



**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE

CZ4034/SC4021 Information Retrieval

Course Assignment Report

Group 9

Team Members

Name	Matriculation Number
ERNEST SIM ZI-EN	U2023460K
GUO CHENRUI	U2120138F
LIYA D/O FELIX SASI KUMAR	U2021437B
NG ZHENG JIE	U2021591H
WONG WEI KAI	U2121476K
YAP WEE JUN	U2023341H

Table of Contents

1. Introduction.....	4
1.1 Topic Selection.....	4
1.2 Problem Statement.....	4
2. Data Crawling.....	5
3. Indexing and Querying.....	8
3.1 Data Index and Query with Apache Solr.....	8
3.1.1 Data Fields.....	8
3.1.2 SOLR Configurations.....	9
3.1.2.1 Stemming Feature.....	9
3.1.2.2 Spell Check Feature.....	11
3.1.3 Data Indexing.....	12
3.1.4 Querying.....	13
3.2 User Interface (UI).....	14
3.2.1 Tools.....	14
3.2.2 Design.....	14
3.2.2.1 Sidebar Filters.....	16
3.2.2.2 Tweets Tab.....	19
3.2.2.3 Graphs Tab.....	20
3.3 Search techniques.....	21
3.3.1 Stemming.....	21
3.3.2 Spellcheck.....	22
3.4 Test Query Result and Performance.....	24
4. Classification.....	32
4.1 Subtasks.....	32
4.2 Covid-19 vaccine tweets dataset.....	33
4.3 Preprocessing of Covid-19 vaccine tweets.....	34
4.4 Machine Learning Models.....	35
4.4.1 Logistic Regression.....	35
4.4.2 Support Vector Machine (SVM).....	36
4.4.3 Naive Bayes.....	36
4.4.4 Random Forest.....	37
4.5 Feature Extraction.....	38
4.5.1 CountVectorizer.....	38
4.5.2 TF-IDF Vectorizer.....	38
4.5.3 N-grams.....	38

4.6 Results.....	38
4.6.1 Subjectivity.....	39
4.6.2 Polarity.....	40
5. Enhancements.....	42
5.1 Ensemble Model.....	42
5.1.1 Subjectivity.....	42
5.1.2 Polarity.....	43
5.2 Grid Search.....	44
5.2.1 Subjectivity.....	45
5.2.2 Polarity.....	46
5.3 Named Entity Recognition.....	47
5.4 Word Sense Disambiguation.....	47
5.5 State of the Art.....	48
5.6 Discussion.....	51
5.6.1 Time Tests.....	51
5.6.2 Accuracy Test.....	52
6. Submission Links.....	54
7. References.....	55

1. Introduction

For our CZ4034/SC4021 Information Retrieval assignment, our group has been tasked to build an opinion search engine for a given topic of our choice. The system should allow users to be able to find relevant opinions on the given topic and perform sentiment analysis on the results. The chosen topic should have enough data and the opinions regarding the topic should be relatively balanced.

1.1 Topic Selection

Our group has chosen the Covid-19 vaccine for our assignment topic, recognising its significance amidst the global pandemic. The Covid-19 vaccine has become the focal point of discussion, with far-reaching impacts on public health, economies, and societal norms. We proceeded with the development of a search engine for this topic, as there was an abundance of available data, which encompasses a spectrum of sentiments regarding Covid-19 vaccines. The abundance of information highlights the importance of our project. Our group aims to provide users with a comprehensive tool to display public perception of vaccines.

1.2 Problem Statement

There are many social media platforms that are used in today's world, allowing users to express their opinions and share facts online that can be accessed by anyone across the world. Usage of social media allowed people from across the globe to be connected and this is especially true when Covid-19 had thrown the world into a lockdown. As the world scrambled to tackle this global pandemic, pharmaceutical companies took on the challenge of developing vaccines in the shortest time possible to help the world return to normalcy. Some vaccines that were used include Moderna, Sinovac and Pfizer. However, due to misinformation, personal experiences and cultural norms, users have taken to Twitter to express their views of Covid vaccines, both positively and negatively.

The objective of our team's project is to be able to build a retrieval system that is able to perform sentiment analysis on tweets about the covid vaccine that has been crawled from Twitter. The

data will be displayed in such a way that it **allows the user to analyze the different perspectives of the general Twitter population on the Covid vaccine.**

2. Data Crawling

For data crawling, the Tweet Harvest tool, a common-line utility that leverages Playwright to scrape tweets from Twitter search results was used. The tool allows for specifying keywords and a date range for focused data collection. The keywords we used to includes:

- *'covid vaccine/vaccination'*: for general information of COVID-19 sentiments
- *'pfizer vaccine/vaccination'*, *'sinovac vaccine/vaccination'* and *'moderna vaccine/vaccination'*: to explore sentiments associated with these three vaccination brands

We focused on collecting data from the year 2021 as this was a pivotal year in the global fight against the Covid-19 pandemic. It marked a time when COVID-19 vaccines were widely distributed and administered around the world. Therefore, it is an important period to analyze public sentiment and discussions around vaccines.

Figures 1 and 2 below shows an example of a query performed using the Tweet Harvest tool. It scrapes tweets based on the specified keywords "Moderna vaccine" and "Moderna vaccination" within a date range from January 1, 2021, to December 31, 2021. The -o option specifies the output filename, the -s option sets the search keyword, the -l option limits the number of tweets to retrieve for each keyword, and the --token option provides the Twitter authentication token needed to access the data. This query is executed using the npx command to run the specific version of Tweet Harvest (v2.2.8).

```
keyword = ['Moderna vaccine', 'Moderna vaccination']  
start_date = datetime(2021, 1, 1)  
end_date = datetime(2021, 12, 31)
```

Figure 1: Keywords and Date Range for Data Crawling

```
!npx --y tweet-harvest@2.2.8 -o "{data_filename}" -s "{search_keyword}" -l {num_tweets_per_tag} --token "{token}"
```

Figure 2: Query Performed Using the Tweet Harvest Tool

The Tweet Harvest tool required a Tweet authentication token for data access. The scraped data was saved as CSV files, for each of the four keywords used. Each CSV file contains rows with the following data fields:

- `created_at`: The date and time the tweet was posted
- `favourite_count`: The number of likes the tweet received
- `full_text`: The complete text of the tweet
- `lang`: The language the tweet is in
- `location`: The user's location
- `quote_count`: The number of times the tweet was quoted
- `reply_count`: The number of replies to the tweet
- `retweet_count`: The number of retweets
- `tweet_url`: The URL link to the tweet
- `username`: The Twitter handle of the user who posted the tweet

The collected data can be used for a variety of applications. For instance, by examining the content of *full_text* and associated metadata such as *favorite_count*, *retweet_count*, and *reply_count*, we can perform sentiment analysis on public opinions about COVID-19 vaccines and specific brands like Pfizer, Sinovac, and Moderna. Furthermore, by evaluating metrics such as *favorite_count*, *retweet_count*, and *quote_count*, we can gain insights into user engagement with specific topics or posts.

Before data preprocessing and duplicate removal, the numbers of tweets obtained for each keyword are as follows:

Table 1: Keywords Used for Crawling and Number of Tweets Retrieved per Keyword

Keywords	Number of Tweets Crawled
covid vaccine, covid vaccination	13,922
pfizer vaccine, pfizer vaccination	11,204
sinovac vaccine, sinovac vaccination	13,079
moderna vaccine, moderna vaccination	13,110

To calculate the number of unique tokens across all the four CSV files obtained through data crawling, we used the `PorterStemmer` from the `nltk.stem` library. The `PorterStemmer` is an algorithm that reduces words to their base or root form (called "stemming"), helping to group similar words together by removing inflectional endings and other derivational forms. By stemming the words, we ensure that similar words are counted as one unique token. Across all the data, we found a total of 93,329 unique tokens.

3. Indexing and Querying

3.1 Data Index and Query with Apache Solr

For our project, we have decided to use Apache Solr to index the crawled tweets. Solr is a widely used and powerful open-source search platform that is built upon Apache Lucene, which is a high-performance full-featured text search engine. Solr provides full-text search, indexing, tokenization, high scalability and spell checking. In this section, we will detail how Solr is implemented to help us achieve the project's objectives.

3.1.1 Data Fields

From the tweets that have been crawled, the selected fields and their content which are added are as follows:

Table 2: Selected Fields and Their Content Added to Solr from Crawled Tweets

Field	Description	Example
Date	Date which the tweet was made	2021-06-04T22:45:37Z
Tweet	Tweet's content	"I received my Moderna vaccine a few weeks ago and today my friend found me in the kitchen cooking chicken noodle soup! Dear gawd what is happening to me???? Am I a zombie now?"
username	Twitter handle of the person who posted	Tomme1
Sentiment	Tone of the tweet	Neutral
id	Unique identifier of the tweet	ada8a6ef-76d8-4e63-b934-e8850440bf7c
Quote_Count	Number of times the tweet has been quoted	0
Reply_Count	Number of replies to the tweet	0

Retweet_Count	Number of retweets	0
Favourite_Count	Number of likes the tweet received	1
Tweet_URL	URL link to tweet	https://twitter.com/Tomme1/status/1400946871779069953

3.1.2 SOLR Configurations

In our project, we have implemented two key Solr configurations to enhance user search queries. First, the stemming feature enables Solr to stem user queries, meaning it reduces words to their root form. This helps display all relevant tweets by matching variations of the same root word. Second, the spell check function provides alternative suggestions for users' misspelled queries. This feature enhances user experience by suggesting correct spellings or closely related terms, allowing users to refine their search queries for more accurate results.

3.1.2.1 Stemming Feature

To implement the stemming function, we used the 'SnowballPorterFilterFactory' from Apache Solr, which was added into the 'managed-schema' file as shown in the code snippet below (Figure 3). This factory applies the Snowball stemming algorithm to user queries and indexed documents. The algorithm reduces words to their base or root form, known as stemming, which helps improve search results by allowing variations of the same word to match. By configuring the filter in 'managed-schema', we enable Solr to apply stemming to the specified fields during both indexing and querying, thus broadening the scope of search results.

```

<fieldType name="text_general" class="solr.TextField" positionIncrementGap="100" multiValued="true">
  <analyzer type="index">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" words="stopwords.txt" ignoreCase="true"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.SnowballPorterFilterFactory"/>
  </analyzer>
  <analyzer type="query">
    <tokenizer class="solr.StandardTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" words="stopwords.txt" ignoreCase="true"/>
    <filter class="solr.SynonymGraphFilterFactory" expand="true" ignoreCase="true" synonyms="synonyms"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.SnowballPorterFilterFactory"/>
  </analyzer>
</fieldType>

```

Figure 3: Implementation of Stemming Function in 'managed-schema' File

The Snowball stemming algorithm can be observed in action through Solr's Analysis feature. When a query such as 'benefiting' is input, the feature shows how Solr processes the query text. From Figure 4 below, we can see that the algorithm retrieves the original text 'benefiting' as well as the stemmed version 'benefit'. This demonstrates how Solr reduces the word to its base form using the Snowball stemming algorithm, improving the ability to match and retrieve relevant documents containing different variations of the same root word.

