement a working version of the Latent Dirichlet Allocation in python.

5.5. Miscellaneous Python libraries

5.5.1. Pandas

Pandas offers a flexible, open-source data analysis tool in Python. It features: an efficient DataFrame object for data manipulation and indexing, tools for reading and writing data structures, flexible reshaping of data sets, label-based slicing and indexing, quick and easy insertion and deletion of columns, time series-functionality and is highly optimized for performance.

5.5.2. Matplotlib

Matplotlib is an impressive library for the creation of interactive visualizations of data and graphical plotting in Python. I have used Matplotlib to visualize the results of my project. Matplotlib also works brilliantly alongside Pandas to allow me to easily create these data visualizations.

5.5.3. Wordcloud

Luvsandorj (2020) wrote an excellent piece on how quick and easy creating a word cloud generator in Python is and showed me how useful and insightful it can be for exploratory analysis. Word clouds are fantastic at visualizing the frequency of words in any given text.

5.5.4. Scikit-learn

Scikit-learn is a machine learning library for Python. It features many algorithms for classification, regression, clustering, and so much more.

6. Data collection

6.1. Subreddit selection

Before I had received ethical approval to begin my data collection, and devising methods to collect the data even, I set about researching what subreddit(s) to use for data collection. I first designed a spreadsheet with a list of the most populated counties across the United Kingdom and checked to verify if they had a corresponding subreddit on Reddit. It did not take long for me to understand how long it would take to extract a year's worth of data from even just the ten most populous counties in the UK. Furthermore, I would have to devise an algorithm to determine if the topic of a submission was centered around coronavirus or restrictions.

Henceforth, I took a different approach and sought to use the subreddit r/coronavirusUK. This is because I felt the ~89,000 members would have sufficient data over the entirety of 2020, and I could be sure that all of the content was focused on the pandemic and its impact in the UK.

6.2. Downloading subreddit comments

PRAW was first experimented with to examine the quality and quantity of data that could be extracted and analyzed. However, using PRAW alone presented no option to return a list of submissions from a subreddit with a specified datetime range. Reddit data dumps have always been a popular choice to process and analyze large volumes of data from Reddit. Unfortunately, there was not enough data to cover the entire first year of the pandemic and downloading ~15 gigabytes of data to only use a few hundred megabytes of it is not a viable method.

This led me to discover a git repository for a Python program created by Simone Guardati called subreddit-comments-dl. It is a simple but intuitive program that downloads any number of submissions and their comments from a specified subreddit, between a given time frame. The resulting process formed two data structures for all submissions and comments separately.

In total, 574,205 comments and 30,530 submissions were both collected. I could then procede with further steps to ensure the quality of my data.

6.3. Removing neutral comments

After sentiment analysis had been performed on the comment data frame, all neutral comments were removed. This is because neutral comments were, more often than not, deleted or removed, or even just automated messages from a subreddit moderator or bot. These neutral comments affected the overall mean sentiment and were also not of interest to this experiment as I am only interested in comments which reflect a positive or negative sentiment.

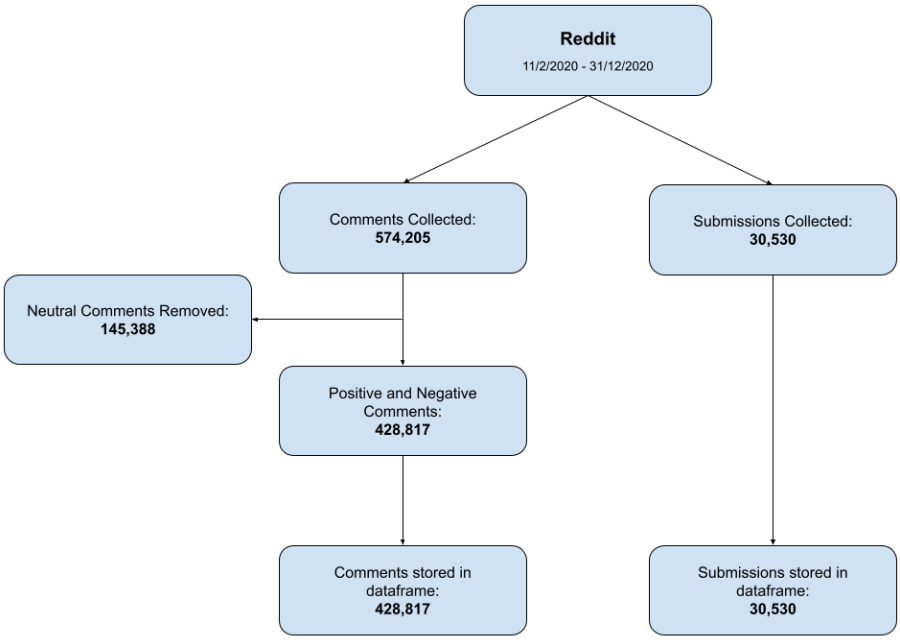

Figure 1. Preprocessing data chart (Personal collection).

6.4. Storing comments and submissions in a data frame

The resulting positive and negative comments, as well as all the submissions they pertain to, are stored in two separate data frames using pandas in Python. The use of pandas enabled me to select and index the data frame quickly and efficiently. The dataframe was saved on my university OneDrive to comply with the university's data storage policy.

id	datetime	body	processed_text	sentiment	compound_score	submission_id	parent_id
gb0v3mc	2020-11-03 17:28:41	You could yes, but that's not the point of the...	you could yes but that's not the point of the ...	positive	0.9186	jndzsq	t1_gb0unis
g9gi4kf	2020-10-20 19:27:51	Fair enough, agree that your particular situat...	fair enough agree that your particular situati...	positive	0.5487	jeojjb	t1_g9ghr6i
ggjcs6b	2020-12-21 00:42:23	I have everything crossable figuratively cross...	i have everything crossable figuratively cross...	positive	0.2484	kh5qfa	t1_ggjcgvj
g2tg5ga	2020-08-25 17:47:46	Consider that the US is in the shit in so many...	consider that the us is in the shit in so many...	positive	0.2500	ig25jx	t1_g2r8h5m
fkr811o	2020-03-17 16:47:15	Oh my parents are the same. I saw them out tod...	oh my parents are the same i saw them out toda...	negative	-0.9433	fk8caw	t3_fk8caw

