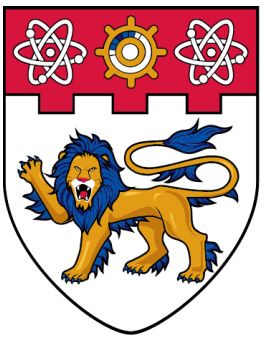


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CZ4034/SC4021 Information Retrieval

Assignment Report

Group 4

AY 2023/2024 Semester 2

Name	Matric. No
Saravanabavan Varsha	U2020628H
Cheam Zhong Wei Caleb	U2021423G
Huang Wei	U2020746H
Darren Chew	U2121505H
Banerjee Tanya	U2023315L
Liew Yew Loong, Jefferson	U2021690C

Table of Contents

1. Introduction

1.1 Background

Cryptocurrency is a digital or virtual form of currency that uses cryptography for security and operates independently of a central authority, such as a government or financial institution. Bitcoin, established in 2009, stands as the foremost cryptocurrency and remains widely recognized today. The creation of cryptocurrencies such as Bitcoin involves a process known as mining, where sophisticated computers solve intricate mathematical problems to authenticate and safeguard transactions on the blockchain. Utilising blockchain technology, which acts as a decentralised ledger recording all transactions across a network of computers, cryptocurrencies ensure transparency and security in financial transactions. The adoption of cryptocurrency has surged among global investors. With advancements in technology and industrialization, digital currencies, notably Bitcoin, are gaining prominence. Cryptocurrency facilitates seamless money transfers without reliance on traditional banking systems or financial intermediaries.

1.2 Objective

The main goal of this project is to provide users with a powerful information retrieval system to navigate the world of cryptocurrencies. By searching through a wide range of Reddit posts, the system helps users explore current sentiments, trends, and discussions about specific cryptocurrencies or broader industry topics. It offers a user-friendly interface that presents search results clearly, providing valuable insights into the sentiments expressed within the cryptocurrency community. Users can gain a better understanding of market dynamics, investor sentiments, and potential investment opportunities through the sentiment analysis results presented.

1.3 Links

Youtube:
https://www.youtube.com/watch?v=FPIRD_AYWkM
Github containing all the source code and datasets:
https://github.com/notvarsha/cz4034-information-retrieval

2. Crawling

2.1 Corpus

This section depicts the method to crawl the data. Web Crawling was done using the Python Reddit API Wrapper (PRAW), and the search was narrowed down to a certain few subreddits. The method to call the API is stated in Figure 2.1.1.

```
import praw
# Initialize the Reddit API client
reddit = praw.Reddit(
    client_id='Your Client ID',
    client_secret='Your Client Secret',
    user_agent='redditdev scraper and bot by u/redditor',
    username="Your Username",
    password="Your Password",
)
```

Figure 2.1 API Call

Each document mainly consists of 1 Post, 1 comment and 1 comment reply. To acquire each document, for each subreddit, the query is limited to the top 20 posts, top-level comments of each post and the 1 respective reply to each top-level comment. This allows for a wide variety of perspectives, avoiding iterative and similar information being pulled. These documents are pulled from cryptocurrency-relevant subreddits as shown in Table 2.1.2. The columns generated from the API calls are PostID, PostURL, Post Title, Post Content, Post Author, Post Upvotes, Post Time, Post Comments, Comment Author, Comment Upvotes, Comment Time, Comment Reply, Reply Author, Reply Upvotes and Reply Time.

CryptoScam	CryptoScams2023	CryptoScams
CryptoScamAwareness	CryptoScamReport	CryptoScamBlacklist
CryptoScamChannels	CryptoCurrency	CryptocurrencyICO
cryptocurrencymemes	CryptoCurrencyMoons	CryptocurrencyReviews
CryptoCurrencyMeta	CryptoCurrencyClassic	CryptoCurrencyTrading
CryptoScammerAbuse	CryptoScamTalk	

Table 2.1 Scrapped Subreddits

2.2 Question 1.2

Use cases involve searching for scams revolving around cryptocurrency such as scams and ponzi schemes so as to avoid those traps. Others can also do fundamental analysis about a few certain cryptocurrencies that they may buy into.

Some potential queries may be:

1) “Should I buy dogecoin?”
2) “Is dogecoin a scam?”
3) “Is Ethereum good long term?”
4) “Is Turtlecoin pump and dump?”

Relevant Reddit posts and comments will then be returned to the user to allow them to make informed decisions before trying to invest.

2.3 Question 1.3

Data Cleaning was made to filter out Reddit Bots and Moderator messages which can directly affect the semantics of the post. Hence, comments and posts created by bots and moderators are removed from the dataset. Emojis were converted to text using a Python library “Demoji”. One example would be converting a smiling emoji

unicode to ":smiling:". Finally, microtext normalisation was performed on acronyms such as "LOL" to laugh out loud or "FOMO" to fear of missing out. To implement this, each word was lowercase for easier micro-text normalisation. Microtext normalisation was referenced from Reddit - Dive into anything along with crypto.com. Table 2.1 indicates the number of words used

Type of Statistic	Count
Total number of documents	37012
Total number of words	6066525
Total number of unique words	37749
Total number of subreddits	17

Table 2.2 Corpus Statistics

3. Indexing

In our project, we used Apache Solr, an acronym for "Searching On Lucene with Replication", which is a free, open-source search engine built upon the Apache Lucene library. It serves as a comprehensive solution for performing efficient search operations and is renowned for its versatility and scalability. Furthermore, Solr also serves as a document-based NoSQL database with transactional support for storage purposes. Primarily written in Java, Solr exposes RESTful APIs in XML/HTTP and JSON formats, along with client libraries for various programming languages, facilitating seamless integration with diverse applications and environments.

Lucene is a high-performance full-featured text search engine library, renowned for its indexing and searching capabilities for text-based data. It offers features such as Boolean queries, fuzzy searches based on edit distance, wildcard searches, phrase queries and more. Lucene can also be used for recommendation systems as well. For instance, Lucene's "MoreLikeThis" class utilises a term vector-based similarity approach to generate recommendations for similar documents, augmenting its utility in diverse use cases.

3.1 Solr Indexing

After crawling and preprocessing the Reddit data, we uploaded the corpus into Solr for indexing. The schema for our data is shown in Figure 3.1. Several tokenizers and filters were added to Solr’s schema configuration to preprocess and index the text data more effectively for search and retrieval. It helps to improve search relevance by normalising and enhancing the text before indexing. The filters applied for indexing and query in this project are shown in Figure 3.2.

```
<field name="CommentAuthor" type="text_general"/>
<field name="CommentReply" type="text_general"/>
<field name="CommentTime" type="pdates"/>
<field name="CommentUpvotes" type="plongs"/>
<field name="PostAuthor" type="text_general"/>
<field name="PostComments" type="text_general"/>
<field name="PostContent" type="text_general"/>
<field name="PostID" type="text_general"/>
<field name="PostTime" type="pdates"/>
<field name="PostTitle" type="text_general"/>
<field name="PostURL" type="text_general"/>
<field name="PostUpvotes" type="plongs"/>
<field name="ReplyAuthor" type="text_general"/>
<field name="ReplyTime" type="pdates"/>
<field name="ReplyUpvotes" type="plongs"/>
<field name="Subreddit" type="text_general"/>
```

Figure 3.1 Schema of the Documents

```
<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" language="English"/>
  </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.txt"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.SnowballPorterFilterFactory" language="English"/>
  </analyzer>
</fieldType>
```

Figure 3.2 Tokenizers and Filters Applied

Tokenizers and Filters applied:

1. **StandardTokenizerFactory**: implements the Unicode Text Segmentation algorithm specified in Unicode Standard Annex #29, which includes rules for splitting text into words and punctuation. It handles punctuation marks, special characters and language-specific rules better, hence it will be more appropriate in this context as our corpus is mainly social media posts.

2. **StopFilterFactory**: used to remove common stop words from the text. Stopwords are words that are very common and do not contribute much to the meaning of the text, such as "and," "the," "of," etc.
3. **SynonymGraphFilterFactory**: used to expand synonyms in the text. It replaces synonyms with their equivalent terms, allowing for more comprehensive search results. For example, "car" might be replaced with "automobile" or "vehicle”.
4. **LowerCaseFilterFactory**: converts all text to lowercase. It's often used to ensure that text is normalised and to make searches case-insensitive.
5. **SnowballPorterFilterFactory**: applies stemming to the text. Stemming is the process of reducing words to their root or base form. For example, "buying" and "buys" would both be stemmed to "buy”.

3.2 Querying

After indexing, Solr can be used for querying. The terms from the user input will be converted into a Lucene query by a query parser to find the matching results. Solr supports various query parameters such as Filter Query (fq) and sort (ascending or descending),

Solr supports the usage of RESTful APIs for querying and retrieving documents based on the query term and parameters. To illustrate an example, Figure 3.3 shows the Query response and the first results obtained from the Solr database for the below query and filters set.

Query: Elon Musk bitcoin
Date Range: 01/01/2022 to 08/04/2024
Sort: Latest posts
Minimum Upvotes: 1000
Subreddits: Cryptocurrency, CryptoTechnology

```
  "responseHeader":{
    "status":0,
    "QTime":3,
    "params":{
      "q":"(PostContent:Elon musk bitcoin OR PostTitle:Elon musk bitcoin OR PostComments:Elon musk bitcoin OR
CommentReply:Elon musk bitcoin) AND (Subreddit:Cryptocurrency OR Subreddit:CryptocurrencyNews OR
Subreddit:CryptoTechnology) AND (PostTime:[2022-01-01T00:00:00Z TO 2024-04-08T23:59:59Z])",
      "indent":"true",
      "q.op":"OR",
      "fq":"PostUpvotes:[1000 TO *]",
      "sort":"PostTime desc",
      "rows":"1000",
      "wt":"json"}},
  "response":{"numFound":24,"start":0,"numFoundExact":true,"docs":[
      {
        "PostID":["tqa89o"],
        "PostURL":
["https://www.reddit.com/r/CryptoCurrency/comments/tqa89o/biden_administration_to_release_2023_budget_today/"],
        "Subreddit":["Cryptocurrency"],
        "PostTitle":["Biden Administration to release 2023 budget today including a new 20% billionaire tax"],
        "PostAuthor":["Yoshie5"],
        "PostUpvotes":[21293],
        "PostTime":["2022-03-28T22:13:00Z"],
        "PostComments":["unrealized gain tax"],
        "CommentAuthor":["Feeling_Ad_411"],
        "CommentUpvotes":[212],
        "CommentTime":["2022-03-28T22:18:00Z"],
        "CommentReply":["Yea it is the unrealized gain tax. It's a slippery slope and an all around terrible idea. If
they want to properly tax billionaires they should focus on removing loopholes and not allowing people to borrow money
against stocks. Edit: unrealized gains isn't real money. Elon musk doesn't have 200B in cash. He owns assets. If you
cannot understand that this brain dead idea is criminal at best you are a lost cause. It's a direct violation of the
constitution as it's the government seizing your assets."],
        "ReplyAuthor":["blindato1"],
        "ReplyUpvotes":[411],
        "ReplyTime":["2022-03-28T22:24:00Z"],
        "score":[0.5267],
        "sentiment":[1],
```


Figure 3.3 Query Response from Solr

The query term “Elon Musk bitcoin” is searched only in relevant columns such as PostContent, PostTitle, PostComments and CommentReply. Boolean operators such as OR and AND are used accordingly, i.e. OR is used to search the query term among all stated fields and subreddits and is concatenated using AND. The latest result that is returned is shown above.

3.3 Innovations

3.3.1 Categorical Search with Subreddits

In Reddit, posts are categorised by topics into user-created boards known as "Subreddits". As mentioned in Section 2, we have collected data from 17 specific subreddits for this project. To allow users to tailor their searches according to their preferences, we have introduced a feature that allows filtering results based on selected subreddits. Users can refine their searches to suit their needs, whether it be cryptocurrency news, scams, markets, or any other topic of interest.

For serious cryptocurrency buyers, they may select the CryptocurrencyMarkets subreddit to view the latest cryptocurrency prices, trends, and discussions. Additionally, for users interested in cryptocurrency trading strategies, analysis, and technical indicators, the subreddit CryptocurrencyTrading would be a valuable resource. Furthermore, cryptocurrency scams are prevalent across the globe, posing significant risks to investors. Many individuals, particularly newcomers to cryptocurrencies, may unknowingly fall victim to scams or encounter compromised accounts and may wish to know more from CryptoCurrencyScam subreddits. Access to relevant information directly from authentic sources on Reddit becomes even more crucial in such circumstances. As depicted in Figure 3.4, users have the option to select one or more subreddits, or they can choose the default option to search across all subreddits.