LCF	text	benefiting
	raw_bytes	[62 65 6e 65 66 69 74 69 6e 67]
	start	0
	end	10
	positionLength	1
	type	<ALPHANUM>
	termFrequency	1
	position	1
SF	text	benefit
	raw_bytes	[62 65 6e 65 66 69 74]
	start	0
	end	10
	positionLength	1
	type	<ALPHANUM>
	termFrequency	1
	position	1
	keyword	false

Figure 4: Demonstration of Stemming Algorithm in Solr's Analysis Feature

3.1.2.2 Spell Check Feature

For the spell checking feature, we utilize the spell check function available in Solr's configuration XML file. We configure this function to focus on the 'Tweet' field, which contains various tweet data, as seen in the code snippet below (Figure 5). When a user query yields fewer than 15 tweets, the spell check function suggests alternative queries based on the corpus and provides the frequency of the suggested words in the corpus. This feature helps users refine their search queries and find more relevant results, thus enhancing their overall experience.

```
<!-- a spellchecker built from a field of the main index -->
<lst name="spellchecker">
  <str name="name">default</str>
  <str name="field">Tweet</str>
  <str name="classname">solr.DirectSolrSpellChecker</str>
```

Figure 5: Configuration of Spell Check Function in Solr's XML File

For instance, if a user misspells the word 'Pfizer' while querying, Solr can provide suggestions of words from our corpus, as demonstrated in Figure 6 below, along with their frequencies. For our project, we will utilize the top two suggested words with the highest frequency. This approach allows users to receive suggestions for their queries, which helps them refine their searches and achieve better results.

```
"suggestion": [{
  "word": "phizer",
  "freq": 88,
  {
    "word": "pfiser",
    "freq": 6,
    {
      "word": "physer",
      "freq": 1,
      {
        "word": "pfizer",
        "freq": 18233,
        {
          "word": "pziifer",
          "freq": 22}}}}],
```

Figure 6: Solr's Suggestions and Frequencies for the Misspelled Query 'Pfizer'

3.1.3 Data Indexing

The data can be indexed after the appropriate pre-processing and configuration steps have been completed. This process organizes the tweets for efficient searching and retrieval.

For example, an indexed tweet might appear as follows:

```
"response":{"numFound":50725,"start":0,"numFoundExact":true,"docs": [
  {
    "Date": ["2021-06-04T22:45:37Z"],
    "Tweet": ["I received my Moderna vaccine a few weeks ago and today my friend four"],
    "username": ["Tomme1"],
    "Sentiment": ["Neutral"],
    "id": ["ada8a6ef-76d8-4e63-b934-e8850440bf7c"],
    "Quote_Count": [0],
    "Reply_Count": [0],
    "Retweet_Count": [0],
    "Favourite_Count": [1],
    "Tweet_URL": ["https://twitter.com/Tomme1/status/1400946871779069953"],
    "_version_": 1796057022344462336},
  {
```

Figure 7: Example of an Indexed Tweet

The results are returned in JSON format, which is a widely-used data interchange format. This makes it easy to integrate with other systems and parse the data for further analysis or visualization.

3.1.4 Querying

Users can perform a search query to retrieve tweets that have been indexed in Solr. The query can target specific fields, such as the 'Tweet' field, allowing users to find content based on specific terms within the tweet text. Additionally, the system offers enhanced functionality, including sorting options. Users can sort the results based on different columns such as date, retweet count, or sentiment. These features give users flexibility and control over how they view and interact with the retrieved tweets.

For instance, a search query for the term 'sinovac' sorted by date descending would show the following results:

```
http://localhost:8983/solr/covid-vaccine/select?indent=true&q.op=OR&q=Tweet%3Asinovac&sort=Date%20desc

{
  "responseHeader": {
    "status": 0,
    "QTime": 1,
    "params": {
      "q": "Tweet:sinovac",
      "indent": "true",
      "q.op": "OR",
      "sort": "Date desc",
      "_": "1712928951923"
    }
  },
  "response": {
    "numFound": 13197,
    "start": 0,
    "numFoundExact": true,
    "docs": [
      {
        "Date": ["2021-12-30T23:33:30Z"],
        "Tweet": ["For those given sinovac 2 doses 2 additional booster doses of mRNA vaccine may be needed"],
        "username": ["gdyagra"],
        "Sentiment": ["Neutral"],
        "id": ["e299553b-12f2-4e0b-884f-883cbfb52b54"],
        "Quote_Count": [1],
        "Reply_Count": [3],
        "Retweet_Count": [18],
        "Favourite_Count": [44],
        "Tweet_URL": ["https://twitter.com/gdyagra/status/1476697985333743617"],
        "_version_": 1796057036328271873
      }
    ]
  }
}
```

Figure 8: Search Query for the Term 'Sinovac' Sorted by Date Descending

3.2 User Interface (UI)

We designed the user interface (UI) with Streamlit to offer an enhanced experience for users seeking to retrieve and access information efficiently. The UI provides a straightforward and user-friendly way to interact with the data, utilizing different filter functions to customize the search results. This user interface queries Solr, enabling users to perform specific searches and sort the results according to their needs. By integrating line graphs and pie charts, the application not only displays query outcomes but also presents them visually for easier analysis and understanding.

3.2.1 Tools

With its intuitive Python-based framework, Streamlit simplifies the process of creating interactive and data-driven web applications. Streamlit facilitates rapid prototyping and iteration and allows code to be written in a linear fashion, with changes reflected in real-time on the webpage. This enables seamless experimentation and refinement of application features.

3.2.2 Design

The webpage comprises four primary sections designed to enhance the user experience, as seen in Figure 9:

1. Search Bar: Located on the main page, the search bar allows users to perform queries efficiently.
2. Sidebar: The sidebar offers a variety of filter functions that allow users to refine and tailor their querying process
3. Tabs: Users can toggle between two tabs—'Tweets' and 'Graphs'—to explore different aspects of the data in the main page:
 - a. Tweets Tab: Displays the tweets retrieved based on user queries on the main page, as seen in Figure 13.
 - b. Graphs Tab: Presents a line graph and pie chart for users to better visualize the data, offering insights into trends and sentiment distribution, as seen in Figure 14.

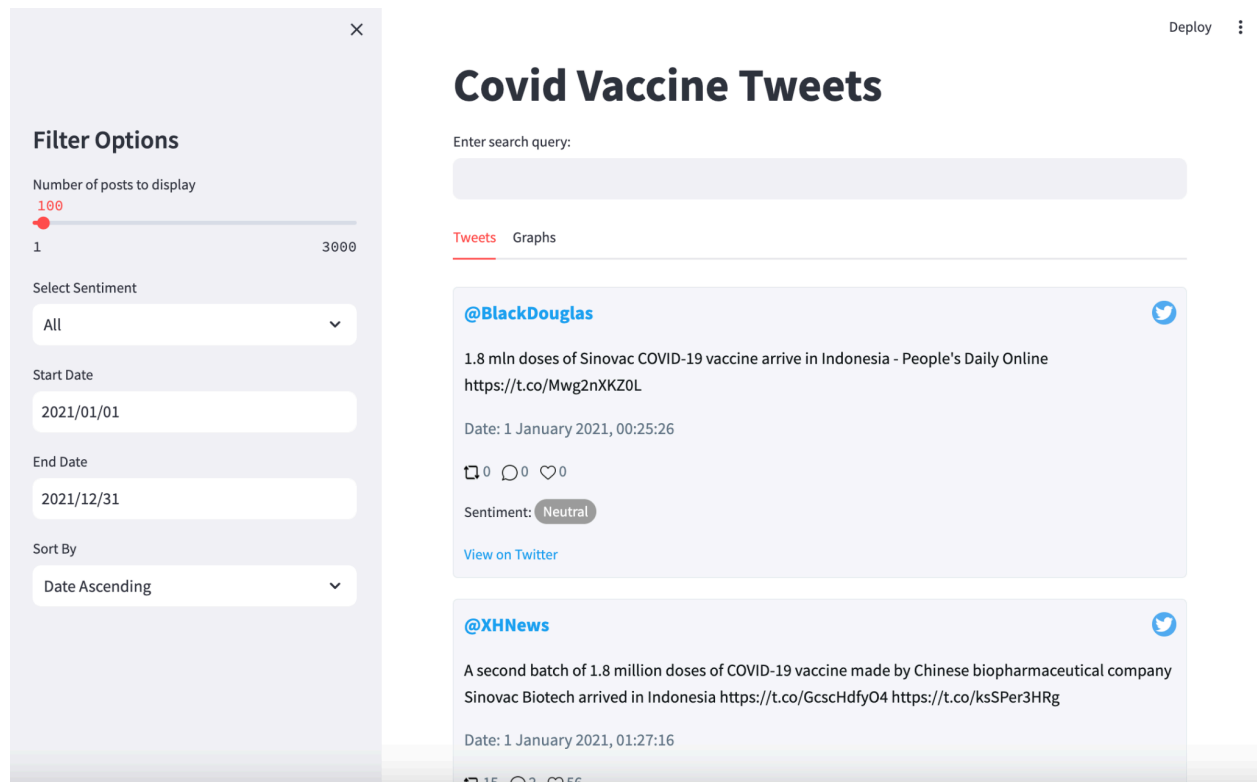


Figure 9: Overview of the User Interface

3.2.2.1 Sidebar Filters

Figure 10 displays the sidebar tab, which contains filtering options for users to customize their querying process. Users can leverage these filtering functions to customize their search results according to their preferences and focus on the most relevant tweets. These options include selecting the number of Tweets to be displayed, filtering by sentiment, specifying a date range, and choosing a preferred sorting method.

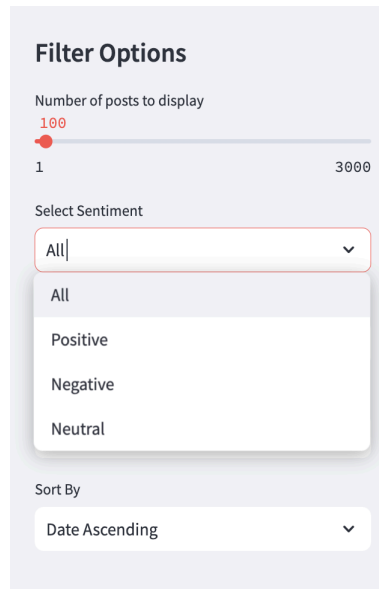
The image shows a sidebar filter panel with a light purple background. It contains the following elements:

- Filter Options**: The title of the panel.
- Number of posts to display**: A slider control. The current value is 100, indicated by a red dot and the number '100' in red. The range is from 1 to 3000.
- Select Sentiment**: A dropdown menu with 'All' selected and a downward arrow.
- Start Date**: A text input field containing '2021/01/01'.
- End Date**: A text input field containing '2021/12/31'.
- Sort By**: A dropdown menu with 'Date Ascending' selected and a downward arrow.

Figure 10: Sidebar Tab with Filtering Options

1. Sentiment filter

For the sentiment filter, users can choose to focus on a specific sentiment such as positive, negative, or neutral. Only tweets that match the selected sentiment will be displayed, allowing users to concentrate on the types of discussions that are most relevant to their interests.