Figure 2. Sample of the comment data frame

7. Implementation

7.1. Text pre-processing

Before I could begin data analysis, I wanted to ensure the quality of my data. Therefore, I undertook text preprocessing steps were taken such as: removing all punctuation, setting all text to lower case, removing emails and newline characters. Figure 3 shows the regular expressions commands I used on the comments to achieve this. It creates a new column in the comment DataFrame called 'processed_text' from the original comment 'body' without any punctuation before converting all that text to lowercase. Emails and newline characters were not removed until the topic modelling algorithm.

```
# TEXT PREPROCESSING
import re

# REMOVING PUNCTUATION
c_df['processed_text'] = c_df['body'].map(lambda x: re.sub('[,\.!?]', '', x))

# SET ALL TEXT TO LOWERCASE
c_df['processed_text'] = c_df['processed_text'].map(lambda x: x.lower())
```

Figure 3. Code for text preprocessing

This newly processed text was stored alongside the original text in the comment data frame so that I could test the results of both. I did this because, in removing punctuation such as exclamation marks and upper-case text, I hypothesized that the sentiment of comments would change, as I would be removing what's known as sentiment amplifiers or intensifiers. For example, the sentence 'I THOUGHT THE FILM WAS GREAT!!!' would result in a higher overall compound score than 'I thought the film was great'.

7.2. Sentiment analysis

Once the preprocessing of the data had finished, I could begin analyzing all the data. The very first thing I did was create a graph to compare the numbers of posts and comments made over time for the entire year. Figure 5 shows the function that would return a list of categorized sentiments and compound scores for each sentiment.

```
def sentiment_analysis(sentences):
    analyser = SentimentIntensityAnalyzer()
    scores = []
    sentiments = []
    for sentence in sentences:
        score = analyser.polarity_scores(sentence)

        if score["compound"] >= 0.05:
            scores.append(score['compound'])
            sentiments.append('positive')
        elif score["compound"] <= -0.05:
            scores.append(score['compound'])
            sentiments.append('negative')
        elif 0.05 > score["compound"] > -0.05:
            scores.append(score['compound'])
            sentiments.append('neutral')

    return [sentiments, scores]
```

Figure 4. Code for sentiment analysis

The body of text from all comments would be sent in the form of a list as a parameter for this function. Pandas would intuitively set the values for both sentiment and compound score columns by iterating through the return value to assign a value for each row. The compound score of a comment was calculated by using the polarity_scores function from VADER's sentiment intensity analyzer.

The idea behind returning the classified sentiment of a comment, as well as the compound score, was so that I could easily and efficiently categorize positive and negative comments without having to calculate it every time.

7.3. Term selection

Scikit-learn has a great primer example on how to implement term selection with chi-square which was perfectly explained by JustGlowing (2014) using sklearn's 20newsgroups datasets. I used their implementation as inspiration to create my own that was fit for purpose to meet the demands of my project.

7.3.1. Preparing the data for term selection

In Figure 5 I copy and transform the comment data frame by adding another column labelled 'target'. A target then had to be assigned to a comment based on its sentiment.

```
### TRANSFORMS COMMENT DATAFRAME INTO DICTIONARY TO BE VECTORIZED

df_to_dict = c_df.copy().reset_index()

df_to_dict['target'] = df_to_dict.apply(lambda x: get_target(x['sentiment']), axis=1)

df_to_dict = df_to_dict.drop(columns=['datetime',
                                      'sentiment',
                                      'compound_score',
                                      'submission_id',
                                      'parent_id'])

df_dict = {'body': list(df_to_dict['body']),
           'target': list(df_to_dict['target'])}
```

Figure 5. Code to transform the comment database into a dictionary

```
def get_target(sentiment):
    if sentiment == 'neutral':
        return 0
    elif sentiment == 'negative':
        return 1
    elif sentiment == 'positive':
        return 2
```

Figure 6. Code to determine what target document a comment would be assigned to

Figure 6 above shows how simple this is to do: 0 for neutral, 1 for negative, and 2 for positive. By separating the comments into separate documents, I can then calculate how unique AND frequent the terms are for each. I further dropped unnecessary columns from the data frame before passing it into the document vectorizer.

This dictionary was finally passed into my document vectorizer with n being the number of chi terms I wanted to return, which was 25. X is a document-term matrix X_{ij} which records the frequency of a term *i* in a document *j*. I can then compute the relevancy of terms based on the columns of this document-term matrix. The code which was used can be found below in Figure 7.

```
def document_vectorizer(document, n):
    X = vectorizer.fit_transform(document['body'])
    chi2score = chi2(X, document['target'])[0]

    wscores = zip(vectorizer.get_feature_names(), chi2score)
    wchi2 = sorted(wscores, key=lambda x:x[1])

    return list(zip(*wchi2[-n:]))
```

Figure 7. Code to return 25 most salient chi^2 terms

7.3.2. Positive vs negative terms

The first experiment I wanted to conduct was to analyze the most frequent terms found at both positive and negative sentiment. Therefore, I passed the entire comment data frame as neutral comments had been completely removed during data collection and pre-processing.

7.3.3. Extreme positive vs extreme negative terms

The second term selection experiment that I thought would give meaningful insight was looking into terms relevant and frequent in extremely positive and negative sentiment. This would be done by collecting comments that fell into the 25^{th} and 75^{th} percentile for compound score.

7.4. Topic modelling

I closely followed the work of Kapadia (2019) and Revert (2018) to understand the background, implementation, and benefits of topic modelling with Latent Dirichlet Allocation for my project. I took inspiration form their own work to implement a working LDA model for my dataset.

7.4.1. Further text-preprocessing

I firstly created a new variable called data which was a list of all previously processed comments. I then use regular expressions to remove all emails, newline characters and single quotes to make the text even more amendable for my analysis and because they can be distracting. This can be seen in Figure 8.

```
# REMOVE EMAILS AND NEWLINE CHARACTERS

data = comments['processed_text'].values.tolist()

# Remove Emails
data = [re.sub('\S*@\S*\s?', '', sent) for sent in data]

# Remove new line characters
data = [re.sub('\s+', ' ', sent) for sent in data]

# Remove distracting single quotes
data = [re.sub("\'", "", sent) for sent in data]
```

Figure 8. Code used for further text-preprocessing

7.4.2. Exploratory analysis

It was important to me to check that my preprocessing was sufficient, so I used the word cloud library in python. Demonstrated by Luvsandorj (2020), I was able to get a visual representation of the most popular terms in the text. Figure 9. showed that it was not necessary to perform any further preprocessing.

