NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CZ4034 Information Retrieval

AY 2023/24 Semester 2

Assignment - Group 29

Name	Matriculation Number
Ingale Omkar	U2020724H
Su Myat Aung	U2021467G
Ng Zheng Xun	U2020299H
Lee Sumi	U2023050B
Neel Kumar	U2023314J
Lim Cheng Ping	U2121712E

Table of Contents

1. Introduction

As cryptocurrencies continue to surge in popularity, it is crucial to stay ahead of the curve and seize opportunities. Yet, amidst the vast expanse of online information, navigating through heaps of data can feel like searching for a needle in a haystack. Furthermore, even after finding the information, it may often prove to be irrelevant and unhelpful.

In this project, we developed a financial information search engine with the use of a simple graphical user interface developed with React, to help enthusiasts, investors, and researchers explore the crypto sphere. Opting to crawl from Reddit API was a natural choice, given its reputation for information sharing, community engagement, and supportive communities. Furthermore, Reddit as a social media platform provides no word count limit unlike Twitter/X, and therefore more data could be potentially scraped for better insights.

Afterwards, we integrated our user interface with Solr and used it as a backend to perform indexing and querying. Lastly, we ran natural language processing tools and deep learning models such as VADER and FinBERT to conceptualise our data and generate sentiment analysis. We also do an ablation study on the various preprocessing techniques, and report the results obtained. The complete architecture and flow of our search app are as follows:

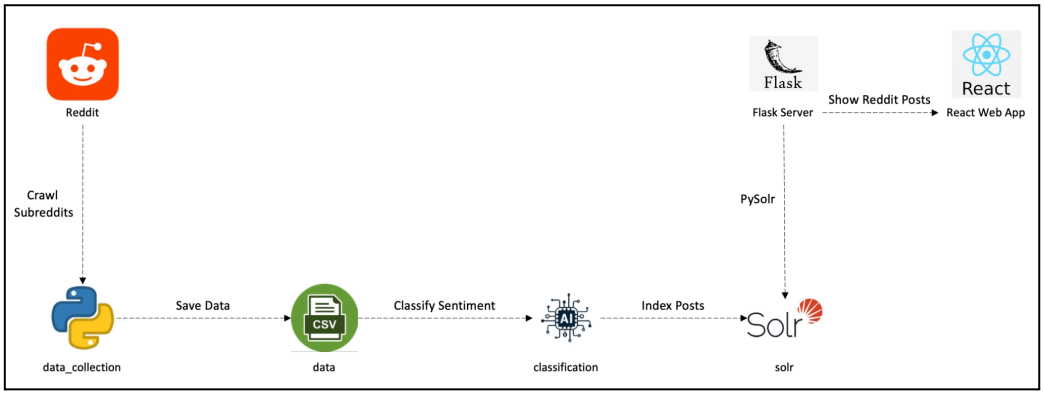


Figure 1: Integration of our application

Links to our project materials are below:

1. YouTube video presentation: https://youtu.be/9TjmPQDuXm8
2. Google Drive with compressed files of crawled text data, queries, and their results, evaluation dataset, automatic classification results, and more data for Questions 3 and 5: https://drive.google.com/drive/folders/14DwqOmrGYQ3YN9UMmzvHc74USu4i-C_5?usp=sharing
3. Google Drive with a compressed file with all source codes and libraries, and a README : https://drive.google.com/drive/folders/1v-Ks1EoNoc08PpkN8OUFS51qp3YVZPqX?usp=sharing

2. Crawling

This section of the report details how the data for our sentiment search engine was obtained. Reddit was chosen as the platform for this search as it hosts several subreddits that facilitate discussion on various financial topics including stocks, markets, indices, cryptocurrencies, and more. To ensure that our search engine served its users' need to obtain vast amounts of data related to financial topics, Reddit seemed like the ideal choice.