The image shows a 'Filter Options' panel with three main controls:

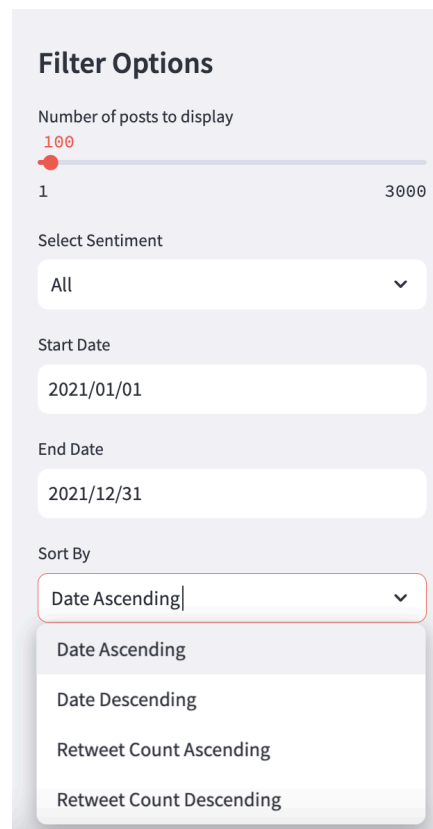
- Number of posts to display:** A slider ranging from 1 to 3000, with a red dot and the number '100' indicating the current selection.
- Select Sentiment:** A dropdown menu currently showing 'All'. The expanded menu lists four options: 'All', 'Positive', 'Negative', and 'Neutral'.
- Sort By:** A dropdown menu currently showing 'Date Ascending'.

Figure 11: Sentiment Filter Options

This filter is especially useful for gauging public opinion and perception of COVID vaccines or vaccine brands. By filtering tweets according to sentiment, users can easily identify prevailing attitudes, concerns, and support for different vaccines. This can provide valuable insights to understand the public's stance on vaccination.

2. Sort method

For the sorting method, users can opt to sort the results by date or retweet count, depending on their preferences. This allows them to prioritize either the most recent tweets or those with the highest retweet counts for greater engagement. Furthermore, users can choose between ascending or descending order, giving them complete control over how the tweets are displayed, as seen in Figure 12.



The image shows a 'Filter Options' form with the following elements:

- Filter Options** (Section Header)
- Number of posts to display**: A slider control with a red dot at 100. The range is from 1 to 3000.
- Select Sentiment**: A dropdown menu currently showing 'All'.
- Start Date**: A text input field containing '2021/01/01'.
- End Date**: A text input field containing '2021/12/31'.
- Sort By**: A dropdown menu currently showing 'Date Ascending'. The dropdown is open, showing four options: 'Date Ascending', 'Date Descending', 'Retweet Count Ascending', and 'Retweet Count Descending'.

Figure 12: Sorting Options by Date or Retweet Count

These sorting options help users focus on the sentiments associated with COVID vaccines or vaccine brands. By fine-tuning their search results, users can quickly identify trends, opinions, and discussions surrounding specific vaccines or brands, allowing for more targeted and insightful analysis.

3.2.2.2 Tweets Tab

The displayed tweets in the 'Tweets' tab include important details such as the username, full text, posting date and time, and counts for retweets, comments, and likes. Users can navigate to other web applications by clicking on the links embedded within the text. Each tweet is presented in a container designed to resemble the appearance of a tweet on popular social media platforms, as seen in Figure 13. This familiar format makes the information easy to understand and navigate, enhancing the user experience.

The sentiment of each tweet is visually conveyed through color-coding: green for positive, red for negative, and gray for neutral. This helps users quickly gauge the overall tone of the tweet.

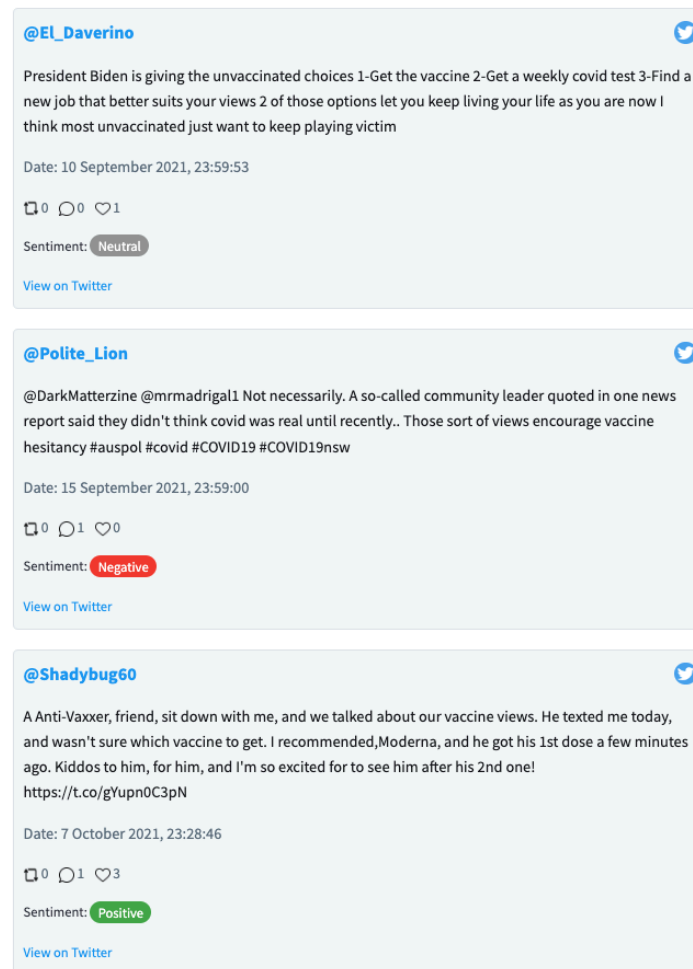


Figure 13: Display of Tweets in the 'Tweets' Tab

3.2.2.3 Graphs Tab

When the user toggles to the 'Graphs' tab, they can view a line graph and a pie chart, as shown in Figure 14. The line graph displays the number of retweets within the selected timeframe, giving users insight into how tweets are being shared over time.

The pie chart shows the distribution of three sentiment labels: Positive, Negative, and Neutral. These labels use consistent colors, with green representing Positive, red for Negative, and gray for Neutral, aligning with the color-coding used in sentiment tags as previously mentioned. This consistency in color coding across the sentiment tags and the pie chart enhances the user interface by providing a cohesive visual experience. It allows users to easily visualize and understand the distribution of sentiments, facilitating a clear analysis of public opinion and trends.

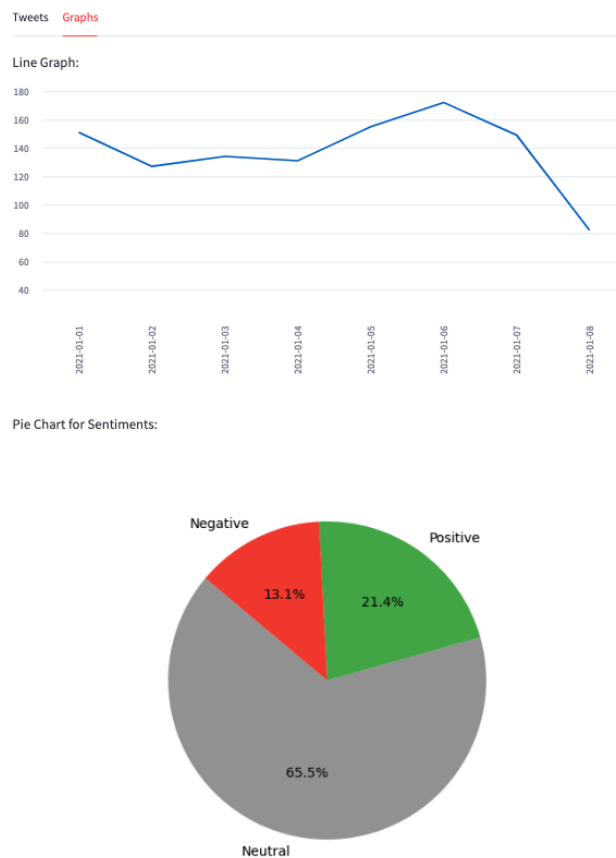


Figure 14: 'Graphs' Tab Showing a Line Graph and Pie Chart

3.3 Search techniques

3.3.1 Stemming

Stemming is a natural language processing technique used to reduce words to their base or root form, known as the stem. The process involves removing suffixes such as prefixes and suffixes from words to normalize them and group together words with similar meanings. For example, stemming converts "benefiting" to the stem "benefit" as shown in Figure 15 below.

As mentioned earlier, we utilize the Solr PorterStemmer filter to apply stemming to user queries and indexed documents. This technique offers several benefits, such as improved search relevance and broader query matching, as it groups variations of a word under a common root. This enhances the user experience by allowing users to retrieve a wider range of related documents with a single query, ultimately leading to more comprehensive and accurate search results.

Covid Vaccine Tweets

Enter search query:

Tweets Graphs

@riotbrdd



@ProfDBernstein 3) Everyone (but especially those at highest risk of death) in the U.S. would have been willing to take the vaccine. Obviously none of that is true. The issues we're having now with distribution completely dwarf the potential benefit of Moderna approval a week earlier. ...

Date: 1 January 2021, 23:31:58

0 1 0

Sentiment: **Positive**

[View on Twitter](#)

Figure 15: Example of Stemming Function

3.3.2 Spellcheck

The spell check function helps address misspelled queries that users may encounter. When a user's query yields fewer than 15 tweets, the spell check function suggests up to two potential corrections that might be the intended query. For example, Figure 16 below shows the outcome when a user misspells the word 'Moderna' in the search bar. The misspelling initially resulted in no tweets being retrieved or displayed. Instead, the spell check function provided alternative suggestions such as 'Moderna' as a possible correction. This feature improves the user experience by guiding users to the correct spelling, allowing them to discover more relevant results and conduct successful searches.



Figure 16: Spell Check Function Offering Suggestions for the Misspelled Query 'Modea'

The suggested words are displayed in the user interface as buttons, allowing users to easily select their preferred suggestion with a single click. Once a user clicks on one of the suggested words, tweets related to the chosen word will be displayed, enhancing the search experience and facilitating quick access to relevant results. This streamlined approach aids users in refining their search queries and finding the information they need efficiently. For instance, Figure 17 below demonstrates the UI after the user clicks on the 'Moderna' button, resulting in the display of tweets related to the query word 'Moderna'.

Covid Vaccine Tweets

Enter search query:

modea

Tweets Graphs

Did you mean:

moderna (Frequency: 15280)

@arirangtvnews



■ MODERNA VACCINE EFFECTIVENESS Moderna says its vaccine remains 93% effective after 6 months
#COVID19 #Moderna #Vaccine <https://t.co/1ry9VG9Khe>

Date: 5 August 2021, 23:50:52

🔄 2 💬 0 ❤️ 4

Sentiment: **Positive**

[View on Twitter](#)

@Tanya07805956



Figure 17: User Interface Showing Suggested Words as Buttons

3.4 Test Query Result and Performance

In this section, we write five test queries and illustrate their results, in detail. QTime is the query time to execute the query.

