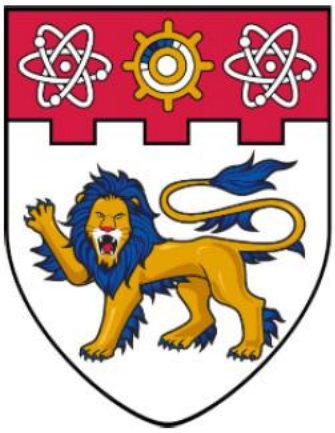


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CZ4034 Information Retrieval

Course Assignment Report

Group 36

Matric No.	Name
U1921026E	Carissa Lim Hui Yi
U1921944C	Hoo Jia Kai
U1920658D	Ng Wee Hsuan
U1921903J	Tan Mei Xuan
U1920270G	Wang Wayne

Table of Contents

1 Introduction

In 2020, the COVID-19 outbreak has been accompanied by the viral spread of fake news, especially on social media. Fake news is considered as disinformation with totally or partially false content, created intentionally to deceive or manipulate the readers.

Examples of COVID-19 fake news can range from the peddling of fake “remedies,” to false conspiracy theories. All this misinformation has a negative impact on the COVID-19 situation. It can distort people’s risk perception of the virus or even the effectiveness of containment strategies made by the government. Furthermore, fake news often imitates the format of real news. This makes it harder for people to detect fake information because fake news often looks like real news. Hence it's difficult for users to find trustworthy and reliable information about COVID-19 when they need it.

To address this issue, we have decided to implement an information retrieval platform whereby users are able to look up covid related news from credible sources. We have crawled COVID-19 related tweets from multiple news channels on Twitter. The news channels' names are ChannelNewsAsia, BBCWorld, straits_times, CNN, YahooNews, Reuters, and nytimes. Our COVID-19 information retrieval platform acts as a one-stop news station for people who want to find trustworthy information regarding COVID-19.

2 Crawling

2.1 Methodology for Crawling of Data

The data is crawled using snscrape library instead of the popular python library Tweepy. This is because Tweepy library has a scraping limit and no access to historical data. However, our 'one-stop' news station requires data that range from the start of covid in the year 2020 to 2022. Snscrape extracts information and returns the discovered items without using Twitter's API. The information can be easily converted to a dataframe or *.csv file.

```
channel_names = ["@ChannelNewsAsia","@BBCWorld" , "@straits_times", "@CNN", "@YahooNews", "@reuters", "@nytimes"]
```

```
for channel in channels:
   # Creating list to append tweet data to
    tweets_list = []


    # Using TwitterSearchScraper to scrape data and append tweets to list
    for i,tweet in enumerate(sntwitter.TwitterSearchScraper('covid 19 OR coronavirus OR #covid19 since:%s until:%s from:%s'\
                                                            %(start,end,channel)).get_items()):
```

Figure 1: Crawling Query with Keywords

As our tweets are only targeted for covid, our specific keyword will be 'covid 19 OR coronavirus'. To get tweets information from different news accounts, we pass different news station usernames in the text query 'from:' and also to get different time periods we specify the date with 'since: until:'. Figure 1 shows an example of the keyword query.

	Datetime	TweetId	Text	RetweetCount	LikeCount	Media	Username
0	2020-12-31 22:57:34	1.34478E+18	WHO lists Pfizer-BioNTech COVID-19 vaccine for emergency	7	10	https://pbs.twimg.com	ChannelNewsAsia
1	2020-12-31 22:27:32	1.34477E+18	Commentary: Pharmaceutical firms saved the world with CO	4	8	https://pbs.twimg.com	ChannelNewsAsia
2	2020-12-31 21:16:03	1.34475E+18	Under fire, France pledges speedier COVID-19 vaccination rc	1	1	https://pbs.twimg.com	ChannelNewsAsia
3	2020-12-31 19:48:03	1.34473E+18	EU reviews BioNTech request for 'extra dose' of COVID-19 va	1	1	https://pbs.twimg.com	ChannelNewsAsia
4	2020-12-31 18:10:34	1.34471E+18	US COVID-19 vaccinations in 2020 fall far short of target of 20	2	1	https://pbs.twimg.com	ChannelNewsAsia
5	2020-12-31 17:28:03	1.3447E+18	Additional COVID-19 measures for marine sector after recen	1	3	https://pbs.twimg.com	ChannelNewsAsia
6	2020-12-31 17:28:02	1.3447E+18	Soccer-Fleetwood Town latest to suspend games after COVI	1	1	https://pbs.twimg.com	ChannelNewsAsia
7	2020-12-31 17:22:03	1.3447E+18	Irish COVID-19 spread worse than formal reporting suggests,	3	7	https://pbs.twimg.com	ChannelNewsAsia
8	2020-12-31 16:55:04	1.34469E+18	Geylang Serai market and 3 restaurants among places visitec	14	17	https://pbs.twimg.com	ChannelNewsAsia
9	2020-12-31 16:49:33	1.34469E+18	Italy reports 555 COVID-19 deaths on Thursday, 23,477 new c	3	4	https://pbs.twimg.com	ChannelNewsAsia
10	2020-12-31 16:04:35	1.34468E+18	New York City aims to vaccinate 1 million against COVID-19 b	2	5	https://pbs.twimg.com	ChannelNewsAsia
11	2020-12-31 15:38:33	1.34467E+18	COVID-19: They were experts in viruses, and now in pitfalls	2	2	https://pbs.twimg.com	ChannelNewsAsia
12	2020-12-31 15:14:05	1.34466E+18	Moderna to supply 40 million doses of COVID-19 vaccine to S	7	12	https://pbs.twimg.com	ChannelNewsAsia
13	2020-12-31 14:47:03	1.34466E+18	Biden inauguration to feature memorial for COVID-19 victim	2	5	https://pbs.twimg.com	ChannelNewsAsia
14	2020-12-31 14:21:34	1.34465E+18	Tennis-Murray pulls out of Delray Beach Open citing COVID-	2	0	https://pbs.twimg.com	ChannelNewsAsia
15	2020-12-31 14:21:33	1.34465E+18	Xi hails China's economic growth despite COVID-19 setback I	3	11	https://pbs.twimg.com	ChannelNewsAsia
16	2020-12-31 13:29:35	1.34464E+18	World begins ushering in locked-down New Year amid COVI	11	36	https://pbs.twimg.com	ChannelNewsAsia
17	2020-12-31 13:04:34	1.34463E+18	Malaysia reports record 2,525 new COVID-19 cases on New Y	35	13	https://pbs.twimg.com	ChannelNewsAsia

Figure 2: Crawled Data in CSV Format

The tweets data is retrieved as a dataframe and exported out as a *.csv file. An example of a *.csv file is shown in Figure 2.

2.2 Sample Queries

Our information retrieval system is focused on the integration of COVID-19 related news from various sources. Therefore, users could query common words that are related to COVID-19 from our information database. For example, users will be able to search for “Pfizer” and get insights into how each media channel reports about the COVID-19 vaccine. Example keyword queries that users might like to search are “Pfizer”, “Moderna”, and “lockdown”.

2.3 Text Corpus Information

Overall, our combined dataset has 10262 tweets crawled from Twitter. The dataset is stored as 3 *.csv files and each *.csv file corresponds to the years 2020, 2021, and 2022. For the year 2022, we only extracted data from 2022-01-01 to 2022-03-12. To make our dataset balanced, we tried to extract the same amount of tweets from each channel. However, some news channels may not have enough tweets to extract from. Table 1 shows the distribution of data from each channel.

Channels	**Year: 2020**	**Year: 2021**	**Year: 2022**
ChannelNewsAsia	650	650	400
BBCWorld	469	231	8
straits_times	650	650	311
CNN	650	650	400
YahooNews	650	650	225
Reuters	650	650	400
nytimes	575	650	93
Total	4294	4131	1837

Table 1: Distribution of Data

Field	Description	Example Content
Datetime	Date & Time the tweet was created	2020-12-31 22:57:34+00:00
TweetId	Unique identifier of tweet	1344779760623230000
Text	Text content of the tweet. The URL is the link to the real tweet on Twitter.	WHO lists Pfizer-BioNTech COVID-19 vaccine for emergency use https://t.co/NPkPodAhQX
RetweetCount	Number of retweets	7
LikeCount	Number of likes of the tweet	10
Media	The image/video URL embedded in the tweets (Some tweets may not have it)	https://pbs.twimg.com/media/EqmeEODUwAA0pcZ?format=jpg&name=large
Username	Twitter account name	ChannelNewsAsia

Table 2: Details of the different columns

Each row in the *.csv file corresponds to a tweet. There are a total of 213,247 words crawled for the “Text” column. After removing stop words and stemming using PorterStemmer, 140,982 words are left. Among those, there are 8509 unique tokens. Our crawled data has 7 columns, the details are shown in Table 2.

3 Indexing of Text Corpus

After we have obtained our tweets data containing the relevant COVID-19 information, we will now index the crawled data using Solr. This creates an index data structure that stores a mapping of content, in this case, words, to its locations in the set of documents, in this case, tweets. The purpose of this is to allow for fast information retrieval, accommodating full-text searches but with a tradeoff of an increased processing time when adding documents to the database.

3.1 Indexing using Solr

Solr is an open-source enterprise search engine that is written in Java. Its main feature which we will use is index creation and full-text search. Under the hood, Solr uses Lucene Java as its core search library within a servlet container Jetty. This allows the support of REST-like HTTP requests and JSON APIs which enables easy deployment of our frontend query application.

The setup process is quite simple to get started quickly:

1. Download the latest Solr binary release from https://solr.apache.org/downloads.html
2. Unpack the solr-8.11.1.tgz file and drag the folder into the working directory
3. Change directory to the Solr directory
   $ cd solr-8.11.1
4. Start the Solr server (Should be running on localhost:8983)
   $ bin/solr start
5. Create the Solr core for tweets using the following command
   $ bin/solr create_core -c tweets
6. Post the documents (CSV) into the Solr core
   $ bin/post -c tweets /Users/wayne.wang/Downloads/covid_data.csv
7. Now we are able to test queries on localhost:8983
   Under q: Text:Pfizer and “execute query”, we can see a number of results.