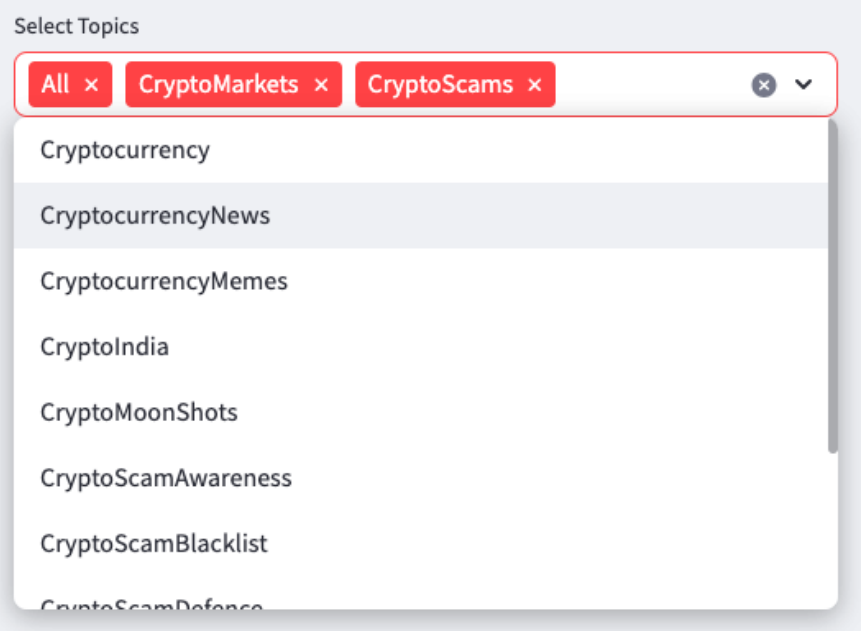


Figure 3.4 Dropdown box to select Subreddits

3.3.2 Timeline Search

For any information retrieval system, filtering by time windows is very crucial, especially in the field of cryptocurrency. Cryptocurrency markets operate 24/7 and are highly volatile, with prices fluctuating rapidly in response to various factors such as news events, regulatory announcements, and market sentiment. Cryptocurrency traders and investors often need to make timely decisions based on current market conditions. Filtering by time windows allows users to retrieve the most up-to-date information relevant to their queries and investors can quickly identify and analyse relevant information that may impact their trading strategies or investment decisions.

In addition to retrieving real-time data, time-based filtering also allows historical analysis of cryptocurrency markets. Users can retrieve posts from past time periods to conduct retrospective analysis, backtest trading strategies, or gain insights into the long-term trends and dynamics of specific cryptocurrencies. Figure 3.5 shows how users can adjust the start and end dates and sort by the latest or oldest posts.

Figure 3.5 Timeline search

3.3.3 Multimodal Search

We have implemented a feature that retrieves not only text-based Reddit posts but also those containing images, videos, polls, and other media formats. By retrieving a wide range of content formats beyond just text, users gain access to a diverse array of information and media related to cryptocurrencies. This includes images depicting price charts, infographics explaining complex concepts, videos showcasing tutorials or market analysis, and polls gauging community sentiment. This diversity enriches the user experience by providing multiple channels through which users can consume information.

Cryptocurrency markets are highly dynamic, and price movements can be better understood through visual representations such as charts and graphs. By retrieving image-based posts, users can access visual data representations that aid in understanding market trends, price movements, and other relevant information more effectively than plain text, as such Figure 3.6.

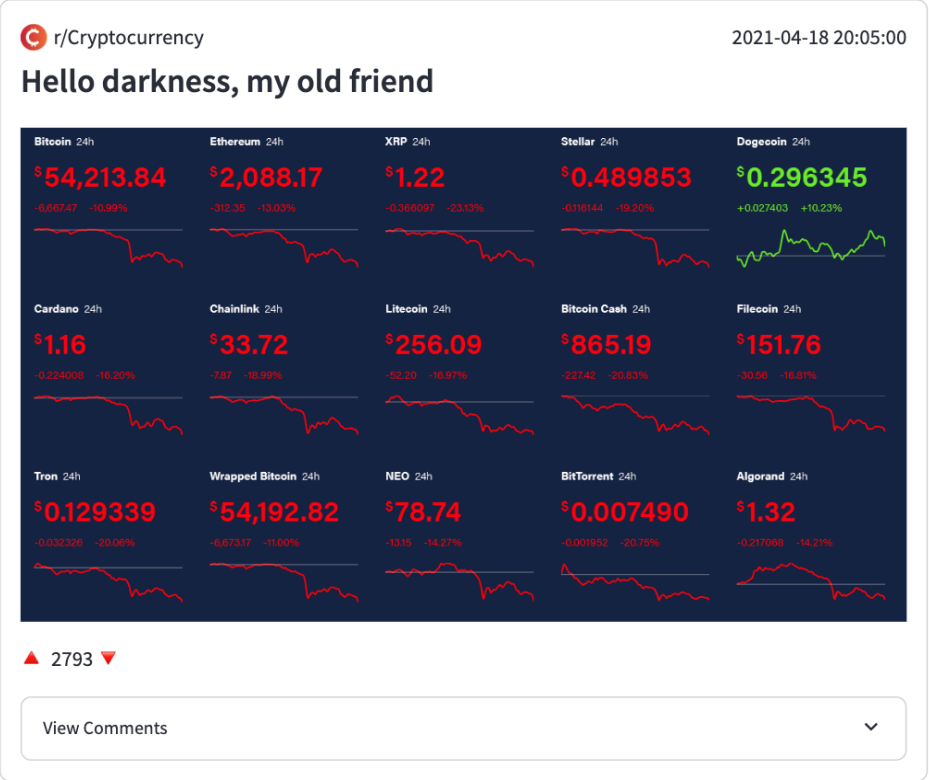


Figure 3.6 Reddit post showing the recent price changes of cryptocurrencies

Memes, gifs, and other visual content add an element of engagement and entertainment to the platform, such as the post in Figure 3.7. While serious discussions are important, incorporating lighter content formats can help maintain user interest and foster a sense of community among users.

Figure 3.7 Reddit post showing a meme on Bitcoin

3.3.4 Spell checks

We have implemented a SpellCheck component in Solr to provide inline query suggestions based on similar terms. Figure 3.8 illustrates the implementation of the DirectSolrSpellChecker using the solrconfig.xml file. This spell checker leverages terms directly from the Solr Index, eliminating the need to build a parallel index each time a query is processed. We have designated the 'PostComments' field for suggestions, as it contains the highest number of words.

The spell checker utilises the Levenshtein edit distance metric to calculate the difference between the query term and similar terms. The parameter maxEdits determines the maximum edit distance allowed between the query term and

suggested terms. Given that most spelling errors are typically only 1 character off, we have set maxEdits to 2 to ensure accurate suggestions.

```
<lst name="spellchecker">
  <str name="name">default</str>
  <str name="field">PostComments</str>
  <str name="classname">solr.DirectSolrSpellChecker</str>
  <!-- the spellcheck distance measure used, the default is the internal levenshtein -->
  <str name="distanceMeasure">internal</str>
  <!-- minimum accuracy needed to be considered a valid spellcheck suggestion -->
  <float name="accuracy">0.5</float>
  <!-- the maximum #edits we consider when enumerating terms: can be 1 or 2 -->
  <int name="maxEdits">2</int>
  <!-- the minimum shared prefix when enumerating terms -->
  <int name="minPrefix">1</int>
  <!-- maximum number of inspections per result. -->
  <int name="maxInspections">5</int>
  <!-- minimum length of a query term to be considered for correction -->
  <int name="minQueryLength">4</int>
  <!-- maximum threshold of documents a query term can appear to be considered for correction -->
  <float name="maxQueryFrequency">0.01</float>
  <!-- uncomment this to require suggestions to occur in 1% of the documents
    <float name="thresholdTokenFrequency">.01</float>
  -->
</lst>
```

Figure 3.8 Implementation of Spell Check in Solr

Figure 3.9 shows the JSON response from Solr when querying with a term that is misspelt. For example, “dogecoin”, a type of cryptocurrency may be misspelt by “dogcoin”. The response shows the frequency of the misspelt word in the PostComments index and suggests the top 5 words that are most similar.

```
{
  "responseHeader":{
    "status":0,
    "QTime":13},
  "response":{"numFound":0,"start":0,"numFoundExact":true,"docs":[]
  },
  "spellcheck":{
    "suggestions":[
      "dogcoin",{
        "numFound":6,
        "startOffset":0,
        "endOffset":7,
        "origFreq":0,
        "suggestion":[{
            "word":"dogecoin",
            "freq":264},
          {
            "word":"dogcoins",
            "freq":1},
          {
            "word":"dogecoins",
            "freq":12},
          {
            "word":"dodgecoin",
            "freq":7},
          {
            "word":"doggycoin",
            "freq":1},
          {
            "word":"dougecoin",
            "freq":1}]}],
    "correctlySpelled":false,
    "collations":[]}}
```


Figure 3.9 Response and Suggestions from Solr API

The top five suggestions for each misspelt word are integrated into our UI as shown in Figure 3.10. When a particular query returns less than 100 results or has no results, the suggestions will be presented to the user if they would like to do another search.

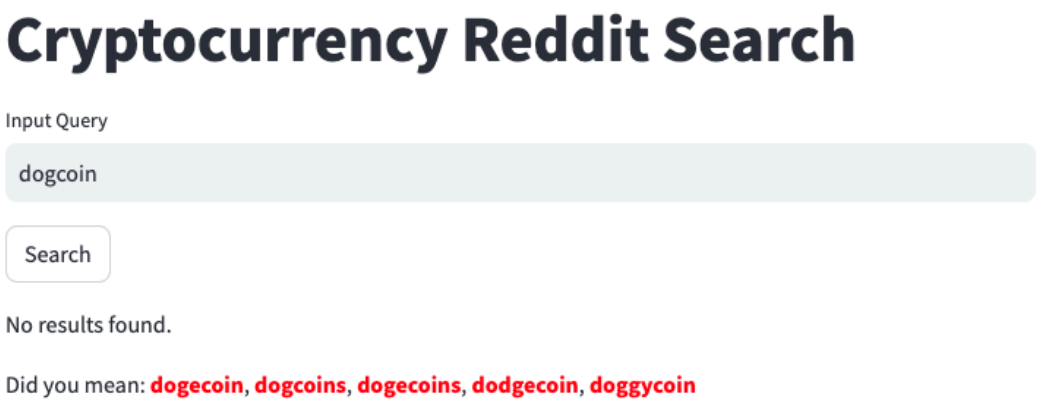


Figure 3.10 “Did you mean” Suggestions on the UI

3.3.5 Word Cloud

Word clouds are visual representations of text data, where the size of each word corresponds to its frequency or importance in the text. They provide a quick and intuitive way to identify the most significant words or themes in a body of text. We have implemented a word cloud feature based on the results of each query.

At first glance, users are able to gauge the overall sentiments of a particular query results from looking at the word cloud generated. Positive sentiments may include words like "bullish," "success," or "profit," while negative sentiments may include words like "crash," "scam," or "loss."

Word clouds can also help identify emerging trends, influential figures or discussions in cryptocurrency. By examining the prominent words in the cloud, users can learn more about upcoming topics, such as new blockchain projects, regulatory developments, market trends, or technological innovations. This can inform decision-making processes, investment strategies, and research initiatives within the cryptocurrency space.

Lastly, cryptocurrency is a complex and technical field, with numerous concepts, terms, and jargon that may be unfamiliar to the average investor or enthusiast. Word clouds can help simplify and communicate complex concepts by visually highlighting key terms or phrases and their relative importance.

In Figure 3.11, the query “Russia Ukraine War” generates the word cloud below. From the headline news that Russian Soldiers surrendering to Ukraine will receive USD $45,000 worth of cryptocurrency, we are able to see relevant terms such as “surrender”, “ruble”, and “45k”. Another query search of “Should I buy dogecoin” generates the word cloud with terms such as “money”, “good”, and “bought” which may guide users to form an informed decision.

Russia Ukraine War

Should I buy dogecoin?

Figure 3.11 Word Cloud generated for each Query

3.4 User Interface

We have built our UI in Python using the Streamlit Library, which is an open-source Python framework built for data visualisation. Our user-friendly interface prioritises easy access to results and offers multiple filters for sorting and organising posts. Figure 3.12 shows the start up page of our UI.