Table 3: Results of the First Test Query and Its Execution Time (QTime)

Query 1	“happy”																																																												
No. of posts to display	100																																																												
Sentiment Filter	all																																																												
Date Filter	2021/01/01 - 2021/12/31																																																												
Sorting Filter	Date Ascending																																																												
Top Tweet and Graphs	<div data-bbox="558 877 1406 1245"> <p>@dragonflies63</p> <p>Ok. Wth. Folks have been crying for vaccines since last March. Yea happy new year! And this bone head destroys over 500 vials. Why? Does he know something we haven't been told? Pharmacist Arrested, Accused Of Destroying More Than 500 Moderna Vaccine Doses https://t.co/XYetS2jDOL</p> <p>Date: 1 January 2021, 23:41:47</p> <p>🔄 0 💬 0 ❤️ 0</p> <p>Sentiment: Negative</p> <p>View on Twitter</p> </div> <p>Line Graph:</p> <table border="1"> <caption>Line Graph Data (Approximate)</caption> <thead> <tr> <th>Date</th> <th>Sentiment Score</th> </tr> </thead> <tbody> <tr><td>2021-01-01</td><td>1.0</td></tr> <tr><td>2021-01-03</td><td>2.0</td></tr> <tr><td>2021-01-07</td><td>1.0</td></tr> <tr><td>2021-01-11</td><td>1.0</td></tr> <tr><td>2021-01-14</td><td>1.0</td></tr> <tr><td>2021-01-21</td><td>2.0</td></tr> <tr><td>2021-01-28</td><td>1.0</td></tr> <tr><td>2021-01-30</td><td>2.0</td></tr> <tr><td>2021-02-02</td><td>1.0</td></tr> <tr><td>2021-02-04</td><td>2.0</td></tr> <tr><td>2021-02-10</td><td>3.0</td></tr> <tr><td>2021-02-13</td><td>2.0</td></tr> <tr><td>2021-02-15</td><td>3.0</td></tr> <tr><td>2021-02-19</td><td>1.0</td></tr> <tr><td>2021-02-22</td><td>2.0</td></tr> <tr><td>2021-02-25</td><td>1.0</td></tr> <tr><td>2021-03-03</td><td>3.0</td></tr> <tr><td>2021-03-06</td><td>1.0</td></tr> <tr><td>2021-03-08</td><td>3.0</td></tr> <tr><td>2021-03-10</td><td>2.0</td></tr> <tr><td>2021-03-12</td><td>2.0</td></tr> <tr><td>2021-03-14</td><td>1.0</td></tr> <tr><td>2021-03-16</td><td>4.0</td></tr> <tr><td>2021-03-20</td><td>2.0</td></tr> <tr><td>2021-03-23</td><td>5.0</td></tr> <tr><td>2021-03-26</td><td>2.0</td></tr> <tr><td>2021-03-28</td><td>3.0</td></tr> <tr><td>2021-03-30</td><td>2.0</td></tr> <tr><td>2021-04-02</td><td>6.0</td></tr> </tbody> </table>	Date	Sentiment Score	2021-01-01	1.0	2021-01-03	2.0	2021-01-07	1.0	2021-01-11	1.0	2021-01-14	1.0	2021-01-21	2.0	2021-01-28	1.0	2021-01-30	2.0	2021-02-02	1.0	2021-02-04	2.0	2021-02-10	3.0	2021-02-13	2.0	2021-02-15	3.0	2021-02-19	1.0	2021-02-22	2.0	2021-02-25	1.0	2021-03-03	3.0	2021-03-06	1.0	2021-03-08	3.0	2021-03-10	2.0	2021-03-12	2.0	2021-03-14	1.0	2021-03-16	4.0	2021-03-20	2.0	2021-03-23	5.0	2021-03-26	2.0	2021-03-28	3.0	2021-03-30	2.0	2021-04-02	6.0
Date	Sentiment Score																																																												
2021-01-01	1.0																																																												
2021-01-03	2.0																																																												
2021-01-07	1.0																																																												
2021-01-11	1.0																																																												
2021-01-14	1.0																																																												
2021-01-21	2.0																																																												
2021-01-28	1.0																																																												
2021-01-30	2.0																																																												
2021-02-02	1.0																																																												
2021-02-04	2.0																																																												
2021-02-10	3.0																																																												
2021-02-13	2.0																																																												
2021-02-15	3.0																																																												
2021-02-19	1.0																																																												
2021-02-22	2.0																																																												
2021-02-25	1.0																																																												
2021-03-03	3.0																																																												
2021-03-06	1.0																																																												
2021-03-08	3.0																																																												
2021-03-10	2.0																																																												
2021-03-12	2.0																																																												
2021-03-14	1.0																																																												
2021-03-16	4.0																																																												
2021-03-20	2.0																																																												
2021-03-23	5.0																																																												
2021-03-26	2.0																																																												
2021-03-28	3.0																																																												
2021-03-30	2.0																																																												
2021-04-02	6.0																																																												

	<p>Pie Chart for Sentiments:</p> <p>A pie chart illustrating the distribution of sentiments. The chart is divided into three segments: a large green segment representing 'Positive' sentiment at 75.0%, a grey segment representing 'Neutral' sentiment at 22.0%, and a small red segment representing 'Negative' sentiment at 3.0%.</p>
QTime (ms)	9

Table 4: Results of the Second Test Query and Its Execution Time (QTime)

Query 2	“medicine”
No. of posts to display	100
Sentiment Filter	all
Date Filter	2021/01/01 - 2021/12/31
Sorting Filter	Date Ascending
Top Tweet and Graphs	<div> <p>@haleymoney</p> <p>Because I am a teacher, I received my first dose of the #PfizerVaccine today! 🙌 So thankful for science and medicine. #flattenthecurve #leadbyexample https://t.co/bS5zeylq2P</p> <p>Date: 2 January 2021, 23:51:05</p> <p>🔄 0 🗨️ 0 ❤️ 2</p> <p>Sentiment: Positive</p> <p>View on Twitter</p> </div>

	<div>Line Graph:</div> <div>Pie Chart for Sentiments:</div> <table border="1"><caption>Sentiment Distribution</caption><thead><tr><th>Sentiment</th><th>Percentage</th></tr></thead><tbody><tr><td>Positive</td><td>20.0%</td></tr><tr><td>Negative</td><td>22.0%</td></tr><tr><td>Neutral</td><td>58.0%</td></tr></tbody></table>	Sentiment	Percentage	Positive	20.0%	Negative	22.0%	Neutral	58.0%
Sentiment	Percentage								
Positive	20.0%								
Negative	22.0%								
Neutral	58.0%								
QTime (ms)	4								

Table 5: Results of the Third Test Query and Its Execution Time (QTime)

Query 3	“death”																																																																												
No. of posts to display	100																																																																												
Sentiment Filter	negative																																																																												
Date Filter	2021/01/01 - 2021/12/31																																																																												
Sorting Filter	Retweet Count Ascending																																																																												
Results	<div data-bbox="548 638 1409 1045"> <p>@jpfitz</p> <p>@EricaWarner16 @mlucey13 @RonFilipkowski Erica, there is so much money being made off the death beds of Covid-19 patients. All three vaxx compare are doing the revolving door executive swap. Look into who left J&J and who joined Moderna. Billions made and now the vaccine is no where near 91% sure to keep out hosp</p> <p>Date: 28 August 2021, 23:29:40</p> <p>🔄 0 💬 1 ❤️ 0</p> <p>Sentiment: Negative</p> <p>View on Twitter</p> </div> <p>Line Graph:</p> <table border="1"> <caption>Approximate data points from the Line Graph</caption> <thead> <tr> <th>Date</th> <th>Sentiment Score</th> </tr> </thead> <tbody> <tr><td>2021-01-05</td><td>1.0</td></tr> <tr><td>2021-01-17</td><td>2.1</td></tr> <tr><td>2021-02-06</td><td>1.0</td></tr> <tr><td>2021-03-11</td><td>2.1</td></tr> <tr><td>2021-03-16</td><td>2.1</td></tr> <tr><td>2021-04-17</td><td>1.0</td></tr> <tr><td>2021-05-28</td><td>1.0</td></tr> <tr><td>2021-06-07</td><td>1.0</td></tr> <tr><td>2021-06-27</td><td>2.1</td></tr> <tr><td>2021-07-04</td><td>1.0</td></tr> <tr><td>2021-07-08</td><td>2.1</td></tr> <tr><td>2021-07-11</td><td>1.0</td></tr> <tr><td>2021-07-15</td><td>2.0</td></tr> <tr><td>2021-07-18</td><td>1.0</td></tr> <tr><td>2021-07-20</td><td>1.0</td></tr> <tr><td>2021-07-22</td><td>3.0</td></tr> <tr><td>2021-08-06</td><td>1.0</td></tr> <tr><td>2021-08-11</td><td>1.0</td></tr> <tr><td>2021-08-18</td><td>2.1</td></tr> <tr><td>2021-08-29</td><td>1.0</td></tr> <tr><td>2021-09-03</td><td>2.0</td></tr> <tr><td>2021-09-09</td><td>1.0</td></tr> <tr><td>2021-09-23</td><td>1.0</td></tr> <tr><td>2021-09-26</td><td>1.0</td></tr> <tr><td>2021-09-28</td><td>2.1</td></tr> <tr><td>2021-10-05</td><td>3.0</td></tr> <tr><td>2021-10-10</td><td>3.0</td></tr> <tr><td>2021-10-20</td><td>1.0</td></tr> <tr><td>2021-10-22</td><td>2.1</td></tr> <tr><td>2021-10-29</td><td>1.0</td></tr> <tr><td>2021-11-03</td><td>2.1</td></tr> <tr><td>2021-11-20</td><td>2.1</td></tr> <tr><td>2021-11-26</td><td>1.0</td></tr> <tr><td>2021-12-07</td><td>2.1</td></tr> <tr><td>2021-12-13</td><td>1.0</td></tr> <tr><td>2021-12-19</td><td>1.0</td></tr> <tr><td>2021-12-27</td><td>1.0</td></tr> </tbody> </table>	Date	Sentiment Score	2021-01-05	1.0	2021-01-17	2.1	2021-02-06	1.0	2021-03-11	2.1	2021-03-16	2.1	2021-04-17	1.0	2021-05-28	1.0	2021-06-07	1.0	2021-06-27	2.1	2021-07-04	1.0	2021-07-08	2.1	2021-07-11	1.0	2021-07-15	2.0	2021-07-18	1.0	2021-07-20	1.0	2021-07-22	3.0	2021-08-06	1.0	2021-08-11	1.0	2021-08-18	2.1	2021-08-29	1.0	2021-09-03	2.0	2021-09-09	1.0	2021-09-23	1.0	2021-09-26	1.0	2021-09-28	2.1	2021-10-05	3.0	2021-10-10	3.0	2021-10-20	1.0	2021-10-22	2.1	2021-10-29	1.0	2021-11-03	2.1	2021-11-20	2.1	2021-11-26	1.0	2021-12-07	2.1	2021-12-13	1.0	2021-12-19	1.0	2021-12-27	1.0
Date	Sentiment Score																																																																												
2021-01-05	1.0																																																																												
2021-01-17	2.1																																																																												
2021-02-06	1.0																																																																												
2021-03-11	2.1																																																																												
2021-03-16	2.1																																																																												
2021-04-17	1.0																																																																												
2021-05-28	1.0																																																																												
2021-06-07	1.0																																																																												
2021-06-27	2.1																																																																												
2021-07-04	1.0																																																																												
2021-07-08	2.1																																																																												
2021-07-11	1.0																																																																												
2021-07-15	2.0																																																																												
2021-07-18	1.0																																																																												
2021-07-20	1.0																																																																												
2021-07-22	3.0																																																																												
2021-08-06	1.0																																																																												
2021-08-11	1.0																																																																												
2021-08-18	2.1																																																																												
2021-08-29	1.0																																																																												
2021-09-03	2.0																																																																												
2021-09-09	1.0																																																																												
2021-09-23	1.0																																																																												
2021-09-26	1.0																																																																												
2021-09-28	2.1																																																																												
2021-10-05	3.0																																																																												
2021-10-10	3.0																																																																												
2021-10-20	1.0																																																																												
2021-10-22	2.1																																																																												
2021-10-29	1.0																																																																												
2021-11-03	2.1																																																																												
2021-11-20	2.1																																																																												
2021-11-26	1.0																																																																												
2021-12-07	2.1																																																																												
2021-12-13	1.0																																																																												
2021-12-19	1.0																																																																												
2021-12-27	1.0																																																																												