2.1 Approach for crawling

To crawl and scrape data from various subreddits, we used the PRAW API. This API allowed us to scrape roughly 1000 posts per query from any given subreddit based on various filters.

```
reddit = praw.Reddit(client_id='99NLbQytYY8eUQmaYiW8gQ', client_secret='hE8yYrSz3hsFwHdfZNw119YLKCo91A', user_agent='irgrpproject')
posts = []


# subreddits to scrape
# changing the limit does not have an effect on number of data points
# all subreddits yield ~900-1000 data points on average
subreddits_names = ['wallstreetbets', 'stocks', 'investing', 'CryptoCurrency', 'pennystocks', 'bitcoin', 'ether', 'shibainucoin', 'StockMarket',
                    'StocksAndTrading', 'StockMarketMovers', 'WallStreetbetsELITE', 'XRP']
for subreddits_name in subreddits_names:
    subreddit = reddit.subreddit(subreddits_name)
    for post in subreddit.new(limit=1000):
        created_date = datetime.datetime.utcfromtimestamp(post.created_utc).strftime('%Y-%m-%d')
        posts.append([post.title, post.score, post.id, post.subreddit, post.url, post.num_comments, post.selftext, created_date])
posts = pd.DataFrame(posts,columns=['title', 'score', 'id', 'subreddit', 'url', 'num_comments', 'body', 'created'])
```

Figure 2: Crawling Implementation using PRAW

For this project, we crawled and scraped data from the following subreddits:

1. r/wallstreetbets
2. r/stocks
3. r/investing
4. r/CryptoCurrency
5. r/pennystocks
6. r/bitcoin
7. r/ether
8. r/shibainucoin
9. r/StockMarket
10. r/StocksAndTrading
11. r/StockMarketMovers

12. r/wallstreetbetsELITE
13. r/XRP

From these subreddits, we collected the following information on posts:

1. Title
2. Body
3. Score
4. ID
5. Subreddit
6. URL
7. Number of Comments
8. Date Posted

The crawled data is saved as a CSV and then into the Apache Solr database. As Solr provides full-text search capabilities using REST APIs, we utilised it for indexing and querying our data.

2.2 Use Cases for Crawled Data

The use cases for our data can vary depending on the users. Some use cases that will benefit the users of this system are as follows:

- Retrieve the sentiment relating to an asset that they might be interested in. Some example queries can be:
    - “Is Bitcoin a good investment?”
    - “Is Nvidia stock overpriced?”
    - “Is a recession likely now?”
- Some users may also like to retrieve detailed analyses posted on these subreddits by other users. This may help them to judge the sentiment of other investors and make informed decisions. Some example queries can be:
    - “Are there antitrust lawsuits currently against Apple?”
    - “Meta’s monopoly on GPUs”

The results of these queries will be posts from subreddits that will ensure that the users of our search engine can understand the current market conditions and make informed investment decisions.

2.3 Analysis of Crawled Data

As we were limited to 1000 posts per query using the PRAW API, we decided to scrape the most recently published posts from the subreddits mentioned before. While preprocessing on raw data was done during sentiment classification, some preliminary statistics about the crawled data can be found below:

Table 1: Analysis of Crawled Data

Statistic	Count
Total subreddits	13
Total posts	10,205
Total words in corpus	1,927,565
Number of unique words	130,418
Number of sentiments	3

Some additional statistics about the data can be found below:

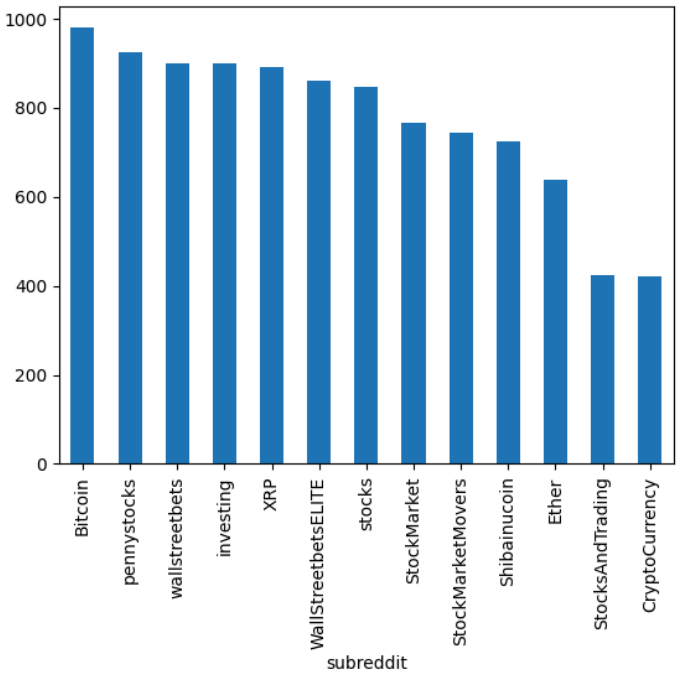


Figure 3: Distribution of Posts by Subreddits

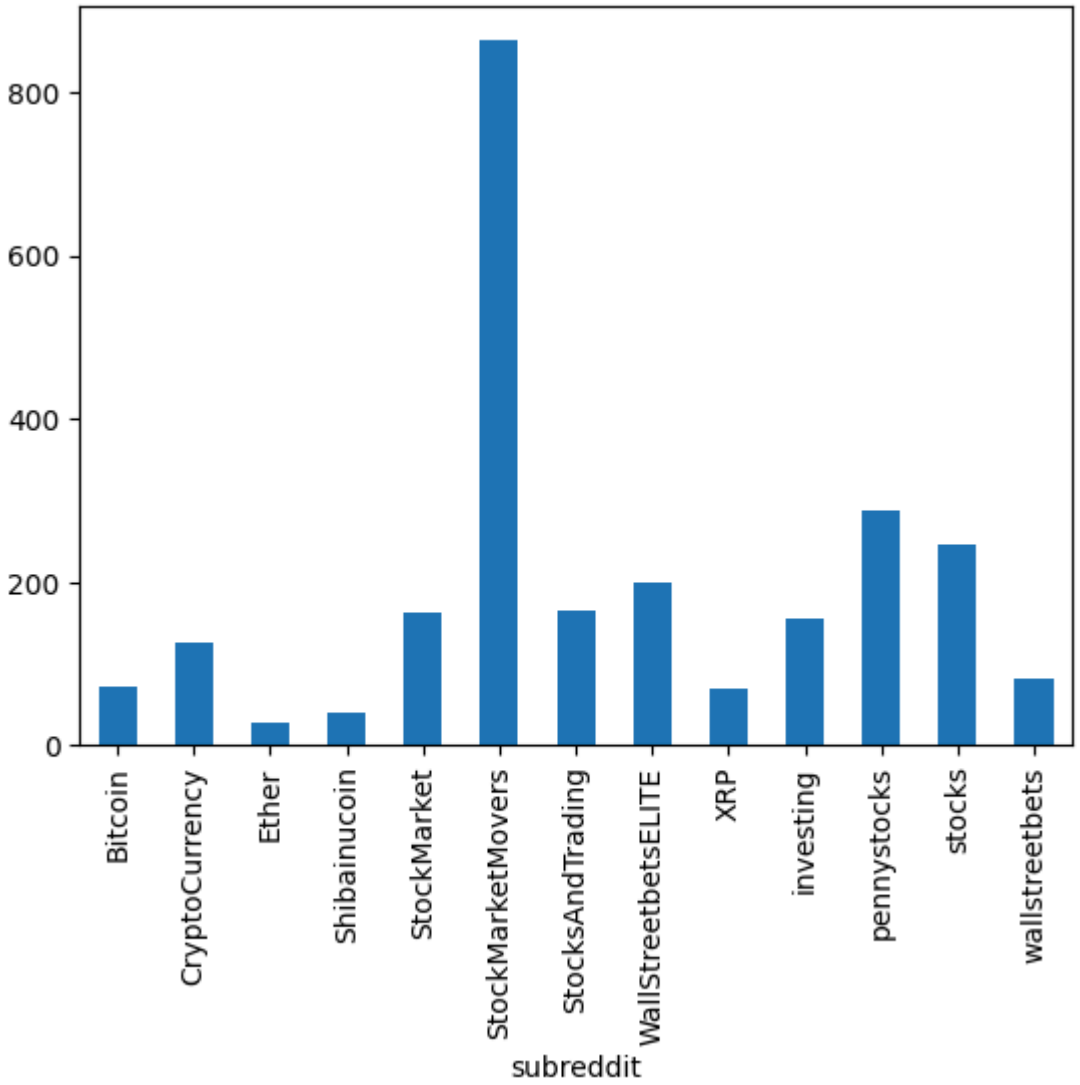


Figure 4: Average Word Count of Posts by Subreddit

A comparison of the data before and after preprocessing techniques were applied can be found in the classification section of this report.

3. Indexing

3.1. Overview

For indexing, we will be using Apache Solr as our backend database to index and store our crawled Reddit posts. Solr is an open-source search platform that provides powerful indexing and searching capabilities for various types of data. It is highly reliable, scalable, and fault-tolerant, providing distributed indexing, replication, load-balancing querying, automated failover and recovery, centralised configuration, and more.