3.2 Search relevancy in Solr

Lucene, or Solr, combines both the Boolean model and Vector Space Model to query for matching documents. Documents that are first “approved” by the Boolean Model are then scored using the Vector Space Model. To rank its search results, a formula called the practical scoring function is used to calculate the search relevance of each document. This function returns a positive floating-point number which represents the relevance of each document.

3.2.1 Boolean Model

For the Boolean Model, the standard algorithm used to determine the similarity between a query and a document is term frequency/inverse document frequency, also known as TF-IDF.

Term frequency represents the extent of occurrence of a search term. The more times the term appears, the more relevant the document is.

$$tf(term\ in\ document) = \sqrt{frequency}$$

Inverse document frequency is a measure of how frequent a word occurs across different documents in the collection. The more often, the lower the weight.

$$idf(term) = 1 + log(\frac{number\ of\ documents}{document\ frequency + 1})$$

Lucene also takes “field-length norm” into account so terms matched in fields with fewer terms have a higher score. The shorter the field, the higher the weight.

$$norm(document) = \frac{1}{\sqrt{number\ of\ terms}}$$

Lucene’s final *Practical Scoring Function* contains the above-mentioned weights, as well as other more advanced customizable weights which are specific to user specifications, and not applicable in this project.

$$score(q,d) = coord(q,d) \cdot queryNorm(q) \cdot \sum_{t\ in\ q} \left(tf(t\ in\ d) \cdot idf(t)^2 \cdot t.getBoost() \cdot norm(t,d) \right)$$

3.2.2 Vector Space Model

As for the Vector Space Model, documents and queries are represented as weighted vectors in a multi-dimensional space, where each unique index term is a vector dimension, and its weight is its TF-IDF value. Lucene uses the cosine similarity of the weighted query and document vectors to rank the relevance.

$$cosine\ similarity(V(q), V(d)) = \frac{V(q) \cdot V(d)}{|V(q)||V(d)|}$$

4 User Interface (UI) for Querying

We built a simple web-based UI for querying our covid related tweets using HTML, javascript and the ReactJS library. Below is a screenshot of our web-based UI.

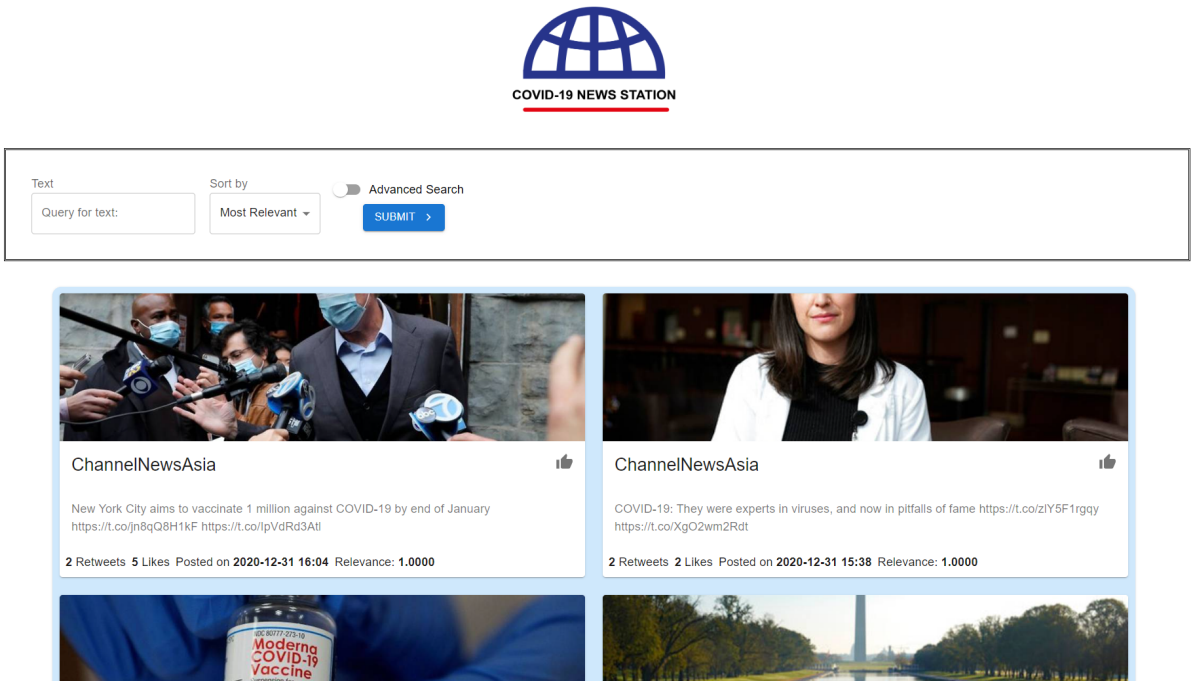


Figure 3: Web Interface

4.1 UI Design

Intuitive UI has been added to assist users in specifying their searches without the need to learn Solr query syntax. The interface provides a sort by field and an advanced search button which when triggered, reveals additional search parameters to allow users to further constrain their query.