Figure 9. Word cloud of comment sample for exploratory analysis

7.4.3. Data tokenization

The next step was to prepare the text in the best format for my LDA model. I tokenized each word into a list of words. Further removing unnecessary characters and punctuation. Figure 10. below shows the code used to prepare the text data for the LDA. This is done using gensim's simple_preprocess function. I have the parameter 'deacc' set to True to remove any remaining punctuation.

```
# Tokenize words and more text clean-up

import gensim
from gensim.utils import simple_preprocess

def sent_to_words(sentences):
    for sentence in sentences:
        yield(gensim.utils.simple_preprocess(str(sentence),
                                             deacc=True))

data_words = list(sent_to_words(data))
```

Figure 10. Code for data tokenization and clean-up

7.4.4. Building bigram and trigram models

A bigram is the name given to two terms that appear frequently together in a document. Likewise, a trigram refers to three terms that occur frequently together in a document. Examples of bigrams could be 'vaccine_trial' or 'high_risk'. Figure 11 below shows how I created the bigram and trigram models using the topic modelling library genism.

```
#Bigram and Trigram Phrase Modelling

bigram = gensim.models.Phrases(data_words,
                               min_count=5,
                               threshold=100
                              )
trigram = gensim.models.Phrases(bigram[data_words],
                                threshold=100
                                )

bigram_mod = gensim.models.phrases.Phraser(bigram)
trigram_mod = gensim.models.phrases.Phraser(trigram)
```

Figure 11. Code to build bigram and trigram models

Now I could begin to define some more functions to remove to stop words, make the bigrams and trigrams, and lemmatize the data. Figure 12. shows the code I used to create all those functions. Lemmatization is the process of combining groups of the inflected forms of words together so that they can be examined as a single entity. In other words, it's an algorithmic for determining the lemma of words based upon their intended meaning.

```
#Define functions for stopwords, bigrams, trigrams and lemmatization

def remove_stopwords(texts):
    return [[word for word in simple_preprocess(str(doc)) if word not in stop_words] for doc in texts]

def make_bigrams(texts):
    return [bigram_mod[doc] for doc in texts]

def make_trigrams(texts):
    return [trigram_mod[bigram_mod[doc]] for doc in texts]

def lemmatization(texts, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV']):
    texts_out = []
    for sent in texts:
        doc = nlp(" ".join(sent))
        texts_out.append([token.lemma_ for token in doc if token.pos_ in allowed_postags])
    return texts_out
```

Figure 12. Code to remove stop words, create bigrams and trigrams and lemmatize data.

7.4.5. Data transformation for the corpus and dictionary

As the two main inputs of an LDA are a corpus and dictionary called id2word, I had to create them with a few simple lines of code, which can be seen in Figure 13 underneath.

```
#Data Tranformation: Corpus and Dictionary

import gensim.corpora as corpora

id2word = corpora.Dictionary(data_lemmatized)

texts = data_lemmatized

corpus = [id2word.doc2bow(text) for text in texts]
```

Figure 13. Code to transform the data and create the corpus and id2word

Essentially, Gensim assigns a unique index for each word in the document. The corpus is a map of each word index and their frequency.

7.4.6. Building LDA model

I created a function which allowed me to build an LDA with any number of topics. This way I could later improve my model to get the most coherent topics. Figure 14 shows the final LDA model that I went with. The two most important parameters for an LDA model are arguably *chunksize* and *passes*. Chunksize dictates how many documents will be loaded into memory each time it trains. Passes is the total number of iterations through the entire corpus. Limiting these two parameters could result in some documents not converging in time. They need to be high enough that I can be sure that the document swill converge so that topics make sense.

```
# Building LDA model

def build_lda(num_topics):
    return gensim.models.LdaModel(corpus=corpus,
                                  id2word=id2word,
                                  num_topics=num_topics,
                                  random_state=100,
                                  update_every=1,
                                  chunksize=10000,
                                  iterations=200,
                                  passes=40,
                                  alpha='auto',
                                  per_word_topics=True)
```

Figure 14. Function to build an LDA

7.4.7. Testing topic coherence scores * Explain Topic Coherence Score

Coherence score in topic modelling is used for evaluating the quality of topics. Therefore, I would ideally prefer to choose the number of topics which would yield the highest coherence score. I investigated a piece by data scientist Ruchirawat (2020) for 6 tips to improve my coherence score for better quality topics. I used this to increase the chunksize and passes parameters in my LDA model to see what kind of scores I could get. I would use these results to test a number of different topics and determine the best fitting model. The code used can be seen below in Figure 15.

```
for n in trange(1, num_topics + 1):

    lda_model = gensim.models.LdaModel(corpus=corpus,
                                       id2word=id2word,
                                       num_topics=n,
                                       random_state=100,
                                       passes=40,
                                       iterations=200,
                                       chunksize=10000,
                                       alpha='auto',
                                       per_word_topics=True,
                                       eval_every=None
                                      )

    # Compute Coherence Score
    coherence_model_lda = CoherenceModel(model=lda_model,
                                         texts=data_lemmatized,
                                         dictionary=id2word,
                                         coherence='c_v')

    topic_coherence.append(coherence_model_lda.get_coherence())
```

Figure 15. Code to test coherence score for n topics

8. Results

8.1. Exploratory analysis

Before beginning any other data analysis, I wanted to not only check the quality of data that I had, but the quantity of the data. I counted each comment and submission by day to visualize activity on the subreddit. Figure 16 very clearly shows how activity on the subreddit changed over time.

There's an initial surge of activity in mid-March not long after the subreddit was first created. There's a gradual decline in comments and posts up until September where there's another increase in activity culminating with a huge peak at the start of November. Lastly there is another spike in activity right on the cusp of the new year.