Python libraries, such as Pysolr were also used, to simplify the integration process with the UI and provide a process of interacting with Solr's REST API. It also takes away the complexities of directly interacting with Solr's HTTP API, providing a more Pythonic interface for working with Solr.

3.2. Approach

Before loading the crawled data into Solr, the core "cz4034" was created and the data fields of the content were added as follows:

Table 2: Fields of data

Field	**Description**
id	Unique identifier of the reddit post obtained from reddit API
title	Title of the reddit post
body	Content of the reddit post
subreddit	Subreddit that the uploaded post belongs to
counter	Counter variable to maintain the number of relevant votes each post received
num_comments	Total number of comments garnered by the post
reddit_score	Number of upvotes minus downvotes that the post has obtained
url	URL leading to the reddit post
created	Date of when the post was created
predicted_sentiment	Predicted sentiments generated from FinBERT

The crawled data is then added to the Solr Core, where the content will be processed by various analysis processes.

In this project, the following analysis processes are applied to index the input data and the query:

1. Tokenizer

The standard tokenizer from the solr.StandardTokenizerFactory was employed during our indexing process. The tokenizer splits our crawled data into tokens by treating whitespaces and punctuations as delimiters. The figure below illustrates how tokenization is performed for the index and query “Best stocks to invest in for NEXT year?”.

ST	text	Best	stocks	to	invest	in	for	NEXT	year
	raw_bytes	[42 65 73 74]	[73 74 6f 63 6b 73]	[74 6f]	[69 6e 76 65 73 74]	[69 6e]	[66 6f 72]	[4e 45 58 54]	[79 65 61 72]
	start	0	5	12	15	22	25	29	34
	end	4	11	14	21	24	28	33	38
	positionLength	1	1	1	1	1	1	1	1
	type	<ALPHANUM>	<ALPHANUM>	<ALPHANUM>	<ALPHANUM>	<ALPHANUM>	<ALPHANUM>	<ALPHANUM>	<ALPHANUM>
	termFrequency	1	1	1	1	1	1	1	1
	position	1	2	3	4	5	6	7	8

Figure 5: Application of tokenizer for “Best stocks to invest in for NEXT year?”

2. Stopword

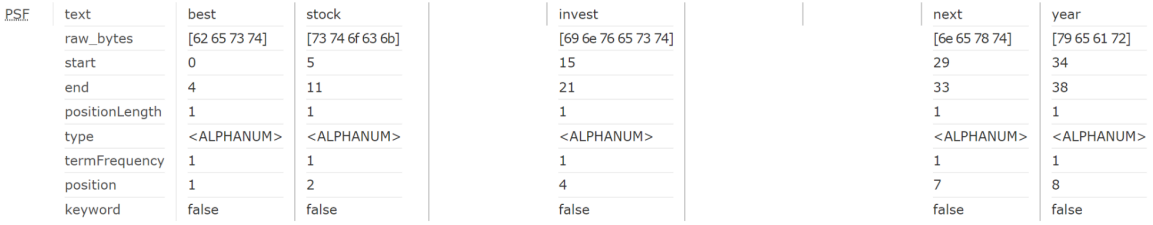
Stopwords are commonly used English words that do not provide useful information. In our project, the solr.StopFilterFactory will discard tokens that are present in the stopwords list.

SF	text	Best	stocks		invest			NEXT	year
	raw_bytes	[42 65 73 74]	[73 74 6f 63 6b 73]		[69 6e 76 65 73 74]			[4e 45 58 54]	[79 65 61 72]
	start	0	5		15			29	34
	end	4	11		21			33	38
	positionLength	1	1		1			1	1
	type	<ALPHANUM>	<ALPHANUM>		<ALPHANUM>			<ALPHANUM>	<ALPHANUM>
	termFrequency	1	1		1			1	1
	position	1	2		4			7	8

Figure 6: Application of stopword filter for “Best stocks to invest in for NEXT year?”

3. Stemming

The Porter Stem Filter was also applied to the tokens generated, allowing us to reduce the size of our dictionary and have a more efficient retrieval. As illustrated, the word “stocks” was indexed as “stock” after the stemming algorithm was applied.