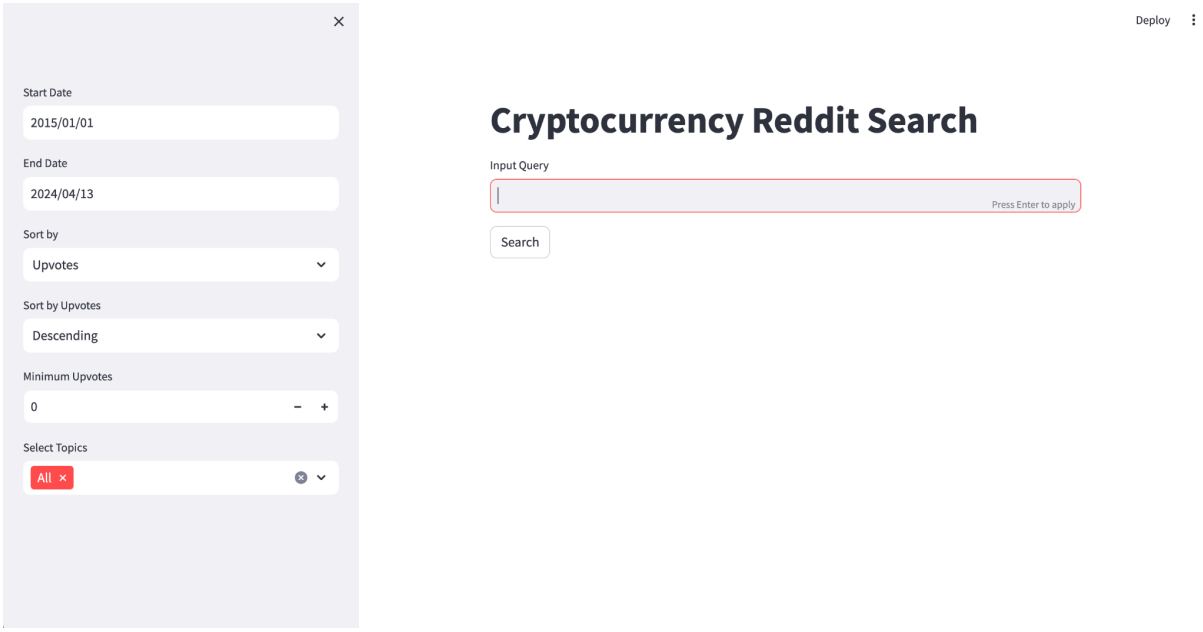


Figure 3.12 Main page of the UI

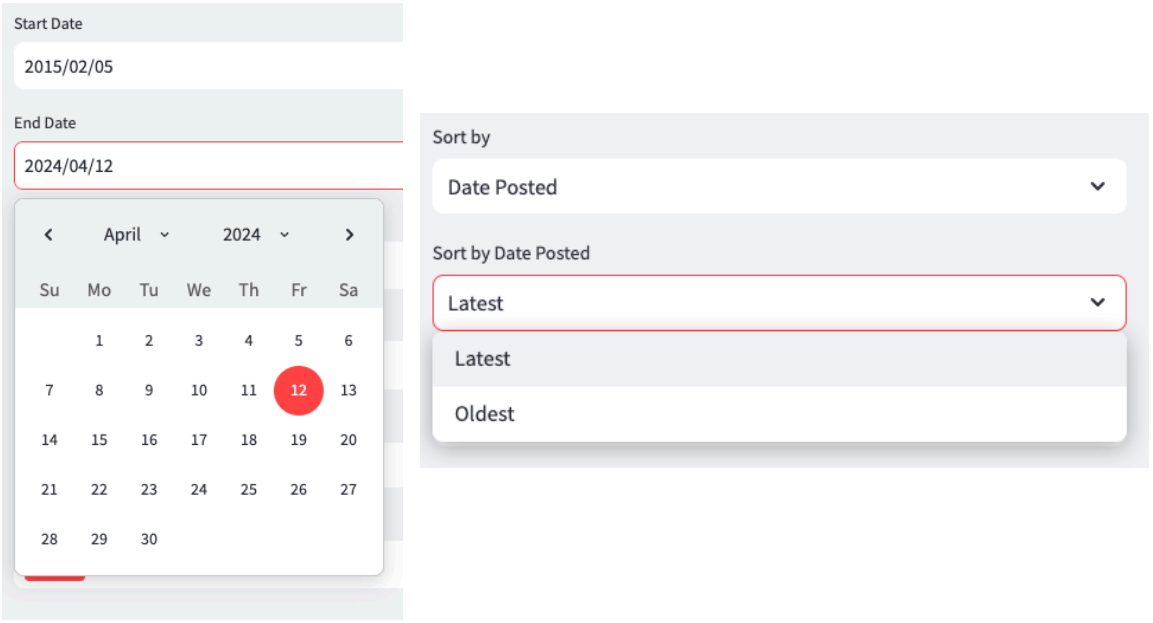


Figure 3.13 Sort and Filter by Dates

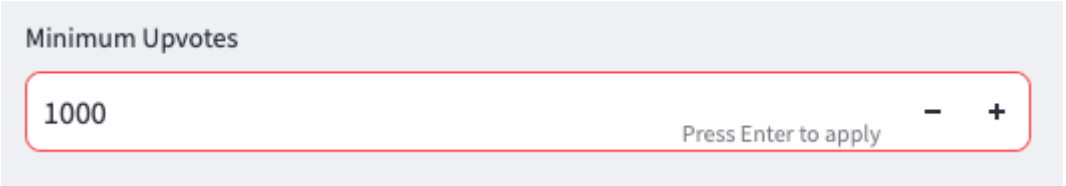


Figure 3.14 Minimum upvotes filter

The sidebar includes filters for date, upvotes, and subreddit selection, allowing users to specify their preferences as seen in Figures 3.13 and 3.14. Users can set date ranges, sort posts by date or upvotes, and further refine sorting options by specifying whether to display posts in the latest/oldest order or by descending/ascending upvote count. Additionally, users can filter posts based on a minimum number of upvotes, and as mentioned previously in the innovations section, users can also select subreddits to their preference, or the default will be all subreddits.

Upon inputting the query term in the search bar, users will be presented with two tabs. The results tab will display all the posts returned in the order of their selection. Each post will display the title, content (text, image or video), number of upvotes, subreddit posted in and data posted. To view the comments under each post, users can expand the dropdown below as seen in Figure 3.15. We believe that comments are most significant as they reflect the true opinions of the general population.

r/Cryptocurrency 2022-03-12 21:46:00

Ukraine says it has spent the nearly $100 million in crypto donations it has received to buy bulletproof jackets, helmets, food and more.

https://www.coindesk.com/policy/2022/03/11/ukraine-details-what-crypto-donations-are-being-spent-on/?fbclid=IwAR0nN5H4PHAhqpVLSD93BdeEpej0Y8-1ed3sDZQSsdBGfO_uRDuj_vk9N5w

▲ 21144 ▼

View Comments

r/Cryptocurrency 2021-06-02 15:02:00

The King of HODL!!!!

▲ 3357 ▼

View Comments

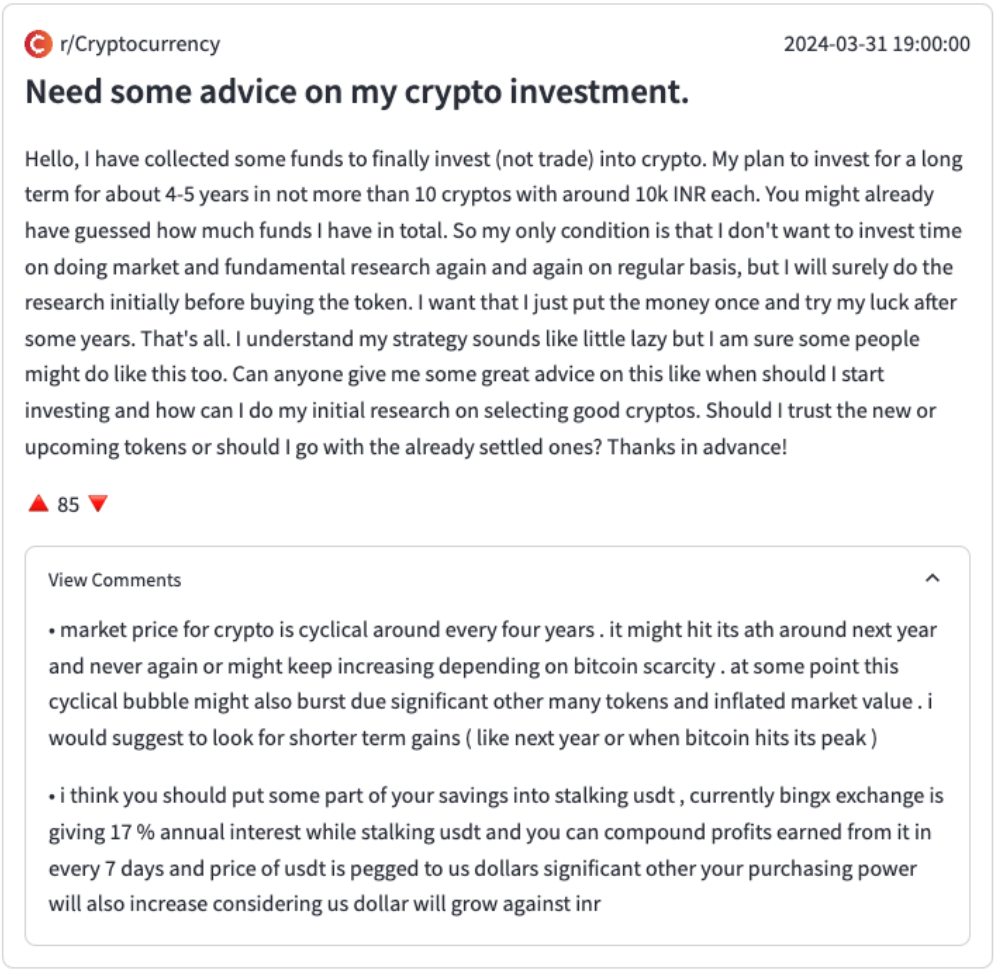


Figure 3.15 Sample posts retrieved for queries

The statistics tab presents the sentiment analysis and word cloud generated for each query as seen in Figure 3.16. Sentiments of each query based on the posts and comments are displayed in a pie chart, with visually easy-to-read colours. Red represents negative sentiments, green for positive and blue for neutral. The word cloud is a visual representation of the most significant terms in the results generated.