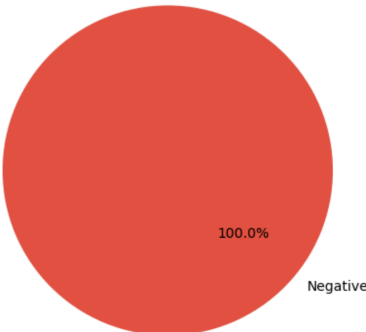
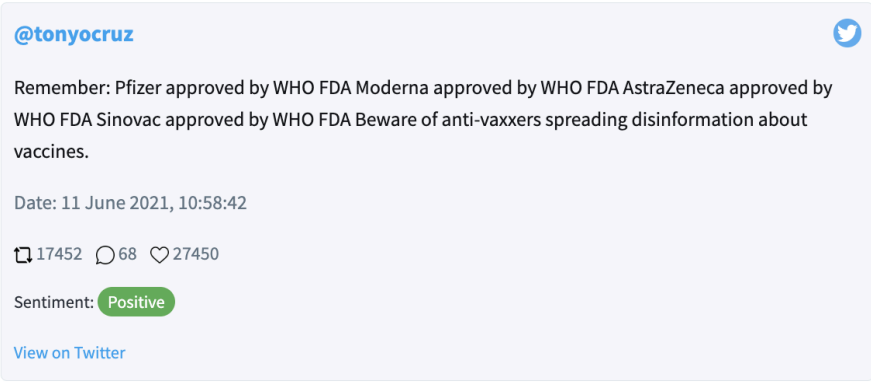
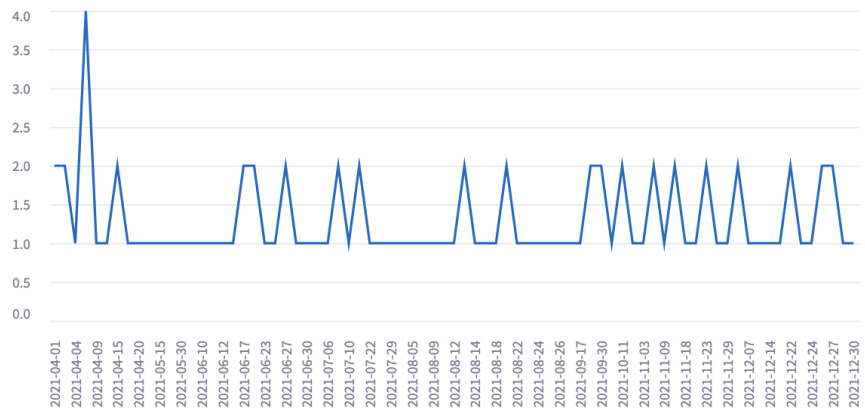
	<p>Pie Chart for Sentiments:</p>  <p>A pie chart with a single red segment representing 100.0% of the data. The segment is labeled 'Negative'.</p>
QTime (ms)	8

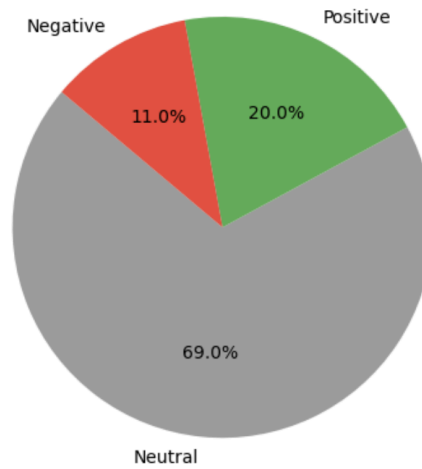
Table 6: Results of the Fourth Test Query and Its Execution Time (QTime)

Query 4	“moderna”
No. of posts to display	100
Sentiment Filter	all
Date Filter	2021/04/01 - 2021/12/31
Sorting Filter	Retweet Count Descending
Results	 <p>A screenshot of a tweet from user @tonyocruz. The tweet text is: "Remember: Pfizer approved by WHO FDA Moderna approved by WHO FDA AstraZeneca approved by WHO FDA Sinovac approved by WHO FDA Beware of anti-vaxxers spreading disinformation about vaccines." The date is 11 June 2021, 10:58:42. It has 17452 retweets, 68 replies, and 27450 likes. The sentiment is labeled as "Positive" in a green box. There is a "View on Twitter" link at the bottom.</p>

Line Graph:



Pie Chart for Sentiments:

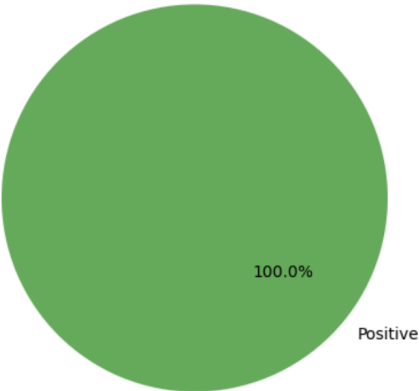


QTime (ms)

4

Table 7: Results of the Fifth Test Query and Its Execution Time (QTime)

Query 5	“pfizer”																																																		
No. of posts to display	300																																																		
Sentiment Filter	positive																																																		
Date Filter	2021/01/01 - 2021/12/01																																																		
Sorting Filter	Date Descending																																																		
Results	<div data-bbox="552 646 1409 1018"> <p>@_nicoleleonard</p> <p>Health providers, medical researchers, public health workers gather in front of the Pfizer Clinical Research Unit in New Haven to call for better global access to COVID-19 vaccines as lower income countries struggle to get doses and remain at risk of deadly outbreaks. https://t.co/kwYyhua0mq</p> <p>Date: 1 December 2021, 23:58:58</p> <p>3 0 4</p> <p>Sentiment: Positive</p> <p>View on Twitter</p> </div> <p>Line Graph:</p> <table border="1"> <caption>Line Graph Data (Approximate)</caption> <thead> <tr> <th>Date</th> <th>Value</th> </tr> </thead> <tbody> <tr><td>2021-10-16</td><td>2</td></tr> <tr><td>2021-10-18</td><td>12</td></tr> <tr><td>2021-10-20</td><td>4</td></tr> <tr><td>2021-10-22</td><td>11</td></tr> <tr><td>2021-10-24</td><td>7</td></tr> <tr><td>2021-10-26</td><td>8</td></tr> <tr><td>2021-10-28</td><td>11</td></tr> <tr><td>2021-10-30</td><td>7</td></tr> <tr><td>2021-11-01</td><td>7</td></tr> <tr><td>2021-11-03</td><td>12</td></tr> <tr><td>2021-11-05</td><td>5</td></tr> <tr><td>2021-11-07</td><td>2</td></tr> <tr><td>2021-11-09</td><td>7</td></tr> <tr><td>2021-11-11</td><td>7</td></tr> <tr><td>2021-11-13</td><td>17</td></tr> <tr><td>2021-11-15</td><td>9</td></tr> <tr><td>2021-11-17</td><td>11</td></tr> <tr><td>2021-11-19</td><td>10</td></tr> <tr><td>2021-11-21</td><td>8</td></tr> <tr><td>2021-11-23</td><td>8</td></tr> <tr><td>2021-11-25</td><td>9</td></tr> <tr><td>2021-11-27</td><td>12</td></tr> <tr><td>2021-11-29</td><td>3</td></tr> <tr><td>2021-12-01</td><td>5</td></tr> </tbody> </table>	Date	Value	2021-10-16	2	2021-10-18	12	2021-10-20	4	2021-10-22	11	2021-10-24	7	2021-10-26	8	2021-10-28	11	2021-10-30	7	2021-11-01	7	2021-11-03	12	2021-11-05	5	2021-11-07	2	2021-11-09	7	2021-11-11	7	2021-11-13	17	2021-11-15	9	2021-11-17	11	2021-11-19	10	2021-11-21	8	2021-11-23	8	2021-11-25	9	2021-11-27	12	2021-11-29	3	2021-12-01	5
Date	Value																																																		
2021-10-16	2																																																		
2021-10-18	12																																																		
2021-10-20	4																																																		
2021-10-22	11																																																		
2021-10-24	7																																																		
2021-10-26	8																																																		
2021-10-28	11																																																		
2021-10-30	7																																																		
2021-11-01	7																																																		
2021-11-03	12																																																		
2021-11-05	5																																																		
2021-11-07	2																																																		
2021-11-09	7																																																		
2021-11-11	7																																																		
2021-11-13	17																																																		
2021-11-15	9																																																		
2021-11-17	11																																																		
2021-11-19	10																																																		
2021-11-21	8																																																		
2021-11-23	8																																																		
2021-11-25	9																																																		
2021-11-27	12																																																		
2021-11-29	3																																																		
2021-12-01	5																																																		

	<p>Pie Chart for Sentiments:</p>  <p>A pie chart titled 'Pie Chart for Sentiments:' is displayed. It consists of a single green circle representing 100.0% of the data. The label '100.0%' is placed inside the circle, and the word 'Positive' is placed to the right of the circle, indicating the sentiment category.</p> <table border="1"><thead><tr><th>Sentiment</th><th>Percentage</th></tr></thead><tbody><tr><td>Positive</td><td>100.0%</td></tr></tbody></table>	Sentiment	Percentage	Positive	100.0%
Sentiment	Percentage				
Positive	100.0%				
QTime (ms)	5				

4. Classification

4.1 Subtasks

Our team had decided to perform sentiment analysis of Covid-19 vaccine tweets by targeting two subtasks: subjectivity and polarity classification.

Subjectivity classification plays an important role in classifying tweets to be either neutral or opinionated labels. As Twitter is a social media platform with a wide range of purposes including news dissemination, this is a necessary step in separating out the objective content in order to retrieve only the relevant opinionated tweets that the information retrieval system user wishes to see. For instance, tweets such as “Winter Storm Delays and Closures COVID-19 Vaccine Sites: Closed 2/2. COVID-19 Test Sites: Plan to reopen at noon 2/2...” are merely for the purpose of news dissemination and do not contain the relevant vaccine sentiments that the users of our information retrieval system wishes to see. They thus need to be flagged as neutral accordingly in this subtask, such that the users are able to apply the relevant filters to exclude them from the returned tweets.