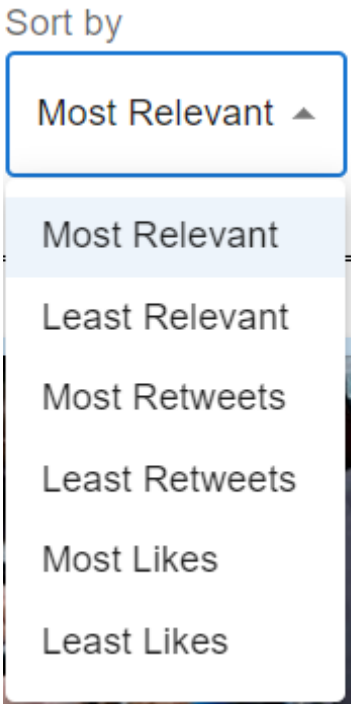


Figure 4: Sorting feature

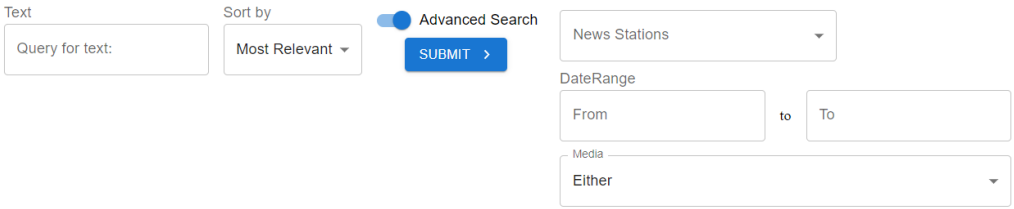


Figure 5: Advanced search feature

4.2 Query Results and Performance

Query 1: *Text:Pfizer*

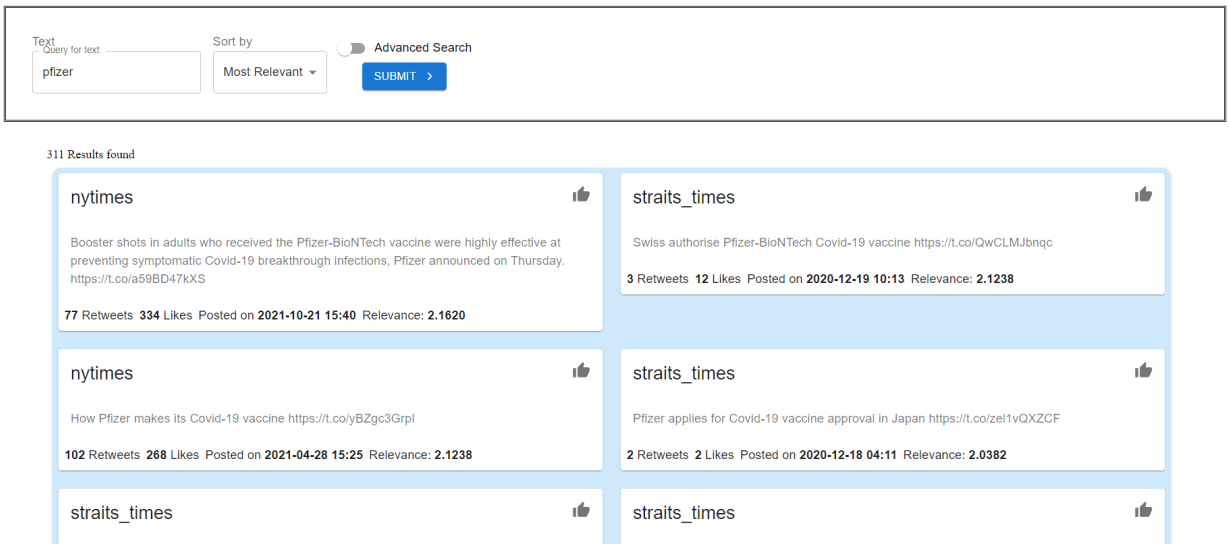


Figure 6: Query 1 results

Query 2: *Text:pfizer AND Username:reuters,channelnewsasia -Media:None*

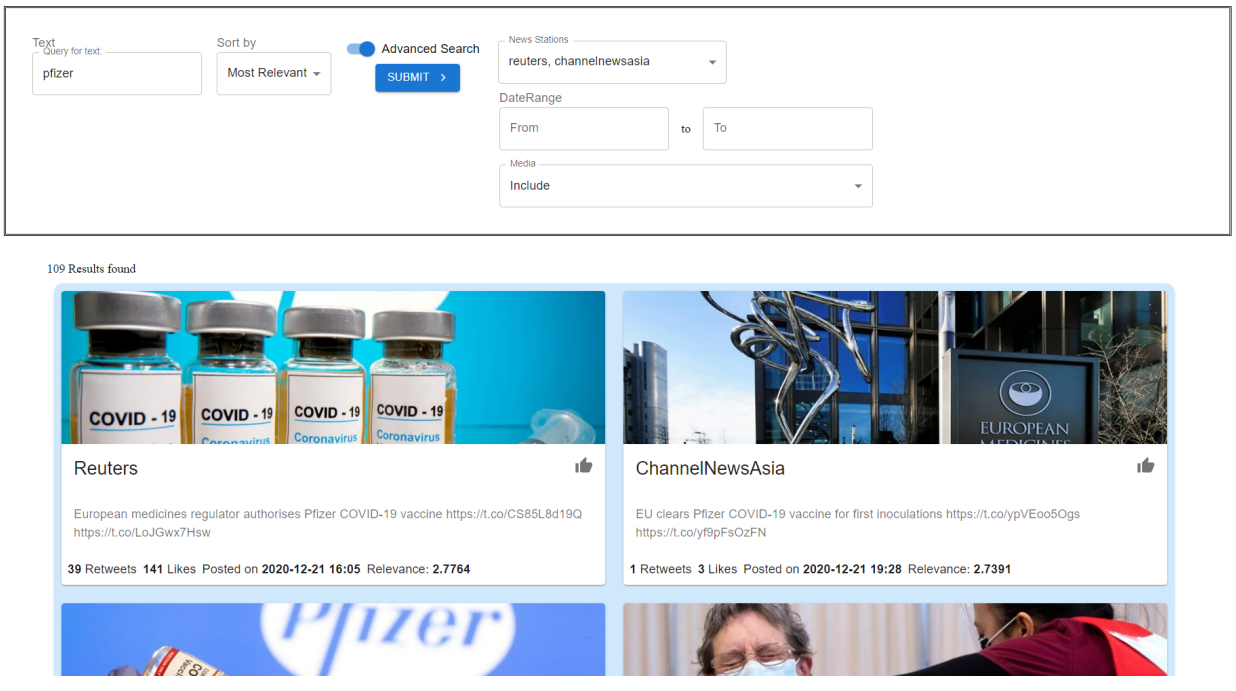


Figure 7: Query 2 results

Query 3: *Text:death toll*

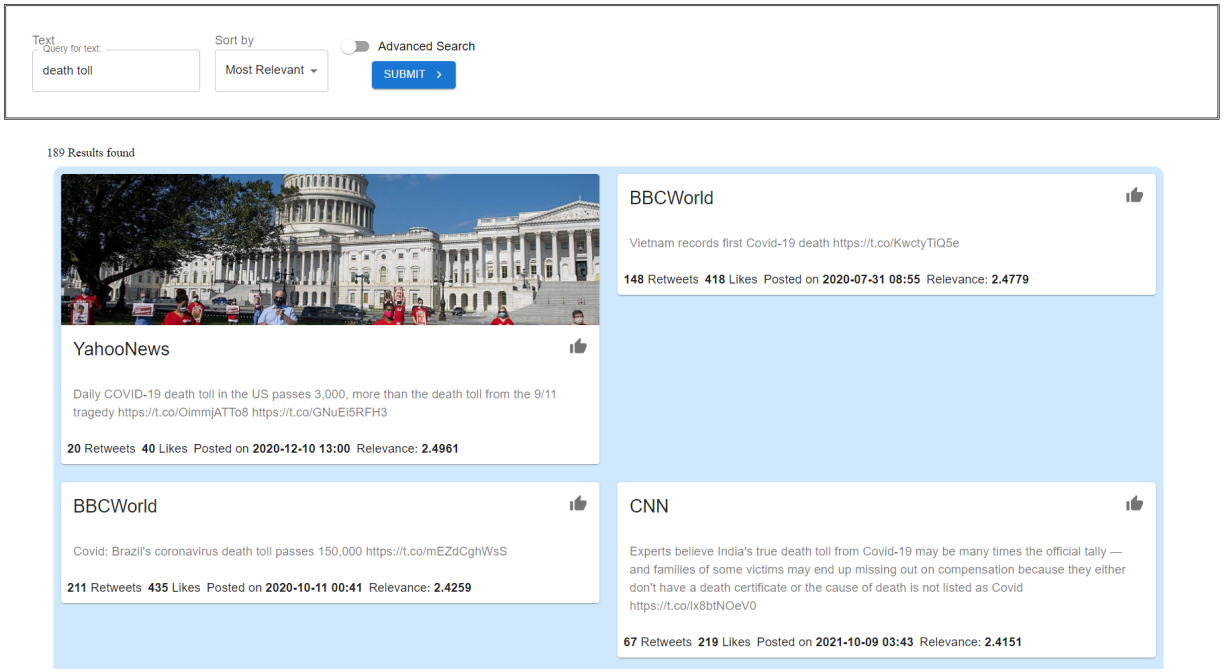


Figure 8: Query 3 results

Query 4: *Text:"death toll" AND Datetime:[2020-12-31T16:00:00.000Z TO 2021-12-30T16:00:00.000Z]*

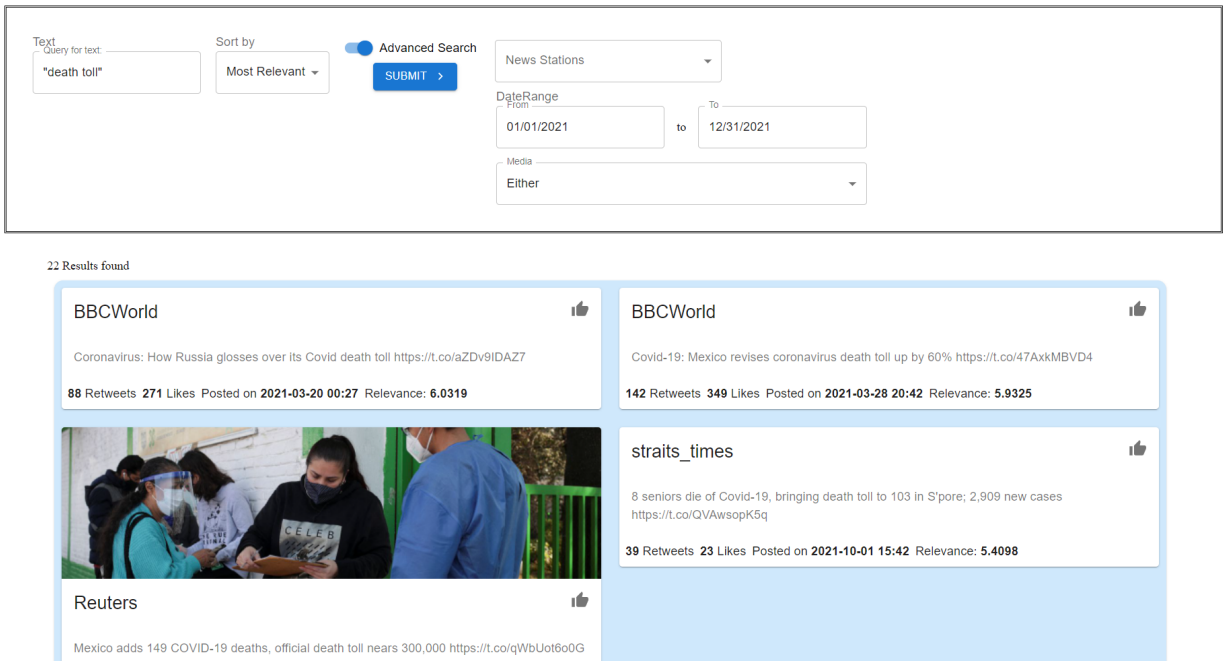
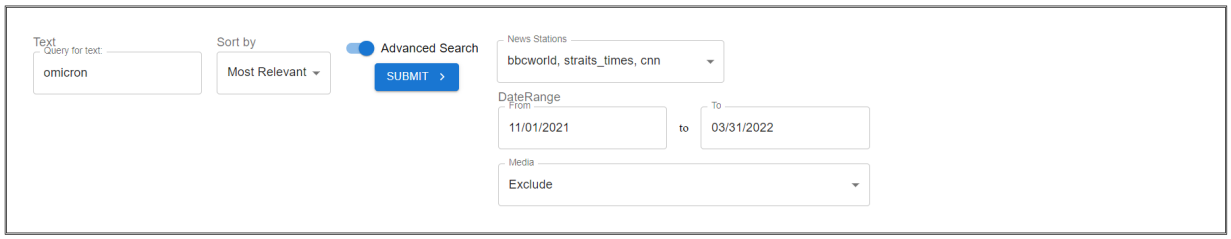


Figure 9: Query 4 results

Query 5: *Text:omicron AND Username:bbcworld,straits_times,cnn AND Datetime:[2021-10-31T16:00:00.000Z TO 2022-03-30T16:00:00.000Z] AND Media:None*



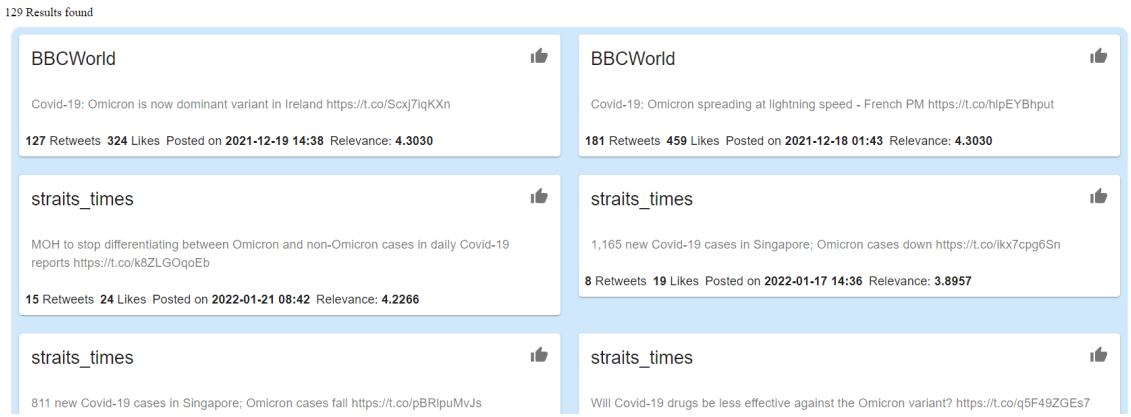


Figure 10: Query 5 results

Query	Time taken (ms)
Text:Pfizer	6
Text:pfizer AND Username:reuters,channelnewsasia -Media:None	7
Text:death toll	8
Text:"death toll" AND Datetime:[2020-12-31T16:00:00.000Z TO 2021-12-30T16:00:00.000Z]	12
Text:omicron AND Username:bbcworld,straits_times,cnn AND Datetime:[2021-10-31T16:00:00.000Z TO 2022-03-30T16:00:00.000Z] AND Media:None	34

Table 3: Sample query speed table

5 Innovations for enhancing the indexing and ranking

5.1 Interactive search

Interactive search allows our interface to refine the search results based on the users' relevance feedback. This was implemented by adding a relevance field to each document in the database. Upon the users' feedback, the "user relevance" field of the document is incremented by 1 each time the user gives the document a "thumbs up".