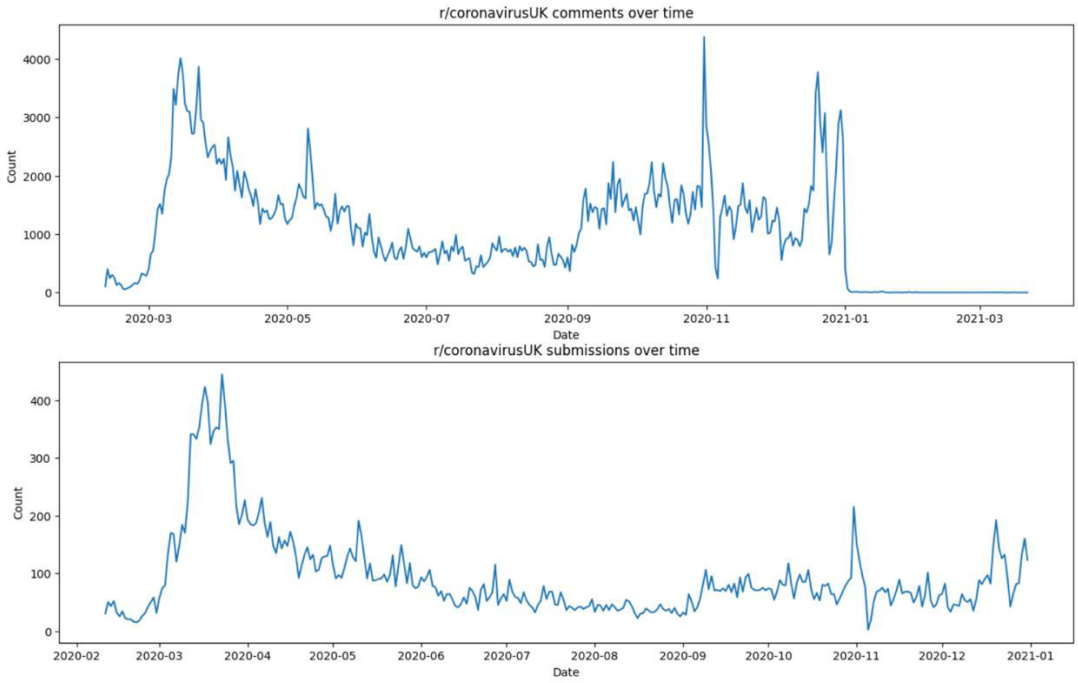

Figure 16. Comments and submissions per day on r/coronavirusUK over time

This data on its own only tells me that there are few peaks throughout the year, therefore I sought to research into why these peaks had occurred. The most obvious choice was to compare the COVID-19 case and death data to this figure.

Figures 17 and 18 below show the daily new cases and deaths in the UK respectively. It may not be as apparent in the figure for daily cases, however, Figure 18 is almost directly correlated to the activity on Reddit. A high daily death rate could be a catalyst to spark more public debate on social media sites such as Reddit. A high cases and death rate could potentially worry the UK population of another lockdown entirely or just more restrictions to control the spread.

These peaks that are separated by troughs in low cases describe the first two waves of the pandemic in the UK. This is visualized in Figure 16 but I believe the lower frequency in posts and comments per day is a result of people not experiencing the same fear of the unknown as they once did in the first wave when all of this very new.

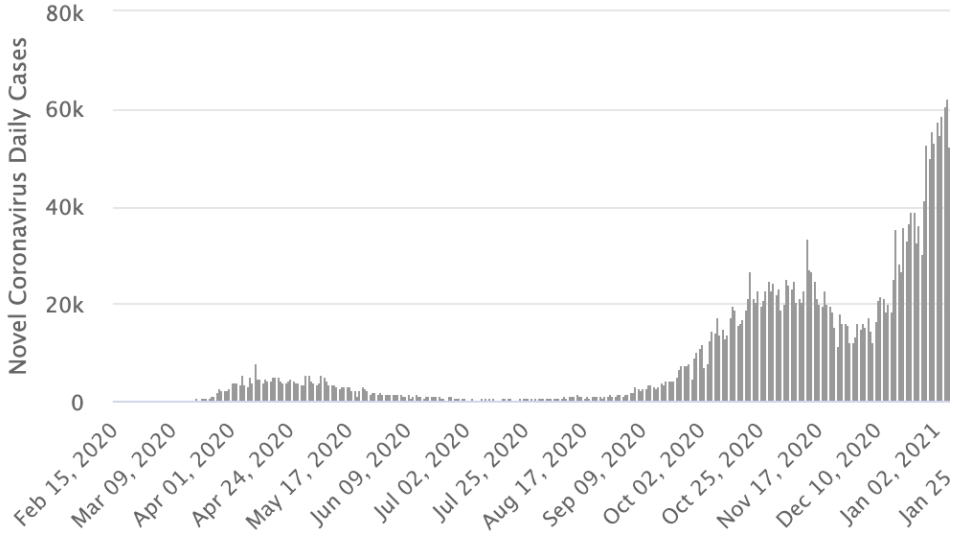

Figure 17. Daily new cases of COVID-19 in the UK (Worldometer, n.d.)

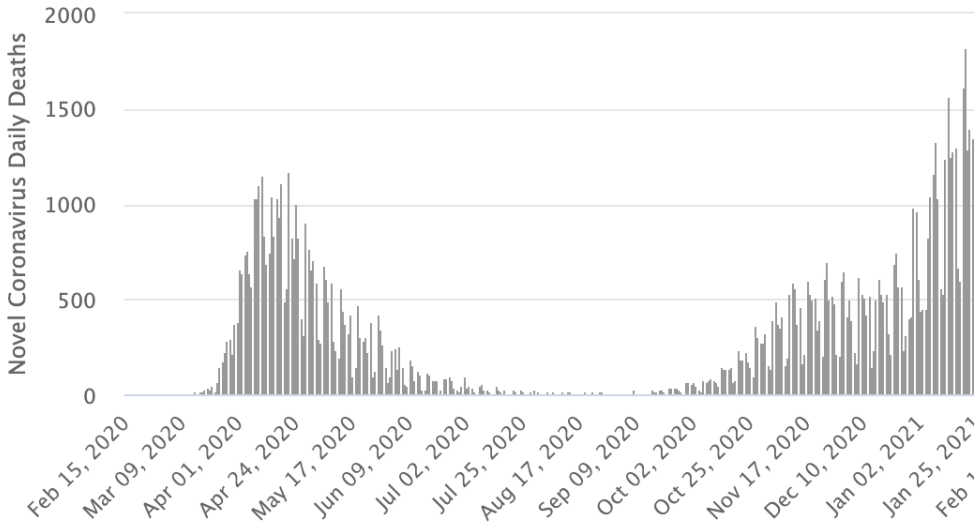

Figure 18. Daily deaths from COVID-19 in the UK (Worldometer, n.d.)

8.2. Sentiment analysis

The results of using the machine learning and natural language processing tool VADER to derive users' thoughts and feelings from the subreddit are not entirely conclusive upon first glance. In Figure 19, I present a graph to show the average sentiment of comments per day. Overall, it shows that the mean sentiment across the period was mildly positive. The reason why there is such a high fluctuation in compound scores towards the end of the graph lies in which the way the data was collected. All of the comments that were collected were pulled from the submissions downloaded by Guardati's subreddit-comment-dl application, which were limited to within the scope of 2020.