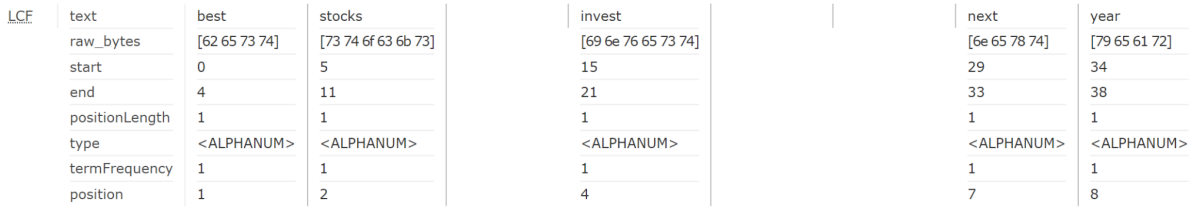


PSF	text	best	stock	invest	next	year
	raw_bytes	[62 65 73 74]	[73 74 6f 63 6b]	[69 6e 76 65 73 74]	[6e 65 78 74]	[79 65 61 72]
	start	0	5	15	29	34
	end	4	11	21	33	38
	positionLength	1	1	1	1	1
	type	<ALPHANUM>	<ALPHANUM>	<ALPHANUM>	<ALPHANUM>	<ALPHANUM>
	termFrequency	1	1	1	1	1
	position	1	2	4	7	8
	keyword	false	false	false	false	false

Figure 7: Application of Porter Stem Filter for “Best stocks to invest in for NEXT year?”

4. Case Folding

Case Folding is also implemented to convert words into lowercase, enhancing the efficiency of the dictionary as we can eliminate the need to distinguish between uppercase and lowercase forms.



LCF	text	best	stocks	invest	next	year
	raw_bytes	[62 65 73 74]	[73 74 6f 63 6b 73]	[69 6e 76 65 73 74]	[6e 65 78 74]	[79 65 61 72]
	start	0	5	15	29	34
	end	4	11	21	33	38
	positionLength	1	1	1	1	1
	type	<ALPHANUM>	<ALPHANUM>	<ALPHANUM>	<ALPHANUM>	<ALPHANUM>
	termFrequency	1	1	1	1	1
	position	1	2	4	7	8

Figure 8: Application of Lower Case Folding for “Best stocks to invest in for NEXT year?”

3.3. Search Engine UI

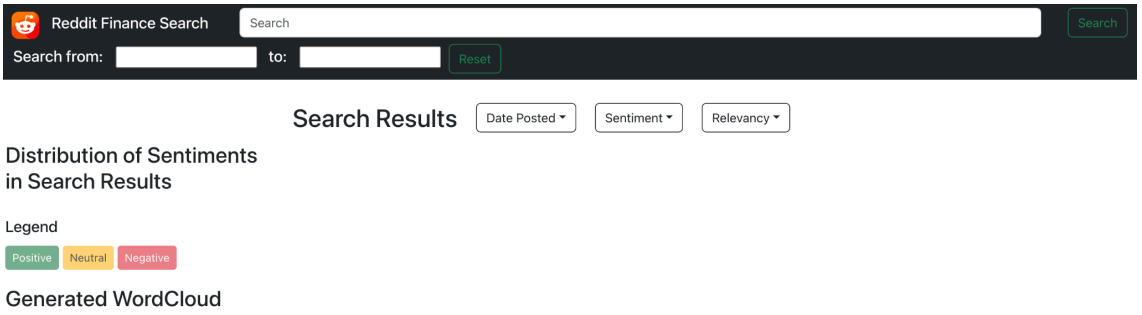


Figure 9: User Interface

The search engine user interface (UI) was built using React. As shown in the figure above, it consists of a search bar and a timeframe for users to input the query settings. Subsequently, search results from the Solr database are displayed below the search field. Multiple filters can be applied such as:

1. "Date Posted" to sort search results by the dates when the posts were created,
2. "Sentiment" to filter the posts by their sentiments, which are labelled through classification, and
3. "Relevancy" to sort the search results by relevance score generated by Solr to indicate how relevant the document is to the query.