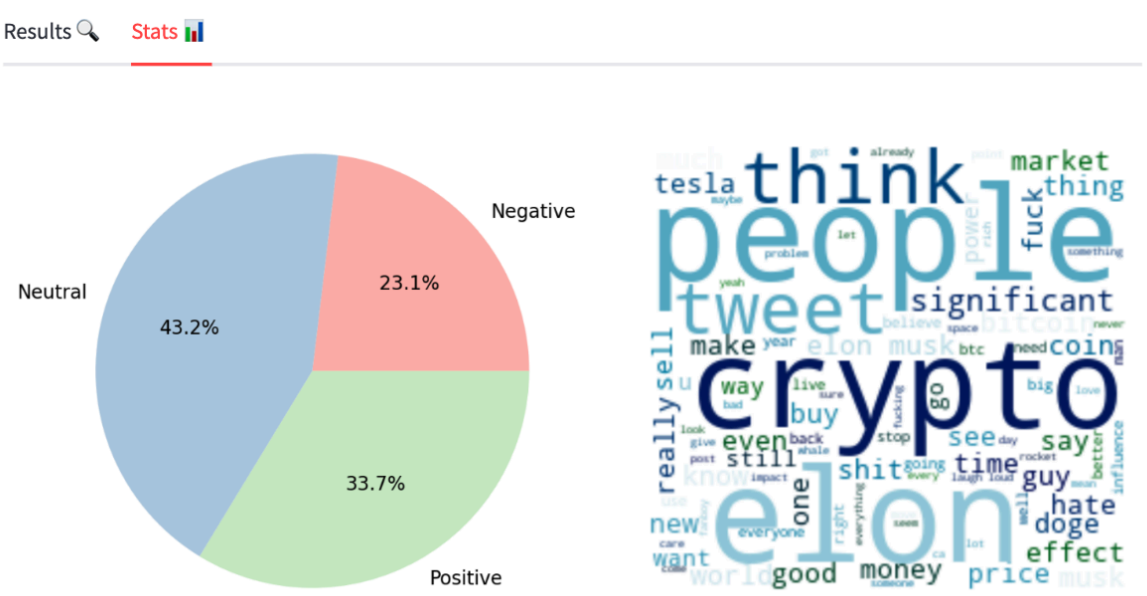


Figure 3.16 Statistics tab showing Sentiments pie chart and Word Cloud

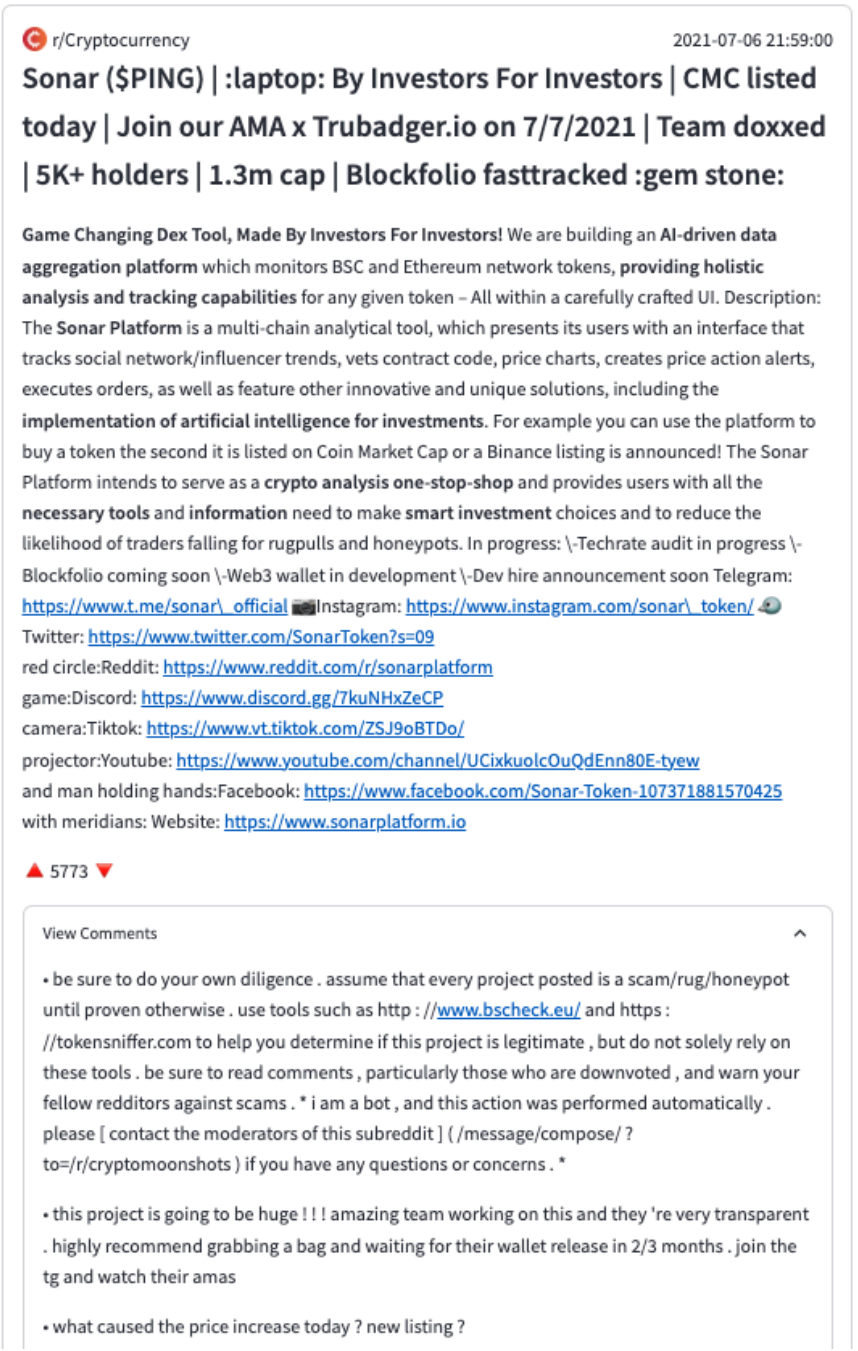
3.4 Sample Queries

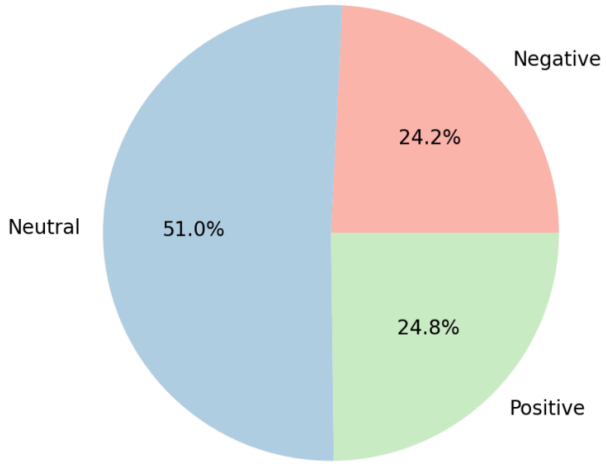
3.4.1 Query 1: “Crypto investment alerts”

URL:http://localhost:8983/solr/new_core/select?q=%28PostContent%3Acrypto+investment+alerts+OR+PostTitle%3Acrypto+investment+alerts+OR+PostComments%3Acrypto+investment+alerts+OR+CommentReply%3Acrypto+investment+alerts%29+AND+%28PostTime%3A%5B2015-01-01T00%3A00%3A00Z+TO+2024-04-12T23%3A59%3A59Z%5D%29&q.op=OR&rows=1000&wt=json&fq=PostUpvotes%3A%5B5000+TO+%2A%5D&sort=PostUpvotes+asc

Speed: 5ms

Top Post:




Negative
24.2%
Neutral
51.0%
24.8%
Positive


node network
people
btc
see
use
back
take
got
even
well
one
say
right
still
energy
transaction
point
price
significant
year
want
sure
coin
nano
make
fee
mean
crypto
wallet
know
buy
scam
post
think
good
much
time
need
work
cost
look

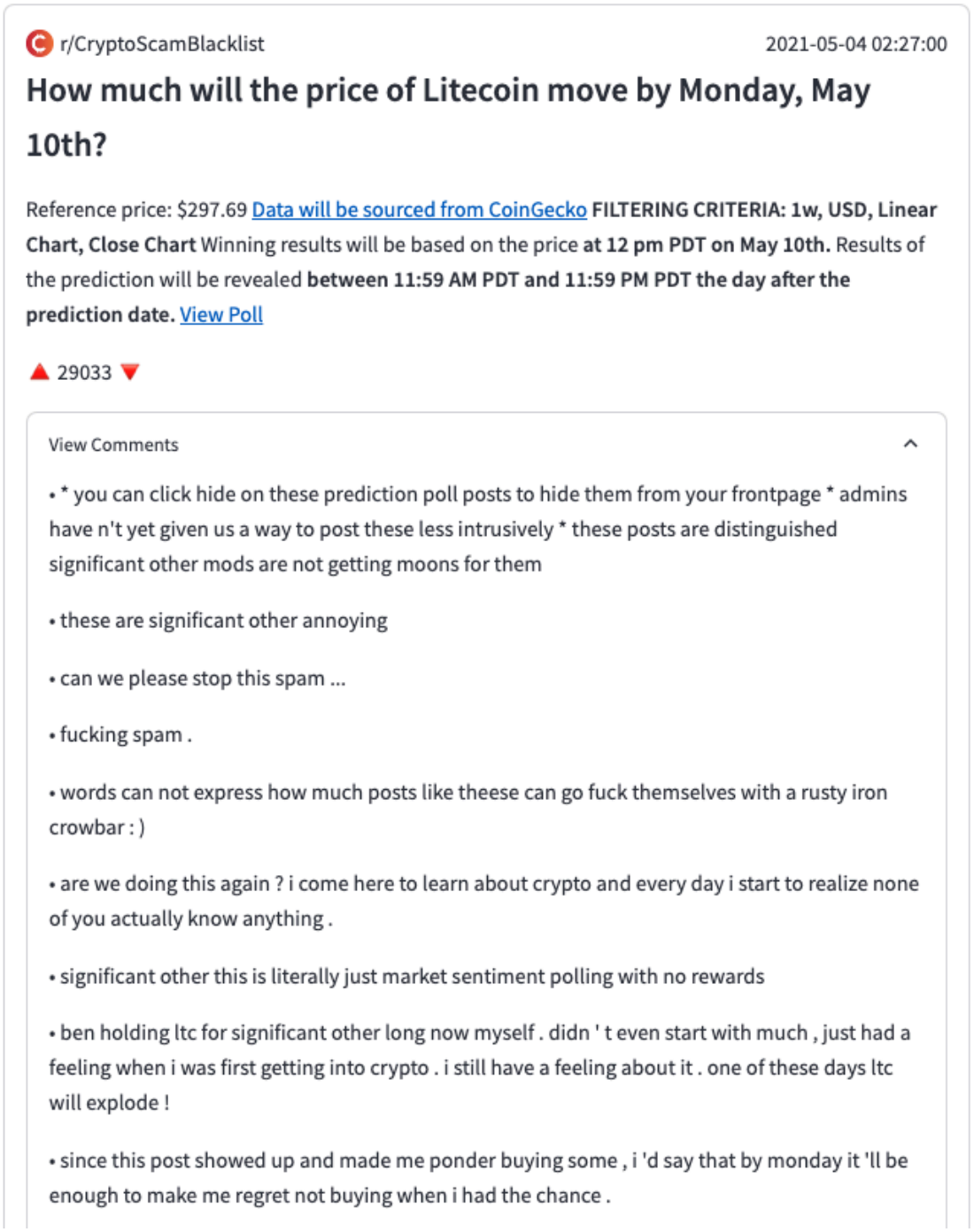
3.4.2 Query 2: “Litecoin price”

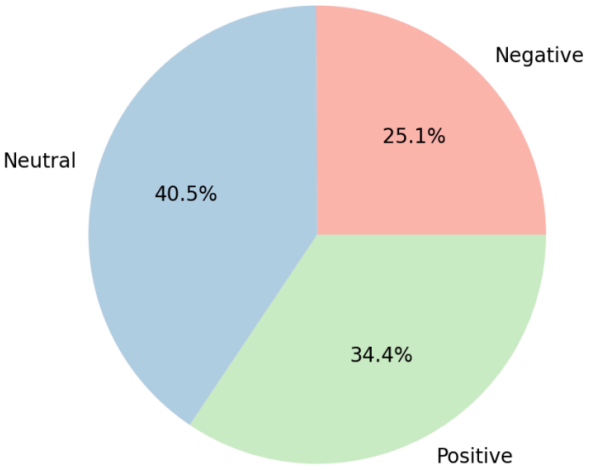
URL:

http://localhost:8983/solr/new_core/select?q=%28PostContent%3Alitecoin+price+OR+PostTitle%3Alitecoin+price+OR+PostComments%3Alitecoin+price+OR+CommentReply%3Alitecoin+price%29+AND+%28PostTime%3A%5B2015-01-01T00%3A00%3A00Z+TO+2024-04-12T23%3A59%3A59Z%5D%29&q.op=OR&rows=1000&wt=json&fq=PostUpvotes%3A%5B0+TO+%2A%5D&sort=PostUpvotes+desc

Speed: 3 ms

Top Post:




Negative
25.1%
Neutral
40.5%
34.4%
Positive

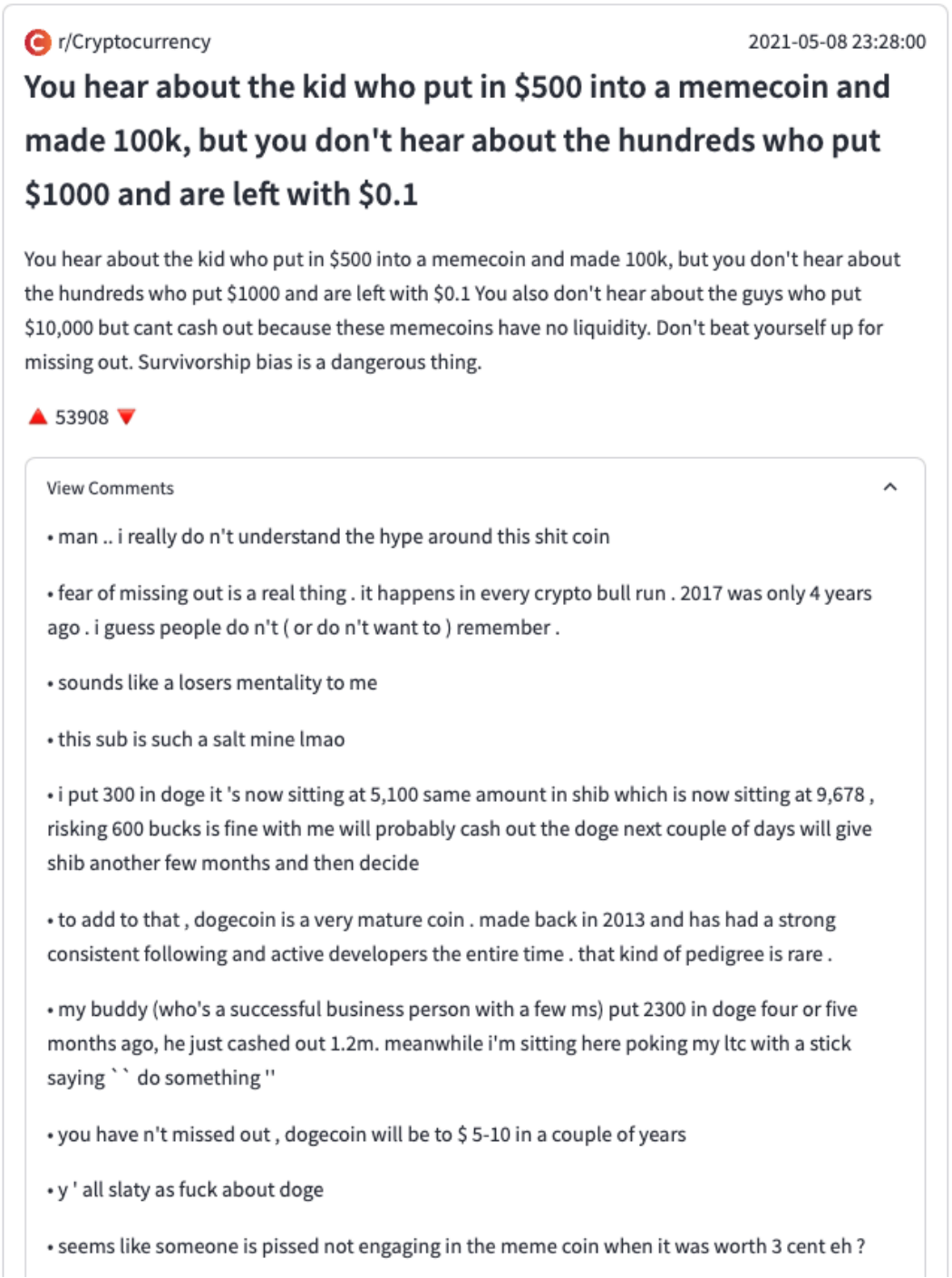

significant
post
made
fuck
time
seem
even
thing
btc
day
one
buy
prediction
really
know
kwh
poll
got
hide
coin
much
bitcoin
well
laugh
loud
good
right
going
mod
use
us
cryptomoon
price
data
back
face
make
see
people
ltc
money
doge
think
point
go
https

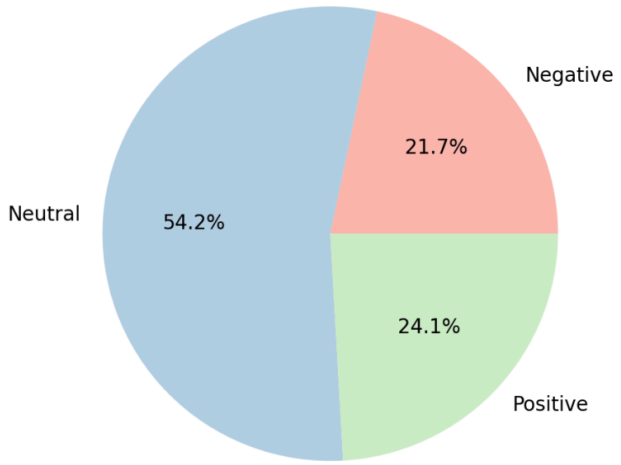
3.4.3 Query 3: “Is memecoin a scam?”