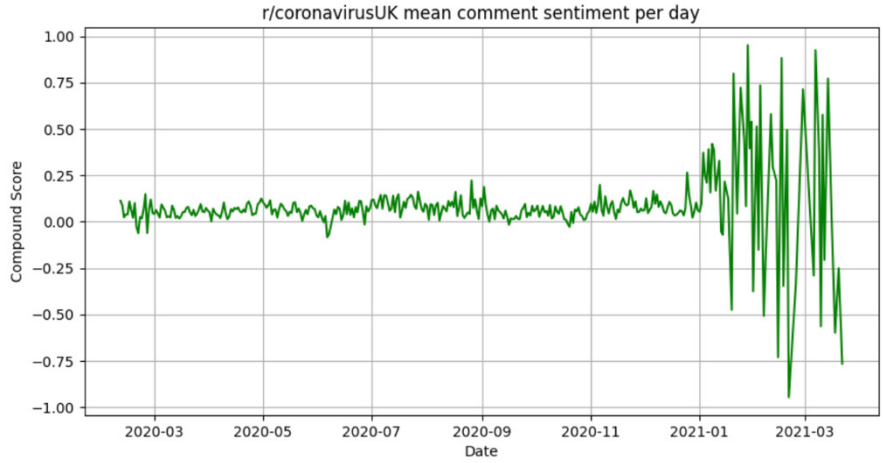

Figure 19. Mean comment sentiment per day

The consequence of this being that there is very little comment data after the 31st of December 2020. Leading to a very high polarity in sentiment after the new year, with Figure 5 showing the drastic drop off in frequency of comments per day after the 31st of December 2020. This has been visualized in Figure 20.

2020-12-26	845
2020-12-27	1540
2020-12-28	2137
2020-12-29	2891
2020-12-30	3119
2020-12-31	2596
2021-01-01	393
2021-01-02	66
2021-01-03	20
2021-01-04	7
2021-01-05	13
2021-01-06	13
2021-01-07	10
2021-01-08	6
2021-01-09	9
2021-01-10	10

Figure 20. Comment count from 26/12/20 – 10/1/21

The solution to this problem as mentioned in the method was to group all comments by their submission ID. My hypothesis was that the discourse and discussion of comments could all be linked back to the topic of the submission. The first graph of which is shown in Figure 21. It tells you that over time, sentiment on the cornavirusUK subreddit was again mildly positive overall with a few peaks from June to September.

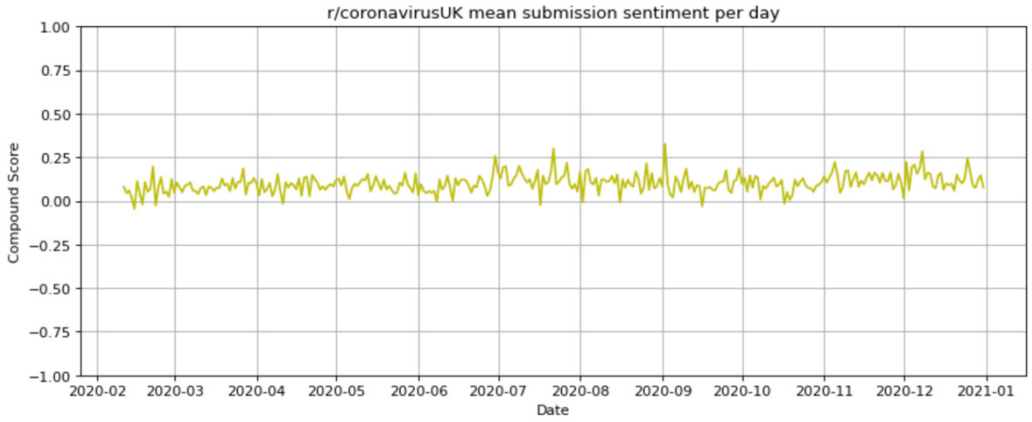

Figure 21. Mean submission sentiment per day

Next, I wanted to visualize the distribution of sentiment across the year. Therefore, instead of plotting the mean sentiment per day, I plotted each compound score individually. The resulting Figure 22 is incredibly interestin. It shows that where there were peaks in activity on the subreddit, there was also a very high polarity in sentiment compound scores.

While one might expect there to be more negative sentiment expressed as a result of lockdowns and a looming pandemic, there could be a fairly simple explanation for this. The figure tells me that while some users could be expressing negative sentiment through the uncertainty of their futures, others may be expressing hope and optimism about theirs.

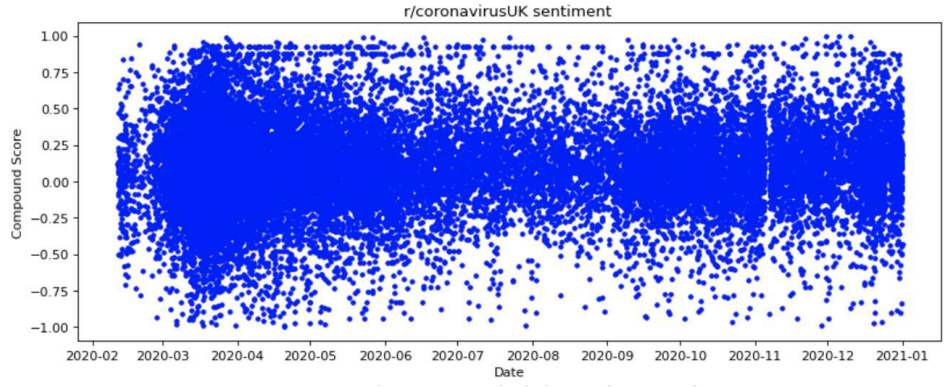

Figure 22. Submission sentiment per day over time

8.3. Most salient chi^2 terms

Figure 23 shows the most salient terms found amongst positive and negative comments in the dataset. It's clear that words like death, die, died, bad, and worse are associated with negative comments and terms like good, thanks, best, and hope are associated with positive comments. Some that might be a little unclear are positive and flu.

Users using the term positive could be someone talking about how they are positive for the future but could also be someone talking about positive COVID-19 cases. Likewise, someone using the term flu could be trying to compare the lethality of COVID-19 to the flu or, on the contrary, using the term to describe just how much more dangerous this coronavirus is.