Then, we performed ranked retrieval using the following heuristic model:

$$netscore(q, d) = score(q, d) + log(relevance(d))$$

The search relevance score is retrieved from the sum of Solr's calculation for their *Practical Scoring Functio*n and the natural logarithm of the user relevance field. This is to balance and scale the weight of the user relevance feedback with the natural term-document matching scores.

Figure 11: Difference between before and after clicking the thumbs up button

As shown in Figure 11, the left and right tweet cards show the difference between the relevance score before and after the user clicks on the thumbs up button. Upon the change of relevance, the order of all the search results is swapped depending on the net score as well.

5.2 Multimodal search

Multimodal search allows our interface to retrieve images from the tweets (if there is any) as well as filter our search based on tweets that have or do not have any images.

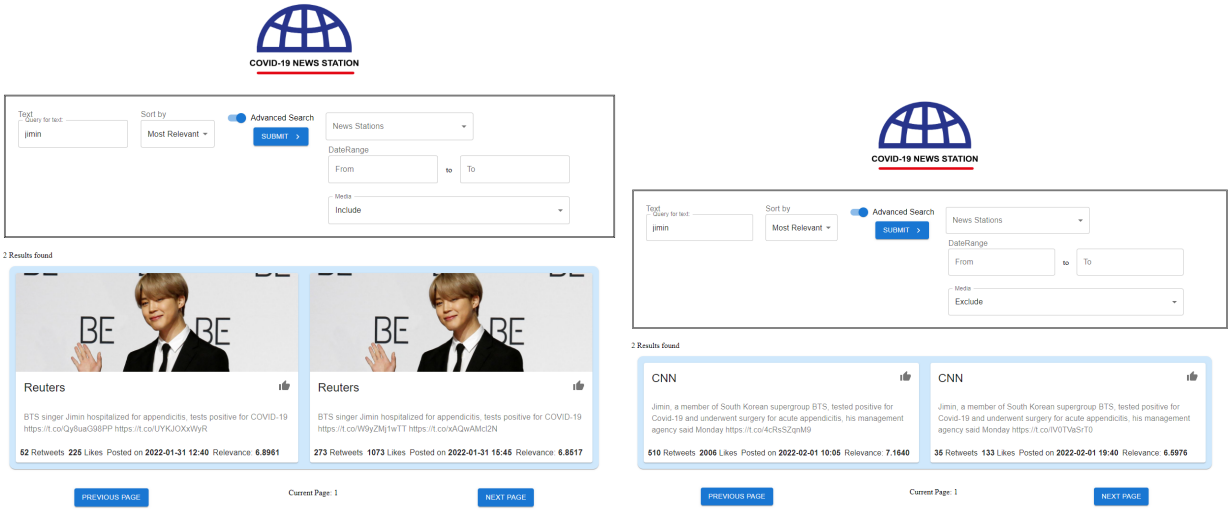


Figure 12: Search by include or exclude media

As seen from Figure 12, the same query for “jimin” has tweets that both contain and do not contain images and we are able to do an advanced search by filtering by this behavior.

5.3 Multifaceted search

Multifaceted search allows users to narrow down search results by applying multiple filters based on faceted classification of the items. In this instance, users can filter by News station and date range

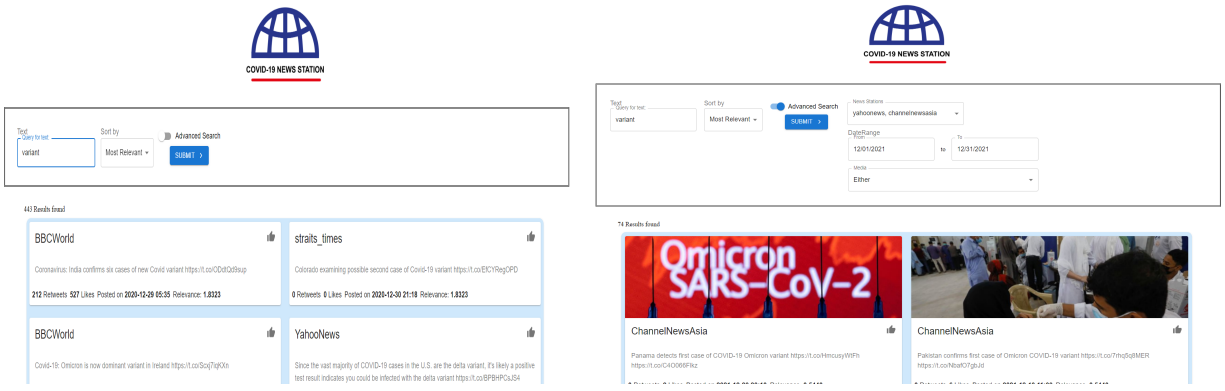


Figure 13: Search by filtering the news channel

As seen from Figure 13, the query for “variant” can be further filtered to only display tweets from yahoo news and channelnewsasia in the month of November 2022.

6 Classification

6.1 Motivate the choice of your classification approach in relation to the state of the art

Our classification task includes subjectivity detection as well as polarity detection. Subjectivity detection involves classifying a tweet as 0 if there is no sentiment feeling (neutral) and 1 if there is sentiment feeling (opinionated). Polarity detection involves classifying the opinionated data as 0 if it is negative and 1 if it is positive. The training dataset our group used for the classification is our manually labeled tweets from the crawled dataset.

Our choice of models to use for classification is Naive Bayes Classification, K-Nearest Neighbors Classification, Support Vector Machine Classification, and Decision Tree Classification.

Naive Bayes Classification

Naive Bayes Classification is easy and quick to predict the class of the test data set. It is mostly used in text classification as it has a higher success rate compared to other algorithms due to it having better results in multi-class problems and independence rule, which is applicable in our context of classifying text tweets from Twitter.

K-Nearest Neighbors Classification

K-Nearest Neighbors Classification makes use of the K-Nearest Neighbors algorithm, which stores all the cases and classifies the cases based on a similarity measure. It classifies based on the distance functions, such as Manhattan and Euclidean distance functions. Since the tweets crawled are all news related to COVID-19, there will be words used which are similar to each other and are used to show sentiment regarding COVID-19. Hence, this model is chosen to calculate the similarity score, and identify the K-Nearest Neighbors before classifying them as positive or negative.

Support Vector Machine Classification

Support Vector Machine Classification is applicable in our sentiment analysis as the sentiment for the news can be positive or negative depending on the content of the news, making it a linearly separable problem. Since SVM uses a hyperplane to differentiate the two classes, it will be suitable to use for our sentiment analysis.

Decision Tree Classification

Decision Tree Classification is a supervised machine learning technique that will split the data based on a certain condition. Since the sentiment can be either positive or negative, we can use the decision tree classification to split the data for our sentiment analysis.

6.2 Pre-processing of Text Corpus

Preprocessing had to be done on each of the tweets as text data from Twitter is “noisy”. Tweets from news outlets often include links to their actual news article, as well as occasional hashtags. We will have to preprocess and clean the tweets to transform them into a more standardized form for the training of the models, in order for the training of models to be conducted without being affected by noise in the data. Hence, preprocessing of the data has to be conducted before conducting classification and analysis.

6.2.1 Merging CSVs

When crawling the data, the data was crawled corresponding to the years 2020, 2021, and 2022. There were multiple CSVs obtained and hence we will have to merge the data together to obtain a single CSV file. Since the structure of the crawled data is the same for all 3 CSV files, we do not have to remove any columns. The merged CSV file consists of the same 7 columns before the merging and no data is lost in the process of merging.

6.2.2 Data Preprocessing

General Cleaning of Data

Since the tweets from news outlets contain URLs which link to their tweet or the original news article, we will have to conduct data cleaning to remove the URLs. Tweets also occasionally contain hashtags or contracted words and hence we will have to remove the hashtags and decontract words. Some tweets also contain images that will create a link within the text and hence they will have to be removed. All the punctuations are also removed and the letters are converted to lowercase to ensure that the text data will only consist of alphanumeric characters and that all the letters are lowercase.

Removal of irrelevant columns

The crawled data contains columns that are irrelevant to our classification and hence the irrelevant columns are dropped to conduct a cleaner analysis. The columns which we will be keeping are “TweetId”, “Text”, “Subjectivity”, and “Polarity”.

6.2.3 Stemming, Lemmatization and Removal of Stopwords

We conducted stemming and lemmatization on the “Text” in order to convert the words into their stem and lemma in order to analyze the usefulness of stemming and lemmatization in our classification. We used the Wordnet lemmatizer from NLTK to lemmatize the words. For words that are not found in Wordnet, the input word will be returned unchanged. Stopwords are also removed using NLTK in order to ignore all the common words in the English language.