An example URL query to the Solr database is http://127.0.0.1:5000/query?q=recession.


```
{
  "results": {
    "documents": [
      {
        "body": [
          "I am beginning to allocate a small percentage of my portfolio towards physical silver. Here's why: \n\n1. I want to diversify my portfolio beyond cash, stocks and crypto.\n\n2.
The gold/silver ratio is at 90. Meaning 90 ounces of silver buys 1 ounce of gold. Historically the average is around 70. Seems like a good time to buy silver.\n\n3. Silver is a hedge
against the USD and US financial markets. If we head into a recession, silver typically performs well. \n\n4. Silver is a non-digital asset. I cannot think of a physical asset that stores
value better than silver, other than other precious metals or land / real estate. It has gone from $5 an ounce to $22 an ounce since 1975. It is important to me to own physical assets
that will appreciate in value. \n\n5. It is a solid vehicle for transferring wealth to my children in the future. \n\n6. It's shiny! Also, it is fun collecting certain coins if you are
willing to pay an extra premium. \n\n7. Value stems from a myriad of industrial and medical uses. \n\nLet me know what you guys think. I know it isn't the best investment in the world,
but it seems sensible to allocate a small percentage of my portfolio to it."
        ],
        "created": [
          "2024-02-06T00:00:00Z"
        ],
        "id": "1akg3bc",
        "reddit_score": 0,
        "score": 5.771188,
        "subreddit": [
          "investing"
        ],
        "title": [
          "Physical silver investment"
        ]
      },
      {
        "body": [
          "Is it a really bad sign we haven't went into a recession even though the yield curve has been inverted?\n\nCall me crazy, but I find it odd we haven't went into a recession
even though we had a yield curve inversion a while back. Look at how the yield curve almost perfectly predicts a recession. \n\nExample- before 2007 and prior recessions. A recession has
to occur right? Rates if they stay jacked up: I don't understand how we don't go into a recession unless they start cutting rates soon."
        ],
        "created": [
          "2024-01-20T00:00:00Z"
        ],
        "id": "19b2xft",
        "reddit_score": 115,
        "score": 3.3590858,
        "subreddit": [
          "stocks"
        ],
        "title": [
          "Yield curve- inverted"
        ]
      },
      {
        "body": [
          "When researching the debt ceiling in correlation to a recession, my only findings point out a recession caused by a debt default and what that would look like. But what about a
debt default caused by a recession? If a recession were to occur from another catalyst, wouldn't the massive debt on the US balance sheet make the situation extremely worse? I thought I
would pose this question to the random strangers on the internet to see your thoughts."
        ],
```


Figure 10: Example query

A query response as shown in Figure 10 is returned from the Solr database, which is then sent to UI for display, as shown below.