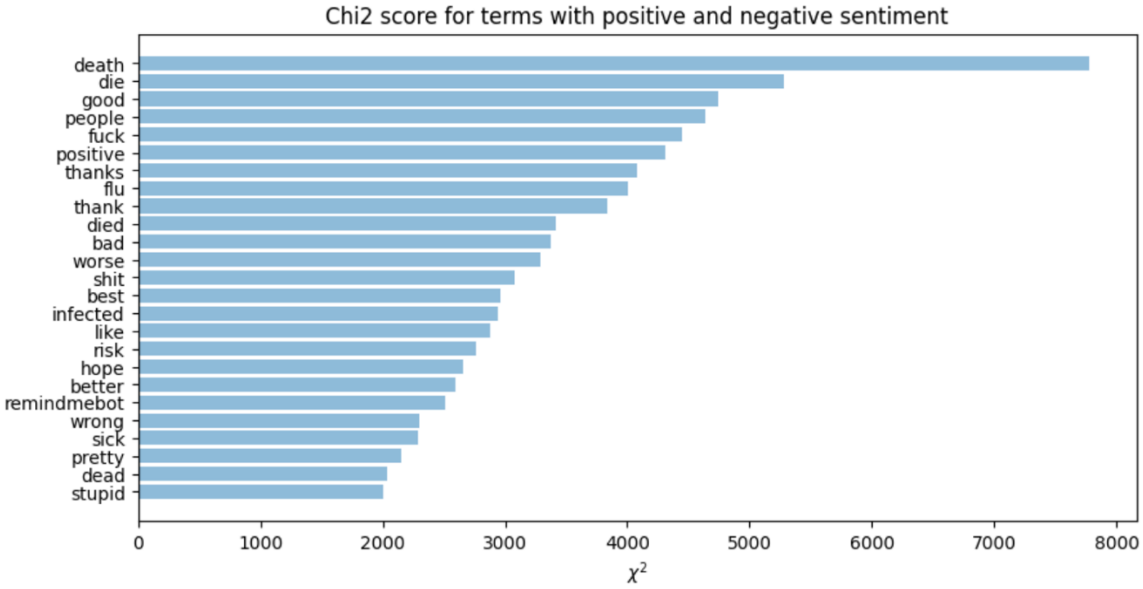

Figure 23. Top 25 most salient chi^2 terms for positive and negative comments

As part of this experiment, I also wanted to look at terms that were salient in extreme positive and negative sentiment. Therefore, I found comments which were found in the upper and lower most quartile of compound score. The results can be seen in Figure 24, which shows a more even spread of salient terms than before. The term Hope is a lot more salient in extremely positive sentiment, while safe and love are two new terms which are most salient in extreme positive sentiment too.

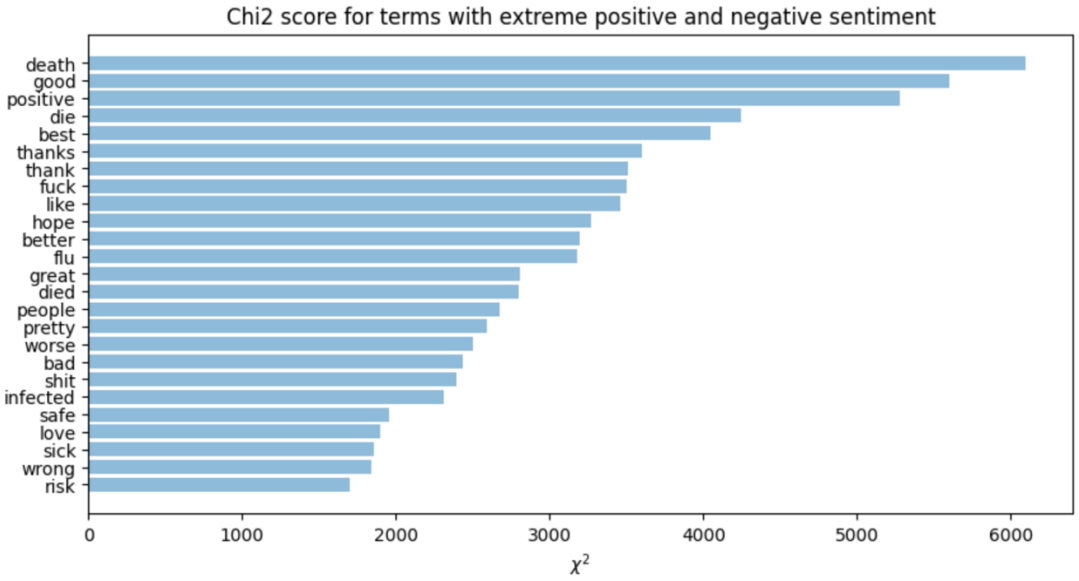

Figure 24. Top 25 most salient chi^2 terms for extreme positive and negative comments

8.4. Topic modelling

The resulting graph for the testing of my topic coherence scores is shown in Figure 25. While selecting the number of topics with the highest coherence score may seem like the logical choice, it is not always best where coherence is highest. Therefore, I tried 10, 12, and 17 topics and visualized them with pyLDAvis.

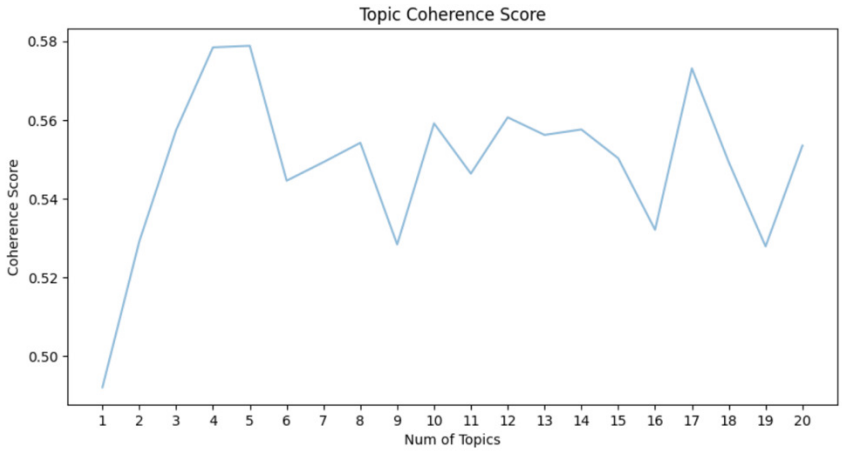

Figure 25. Coherence score testing on the comment data frame

Ultimately, I found that 17 was the optimal number of topics. I found that increasing the number of topics often split up larger ambiguous topics into smaller more related topics. Figure 26 shows the intertopic distance between my visualized topics. While an ideal LDA model should contain no overlapping topics, I found that there were always topics which were not going to the general nature of Reddit and its community.