6.2.4 Word Count

The word count after each data preprocessing process is as follows:

```
There are 49110 words in the corpus.
There are 49115 words in the corpus after stemming.
There are 33245 words in the corpus after stemming and removal of stopwords.
There are 49110 words in the corpus after lemmatization.
There are 32833 words in the corpus after lemmatization and removal of stopwords.
There are 32434 words in the corpus after removal of stopwords.
```

Figure 14: Word count after data preprocessing

6.2.5 Vectorization and N-grams

The text data is converted into vectors using Count Vectorization and TF-IDF Vectorization. Countvectorizer converts a given set of strings into a frequency representation. It understands the type of text by the frequency of words in it. TF-IDF (Term Frequency - Inverse Document Frequency) is a statistic that is based on the frequency of a word and provides a numerical representation of how important a word is for statistical analysis. To further improve the accuracy, we also use a combination of bigrams and unigrams.

6.3 Manual Labeling of Collected Data

The team has manually labeled a total of 2,729 rows of data out of 10,261 rows of collected data. The inter-annotator agreement between the 2 raters is 0.961.

```
rater_1 = labelled2raters_df['Rater_1'].to_numpy()
rater_2 = labelled2raters_df['Rater_2'].to_numpy()

from sklearn.metrics import cohen_kappa_score
kappa_boi = cohen_kappa_score(rater_1, rater_2)
print("The inter-annotator score between 2 raters is {}".format(kappa_boi))
```

```
The inter-annotator score between 2 raters is 0.9610651845332726
```

Figure 15: Inner-annotator score of manually labeled data

6.4 Evaluation Metrics

After preprocessing the data, our data is then split into 80% training data and 20% testing data. The training data is used to train the models and the testing data is then used to evaluate the model's performance. For each model, we experimented with the different preprocessing methods (stemming, lemmatization, stopwords removal) to determine the best preprocessing method for each model. We summarized all the preprocessing methods for each model below.

6.4.1 Naive Bayes Classification

Subjectivity Classification

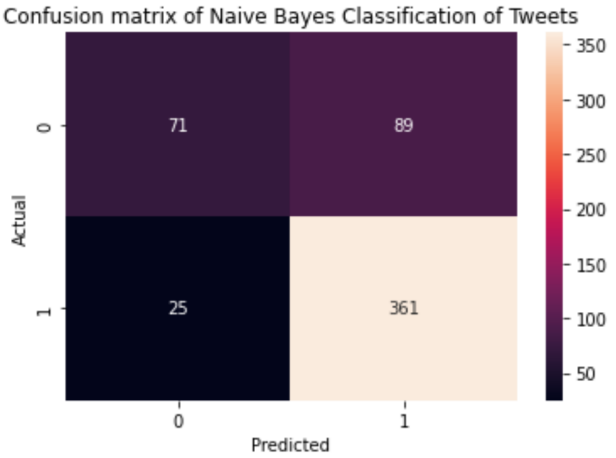


Figure 16: Confusion matrix of Subjectivity Classification for Naive Bayes Classification of Tweets

Optimization of Naive Bayes Classifier - Subjectivity

Vectorizer	Preprocessing	F1 Score	Precision Score	Recall Score	Difference of F1 score from base
Count	None	0.85	0.87	0.832	base
Count	Stemming	0.828	0.844	0.813	-0.022
Count	Lemmatization	0.834	0.864	0.806	-0.016
Count	Stopwords	0.83	0.852	0.808	-0.02
Count	Lemmatization + Stopwords	0.822	0.844	0.801	-0.028
Count	Stemming + Stopwords	0.811	0.829	0.793	-0.039
Tf-Idf	None	0.864	0.802	0.935	base
Tf-Idf	Stemming	0.844	0.772	0.93	-0.02
Tf-Idf	Lemmatization	0.848	0.783	0.925	-0.016
Tf-Idf	Lemmatization + Stopwords	0.842	0.788	0.904	-0.022
Tf-Idf	Stemming + Stopwords	0.843	0.78	0.917	-0.021

Table 4: Summary table of preprocessing functions (Subjectivity)

From Figure 4, we can deduce that the Tf-Idf vectorizer performs better than the count vectorizer. We can also observe that no preprocessing provides the best scores for the Naive Bayes Classification. Hence, we will be using the Tf-Idf vectorizer and no preprocessing for the remaining models on Subjectivity Classification - K-Nearest Neighbour, Support Vector Machine and Decision Tree.

Polarity Classification

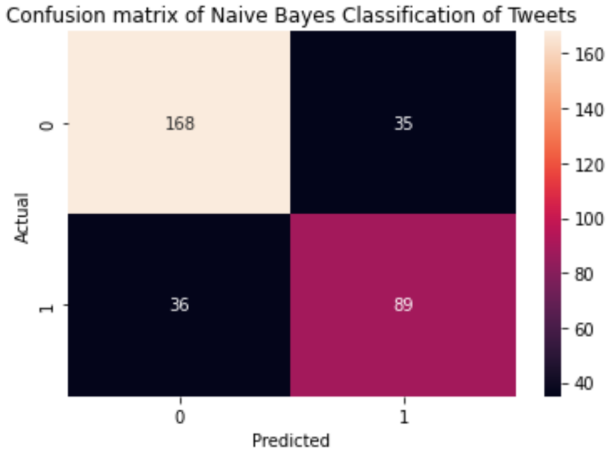


Figure 17: Confusion matrix of Polarity Classification for Naive Bayes Classification of Tweets

Optimization of Naive Bayes Classifier - Polarity

Vectorizer	Preprocessing	F1 Score	Precision Score	Recall Score	Difference of F1 score from base
Count	None	0.715	0.718	0.712	base
Count	Stemming	0.699	0.711	0.688	-0.016
Count	Lemmatization	0.697	0.714	0.68	-0.018
Count	Stopwords	0.667	0.661	0.672	-0.048
Count	Lemmatization + Stopwords	0.704	0.713	0.696	-0.011
Count	Stemming + Stopwords	0.686	0.709	0.664	-0.029
Tf-Idf	None	0.699	0.782	0.632	base
Tf-Idf	Stemming	0.636	0.75	0.552	-0.063
Tf-Idf	Lemmatization	0.667	0.763	0.592	-0.032
Tf-Idf	Lemmatization + Stopwords	0.627	0.739	0.544	-0.072
Tf-Idf	Stemming + Stopwords	0.625	0.707	0.56	-0.074

Table 5: Summary table of preprocessing functions (Polarity)

From Table 5, we can deduce that the count vectorizer performs better than the TF-IDF. We can also observe that no preprocessing provides the best scores for the Naive Bayes Classification. Hence, we will be using the count vectorizer and no preprocessing for the remaining models on Polarity Classification - K-Nearest Neighbour, Support Vector Machine and Decision Tree.