Speed: 6ms

Results:http://localhost:8983/solr/new_core/select?q=%28PostContent%3AIs+memecoin+a+scam%3F+OR+PostTitle%3AIs+memecoin+a+scam%3F+OR+PostComments%3AIs+memecoin+a+scam%3F+OR+CommentReply%3AIs+memecoin+a+scam%3F%29+AND+%28PostTime%3A%5B2015-04-02T00%3A00%3A00Z+TO+2024-04-13T23%3A59%3A59Z%5D%29&q.op=OR&rows=1000&wt=json&fq=PostUpvotes%3A%5B0+TO+%2A%5D&sort=PostUpvotes+desc

Top Post:




Negative
21.7%
Neutral
54.2%
24.1%
Positive

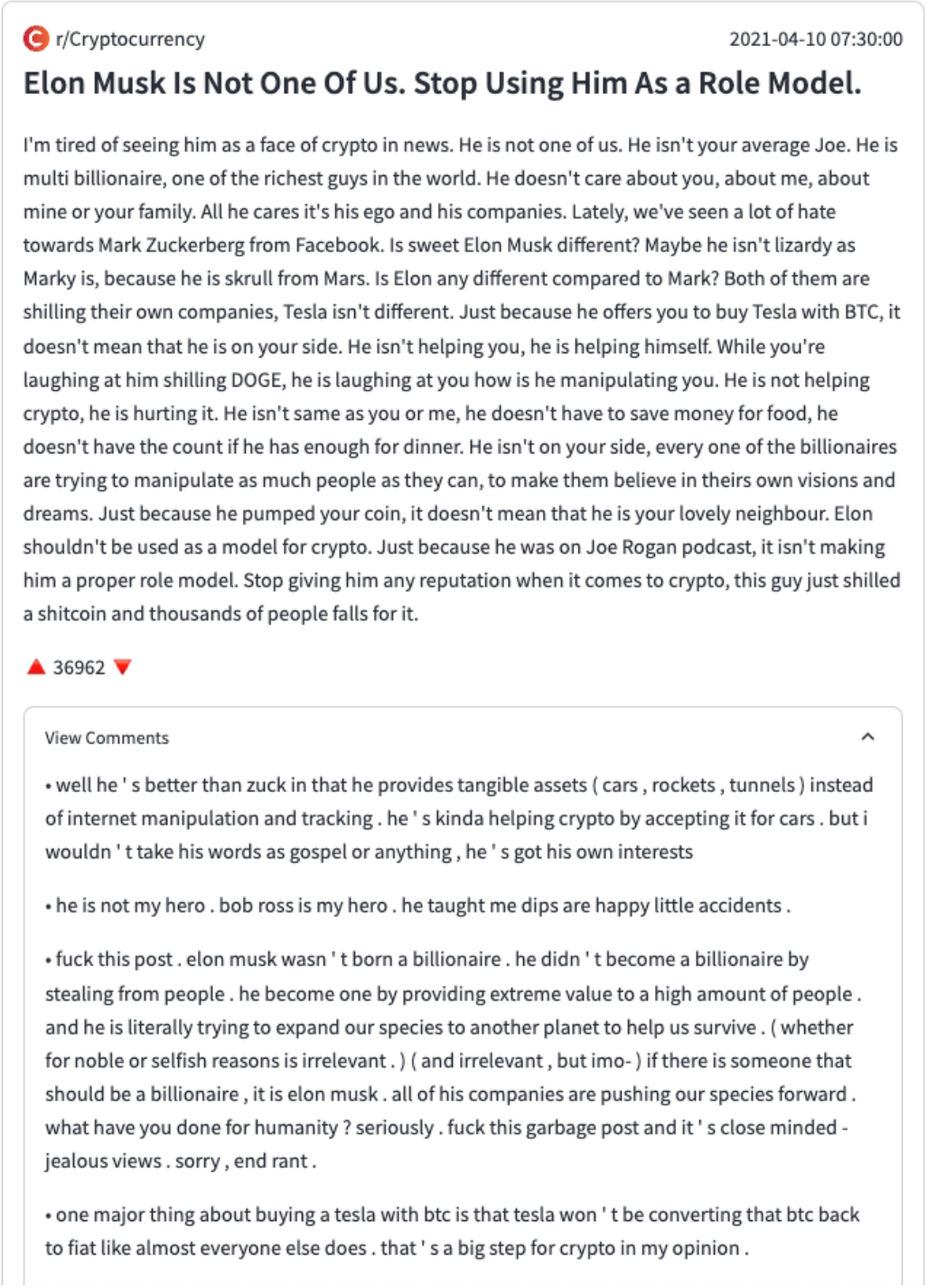

significant
lose
way
sell
back
guy
think
see
money
coin
still
even
dogecoin
bought
doge
make
time
every
gain
know
going
buy
hear
want
people
post
market
something
thing
one
made
got
put
crypto
year
much
right
good
sold
investment

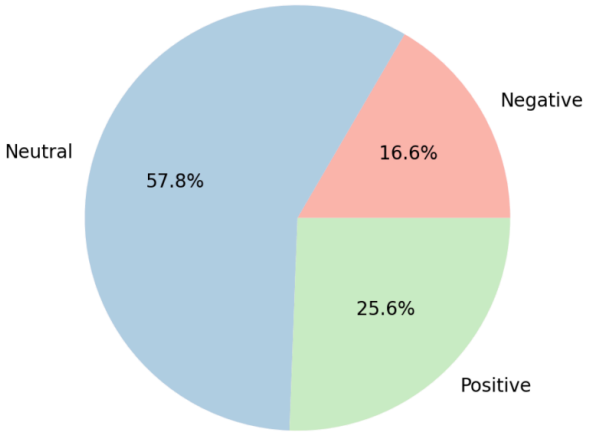
3.4.4 Query 4: “Tesla bitcoin holdings”

URL:http://localhost:8983/solr/new_core/select?q=%28PostContent%3ATesla+bitcoin+holdings+OR+PostTitle%3ATesla+bitcoin+holdings+OR+PostComments%3ATesla+bitcoin+holdings+OR+CommentReply%3ATesla+bitcoin+holdings%29+AND+%28PostTime%3A%5B2015-01-01T00%3A00%3A00Z+TO+2024-04-12T23%3A59%3A59Z%5D%29&q.op=OR&rows=1000&wt=json&fq=PostUpvotes%3A%5B0+TO+%2A%5D&sort=PostUpvotes+desc

Speed: 8 ms

Top Post:




Neutral
57.8%
Negative
16.6%
25.6%
Positive

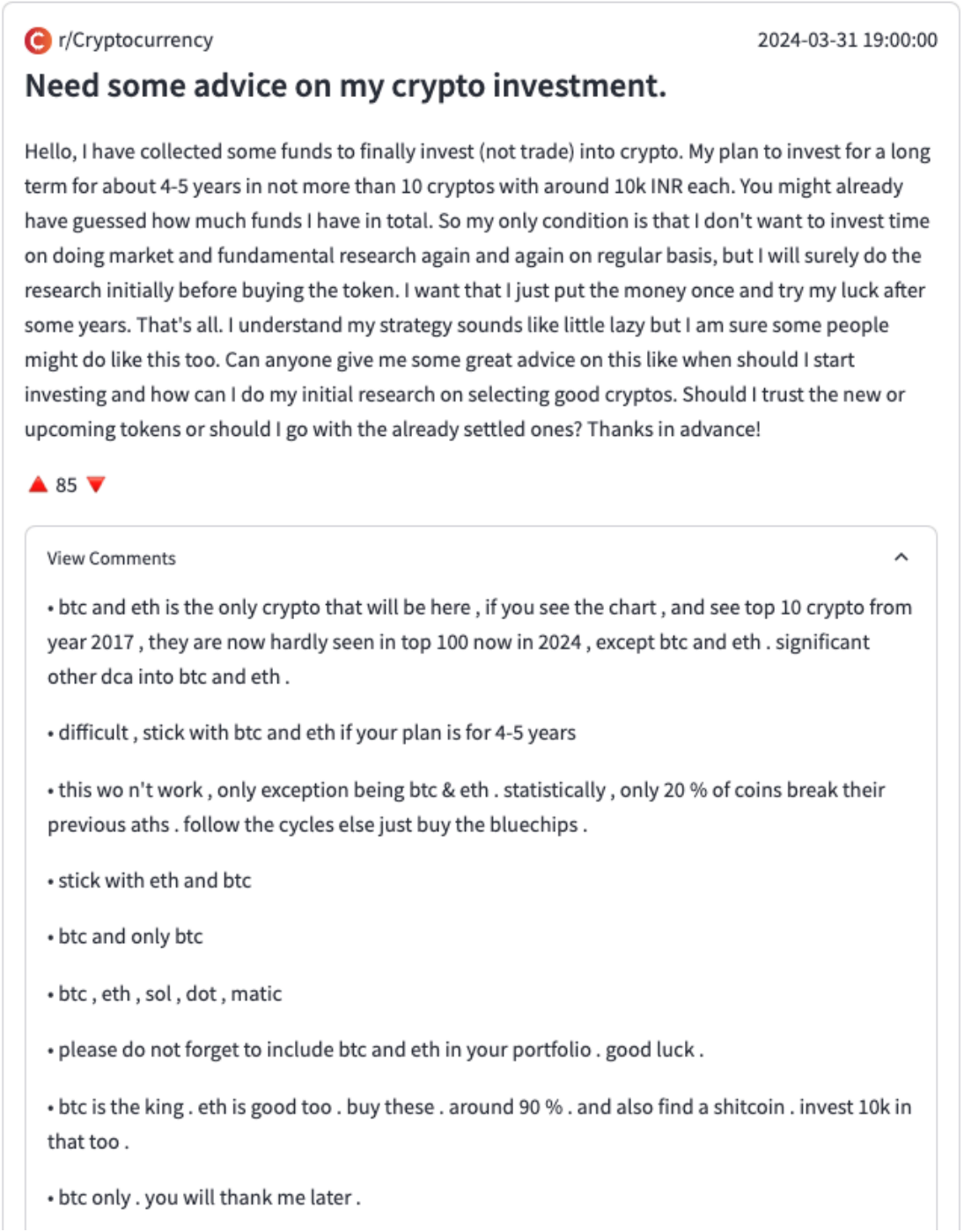

elon
people
crypto
significant
know
tesla
elon
musk
billionaire
fuck
world
thing
doge
good
way
rich
money
post
make
bitcoin
really
much
one
see
even
musk
care
guy
role model
think

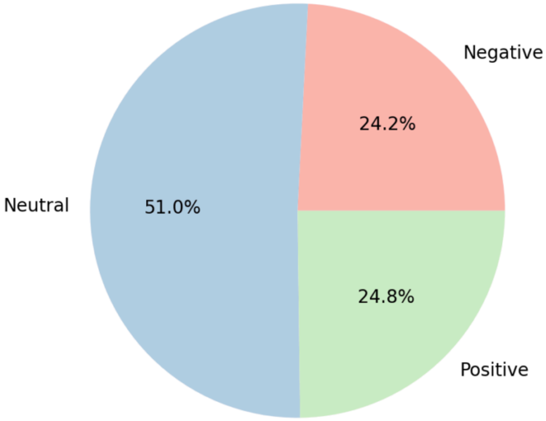
3.4.5 Query 5: “BTC vs ETH”

URL:http://localhost:8983/solr/new_core/select?q=%28PostContent%3Abtc+vs+eth+OR+PostTitle%3Abtc+vs+eth+OR+PostComments%3Abtc+vs+eth+OR+CommentReply%3Abtc+vs+eth%29+AND+%28PostTime%3A%5B2015-01-01T00%3A00%3A00Z+TO+2024-04-12T23%3A59%3A59Z%5D%29&q.op=OR&rows=1000&wt=json&fq=PostUpvotes%3A%5B0+TO+%2A%5D&sort=PostTime+desc

Speed: 6 ms

Top post:




Negative
24.2%
Neutral
51.0%
24.8%
Positive


node network
people
btc
see
use
back
take
market
got
even
well
one
right
still
say
transaction
price
point
significant
year
sure
want
coin
bitcoin
nano
make
fee
mean
crypto
money
know
wallet
buy
scam
way
think
post
cost
need
time
good
much
look
work

4. Classification

In the dynamic domain of cryptocurrencies, understanding public sentiment is crucial for both individual investors and financial analysts. The unstructured data available on forums such as Reddit provides a fertile ground for extracting insights that capture public opinion. However, manually navigating through this sea of information is logistically impractical. This is where the power of information retrieval, combined with advanced sentiment analysis techniques can be paramount for a system.