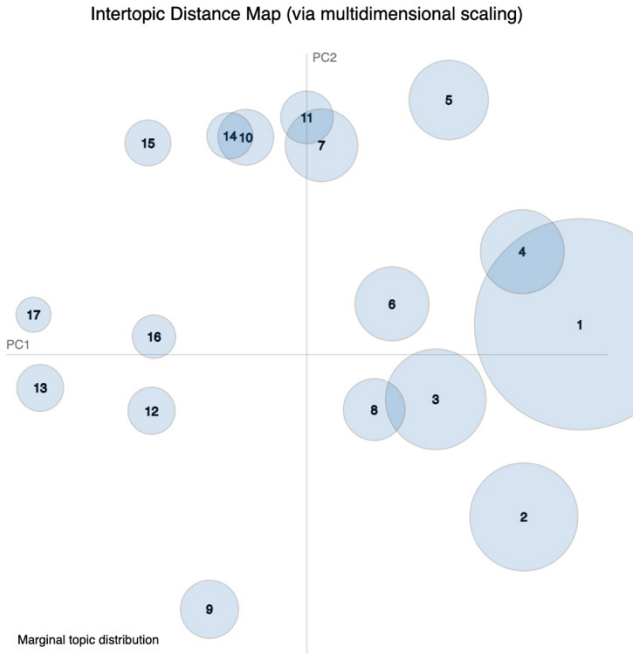

Figure 26. Intertopic distance map for LDA model with 17 topics

To better understand what Figure 26 is actually showing: The circles represent each topic, and the distance between each topic visualizes how related they are. The model tells me how unique each topic is that I have discovered. While I was aiming to minimize the number of overlapping topics as I wanted them to be as unique as possible, I always found that it would detect topics which contained reddit slang or features, which no coherent themes could be derived from.

Of the 17 topics, there were only 14 which had clear themes amongst them.

- Topic 2: Increasing rate of cases and deaths and a potential lockdown.
- Topic 3: Debate around Boris and his governments scientific evidence in response to the pandemic.
- Topic 4: The damage of lockdowns on the economy and society.
- Topic 5: Uncertainty at whether: pubs and gyms would be open, and if people could visit their families for Christmas.
- Topic 6: The NHS and the deaths of vulnerable, elderly people in care.
- Topic 7: Panic buying of essential items.
- Topic 8: Transmission of the virus.
- Topic 9: Figures and reports of daily figures.
- Topic 10: Discourse surrounding mandatory mask wearing.
- Topic 11: Education and online-learning.
- Topic 12: New test and trace application to track positive cases.
- Topic 14: Guidance for people who can't work from home who are self-isolating.
- Topic 16: Discussion and debate surrounding the news of potential vaccines.
- Topic 17: Potential new strains of coronavirus immune to vaccine.

Figure 27 contains a table of all topics which I could derive a clear theme below from based on the association of the words contained within the topics.

Topic 2	case	death	number	lockdown	country	rate	low	population	infection	measure	increase	high	compare	rise
Topic 3	government	public	opinion	claim	medium	science	response	boris	blame	state	expert	decision	clearly	argument
Topic 4	economy	life	society	benefit	mental_health	damage	economic	save	world	furlough	industry	tax	thousand	solution
Topic 5	rule	pub	allow	open	christmas	police	house	gym	visit	close	protest	meet	enforce	christma
Topic 6	nhs	hospital	die	flu	vulnerable	covid	condition	patient	treatment	doctor	age	care	nurse	health
Topic 7	food	buy	hour	eat	store	shop	delivery	supermarket	stock	supply	dog	water	order	essential
Topic 8	virus	spread	transmission	infect	asymptomatic	disease	contain	deadly	reduce	surface	risk	expose	viral	infectious
Topic 9	https	report	yesterday	england	coronavirus	nhttps	update	scotland	datum	figure	ons	date	daily	today
Topic 10	mask	wear	face	touch	wash	hand	bus	idiot	mandatory	wear_mask	glove	mouth	nose	exempt
Topic 11	school	kid	child	parent	student	teacher	class	education	send	adult	son	university	teach	daughter
Topic 12	test	symptom	positive	isolate	trace	negative	app	testing	lol	track	flight	result	sunday	symptomatic
Topic 14	tier	office	guidance	employer	wfh	household	law	area	workplace	advice	regulation	legal	self_isolation	support_bubble
Topic 16	vaccine	vaccinate	dose	trial	vaccination	oxford	approve	coronavirusuk_comment	pfizer	remind	delete	roll	remindmebot	efficacy
Topic 17	thank	reply	immune	strain	zoe	mutation	interesting	mutate	doom	variant	parliament	wtf	excellent	curious

Figure 27. Selected topics with clear themes and their 14 most relevant terms

9. Evaluation

9.1. Text pre-processing

It is very clear to me that the text pre-processing was very successful and important in the results. I made sure to keep both the original body of text from each comment with little pre-processing as well as the final processed text.

Figure 9 showed me that text pre-processing was successful because of the quality of the most popular terms within the text. Removing neutral comments after data collection was important because

9.2. Sentiment analysis

Sentiment analysis on Reddit data was a huge success. The results from Figure 19 were not conclusive and only really proved that the sentiment across the year was fairly neutral.

I then aggregated the mean compound score for each comment by their submission ID, so that I could give each submission a compound score based on the mean score of all their comments. This was I could plot the results as seen in Figure 21 and avoid the fluctuation in results after 2020. The results only proved to be more mildly positive throughout the year with some fluctuation between June and October.

I also used this aggregated submission compound score to plot Figure 22 which showed me that there was a larger polarity in sentiment where the frequency of comments was greater. I found this to correlate with the case and death figures in the UK throughout the year, as whenever cases and particularly deaths were on the rise, activity on the subreddit also rose with it. This can be seen in Figures 17 and 18.

9.3. Term selection

I looked at both positive and negative, and extreme positive and negative comments to see which terms appeared most frequently and exclusively to both. I found the results to be quite impressive in showing that people commonly used the terms death, die, died, bad, worse in negative comments and good, people, thanks, and hope in positive. There are some which are ambiguous such as 'positive' and 'flu' like I mentioned in the results which could be from either.

It was interesting to me to find the words hope, safe, and love had a much higher saliency in extreme positive and opposed to all positive comments. Seen in Figure 24.

9.4. Topic extraction

Of all my experiments, topic modelling has been the one I am most proud of. It is an area of machine-learning that I had not known of before the project began. It was very insightful to be able to model the different themes of discussion within the text. Topic modelling was possibly the most time and resource consuming aspect of the project, with which results I believe could be improved upon with more advanced computational power.

10. Future Work

There are many extra features I would love to have been able to explore and implement if I had more time and the processing power. One such feature would have been geocoding the sentiment and public discourse based on geographical location data found within the text. This is something I had tried early on in the project but was unsuccessful with as not nearly enough submissions actually contained and geo-parseable data. One method of overcoming this would be to search for COVID-19 related submissions and comments on localized subreddits such as r/London or r/Cardiff.