6.4.2 K-Nearest Neighbor Classification (KNN)

Subjectivity Classification

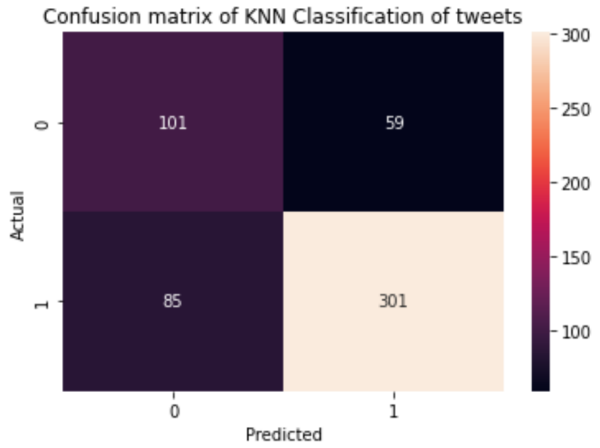


Figure 18: Confusion matrix of Subjectivity Classification for KNN

Polarity Classification

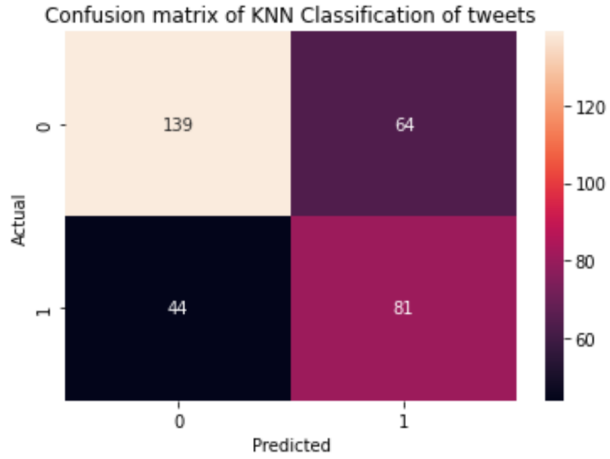


Figure 19: Confusion matrix of Polarity Classification for KNN

6.4.3 Support Vector Machine Classification (SVM)

Subjectivity Classification

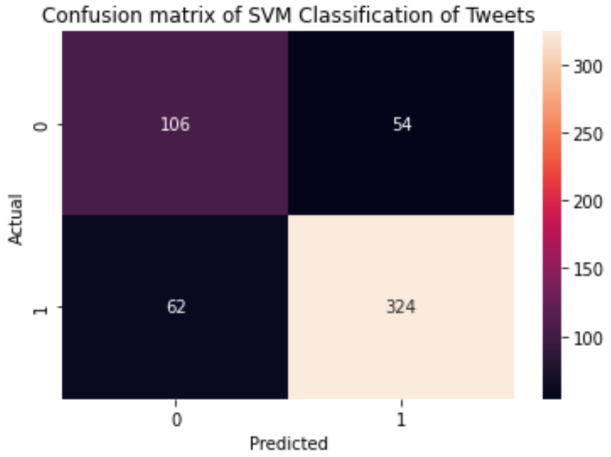


Figure 20: Confusion matrix of Subjectivity Classification for SVM

Polarity Classification

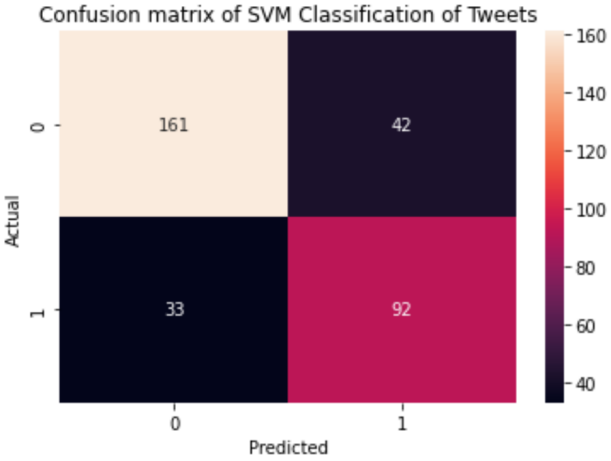


Figure 21: Confusion matrix of Polarity Classification for SVM

6.4.4 Decision Tree Classification

Subjectivity Classification

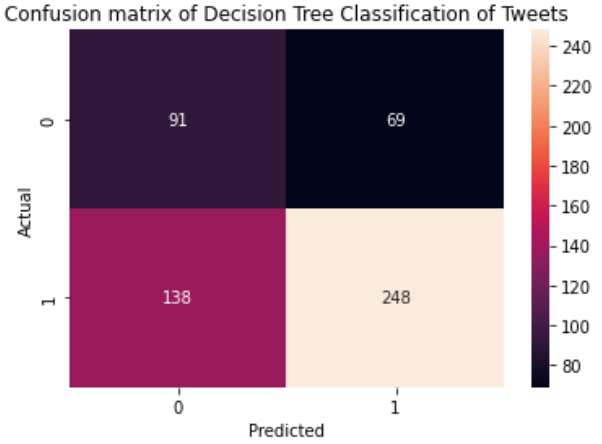


Figure 22: Confusion matrix of Subjectivity Classification for Decision Tree

Polarity Classification

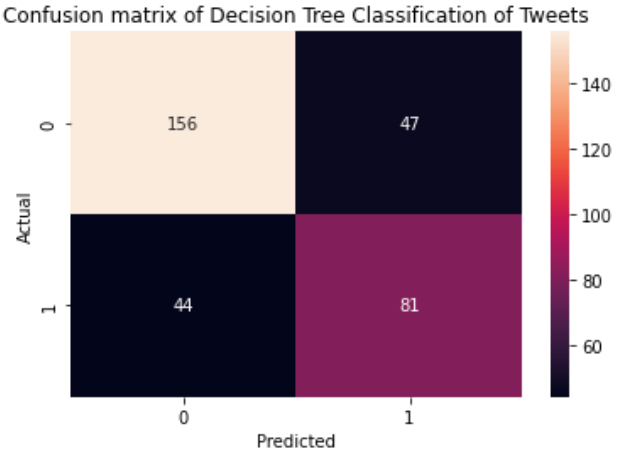


Figure 23: Confusion matrix of Polarity Classification for Decision Tree

6.4.5 Summary

Subjectivity Classification (TF-IDF)

Model	F1 Score	Precision Score	Recall Score	CV Folds
Naive Bayes Classification	0.829	0.709	0.997	5
K-Nearest Neighbor Classification	0.847	0.766	0.948	5
Support Vector Machine Classification	0.86	0.766	0.982	5
Decision Tree Classification	0.802	0.775	0.855	5

Table 6: Summary table for Subjectivity Classification

Above shows the summary table after optimization of the model. We can observe that the SVM model generates the highest F1 score among all the models, with an F1 score of 0.86. The KNN model generates the next highest F1 score of 0.847.

The steps taken to optimize our models includes:

1. Choosing the best preprocessing method for each individual classification model
2. Choosing the best hyperparameters using GridSearchCV
3. Performing 5-fold cross-validation

The steps taken to optimize our models will be discussed in the later part of our report under innovations for enhancing classification.

Polarity Classification (Count)

Model	F1 Score	Precision Score	Recall Score	CV Folds
Naive Bayes Classification	0.584	0.717	0.496	5
K-Nearest Neighbor Classification	0.596	0.623	0.584	5
Support Vector Machine Classification	0.655	0.725	0.600	5
Decision Tree Classification	0.566	0.592	0.608	5

Table 7: Summary table for Polarity Classification

Above shows the summary table after optimization of the model. We can observe that the SVM model generates the highest F1 score among all the models, with an F1 score of 0.655. The KNN model generates the next highest F1 score of 0.596.

The steps taken to optimize our models includes:

1. Choosing the best preprocessing method for each individual classification model
2. Choosing the best hyperparameters using GridSearchCV
3. Performing 5-fold cross-validation

The steps taken to optimize our models will be discussed in the later part of our report under innovations for enhancing classification.

6.5 Performance Metrics

6.5.1 Average number of words

The average number of words in a tweet is 17.10.

```
wordcount = df['Text'].apply(lambda x: len(x.split())).sum()

wordlength = wordcount / df.shape[0]
print ("The average number of words in a tweet is %.2f" % wordlength)
```

```
The average number of words in a tweet is 17.10
```

Figure 24: Average words in a tweet

6.5.2 Tweets classified per second

According to our summary of the models, we used the top 2 models which performed the best in terms of F1 score and calculated the number of tweets classified per second as well as the time taken to train the model.

Subjectivity Classification

Model	F1 Score	Tweets classified per second	Time taken to train the model
K-Nearest Neighbor Classification	0.847	5071.7455	0.0806 seconds
Naive Bayes Classification	0.829	4783.2685	0.0766 seconds

Table 8: Tweets classified per second (Subjectivity Classification)

From the table above, we can observe that KNN Classification, which has a higher F1 score, was able to classify more tweets per second, but it is slightly slower than the Naive Bayes Classification in terms of training the model. If the training were to be performed on a large dataset, the difference in time taken to train the model might be more significant hence Naive Bayes might be preferred in the case of a large dataset.

Polarity Classification

Model	F1 Score	Tweets classified per second	Time taken to train the model
K-Nearest Neighbor Classification	0.596	5095.9025	0.0848 seconds
Support Vector Machine Classification	0.584	5140.6427	0.0811 seconds

Table 9: Tweets classified per second (Polarity Classification)

From the table above, we can observe that KNN Classification classified fewer tweets per second as compared to the SVM Classification. However, compared to SVM Classification, the KNN Classification took less time to train the model, which will work better on a large dataset.

6.6 Visualizing Text Data

6.6.1 Word Cloud

A word cloud is a collection of words depicted in different sizes. The bigger and bolder the word appears, the more often it is mentioned within a given text and the more important it is. To visualize the classified data, our group decided to generate word clouds to show the classification of tweets.

Word Clouds based on Subjectivity

Figure 25: Opinionated word cloud

From Figure 25, we can see that most opinionated tweets include words like vaccination, speedier, pledges, and pharmaceutical, which are possibly associated with tweets that mention higher vaccination rates in countries or having a speedier roll out of vaccination schemes.

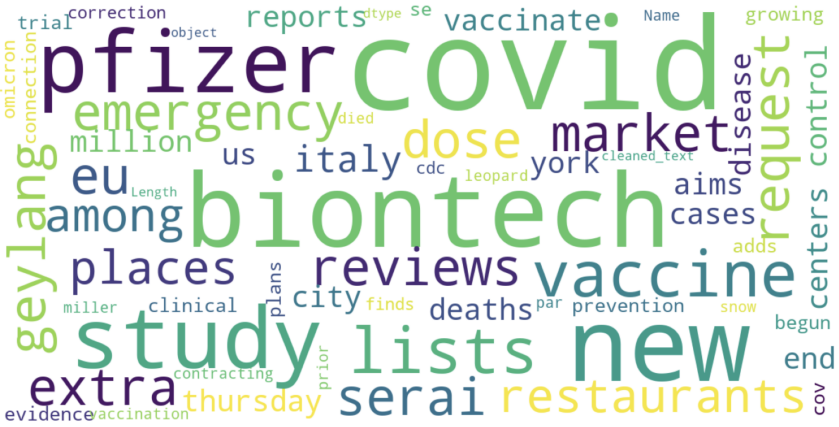


Figure 26: Neutral word cloud

From Figure 26, we can see that most neutral tweets include words like reviews and studies, which are possibly associated with reviews and studies done on people or vaccines, which have a neutral sentiment to them.

Word Clouds based on Polarity

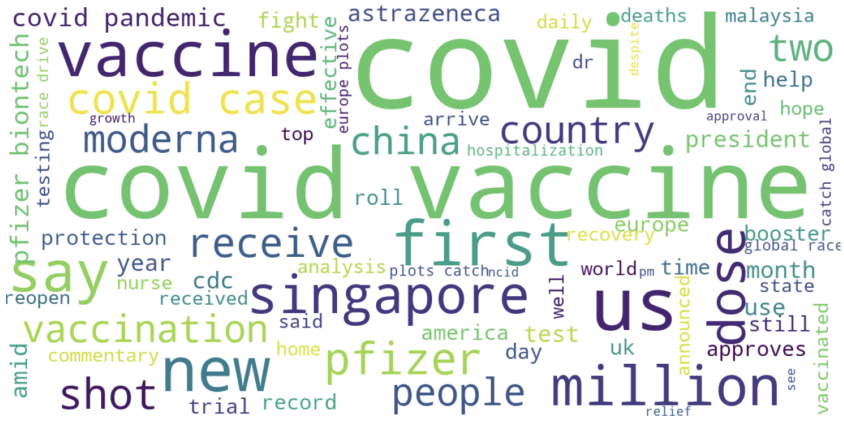


Figure 27: Positive word cloud

From Figure 27, we can see that most positive tweets include words like first, vaccine, new, which are possibly associated with new vaccines or the first batch of vaccines being delivered.