This portion aims to harness the capabilities of machine learning (ML) algorithms to infer the sentiment of comments scraped from cryptocurrency-related subreddits. By applying a range of classification algorithms, ranging from traditional ML models like Support Vector Machines (SVM) and Random Forests (RF) to more advanced Deep Learning (DL) architectures such as Roberta, we seek to categorise comments into sentiment classes (positive, negative, neutral). This automated sentiment analysis serves multiple purposes within the broader scope of information retrieval as stated in the table below.

Use Case	Description
Enhanced Search Efficiency	By tagging comments with their inferred sentiment, we can enable more nuanced search capabilities. Users can filter or prioritise search results based on the overall sentiment, making it easier to identify relevant discussions.
Trend Analysis	Sentiment trends over time can be an indicator of changing public opinion towards specific cryptocurrencies or the market in general. This can inform both short-term trading strategies and long-term market analysis.

Therefore, integrating sentiment analysis with information retrieval presents a powerful tool for navigating the complex landscape of cryptocurrency discussions on Reddit. The following sections will give a summary of our experiment results, machine learning pipeline, as well as additional mathematical enhancements to improve the performance of our classification algorithm.

4.1 Data preprocessing

The data scraped from various Reddit pages had columns such as Post ID, Post Comments, UpVote Counts, Comment Replies etc, with about 33000 rows, even though there were only about 200 unique posts. To prevent too many comments coming from duplicate posts and remove bias, we first limited our dataset to about 100 comments for each post, only taking those that had the highest interactions(sorted by Upvotes and Comment Replies).

Our text processing pipeline is meticulously crafted to convert raw Reddit comments into a clean, standardised format conducive to machine learning analysis. Recognizing the informal and diverse nature of Reddit discourse, especially within cryptocurrency communities, our pipeline integrates both conventional text preprocessing techniques and specific adjustments to address the unique elements of social media text. The following steps outline our comprehensive approach:

Case Folding: We begin by converting all text to lowercase to ensure uniformity and prevent the same words in different cases from being interpreted as distinct tokens.

Removal of URLs/links: URL links should be removed as it is not uncommon to find them in comments and they tend to provide little to no relevance for the sentimental analysis. Irrelevant links add unnecessary noise to the data and can potentially distract the sentiment of the content.

Removal of duplicates: Comments in Reddit are quoted and could appear multiple times when another individual replies to the original comment during the scrapping process. As such, these duplicated comments should be removed.

Removal of comments that no longer exist: It is not uncommon to find deleted and removed Reddit comments as they could have been offensive and removed from the moderators. When they are removed, the original text of the comment will be changed to ‘[removed]’ or ‘[deleted]’. These comments provide no relevance to the sentimental analysis and should be removed during the processing of text for sentiment analysis.

Removal of Accented Characters: Accented characters are normalised to their closest ASCII counterparts. This step helps in reducing the complexity of the dataset by consolidating variations of the same word that might only differ in accentuation.

Expanding Contractions: Given the informal nature of Reddit comments, contractions are prevalent. We systematically expand contractions (e.g., converting "can't" to "cannot", and "they're" to "they are") to improve the model's ability to understand and process these expressions.

Emoji Conversion: Recognizing the significance of emojis in conveying sentiment, we employ an emoji-to-text conversion process. This step translates emojis into corresponding descriptive texts, thereby preserving their emotional or contextual significance within the comment text.

Removal of Stopwords: Stopwords (e.g., "the", "is", "in") are removed from the text. While these words are essential for sentence structure, they offer little value in the context of sentiment analysis and can unnecessarily increase the dimensionality of the dataset.

Handling Special Characters and Punctuation: Non-alphanumeric characters, including punctuation, are removed. This cleanup is crucial for reducing noise in the data. However, special attention is paid to symbols that might convey sentiment (e.g., "!") to ensure their impact is not entirely lost in preprocessing.

Tokenization: The cleaned text is then tokenized, breaking it down into individual words or tokens. This step is fundamental for transforming the text into a format that machine learning algorithms can interpret and analyse.

Lemmatization: This process further reduces words to their base or root form, aiding in the consolidation of different forms of a word into a single representative token and can potentially improve the robustness of our TF-IDF matrix.

With the above steps that are carefully done in the context of our use case, we now have clean, formatted texts that are available for our machine learning models to train on.

4.1.1 Manually Labelled Sentiments

The evaluation data set that will be used to evaluate the accuracy, precision, recall, F1 score will be manually labelled. This requires a set of guidelines to follow for labelling the sentiments of each comment to ensure there is consistency in the comments being labelled in the 3 categories of positive, neutral and negative

The guidelines are:

1) **Reading and understanding the text:** The text has to be fully read to understand the context and tone of the text. If the text is broken down into segments, some parts could have a different sentiment but the overall sentiment of the entire text has to be considered.

2) **Identifying important sentiment keywords:** Certain words and phrases are more indicative of the sentiment of the text. For example, ‘happy’, ‘thanks’ are positive, ‘sad’, ‘shame’, ‘sucks’ carry a negative sentiment.

3) **Profanities:** Although profanities usually carry a negative sentiment, it is important to consider the overall context and noun following the profanity.

4) **Objective sentiment annotators:** At least 3 individuals participated in manually labelling the sentiments and the modal sentiment was taken as the final sentiment. This ensures objectivity in the sentiment and thus a more accurate reflection of the sentiment.

While these guidelines provide a clear and concise method to label the sentiments of the texts, it is also important to ensure that there is consistency in following the guidelines when labelling the sentiments.

4.1.2 Natural Language Toolkit(NLTK)

NLTK is an open source library on Python and it offers a platform for working with human language texts and data in natural language processing. It offers tools for processing, analysing, normalising data, tokenization, lemmantising, stemming, stopword removals and classification.

After processing the comments through the various processing methods, the data is analysed through NLTK's sentiment analysis. Each comment will be given a compound score after being analysed by NLTK's sentiment analysis. The compound score is a score that sums up and calculates the normalised lexicon ratings of the comment. Compound scores of more than 0.05 will return a positive sentiment, compound scores of less than -0.05 will return a negative sentiment, and compound scores of between -0.05 and 0.05 will return a neutral sentiment.

```
# get compound scores function
def get_compoundscore(text_data):
    sentiment_analyzer = SentimentIntensityAnalyzer()

    scores = sentiment_analyzer.polarity_scores(text_data)
    compoundscore = scores['compound']
    return compoundscore
```

Figure 4.1 GetCompoudScore Function

```
# get_sentiment function
def get_sentiment(text_data):
    sentiment_analyzer = SentimentIntensityAnalyzer()

    scores = sentiment_analyzer.polarity_scores(text_data)
    if (scores['compound'] >= 0.05):
        sentiment = 1
    elif (scores['compound'] <= -0.05):
        sentiment = -1
    else:
        sentiment = 0

    return sentiment
```

Figure 4.2 GetSentiment Function

While the sentiment received from NLTK's sentiment analysis provides a basic sentiment of the text, it lacks a deeper understanding of the nuances of the human language such as hidden humour, abbreviation, slang and sarcasm. It also suffers from the lack of machine and deep learning analysis to improve its analysis performance. Hence, the sentiment received will be used to feed the machine learning algorithm later on to yield a more accurate result for the sentiments from the comments.

4.2 Classification approaches

The following sections will discuss the implementations, experiments and discussion of results for various classification algorithms used.

4.2.1 Machine Learning Approach

Random Forest and Support Vector Machine (SVM) models have each demonstrated their strengths in the realm of text classification, contributing valuable tools for natural language processing tasks. For both aforementioned algorithms, we start off with the benchmark of using a simple pipeline using tf-idf vectorizer library to format our Reddit comments as seen below.

```
pipeline = pipeline = make_pipeline(TfidfVectorizer(), SVC(kernel='linear', probability=True))
```

We will discuss briefly why we chose the machine learning algorithms we used to classify sentiments along with the results.

4.2.1.1 Random Forest

Random Forest, a powerful ensemble learning method, operates by constructing a multitude of decision trees during the training phase and outputting the class that is the mode of the classes of the individual trees, with a loss function that tries to reduce class impurity [Equation 1]. Its ability to handle high-dimensional data makes it a suitable candidate for text classification tasks. A study by Fernandez-Delgado et al. (2014) found that Random Forests are among the best classification algorithms for a variety of datasets, highlighting their versatility and robustness. However, the performance of Random Forest can sometimes be eclipsed by the computational efficiency and effectiveness of other methods in high-dimensional spaces, typical of text data.

$$G(S,\theta) = \sum_{i=1}^{k} \frac{n_i}{N} H(S_i) \quad \text{.....Eq (1)}$$

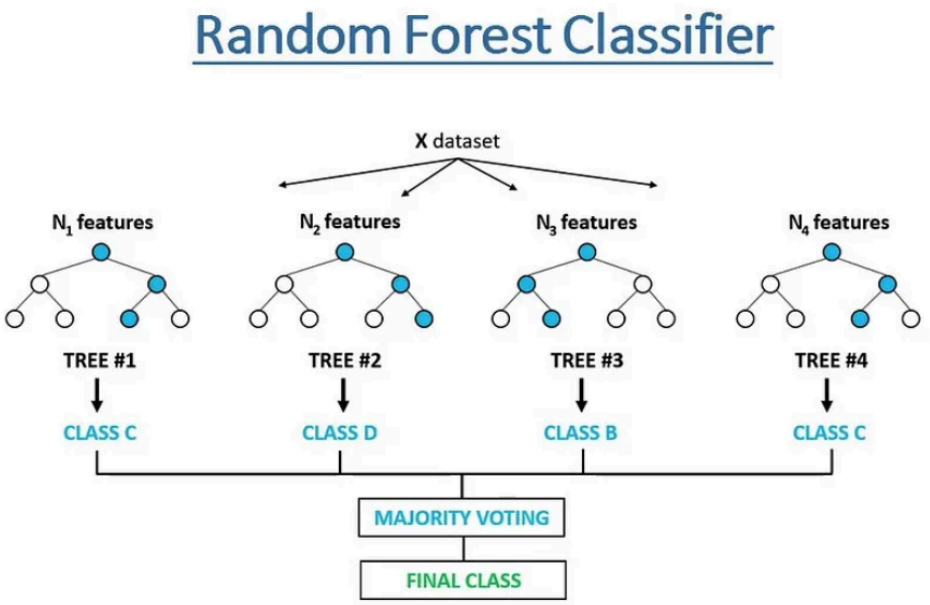


Figure 4.3 Random Forest Classifier Summary

Figure 4.3 summarises the random forest classifier, where the features are tf-idf scores for terms extracted from each document (1 Reddit comment). Figures 4.4 and 4.4 depict the results and confusion matrix obtained for Random Forest. It has achieved an accuracy of 0.755.

```
Accuracy: 0.7547058823529412
Precision: 0.768973790693774
Recall: 0.7547058823529412
F1 Score: 0.748754830271355
              precision    recall  f1-score   support

          -1       0.87      0.58      0.69       568
           0       0.57      0.63      0.60       228
           1       0.76      0.90      0.82       904

    accuracy                           0.75      1700
   macro avg       0.73      0.70      0.70      1700
weighted avg       0.77      0.75      0.75      1700
```

Figure 4.4 Random Forest Results

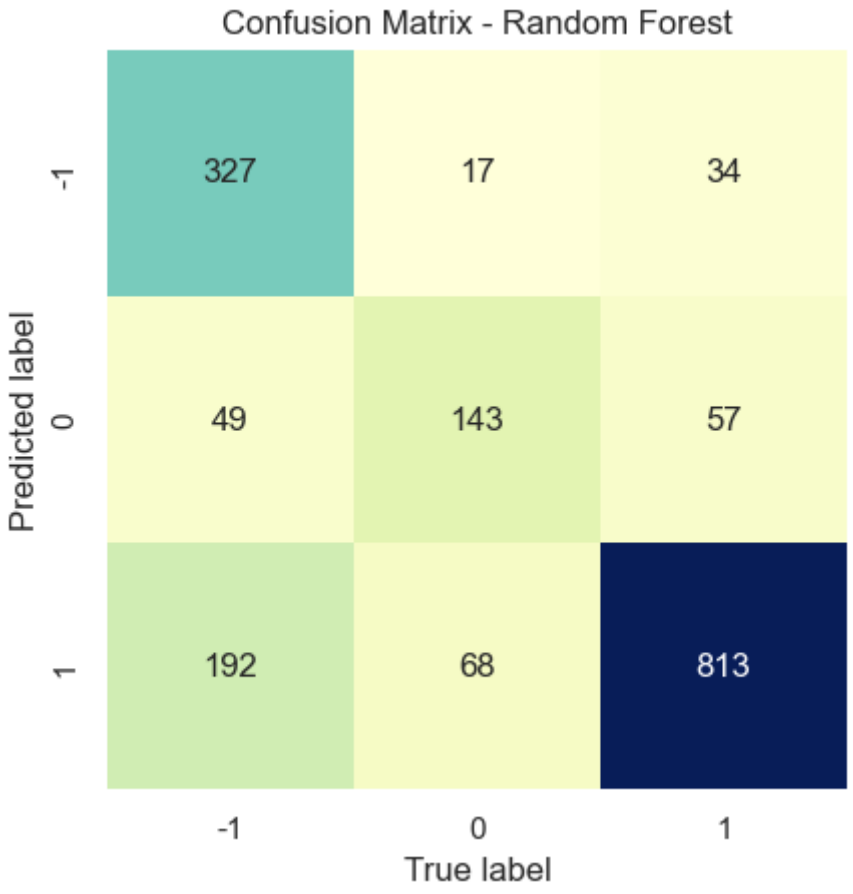


Figure 4.5 Random Forest Confusion Matrix

4.2.1.2 Support Vector Machines

On the other hand, Support Vector Machines (SVMs) are particularly well-suited for text classification due to their foundation in the principle of structural risk minimization, which aims to generalise well to unseen data. SVMs excel in high-dimensional spaces, making them ideal for dealing with sparse data sets, a common characteristic of text data. The efficacy of SVMs in text classification has been empirically supported by Joachims (1998), who demonstrated that SVMs perform exceptionally well on text categorization tasks compared to traditional methods, attributing this success to their ability to handle large feature spaces and their robustness to overfitting.