For tweets that are classified as opinionated, they will be used for polarity classification, further distinguishing the sentiments into negative or positive labels. These subtasks can be challenging due to the nuanced nature of the human language. In our case, we are dealing with the context of Covid-19 vaccine sentiments which is different from the typical sentiment analysis of any given sentence. This makes the subtask more complex, as a tweet with a negative sentiment may be overall in support of Covid vaccines. An example of such a tweet is “Not for Covid. How the hell are you a US Senator? Why do u hate vaccines? It’s so stupid.” where the Twitter user is exhibiting a negative tone against vaccine doubters, which reinforces support and trust in the Covid vaccines. Similarly, the converse may be true where the Twitter user expresses praise and support for antivaccine sentiments. Due to this increased complexity in polarity classification, our team had to manually label the crawled corpus for the training of our classification models.

4.2 Covid-19 vaccine tweets dataset

After consolidating the tweets that are crawled from the four keywords, the total number of tweets amounts to 51315. However, as some of the tweets may be retrieved more than once from different keywords, all the duplicate tweets are first identified using a combination of 12 features including “created_at”, “id_str” and “full_text”, and dropped. In total, 576 duplicate tweets are identified and removed from the consolidated tweet dataset, leaving behind 50739 unique tweets. This consolidation of tweets and dropping of duplicates is performed in the 1 Consolidate Tweets.ipynb notebook.

With the consolidated tweet dataset, a total of 3000 tweets are manually labelled. Each of these tweets is labelled by two different annotators, reaching an interannotator agreement of 81.6%. The breakdown of the labels are provided in Figure 18. Note that the column “Sentiment Analysis (Label)” is from the first annotator, while “label2” is from the second annotator. This step is performed in the 2 Finetune Huggingface.ipynb notebook.

	label2	Negative	Neutral	Positive
Sentiment Analysis (Label)				
Negative		465	127	71
Neutral		90	1315	149
Positive		15	101	667

Figure 18: Confusion Matrix of the two manual annotators on the 3000 labelled tweets

These 3000 double labelled tweets are split into 1000 for the evaluation dataset, and 2000 for the train dataset. For the purpose of training and evaluation that requires only one label per tweet, only the first label “Sentiment Analysis (Label)” is retained. This preliminary train set is relatively well balanced, with 1068 neutral labels and 932 opinionated labels, which are further split into 413 negative and 519 positive labels.

When training the machine learning models, we discovered that the 2000 manually labelled tweets are insufficient for the models to sufficiently capture the language patterns and sentiment in the text. Hence, to obtain more labelled tweets for training, we leveraged a state-of-the-art (SOTA) transformer based RoBERTa (Robustly Optimized BERT Pretraining Approach) model

called BERTweet which was pretrained on english tweets (Nguyen et al., EMNLP 2020). After fine tuning the SOTA model on the 2000 manually labelled tweets in the train set, it is used to auto-label an additional 5000 tweets to yield 7000 labelled tweets for training the machine learning models. The distribution of this train dataset for subjectivity is relatively balanced, with 3855 neutral labels and 3145 opinionated labels. The distribution for polarity is relatively balanced with 1376 negative labels and 1769 positive labels. More details on the finetuning and evaluation of the SOTA model can be found in section 5.5.

4.3 Preprocessing of Covid-19 vaccine tweets

Tweets refer to short messages posted on the Twitter platform. Compared to normal sentences, tweets are limited to a maximum of 280 characters per tweet. Tweets have several distinct characteristics that are crucial to be accounted for in preprocessing compared to the normal sentence. The preprocessing steps we used are listed as follows.

```
import re
import contractions
import emoji
from nltk.stem import WordNetLemmatizer

def clean_text(text):

    # Step 1: Expand contractions
    text = contractions.fix(text)

    # Step 2: Map emojis into its word meaning
    text = emoji.demojize(text)

    # Step 3: Remove mentions, hashtags, numbers and links
    pattern = r'@[A-Za-z0-9_]+|#[A-Za-z0-9_]+|\d+|https?://\S+|[\^\w\s]'
    text = re.sub(pattern, '', text)

    # Step 4: Lemmatize words
    wordnet_lemmatizer = WordNetLemmatizer()
    words = text.split()
    words = [wordnet_lemmatizer.lemmatize(word, pos="v") for word in words]
    text = ' '.join(words)

    return text
```

Figure 19: Code for Preprocessing Tweets

1. Presence of contractions. Due to the informal nature of tweets, there can be many contractions used in tweets. Contractions are expanded.
2. Presence of emojis. As Twitter is a social media platform where tweets are character constrained, many users turn to emojis to convey their feelings. Emojis can play an

important role in sentiment analysis, as such it will be processed into its word meaning through a python library, emoji.

3. Presence of mentions @, hashtags #, numbers and links. Mentions, hashtags, numbers and links often do not contain enough information for sentiment analysis and can be abused by bots via spamming on Twitter's platform for views and engagement. Thus, it is essential that mentions and hashtags are removed.
4. Lemmatization, this is a common preprocessing step in Natural Language Processing. It normalizes words to their base form which reduces the dimensionality of feature space and improves the generalization of Machine Learning models.

4.4 Machine Learning Models

4.4.1 Logistic Regression

Logistic Regression is a statistical model widely employed in supervised learning tasks, primarily in binary classification. The goal of a Logistic Regression model is to predict the probability of a given sample belonging to a certain class. This is possible due to a sigmoid function being applied which outputs values in the range of 0 to 1.

$$S(x) = \frac{1}{1 + e^{-x}}$$

$S(x)$ = sigmoid function

e = Euler's number

Figure 20: Sigmoid Function

A threshold value, usually 0.5, is applied such that when the output probability is more than 0.5, the sample will be classified as class 1 whereas an output probability of less than and equal to 0.5 will be classified as class 0.

4.4.2 Support Vector Machine (SVM)

Support Vector Machine is a supervised learning model used for classification and regression tasks. The main objective of SVM is to find an optimal line or hyperplane that maximizes the distance between each class in an N-dimensional space. The number of input features determine if the hyperplane is in a 2 dimensional space or n-dimensional space. Support vectors are the closest data points to the hyperplane, which plays a critical role in deciding the hyperplane and margin.

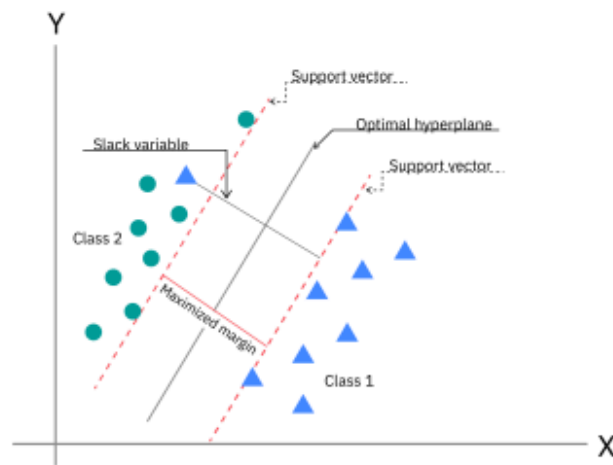


Figure 21: SVM Graph Visualisation (Image credits: IBM)

4.4.3 Naive Bayes

Naive Bayes classifier is a supervised machine learning model used for classification tasks. The objective of Naive Bayes classifier is to give a probability output of a certain class given a sample. The algorithm applies Bayes Theorem of conditional probability and makes an

assumption of conditional independence among features.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

Likelihood
Class Prior Probability
Posterior Probability
Predictor Prior Probability

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \dots \times P(x_n|c) \times P(c)$$

Figure 22: Naive Bayes Posterior Probability

4.4.4 Random Forest

Random Forest is an ensemble learning method developed by Leo Breiman and Adele Cutler for classification and regressions tasks. It combines the output of multiple decision trees. In classification tasks, the output of the Random Forest is the class selected by the majority of the trees.

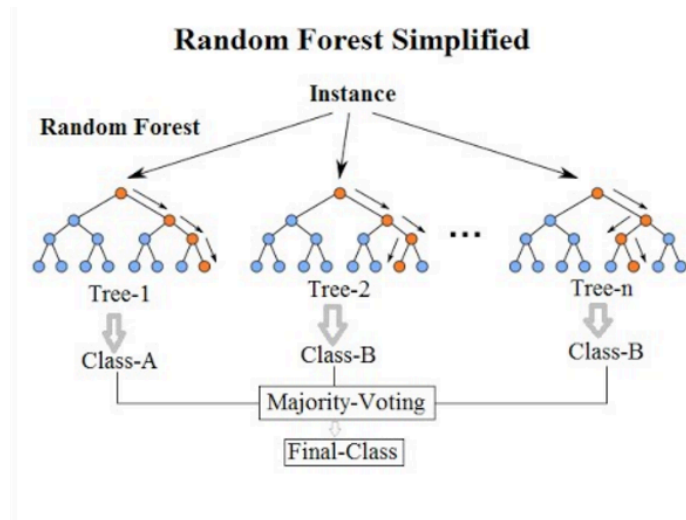


Figure 23: Random Forest Visualisation (Image credits: Towards Data Science)

4.5 Feature Extraction

Preprocessed tweets are tokenized using NLTK tokenizer from the NLTK library in python before being passed into a Vectorizer from the sk-learn library.

4.5.1 CountVectorizer

CountVectorizer is a Vectorizer from the sk-learn library which converts a collection of text documents into a matrix of token count.

4.5.2 TF-IDF Vectorizer

TfidfVectorizer is a Vectorizer from sk-learn library which converts a collection of raw documents to a matrix of TF-IDF features. TF-IDF refers to Term Frequency * Inverse Document Frequency. The term frequency of a term t in document d is defined as the number of times that t occurs in d . The document frequency of a term t is defined as the number of documents which contains the term t . The IDF of a term t is can be calculated using $\log(N/df_t)$ where N is the total number of documents. There are different variants in calculating TF-IDF scores.

4.5.3 N-grams

N-grams are contiguous sequences of n items, in our case words. $N=1$ refers to unigrams, $N=2$ refers to bigrams. Bigrams can be useful in picking up sentiments as when the sentiments are made up of two consecutive words.

4.6 Results

The performance of the models during training is evaluated using K-Fold cross validation, employing a 5-fold approach. After training, the models are evaluated with an unseen test dataset. The metrics used are accuracy, precision, recall and F1-score. The models are trained with unigrams and (unigrams and bigrams) inputs to find the best models. For bigrams and (bigrams and trigrams) the results were found to be poor.