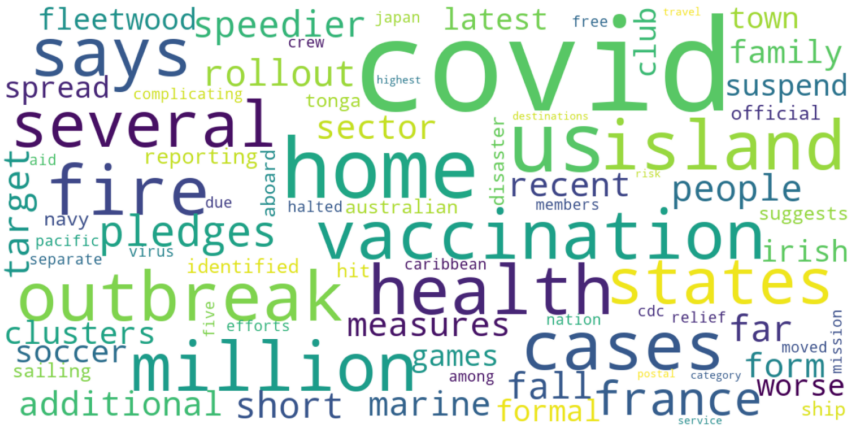


Figure 28: Negative word cloud

From Figure 28, we can see that most negative tweets include words like outbreak, clusters, million, which are possibly associated with outbreaks happening in regions, more clusters being formed, or over a million covid cases in the country.

6.6.2 Latent Dirichlet Allocation (LDA) Topic Modeling

Topic modeling provides methods to automatically organize and classify documents into different themes. LDA is a popular topic modeling technique that can extract topics from a given corpus. LDA uses Gibbs sampling to assign topics to documents.

The data was preprocessed into tokens and lemmatization and stopwords removal were conducted. The data was then converted into a bag-of-words corpus and placed into a dictionary.

Interpretation of the visualization

pyLDAvis is a library that can help users interpret the topics in a topic model that has been fit to a corpus of text data. It extracts information from a fitted LDA topic model to inform an interactive web-based visualization.

1. The blue bars represent the overall term frequency of the word.
2. The red bars represent the estimated term frequency within the selected topic.
3. The size of the circles represents the marginal topic distribution.
4. Distance between each topic represents the relationship between the topics. If two topics are closer to each other, they are semantically related.
5. When the value of lambda is set to 0, we can see words that are exclusive to the topic.
6. When we hover over the bars for each word, we can see which other topics the words are used in.

The figure below shows 25 topics on the intertopic distance map on the left, and the top 30 most salient terms on the right.

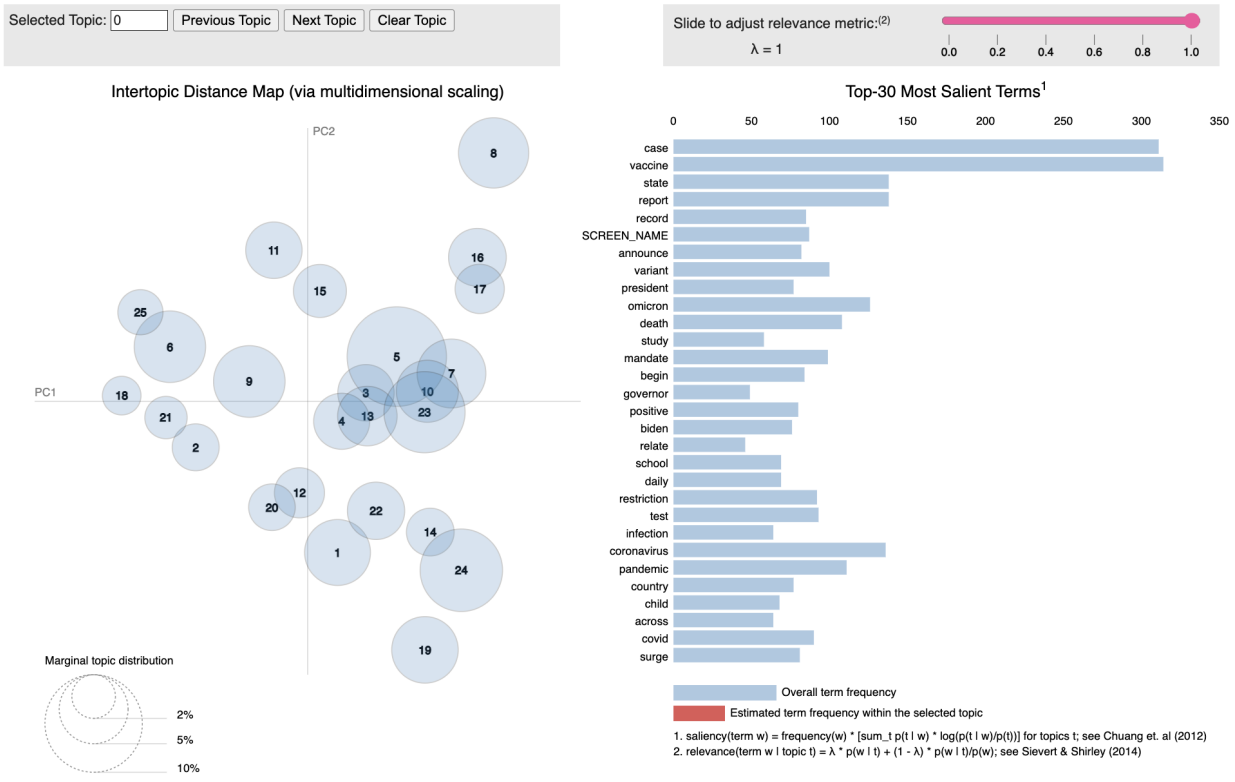


Figure 29: Visualization of intertopic distance map and salient terms (1)

From the news tweets, we could see many tweets which talked about the release of a new vaccine, or a new batch of vaccines being sent to a country. By hovering over the word vaccines, we can see that there are many topics that relate to the word ‘vaccine’.

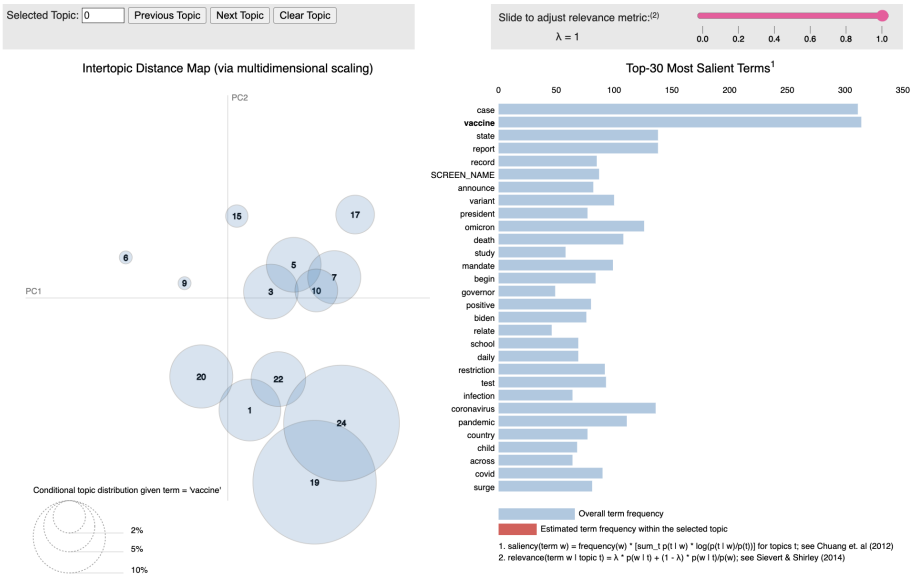


Figure 30: Visualization of intertopic distance map and salient terms (2)

From the intertopic distance map below, we can see that the topic which has the largest marginal distribution, topic 5, has the terms ‘covid-19’ and ‘case’, which is probably due to there being many news tweets that gave reports on the daily number of covid cases. We can see that most of the words have a negative sentiment to them such as ‘spike’, ‘rapid’, and ‘lockdown’ possibly due to the spike in the number of cases resulting in a lockdown.

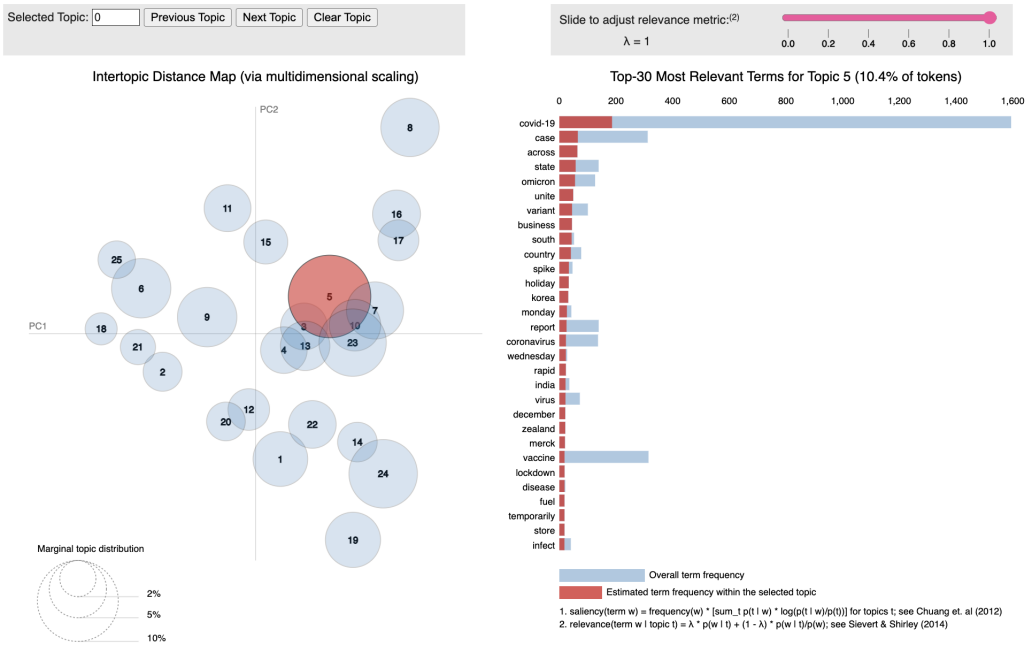


Figure 31: Visualization of intertopic distance map and salient terms (3)

7 Innovations for enhancing classification

7.1 Ensemble classification - Random Forest Classification (Subjectivity)

One major problem encountered when running the various models is the high percentage of wrong predictions and one way to solve this problem would be to use random forest classification which consists of a large number of individual decision trees that work as an ensemble. The main concept of the random forest classification would be the “wisdom of crowds” [1] where a large number of individual decision trees combined together is smarter than a single tree. This is proven by the scores below where random forest classification brings about a higher F1 Score.