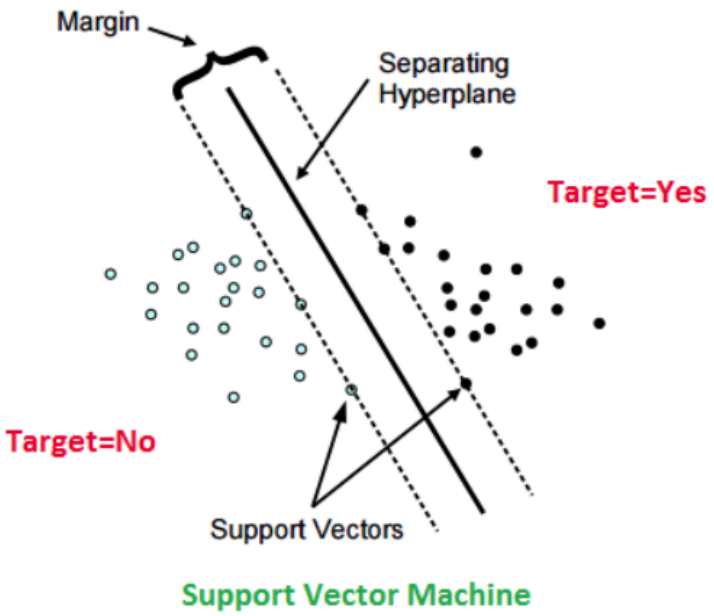


Figure 4.6 Support Vector Machine

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \quad \text{subject to } y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i, \xi_i \geq 0 \qquad \text{.....(2)}$$

Equation 2 represents the optimization problem for the SVM, aiming to minimise the norm of the weight vector w (thus maximising the margin) while controlling the penalty imposed by misclassifications. C is the penalty parameter, ξ i are the slack variables allowing margin violation, yi are the class labels, and x i are the feature vectors of the data points.

```
Accuracy: 0.7635294117647059
Precision: 0.763316121261703
Recall: 0.7635294117647059
F1 Score: 0.7480290775609876
              precision    recall  f1-score   support

          -1       0.82      0.68      0.74       568
           0       0.69      0.33      0.45       228
           1       0.75      0.92      0.83       904

    accuracy                           0.76      1700
   macro avg       0.75      0.65      0.67      1700
weighted avg       0.76      0.76      0.75      1700
```

Figure 4.7 The Overview of Results for Support Vector Machine

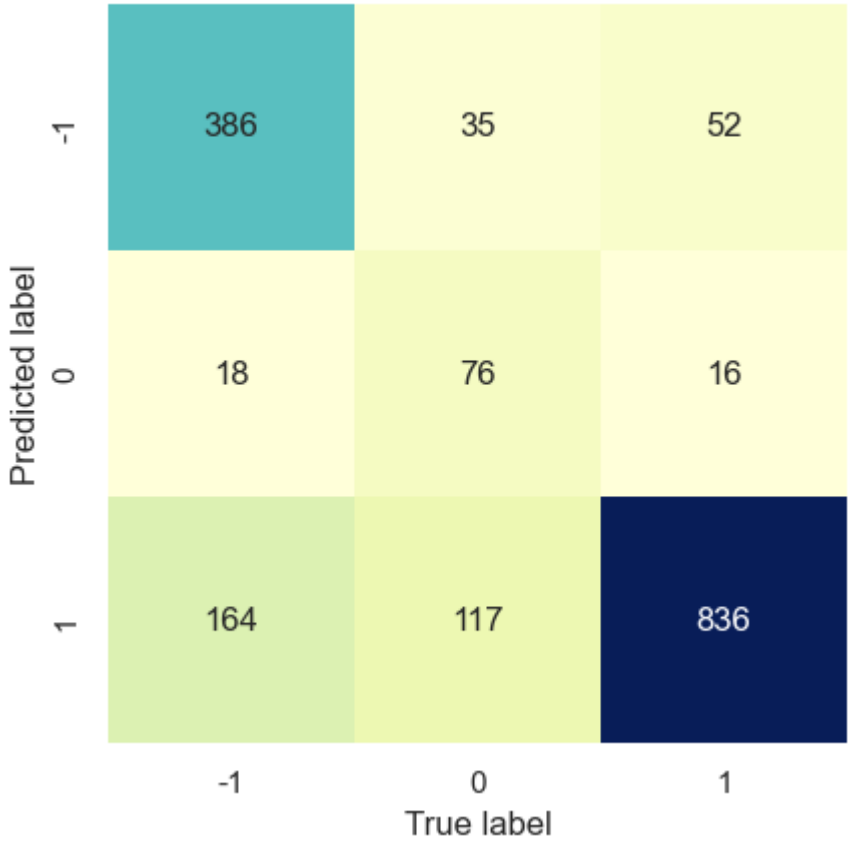


Figure 4.8 The Confusion Matrix Results for Support Vector Machine

Figures 4.6 and 4.7 depict the results and confusion matrix obtained for SVM. It has achieved an accuracy of 0.764.

As seen in the above experiments, SVM models often outperform Random Forest in text classification contexts. The superiority of SVMs can be attributed to their ability to find the optimal hyperplane that separates different classes in a high-dimensional space, which is crucial for handling the sparse and high-dimensional nature of textual data. Moreover, the kernel trick employed by SVMs allows them to efficiently handle nonlinear relationships within the text data, providing a significant advantage over Random Forest models that might struggle with non-linear separability without extensive tuning. Empirical evidence supporting the superiority of SVMs for text classification includes numerous studies and benchmarks where SVMs consistently achieve higher accuracy and F1 scores across diverse text datasets and classification tasks, as noted by Yang and Liu (1999) in their comprehensive comparison of text classification algorithms.

4.2.2 Deep Learning Approach

4.2.2.1 Roberta Model

RoBERTa, which stands for Robustly Optimised BERT Pretraining Approach, is a modification of the original BERT (Bidirectional Encoder Representations from Transformers) model, which itself revolutionised the field of natural language processing (NLP). BERT models, including RoBERTa, leverage the transformer architecture, which relies on attention mechanisms to capture the context of a word within its surrounding text.

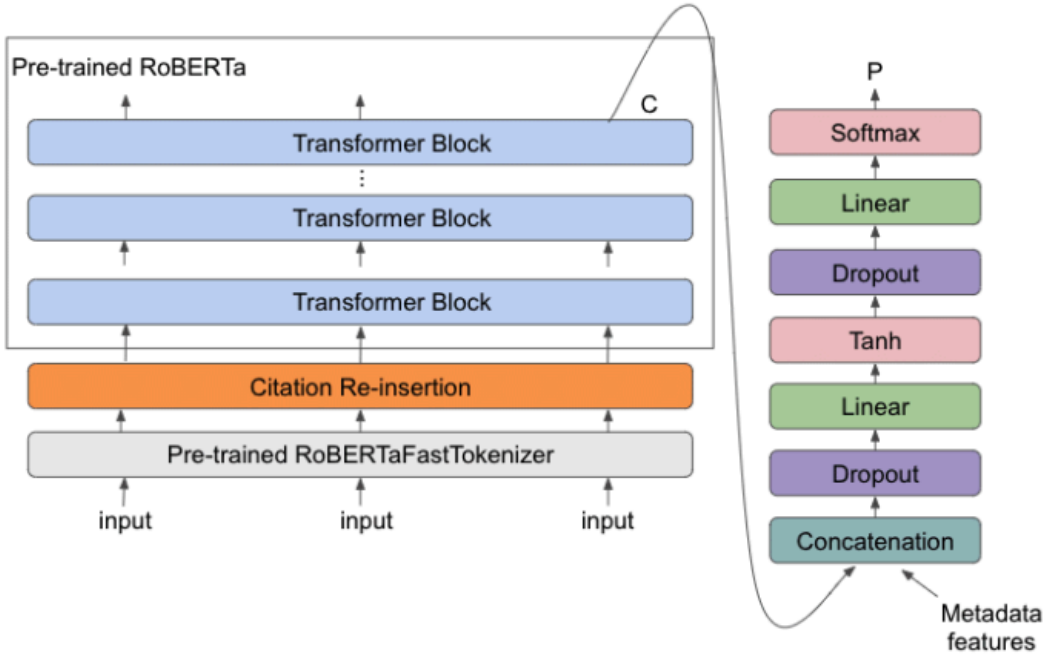


Figure 4.9 The RoBERTa model

RoBERTa, similar to BERT, is based on a multi-layer bidirectional transformer encoder. Each layer in Roberta is composed of two main subcomponents: a multi-head self-attention mechanism and a fully connected feed-forward network. This architecture allows the model to weigh the influence of different words within the sentence, enabling it to capture nuanced semantic relationships.

We decided to experiment our text classification task with this model due to its advantages over conventional deep learning models for NLP and even its predecessor, BERT. These advantages include improved contextual understanding

and dynamic masking all of which contribute to the robustness of the model in inferring unseen data.

```python
def tokenize(batch):
    return tokenizer(batch["ProcessedComments"], padding=True, truncation=True, max_length=512)

train_dataset = train_dataset.map(tokenize, batched=True)
test_dataset = test_dataset.map(tokenize, batched=True)


train_dataset = train_dataset.rename_column("sentiment", "labels")
test_dataset = test_dataset.rename_column("sentiment", "labels")
train_dataset.set_format("torch", columns=["input_ids", "attention_mask", "labels"])
test_dataset.set_format("torch", columns=["input_ids", "attention_mask", "labels"])


from transformers import DataCollatorWithPadding


data_collator = DataCollatorWithPadding(tokenizer=tokenizer, return_tensors="pt")


model = AutoModelForSequenceClassification.from_pretrained(model_checkpoint, num_labels=3)


training_args = TrainingArguments(
    output_dir="./results",
    num_train_epochs=5,
    per_device_train_batch_size=8,
    per_device_eval_batch_size=8,
    learning_rate=5e-5,
    evaluation_strategy="epoch",
    save_strategy="epoch",
    load_best_model_at_end=True,
)


def compute_metrics(p):
    preds = np.argmax(p.predictions, axis=1)
    labels = p.label_ids
    precision, recall, f1, _ = precision_recall_fscore_support(labels, preds, average='weighted')
    acc = accuracy_score(labels, preds)
    return {"accuracy": acc, "precision": precision, "recall": recall, "f1": f1}


trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
    tokenizer=tokenizer,
    data_collator=data_collator,
    compute_metrics=compute_metrics,
)


trainer.train()

```

Figure 4.10 Code Snippet of RoBERTa model

The following pipeline of 5 epochs was used to train the Roberta model with minimal hyperparameter tuning using Google Collabs free tier GPU.

Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1
1	0.720900	0.671445	0.800000	0.807693	0.800000	0.791637
2	0.428400	0.745634	0.817647	0.825476	0.817647	0.812263
3	0.323900	0.721231	0.833529	0.835209	0.833529	0.829809
4	0.256200	0.859944	0.837059	0.837781	0.837059	0.832446
5	0.182700	0.830657	0.839412	0.838763	0.839412	0.836497

Figure 4.11 RoBERTa model training

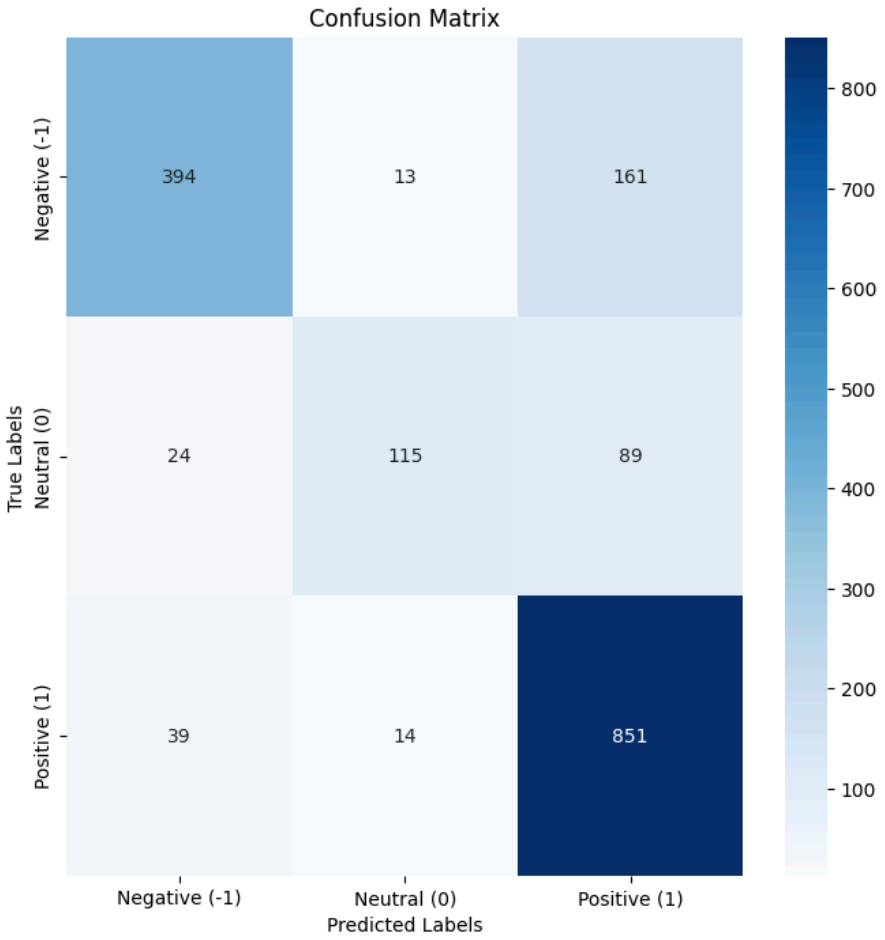


Figure 4.12 RoBERTa model confusion matrix

We clearly see that the precision and recall are both high at about 83%, which significantly beats our traditional machine-learning model algorithms. With further hyperparameter tuning and more computational power, these evaluation metrics could be improved.

4.3 Enhancements of Machine Learning Classification

For this portion, we attempt to improve the accuracy of our machine learning model, namely the SVM model by using feature selection as well as dimensionality reduction techniques.

4.3.1 Chi-Square Feature Selection

Firstly, we use Chi Square select K best for feature selection, which extracts terms that have a relationship with the target variable(sentiment). The Chi-squared test measures the lack of independence between a term t and a class c. A high Chi-squared score for a term means that the occurrence of that term is more correlated with the occurrence of the class c. By selecting the top k terms with the highest Chi-squared scores, the pipeline ensures that the features used for classification are those that are most statistically significant and relevant to the target variable. This enhances the model by potentially improving accuracy and reducing overfitting since irrelevant or less relevant features are removed. The Equation below is the mathematical representation of calculating the chi-square score for term t.

$$\chi^2(t,c) = \sum_{i=1}^{N} \frac{(O_i - E_i)^2}{E_i} \quad \text{.....(3)}$$

4.3.2 Truncated SVD

Following this we apply decomposition on our matrix using truncated SVD algorithms. Truncated SVD reduces the dimensionality of the feature space by approximating the original TF-IDF matrix with a matrix of lower rank that captures the most significant variance in the data. By doing so, it helps in mitigating the curse of dimensionality and reduces noise. It also makes the computation more efficient by decreasing the number of features that need to be processed. Furthermore, SVD can uncover latent structures in the text data, such as topics or concepts, that may not be immediately apparent but could be useful for classification. This enhances the model's performance by focusing on the most informative aspects of the data and often leads to better generalisation on unseen data. The Equation below represents the mathematical representation of reducing the tf-idf matrix represented by matrix A, into k dimensions.

$$A \approx A_k = U_k \Sigma_k V_k^T \quad \text{.....(4)}$$

4.3.3 Results

```
Shape of TF-IDF Matrix: (6800, 10985)
Accuracy: 0.7611764705882353
Precision: 0.7597329704872027
Recall: 0.7611764705882353
F1 Score: 0.7451522336082738
              precision    recall  f1-score   support

          -1       0.83      0.67      0.74       568
           0       0.65      0.32      0.43       228
           1       0.74      0.93      0.83       904

    accuracy                           0.76      1700
   macro avg       0.74      0.64      0.67      1700
weighted avg       0.76      0.76      0.75      1700
```

Figure 4.13 Results of RoBERTa after Feature Selection and Truncated SVD

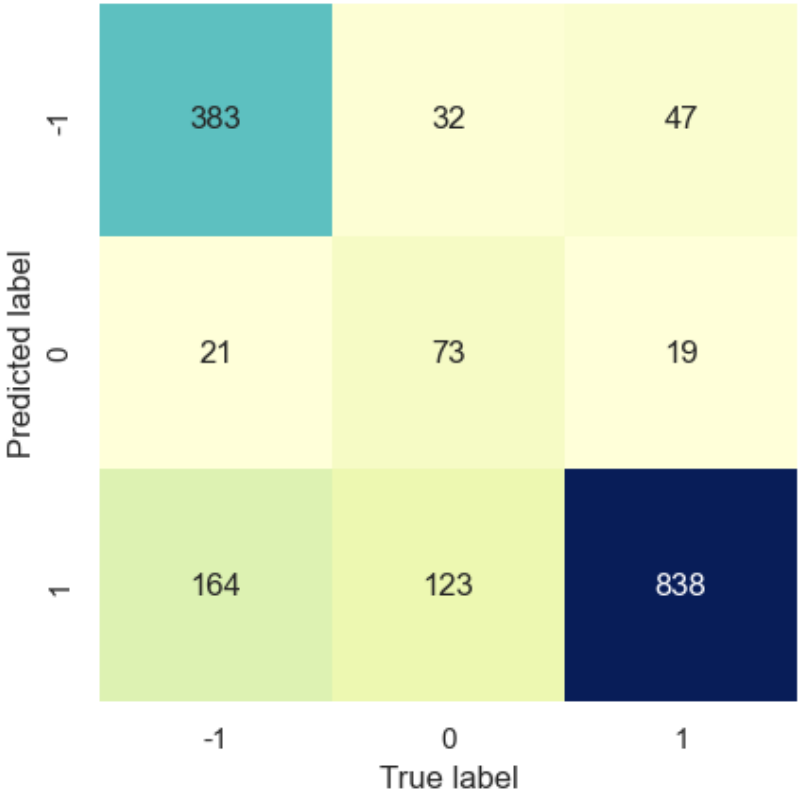


Figure 4.14 Confusion matrix of RoBERTa after Feature Selection and Truncated SVD

Following the implementation of the Chi-squared feature selection and Truncated SVD for dimensionality reduction in our text classification pipeline, we observed a modest improvement in the model's performance. This incremental gain can be attributed to the more judicious feature set curated through the Chi-squared test, which helped isolate the most statistically significant terms for class prediction. Moreover, the application of Truncated SVD effectively condensed the feature space, mitigating the curse of dimensionality and enhancing computational efficiency. By distilling the text data into its most informative components, the model could focus on the underlying structure and semantic content that are most relevant for classification. Although the enhancements led to only a slight uptick in results, this underscores the potential for further tuning and exploration of feature engineering to maximise the predictive power of the model. Such refinements align with the iterative nature of machine learning workflows, where each step, no matter how small, contributes to the journey towards optimal performance.

4.4 Innovations: Sarcasm Detection

Reddit comments are often written in a sarcastic tone. Hence, sarcasm detection was performed using the SenticNet API. The model architecture used by SenticNet is given below. It consists of multiple convolution layers to extract different features from the text followed by max pooling. The features are put together and classification is performed on the resulting data.

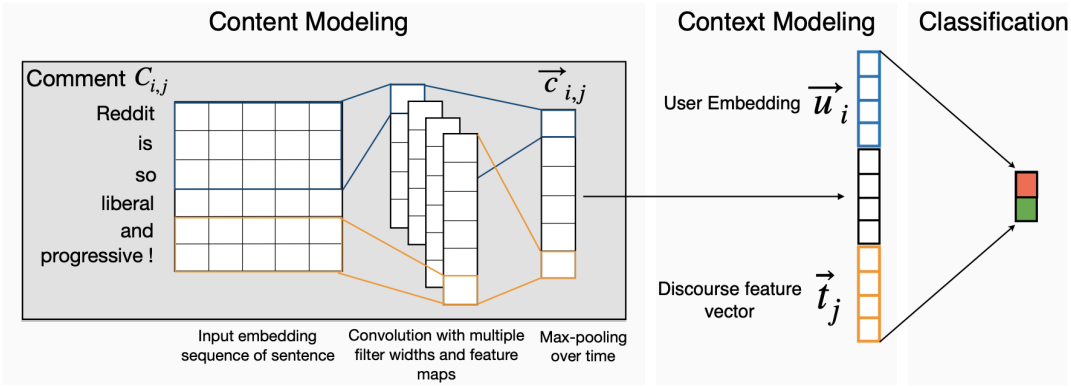


Figure 4.15 Architecture of SenticNet sarcasm detection model

Out of 34690 relevant comments, 2294 were found to have a sarcasm score greater than 0, with the most common range being from 0.6 to 0.8. This is about 6.61% of the comments.

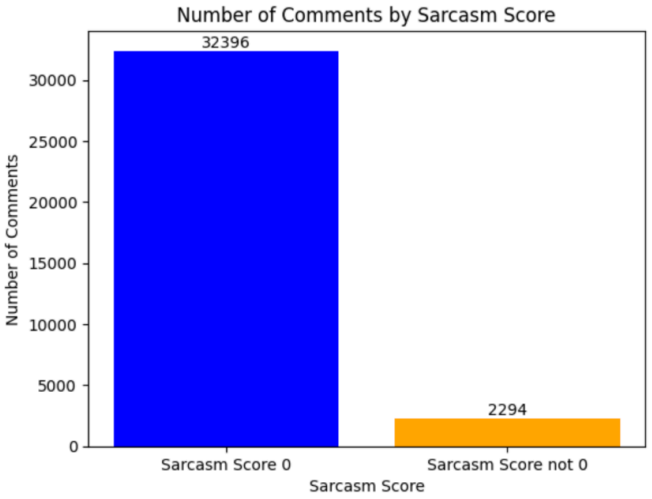


Figure 4.16 Comments with vs without sarcasm

When classification was performed using SVM, the performance improved significantly. The following scores and confusion matrix were obtained.

```
Accuracy: 0.8215271389144434
Precision: 0.82163605939907
Recall: 0.8215271389144434
F1 Score: 0.8206865751686837
```

Figure 4.17 Performance metrics after sarcasm detection

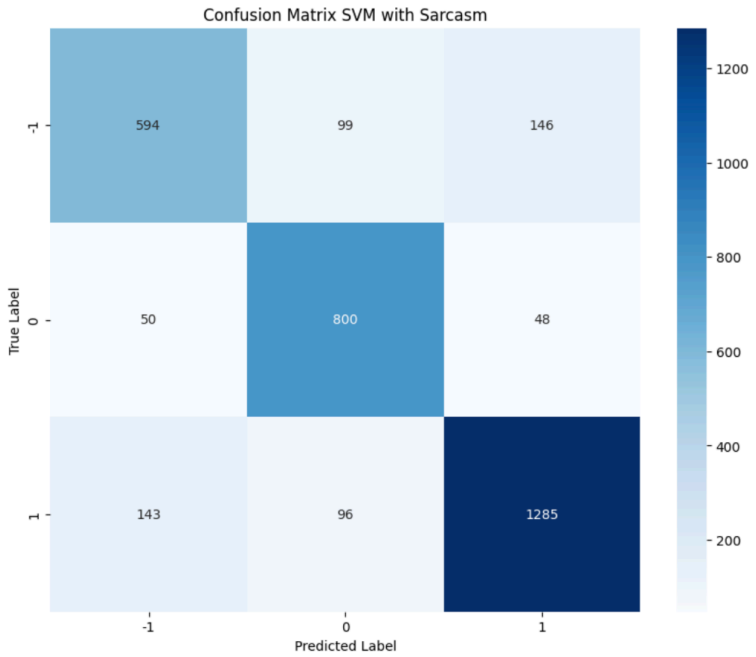


Figure 4.18 Confusion matrix after sarcasm detection

4.4 Conclusion

From the results of our project, RoBERTa outperforms classical machine learning models in text analysis tasks due to its deep understanding of language context derived from its transformer architecture. Classical models like SVMs or Random Forests are limited to the features they are provided and cannot inherently understand language nuances. Roberta, however, uses self-attention mechanisms to weigh the importance of each word within its context, allowing for a sophisticated understanding of the text. It's pre-trained on vast corpora, which enables it to grasp a wide array of linguistic patterns and idioms. Moreover, unlike classical models that require extensive feature engineering to handle different aspects of language, RoBERTa learns representations automatically, capturing subtleties that are often missed by traditional models. This ability to model complex dependencies and interpret the semantics of text leads to a substantial improvement in performance across a variety of natural language processing tasks.

Added features can be beneficial for a model to ‘understand’ the text better, for example, a sarcasm score. The following comment is found in our dataset: “Yet people pay for his course. Indeed, what a world.” It has a sarcasm score of 0.84. Without this knowledge, a model may classify it as neutral or even positive, but with the sarcasm score, it is more likely to classify it correctly. As seen in the performance improvement, this helped the model make better predictions.