The abbreviated forms used in the tables and following sections are as follows:

- CV: CountVectorizer
- TFIDF: Term Frequency-Inverse Document Frequency
- LR: Logistic Regression
- SVM: Support Vector Machine
- NB: Naive Bayes
- RF: Random Forest
- K-Acc: Mean K-Fold accuracy
- K-Pre: Mean K-Fold Precision
- K-Rec: Mean K-Fold Recall
- K-F1: Mean K-Fold F1-Score
- Test-Acc: Test Accuracy
- Test-Pre: Test Precision
- Test-Rec: Test Recall
- Test-F1: Test F1-Score

4.6.1 Subjectivity

Table 8: Results for Unigrams

Metrics	CV + LR	CV + SVM	CV + NB	CV + RF	TFIDF + LR	TFIDF + SVM	TFIDF + NB	TFIDF + RF
K-Acc	0.6855	0.6835	0.6802	0.6855	0.6899	0.6936	0.6882	0.6785
K-Pre	0.6842	0.6829	0.6897	0.6847	0.6892	0.6932	0.6887	0.6776
K-Rec	0.6855	0.6835	0.6802	0.6855	0.6899	0.6936	0.6882	0.6785
K-F1	0.6836	0.6801	0.6809	0.6841	0.6873	0.6917	0.6872	0.6762
T-Acc	0.657	0.665	0.679	0.657	0.688	0.678	0.664	0.647
T-Pre	0.6656	0.6787	0.6791	0.6644	0.6975	0.6856	0.6675	0.6593

T-Rec	0.657	0.665	0.679	0.657	0.688	0.678	0.664	0.647
T-F1	0.6545	0.6607	0.6784	0.6549	0.6857	0.6762	0.6634	0.6427

Table 9: Results for Unigrams + Bigrams

Metrics	CV + LR	CV + SVM	CV + NB	CV + RF	TFIDF + LR	TFIDF + SVM	TFIDF + NB	TFIDF + RF
K-Acc	0.6943	0.6853	0.6885	0.6771	0.6909	0.6905	0.6921	0.6756
K-Pre	0.6936	0.6847	0.6988	0.6768	0.6904	0.6902	0.6928	0.6745
K-Rec	0.6943	0.6853	0.6885	0.6771	0.6909	0.6905	0.6921	0.6756
K-F1	0.6925	0.6818	0.6891	0.6732	0.6901	0.6899	0.6889	0.6736
T-Acc	0.689	0.681	0.68	0.652	0.682	0.676	0.67	0.668
T-Pre	0.697	0.691	0.68	0.666	0.686	0.679	0.6799	0.6756
T-Rec	0.689	0.681	0.68	0.652	0.682	0.676	0.67	0.668
T-F1	0.6872	0.6785	0.6789	0.6472	0.6813	0.6754	0.6673	0.666

The best 3 models for subjectivity classification given the trade-offs are TFIDF + LR (unigrams), CV + LR (unigrams + bigrams) and TFIDF + LR (unigrams + bigrams).

4.6.2 Polarity

Table 10: Results for Unigrams

Metrics	CV + LR	CV + SVM	CV + NB	CV + RF	TFIDF + LR	TFIDF + SVM	TFIDF + NB	TFIDF + RF
K-Acc	0.7037	0.6922	0.7091	0.6881	0.7107	0.7202	0.6464	0.6855
K-Pre	0.7026	0.6915	0.7117	0.6919	0.7116	0.7211	0.7199	0.6879

K-Rec	0.7037	0.6922	0.7091	0.6881	0.7107	0.7202	0.6464	0.6855
K-F1	0.7024	0.6872	0.701	0.6769	0.7048	0.7152	0.5845	0.6748
T-Acc	0.6693	0.6576	0.6401	0.6381	0.6693	0.6887	0.5798	0.6518
T-Pre	0.6693	0.6594	0.6471	0.6477	0.6721	0.6894	0.6459	0.6616
T-Rec	0.6693	0.6576	0.6401	0.6381	0.6693	0.6887	0.5798	0.6518
T-F1	0.6688	0.6552	0.6329	0.6289	0.6665	0.6878	0.515	0.6436

Table 11: Results for Unigrams + Bigrams

Metrics	CV + LR	CV + SVM	CV + NB	CV + RF	TFIDF + LR	TFIDF + SVM	TFIDF + NB	TFIDF + RF
K-Acc	0.7237	0.6967	0.7049	0.6843	0.7145	0.7103	0.6114	0.6763
K-Pre	0.7226	0.6972	0.7212	0.6958	0.7237	0.7246	0.7247	0.6791
K-Rec	0.7237	0.6967	0.7049	0.6843	0.7145	0.7103	0.6114	0.6763
K-F1	0.7219	0.6904	0.6875	0.6667	0.7025	0.6951	0.5144	0.6646
T-Acc	0.7218	0.6693	0.6595	0.6362	0.6848	0.6887	0.5661	0.6206
T-Pre	0.7218	0.6737	0.6825	0.6549	0.6957	0.7025	0.6905	0.6318
T-Rec	0.7218	0.6693	0.6595	0.6362	0.6848	0.6887	0.5661	0.6206
T-F1	0.7216	0.6655	0.645	0.6206	0.6784	0.6812	0.4697	0.6081

The best 3 models for subjectivity classification given the trade-offs are TFIDF + SVM (unigrams), CV + LR (unigrams + bigrams) and TFIDF + SVM (unigrams + bigrams).

5. Enhancements

5.1 Ensemble Model

Ensemble model in machine learning is a technique of combining multiple base models to achieve improvements and robustness in predictive performance. Each model may have different predictions given the same sample. Having multiple models and a hard voting system (majority vote) in an ensemble model could help to improve predictive performance. The results from 4.6 showed trade-offs between metrics and we may not have a single model that performs the best in all metrics, therefore an ensemble model is applied to explore improvements and robustness.

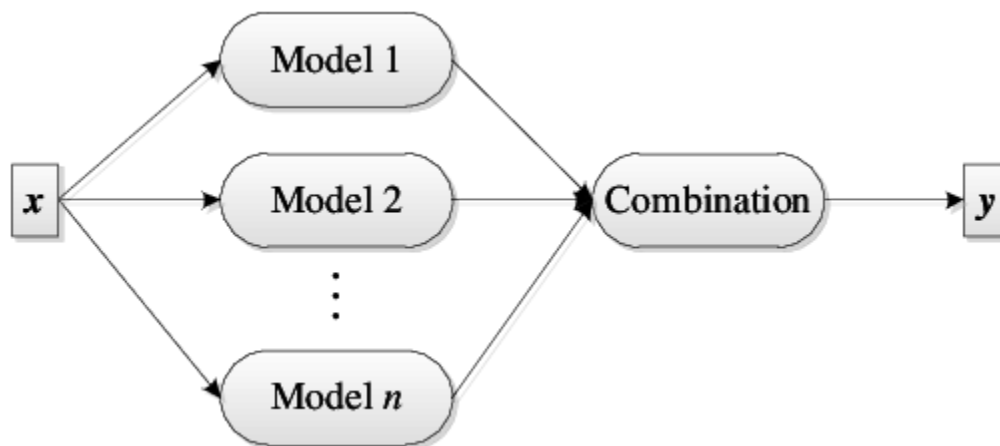


Figure 24 : Ensemble Model Visualisation (Image credits: Michael Affenzeller)

5.1.1 Subjectivity

The best 3 models for subjectivity classification given the trade-offs are TFIDF + LR (unigrams), CV + LR (unigrams + bigrams) and TFIDF + LR (unigrams + bigrams), the ensemble model will consists of the 3 models and a hard voting system.

Table 12: Results for Subjectivity

Metrics	TFIDF + LR (unigrams)	CV + LR (unigrams + bigrams)	TFIDF + LR (unigrams + bigrams)	Ensemble
K-Acc	0.6899	0.6943	0.6909	0.6942
K-Pre	0.6892	0.6936	0.6904	0.6935
K-Rec	0.6899	0.6943	0.6909	0.6942
K-F1	0.6873	0.6925	0.6901	0.6926
T-Acc	0.688	0.689	0.682	0.691
T-Pre	0.6975	0.697	0.686	0.6988
T-Rec	0.688	0.689	0.682	0.691
T-F1	0.6857	0.6872	0.681	0.6893

The ensemble model showed improvements compared to using a single model. It displayed the best Test metrics scores and its performance is on-par with the best mean K-Fold metrics. The ensemble model will be used for subjectivity classification.

5.1.2 Polarity

The best 3 models for subjectivity classification given the trade-offs are TFIDF + SVM (unigrams), CV + LR (unigrams + bigrams) and TFIDF + SVM (unigrams + bigrams), the ensemble model will consists of the 3 models and a hard voting system.

Table 13: Results for Polarity

Metrics	TFIDF + SVM (unigrams)	TFIDF + SVM (unigrams + bigrams)	CV + LR (unigrams + bigrams)	Ensemble
---------	---------------------------	--	------------------------------------	----------

K-Acc	0.7202	0.7103	0.7215	0.7189
K-Pre	0.7211	0.7246	0.7204	0.7224
K-Rec	0.7202	0.7103	0.7215	0.7189
K-F1	0.7152	0.6951	0.7193	0.7115
T-Acc	0.6887	0.6887	0.7335	0.7062
T-Pre	0.6894	0.7025	0.7337	0.7087
T-Rec	0.6887	0.6887	0.7335	0.7062
T-F1	0.6878	0.6812	0.7331	0.7045

The ensemble model showed improvements when compared to TFIDF + SVM (unigrams) and CV + LR (unigrams + bigrams). However its performance is lower than TFIDF + SVM (unigrams + bigrams). The decreased performance can be attributed to less performing models being the majority vote. CV + LR (unigrams + bigrams) will be used for polarity classification.

5.2 Grid Search

Grid Search is a tuning technique that attempts to find the optimum hyperparameters of a model which results in the most accurate predictions. The grid search function from sk learn model selection scores the model based on K-Folds cross validation using 5 folds. For Logistic Regression models, the hyperparameter we tuned are the regularization parameter (C) and the solver algorithm. The regularization parameter controls the balance between fitting the training data and preventing overfitting, the values used are 0.1, 1, 10. The solver algorithm is the algorithm used in the optimization of coefficients, the algorithms used are 'lbfgs', 'liblinear', 'newton-cg', 'sag'.

```

# Define the parameter grid for grid search for logistic regression model
lr_param_grid = {
    'classifier__C': [0.1, 1, 10], # Parameters for logistic regression in lr_model
    'classifier__solver': ['lbfgs', 'liblinear', 'newton-cg', 'sag'], # Solver options for Logistic regression in lr_model
}

# Perform grid search for logistic regression model with TF-IDF vectorizer
lr_grid_search = GridSearchCV(lr_cv_ngram_model, lr_param_grid, cv=5)
lr_grid_search.fit(X_train, y_train)

# Get best parameters and score for logistic regression model with TF-IDF vectorizer
print("Best Parameters (LR cv):", lr_grid_search.best_params_)
print("Best Score (LR cv):", lr_grid_search.best_score_)

# Evaluate the best model found by grid search on test data
test_accuracy = lr_grid_search.score(X_test, y_test)
print("Test Accuracy (LR cv):", test_accuracy)

```

Figure 25: Code for Grid Search

5.2.1 Subjectivity

The hyperparameters found for TFIDF + LR (unigrams + bigrams) are $C = 10$ and solver = 'lbfgs'. The ensemble model was updated to compare against the original results.

Table 14: Results for Ensemble with Grid Search

Metrics	Original Ensemble	Updated Ensemble
Mean K-Fold accuracy	0.6942	0.7018
Mean K-Fold Precision	0.6935	0.7013
Mean K-Fold Recall	0.6942	0.7018
Mean K-Fold F1-Score	0.6926	0.7009
Test Accuracy	0.691	0.698
Test Precision	0.6988	0.7039
Test Recall	0.691	0.698
Test F1-Score	0.6893	0.6969

The updated ensemble model for subjectivity classification has shown improved performance for all metrics due to grid search.

5.2.2 Polarity

The hyperparameters found for CV + LR (unigrams + bigrams) are $C = 10$ and solver = 'sag'. The CV + LR (unigrams + bigrams) model was updated to compare against the original results.