Model	F1 Score	Precision Score	Recall Score	CV Folds
Naive Bayes Classification	0.829	0.709	0.997	5
K-Nearest Neighbor Classification	0.847	0.766	0.948	5
Support Vector Machine Classification	0.86	0.766	0.982	5
Decision Tree Classification	0.802	0.775	0.855	5
Random Forest Classification	0.867	0.778	0.984	5

Table 10: Ensemble Classification Summary for Subjectivity

7.2 GridSearchCV

To enhance the classification models, we made use of the GridSearchCV function which helps us determine the best hyperparameters for a model by passing in a range of the hyperparameters' values. The GridSearchCV function is conducted on all the models in order to find the best hyperparameters for each model to ensure that all models are enhanced.

K-Nearest Neighbors Classification

Upon testing various distance metrics such as Euclidean, Manhattan, and Cosine, we have found that Euclidean distance works the best for the model. A list of values ranging from 1 to 50 for the number of neighbors and leaf size was passed into the function to determine the best number of clusters and leaf size.

Subjectivity Classification:

```
================================================
Model: K-Nearest Neighbour Classification
Best parameters: {'leaf_size': 1, 'metric': 'euclidean', 'n_neighbors': 20, 'weights': 'distance'}
Preprocessing Function:  Text
================================================
```

Figure 31: GridSearchCV and KNN (Subjectivity Classification)

Polarity Classification:

```
================================================
Model: K-Nearest Neighbour Classification
Best parameters: {'leaf_size': 1, 'metric': 'euclidean', 'n_neighbors': 18, 'weights': 'distance'}
Preprocessing Function:  Text
================================================
```

Figure 32: GridSearchCV and KNN (Polarity Classification)

Support Vector Machine Classification

For SVM, we have used 4 hyperparameters - C, gamma, kernel type, and degree. C is the penalty parameter that represents the misclassification or error term. The misclassification or error term tells the SVM optimisation how much error is bearable. Gamma is the measurement of the plausible line of separation. Kernel is the parameter that selects the type of hyperplane used to separate the data. Degree is a parameter used for the polynomial kernel type and indicates the degree of the polynomial to be used.

Subjectivity Classification:

```
================================================
Model: Support Vector Machine Classification
Preprocessing Function:  Text
Best parameters: {'C': 10, 'degree': 0, 'gamma': 0.05, 'kernel': 'rbf'}
================================================
```

Figure 33: GridSearchCV and Support Vector Machine (Subjectivity Classification)

Polarity Classification:

```
==================================================
Model: Support Vector Machine Classification
Preprocessing Function:  Text
Best parameters: {'C': 1, 'degree': 1, 'gamma': 0.2, 'kernel': 'poly'}
==================================================
```

Figure 34: GridSearchCV and Support Vector Machine (Polarity Classification)

Decision Tree Classification

For Decision Trees, the quality of the split can be measured by gini for gini impurity or entropy for information gain. This can be influenced by the hyperparameter criterion in the DecisionTreeClassifier() function. Splitter is also tested which determines the split at each node and the maximum number of features to consider when looking for the best split.

Subjectivity Classification:

```
================================================
Model: Decision Tree Classification
Preprocessing Function:  Text
Best parameters: {'criterion': 'entropy', 'max_features': 'sqrt', 'splitter': 'random'}
================================================
```

Figure 35: GridSearchCV and Decision Tree (Subjectivity Classification)

Polarity Classification

```
================================================
Model: Decision Tree Classification
Preprocessing Function:  Text
Best parameters: {'criterion': 'gini', 'max_features': 'sqrt', 'splitter': 'best'}
================================================
```

Figure 36: GridSearchCV and Decision Tree (Subjectivity Classification)

Random Forest Classification

Random Forest classification is a model consisting of multiple Decision Trees. We have decided to modify the maximum number of features and the number of decision trees, represented by n_estimators.

Subjectivity Classification:

```
================================================
Model: Enhanced Classification
Preprocessing Function:  Text
Best parameters: {'bootstrap': False, 'max_features': 'auto', 'n_estimators': 100}
================================================
```

Figure 37: GridSearchCV and Random Forest (Subjectivity Classification)

Polarity Classification:

```
==================================================
Model: Enhanced Classification
Preprocessing Function:  Text
Best parameters: {'bootstrap': True, 'max_features': 'log2', 'n_estimators': 100}
==================================================
```

Figure 38: GridSearchCV and Random Forest (Polarity Classification)

7.3 k-Fold Cross-Validation (Subjectivity)

To prevent possible overfitting issues, our group performed k-Fold Cross Validation together with GridSearchCV. k-Fold Cross-Validation is used to evaluate the models by estimating how the model is expected to perform in general when used to make predictions on testing data. The steps involve shuffling the dataset randomly and splitting the dataset into k groups. For each group, it will be used as testing data and the remaining groups will be used as training data. The models will be fitted on the training set and evaluated on the testing set. Our group has decided on a 5-fold cross-validation since it has been shown empirically to yield test error rate estimates that do not suffer from excessively high bias or very high variance.

By conducting k-Fold Cross-Validation, we will be able to obtain results that are less biased and more consistent as the training and testing datasets change. We will then obtain the mean value for the F1 Score, Precision Score, and Recall Score to get a less biased result. The table below compares the results with and without 5-Fold Cross-Validation.

Model	**F1 Score**		**Precision Score**		**Recall Score**	
	Without	With	WIthout	With	WIthout	WIth
Naive Bayes Classification	0.864	0.829	0.802	0.709	0.935	0.997
K-Nearest Neighbor Classification	0.783	0.834	0.827	0.786	0.744	0.888
Support Vector Machine Classification	0.828	0.828	0.707	0.707	1	1
Decision Tree Classification	0.759	0.798	0.833	0.775	0.697	0.832
Random Forest Classification	0.86	0.865	0.876	0.774	0.845	0.977

Table 11: k-Fold Cross Validation Summary for Subjectivity Classification

7.4 Error Analysis

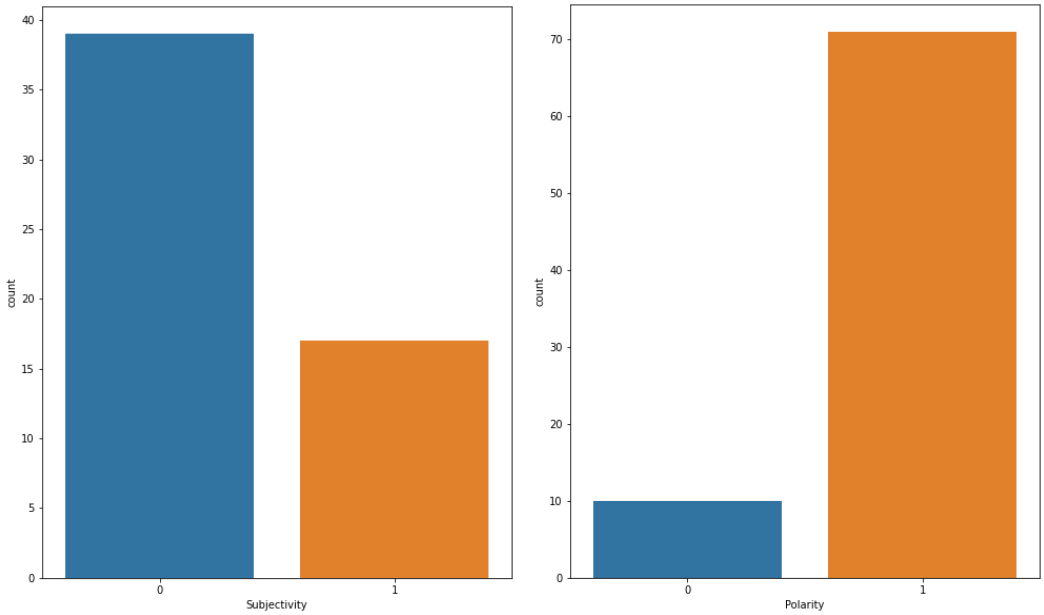


Figure 39: Count of subjectivity and polarity classification tweets

Out of the test data of around 819 rows, about 100 of them are wrongly classified for both subjectivity and polarity. The reason why subjectivity has a higher proportion of wrongly classified data as compared to polarity could be due to subjectivity having more data. This could be because the news channel aims to be as neutral as possible, thus leading to less data for polarity (positive and negative opinion).

Upon closer inspection of the wrongly_classified *.csv of both subjectivity and polarity, we realized that some of the tweets contain unknown words which could be a result of the original tweet containing emojis or non-english words such as the one below. This may have an effect on the classification model.

Figure 40: Example of tweet with emojis

8 Submission Links

A YouTube link to a video presentation of up to 5 minutes

https://www.youtube.com/watch?v=J01LJOMD2qU

A Dropbox (or Google Drive) link to a compressed (e.g., zip) file with crawled text data, queries and their results, manual classifications, automatic classification results, and any other data for Questions 3 and 5

Readme document is provided inside the folder

https://drive.google.com/file/d/1OZSjEFJC0oibXORAabTgi6yzEHoTeEVn/view?usp=sharing

A Dropbox (or Google Drive) link to a compressed (e.g., zip) file with all your source codes and libraries, with a readme file that explains how to compile and run the source codes

Readme document is provided inside the folder

https://drive.google.com/file/d/1FzeMd-q7Wq568Z0t7KPX3_c5ZveU1B5B/view?usp=sharing