While being able to see how positive and negative sentiment towards COVID-19 restrictions changed over time was incredibly useful, I could not help but feel I was only scratching the surface into examining the public's emotional response to the pandemic in the UK. Therefore, I would have liked to have investigated a method in which I could detect emotions from textual data. Band (2020) introduces the python package text2emotion which aims to do just that. It processes any given text, analyzes it to extract embedded emotions, and outputs a dictionary. This dictionary contains five key emotions: Happiness, Anger, Sadness, Surprise, and Fear. I did a short test and found that with my computer's processing power it would take 111 hours to compute the emotions from all 428,817 comments. Therefore, it was not feasible to try and implement this with my currently available resources.

I would also really like to look more in-depth into the topics and themes I had discovered within the text. I think that looking more into the sentiment expressed within each topic could give an even clearer picture into the public debate around COVID-19 and the related restrictions. For example, the sentiment expressed within the themes: The damage of lockdowns on the economy and society, closures of pubs and gyms and restrictions around Christmas, mandatory mask-wearing, online-learning, and working from home could help to paint a much clearer picture of how really felt about COVID-19 restrictions in the UK.

11. Conclusion

This project took a different path from what was initially intended from the initial report. Limitations in the metadata that Reddit stores on comments and submissions made it incredibly difficult to try and geocode the data. However, I am incredibly satisfied with the outcome.

Overall, Reddit is useful to a large extent when used as a tool to measure sentiment and public debate of COVID-19 and related restrictions. This is because main objectives of my project were to: collect submission and comment data from Reddit, perform sentiment analysis on the comments, visualize how sentiment towards COVID-19 discourse changed over time, find the most popular terms in positive and negative sentiment, and derive themes of discussion within the text, and I feel like I successfully achieved all of those aims.

Data collection went incredibly smoothly with the help of Guardati (n.d.) and I was able to construct a large dataset, sufficient in quantity and quality for my data analysis to come later.

Sentiment analysis, with the help of VADER, produced some very interesting results which were later used to help select the most salient terms with chi-square.

It was very insightful for me to see the most frequent and unique terms to both positive and negative sentiment comments. Chi-square helped solve the term selection problem quickly and efficiently, while

also providing me with a model to train future machine-learning algorithms faster. This is because it reduces the complexity of a model as it only features the most salient terms. Term selection was very useful in analyzing the most popular and unique terms in different sentiments. This could be used in future to build a model that is easier to understand as it will only feature the most salient features.

Topic extraction and modelling has by far been my most proud experiment. It has taught me a different field of natural language processing and machine-learning that allowed me to really challenge my programming skills and the natural language processing theory I have built up during my time at Cardiff. The results of this experiment can definitely be improved upon to find more coherent topics and to use them to measure the sentiment and discourse within the different topics. However, topic modelling increased the utility of Reddit being used to model sentiment and public debate towards COVID-19 and the related restrictions.

This project has been an immense challenge for myself, one which I have already reaped several benefits from. I believe this project has a lot more potential that I implore to explore as there are many possibilities on Reddit for the data science field. My problem solving and time management skills have also greatly benefited from this project too; I have faced many coding problems which I have not yet experienced while studying at Cardiff such as term selection and topic modelling and have been given the opportunity to work on a project with a scale that I have not been used to.

12. References

Band A. (2020), *Text2emotion: Python package to detect emotions from textual data,* Available at: https://towardsdatascience.com/text2emotion-python-package-to-detect-emotions-from-textual-data-b2e7b7ce1153, [Accessed: 12/3/2021].

BBC (2020), *Coronavirus: Two cases confirmed in the UK*, Available at: https://www.bbc.co.uk/news/health-51325192, [Accessed: 1/3/2021].

Beri A. (2020), *SENTIMENT ANALYSIS USING VADER, interpretation and classification of emotions*, Available at: https://towardsdatascience.com/sentimental-analysis-using-vader-a3415fef7664 [Accessed 1/3/2021].

Chen C., Chen J., Li S., Xue J., Zheng C., Zhu T. (2020), *Public discourse and sentiment during COVID 19 pandemic: Using Latent Dirichlet Allocation for topic modelling on Twitter*, Available at: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0239441, [Accessed: 5/3/2021].

Gajawada S.K (2019), *Chi-Square Test for Feature Selection in Machine Learning*, Available at: https://towardsdatascience.com/chi-square-test-for-feature-selection-in-machine-learning-206b1f0b8223, [Accessed 21/3/2021].

Guardati S (n.d)., *subreddit-comments-dl*, Available at: https://github.com/pistocop/subreddit-comments-d, [Accessed: 20/3/2021].

Just Glowing Python (2014), *Term selection with chi-square*, Available at: https://glowingpython.blogspot.com/2014/02/terms-selection-with-chi-square.html, [Accessed: 12/4/2021].

Kapadia S. (2019), *Topic Modelling in Python: Latent Dirichlet Allocation (LDA)*, Available at: https://towardsdatascience.com/end-to-end-topic-modeling-in-python-latent-dirichlet-allocation-lda-35ce4ed6b3e0, [Accessed: 20/4/2021].

Luvsandorj Z. (2020), *Simple word cloud in Python,* Available at: https://towardsdatascience.com/simple-wordcloud-in-python-2ae54a9f58e5, [Accessed: 2/3/2021].

Pandey P (2018), *Simplifying Sentiment Analysis using VADER in Python (on Social Media Text)*, Available at: https://medium.com/analytics-vidhya/simplifying-social-media-sentiment-analysis-using-vader-in-python-f9e6ec6fc52f, [Accessed: 28/2/2021].

Rajan M. (2020), *TV watching and online streaming surge during lockdown*, Available at: https://www.bbc.co.uk/news/entertainment-arts-53637305, [Accessed: 14/4/2021].

Revert F. (2018), An overview of topics extraction in Python with LDA, Available at: https://towardsdatascience.com/the-complete-guide-for-topics-extraction-in-python-a6aaa6cedbbc, [Accessed 20/04/2021].

Ruchirawat N. (2020), *6 Tips for Interpretable Topic Models*, Available at: https://towardsdatascience.com/6-tips-to-optimize-an-nlp-topic-model-for-interpretability-20742f3047e2, [Accessed: 3/5/2021].

Subredditstats (n.d.), *r/CoronavirusUK stats*, Available at: https://subredditstats.com/r/coronavirusuk, [Accessed: 20/4/2021].

Widman J. (2021), *What is Reddit?*, Available at: https://www.digitaltrends.com/web/what-is-reddit/, [Accessed: 20/4/2021].

Worldometer (n.d.), *United Kingdom COVID*, Available at: https://www.worldometers.info/coronavirus/country/uk/, [Accessed: 13/5/2021].