Table 15: Results for Polarity with Grid Search

Metrics	Original CV + LR (unigrams + bigrams)	Updated CV + LR (unigrams + bigrams)
Mean K-Fold accuracy	0.7237	0.7208
Mean K-Fold Precision	0.7226	0.7201
Mean K-Fold Recall	0.7237	0.7208
Mean K-Fold F1-Score	0.7219	0.7196
Test Accuracy	0.7218	0.7237
Test Precision	0.7218	0.7237
Test Recall	0.7218	0.7237
Test F1-Score	0.7216	0.7236

The updated model by grid search shows a slight increase performance in Test metrics but a slight decrease in performance for K-Fold metrics, this is due to different shuffles in K-Fold used by grid search function from sk learn compared to our K-Fold evaluation method that resulted in different performance. We will use the original model as it produces results that are more consistent with each other.

5.3 Named Entity Recognition

Named Entity Recognition is a Natural Language Processing method to extract named entities. Named entities can include, but not limited to, names of individuals, companies, locations, times, monetary values and percentages. Applying named entity recognition in our case to classify sentiments of Covid-19 vaccine tweets yielded poor results. This could be due to our specific context of Covid-19 vaccine tweets where the named entities revolve around vaccine brand names which do not convey the sentiment of Covid-19 vaccines.

Table 16: Results for NER

Metrics	NER model
Mean K-Fold accuracy	0.5506
Mean K-Fold Precision	0.3034
Mean K-Fold Recall	0.5506
Mean K-Fold F1-Score	0.3912
Test Accuracy	0.486
Test Precision	0.2362
Test Recall	0.486
Test F1-Score	0.3179

5.4 Word Sense Disambiguation

Word Sense Disambiguation (WSD) is a classical problem in Natural Language Processing of identifying which sense (meaning) of a word is used in the context of a sentence. A common algorithm used to perform WSD is the Lesk algorithm which compares the dictionary definitions of an ambiguous word with the surrounding context sentence. The application of WSD on the train corpus using NLTK lesk, resulted in extremely long training times with no results generated after an hour. Given the characteristics of tweets for Covid-19 vaccine sentiment where there

may not be many ambiguous words to make sense of and WSD's long training time, applying WSD was not feasible in our case.

5.5 State of the Art

Our main motivation for using basic machine learning approaches like logistic regression, SVM, Naive Bayes, and random forest classifiers are their simplicity in model architecture and their ability to scale efficiently. These methods have straightforward architectures with relatively few parameters compared to more complex natural language processing models. This simplicity contributes significantly to their scalability in terms of faster training and deployment and their use of fewer computational resources to finetune and make classifications, making them suitable for handling larger datasets. Enhancements in terms of ensemble models can also relatively inexpensively improve the accuracy and robustness of this approach. However, when implementing this approach for our use case, we observed significant challenges in model training, with 2000 labelled tweets not being sufficient to adequately train the models. The classification accuracy also seems to hit a low ceiling indicating the challenge in capturing complex relationships and nuances present in tweets. This is in part due to the complexity of working with natural language, which was further elevated since the polarity is based not on word choice but also relating to the context of covid vaccines. This necessitates the search for a more advanced transformer-based model that leverages self-attention mechanisms and large-scale training which is better adapted for natural language classification tasks.

In applying our chosen pretrained transformer-based BERTweet model that directly classifies tweets into neutral, positive and negative labels, we are able to combine the subjectivity and polarity detection steps together and perform multitask classification thanks to the model's versatility enabled by its complex architecture. We are also able to apply transfer learning through finetuning the pretrained model on our own classification rules (sentiment with respect to vaccines and not tone) to transfer over the conversational patterns learned from the average tweet to improve model generalisation. This makes things convenient for us as the BERTweet model is designed around tweets, with its own tweet preprocessing steps in place, making it not necessary for us to do it again. However, even though the model is able to perform very well for

this assignment, it suffers from high computational and time complexities, reducing its scalability and applicability in real life situations that deal with large inflows of data.

The use of the SOTA model in our assignment is twofold: generating 5000 more autolabelled data for training the other machine learning models (see section 4.2), and performing the labelling of the remaining 42000 over unlabelled tweets for our retrieval system.

SOTA Model Finetuning

Before using the model to make any classifications, it needs to be finetuned to learn our specific use case, which is the labelling of tweets with respect to the Twitter user's sentiment towards Covid vaccines. As this step is computationally intensive, it is performed on Google Colab to make use of their advanced GPUs. For the purpose of finetuning, we performed an 80-20 train test split on the existing train set of 2000 manually labelled tweets. Through some investigations, we were able to determine that 8 epochs is optimal in reaching a small enough loss on our manually labelled train set. The additional finetuning diagnostics are displayed in Table 17. This step is performed in the 2 Finetuning Huggingface.ipynb notebook. Note that the column "Sentiment Analysis (Label)" is the manual label, while "label2" is from the finetuned SOTA model.

Table 17: Diagnostics for the Finetuning of the SOTA Model

Metrics	SOTA Model Finetuning Diagnostics
Train Test Ratio	80:20
Number of Epochs	8
Training Size	12800 (1600 tweets over 8 epochs)
Initial Training Loss (Step 10)	0.969700
Final Training Loss (Step 800)	0.013300
Total Runtime (s)	127.93
Training time per sample (s)	0.00999

Train samples per second	100.05
--------------------------	--------

Evaluation of Finetuned SOTA Model

After finetuning, the model's performance is evaluated on the evaluation dataset. This step is also performed in the 2 Finetuning Huggingface.ipynb notebook, and executed on Google Colab. Note that the column "Sentiment Analysis (Label)" is the manual label, while "label2" is from the finetuned SOTA model. The different metrics of precision, recall, and F1 score are the weighted average values for all three labels (Negative, Neutral, Positive) in each case.

label2	Negative	Neutral	Positive
Sentiment Analysis (Label)			
Negative	228	11	11
Neutral	12	461	13
Positive	3	23	238

Figure 26: Confusion Matrix of the Finetuned SOTA Model on the Evaluation Dataset

Table 18: Summarised Results for Finetuned SOTA Model

Metrics	Finetuned SOTA model
Test Accuracy	0.927
Test Precision	0.927
Test Recall	0.927
Test F1-Score	0.927

Labelling of Unlabelled Tweets

The finetuned model is then used to label the remaining dataset. The labelling of the first 5000 tweets to augment the existing train set is done in the 2 Finetuning Huggingface.ipynb notebook,

and executed on Google Colab, while the labelling of the remaining 42739 tweets is done in the 3 Label Entire Corpus.ipynb notebook, and executed on my PC. The labelling performance of the SOTA model is as follows.

Table 19: Results for SOTA Time Tests

Metrics	Finetuned SOTA model (Google Colab)	Finetuned SOTA model (Personal Computer)
Inference time (s)	249.18	2119.98
Inference size	5000	42739
Inference time per sample (s)	0.049836	0.049603
Inference samples per second	20.07	20.16

5.6 Discussion

5.6.1 Time Tests

Table 20: Consolidated Results for Time Tests

Metrics	Machine Learning Models (Subjectivity)	Machine Learning Models (Polarity)	Finetuned SOTA Model
Training time (s)	6.1089	1.1484	127.93
Training size	6998	3145	12800
Training time per sample (s)	0.0009	0.000365	0.00999
Train samples per second	1145.547	2738.61	100.05
Inference time (s)	0.375	0.075	2369.16 (total)
Inference size	1000	514	47739 (total)
Inference time per	0.0004	0.000146	0.049627 (average)

sample (s)			
Inference samples per second	2666.67	6853.33	20.15018 (average)

The machine learning approach is much more scalable than the finetuned SOTA model due to its fast Training and Inference time per second, especially so when the selected models turned out to be Logistic Regression models. Logistic regression models are able to process large volumes of data at high speeds as they require less computational capacity, such as memory and processing power. This makes them ideal for scalability.

5.6.2 Accuracy Test

Table 21: Consolidated Results for Accuracy Test

Metrics	Machine Learning Models (Subjectivity)	Machine Learning Models (Polarity)	Finetuned SOTA Model
Test Accuracy	0.698	0.7218	0.927
Test Precision	0.7039	0.7218	0.927
Test Recall	0.698	0.7218	0.927
Test F1-Score	0.6969	0.7216	0.927

The predictive accuracy of the finetuned SOTA model is much better than the machine learning approach, despite the enhancements from ensemble methods and grid search. Despite so, in a real life situation, the fast inference speed of the machine learning models make them much more desirable for most users of information retrieval systems. While the finetuned SOTA model can take in excess of 40 minutes to label the over 47000 tweets, the machine learning models can do the same in a little over 20 seconds (assuming a mix of 50% neutral and 50% subjective tweets) despite necessitating two separate steps of subjectivity detection and polarity detection. Perhaps with the advancement of technology and increased affordability of required hardware, the SOTA model could replace the machine learning models more feasibly.

```
# Make predictions using the Loaded model
tweet_1 = 'The vaccine is bad, do not take it.'
tweet = preprocess(tweet_1)
predictions = loaded_model.predict([tweet])
print(tweet_1, decode(predictions) )

tweet_2 = 'The vaccine is good for everyone even though it hurts badly, please take it.'
tweet = preprocess(tweet_2)
predictions = loaded_model.predict([tweet])
print(tweet_2, decode(predictions) )

tweet_3 = 'Covid vaccine is available for everyone to take for free.'
tweet = preprocess(tweet_3)
predictions = loaded_model.predict([tweet])
print(tweet_3, decode(predictions) )

tweet_4 = 'Covid vaccine uses mrna technology to combat the virus.'
tweet = preprocess(tweet_4)
predictions = loaded_model.predict([tweet])
print(tweet_4, decode(predictions) )
```

The vaccine is bad, do not take it. Opinionated
The vaccine is good for everyone even though it hurts badly, please take it. Opinionated
Covid vaccine is available for everyone to take for free. Neutral
Covid vaccine uses mrna technology to combat the virus. Neutral

```
# Make predictions using the Loaded model
tweet_1 = 'The vaccine is bad, do not take it.'
tweet = preprocess(tweet_1)
predictions = loaded_model.predict([tweet])
print(tweet_1, decode(predictions) )

tweet_2 = 'The vaccine is good for everyone even though it hurts badly.'
tweet = preprocess(tweet_2)
predictions = loaded_model.predict([tweet])
print(tweet_2, decode(predictions) )
```

The vaccine is bad, do not take it. Negative
The vaccine is good for everyone even though it hurts badly. Positive

Figure 27: Code for Random Sample Testing

The machine learning models are able to classify the random samples well. For the sample, “The vaccine is good for everyone even though it hurts badly.”, the presence of "hurts badly" did not trick the polarity classifier into classifying the tweet as negative. For the sample, “Covid vaccine is available for everyone to take for free.” is also classified correctly as neutral despite “free” generally being associated with positive. The usage of (unigrams and bigrams) as inputs helped to represent sentiments more accurately.

6. Submission Links

A YouTube link to a video presentation:

<https://www.youtube.com/watch?v=QluaayQfO78>

Link to Github for submission:

<https://github.com/liya0017/CZ4034-Group-9>

7. References

Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. BERTweet: A pre-trained language model for English Tweets. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14, Online. Association for Computational Linguistics.