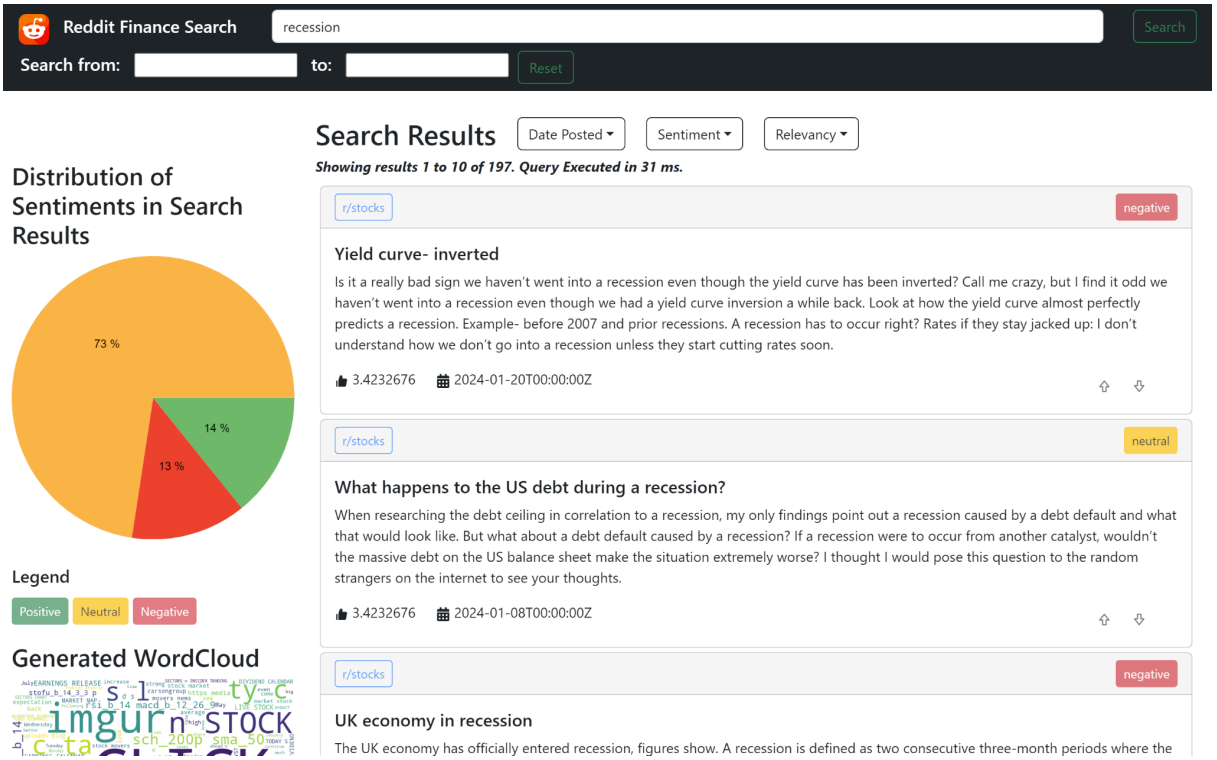


Figure 11: Search result for "recession"

A Pagination component, as shown below, is implemented such that 10 search results are displayed to the user at a time. Users can navigate and browse through more Reddit posts retrieved from the database.

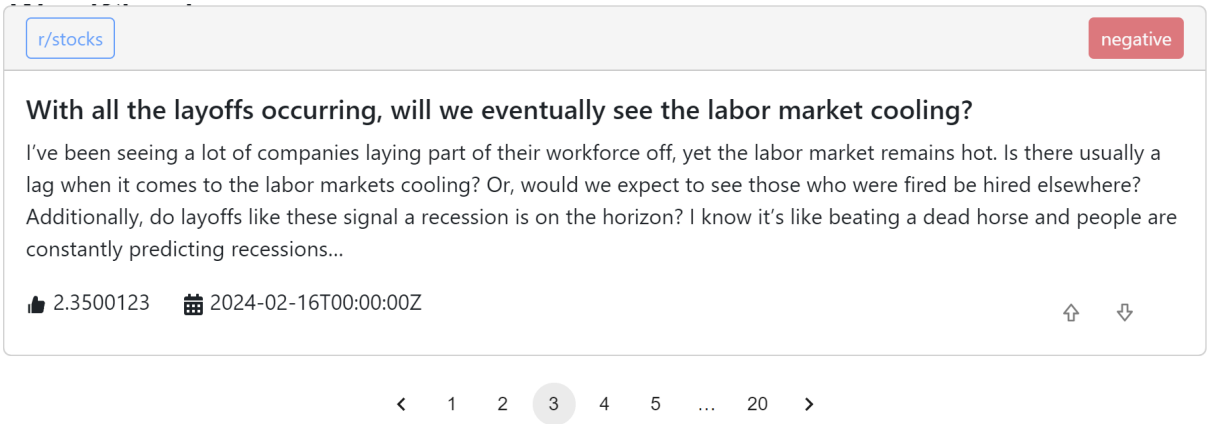


Figure 12: Pagination Component

A table of example queries and their respective number of results and time taken for retrieval of the results for the queries (QTime) are presented in the table below.

Table 3: Query with search result and query time

Query	Top Post Title	Number of Results	QTime (ms)
recession	Yield curve- Inverted	197	2
recession forecast for next year	Recession fears evaporate in new forecast of top economists	5601	6
recetion forcas for nxt year	Sofi's First Profitable Quarter with Q4 EPS beyond the estimated EPS. Strong 2024 Guidance that is higher than the current estimates.	5433	32
nvda bad	NVDA at the UK open	611	9
best stock to invest now	3 Best Asian Stocks to Buy Right Now - Newsblare	6868	4

The time taken for each query is relatively fast. However, the third query that contains misspelt words had a generally longer query time due to the spell check that has to be conducted. In addition, queries containing multiple words generally yield more results as we have set Solr to execute an 'OR' operation between each word.

3.4. Enhancement of Indexing and Ranking

To further improve the querying abilities of the search engine, we have integrated some innovative approaches to make it more relevant to the users.

3.4.1. Timeline Search

Users could analyse the timeline to gain insights into forecasting and predicting future events, allowing educated guesses, which is vital in the stock market.

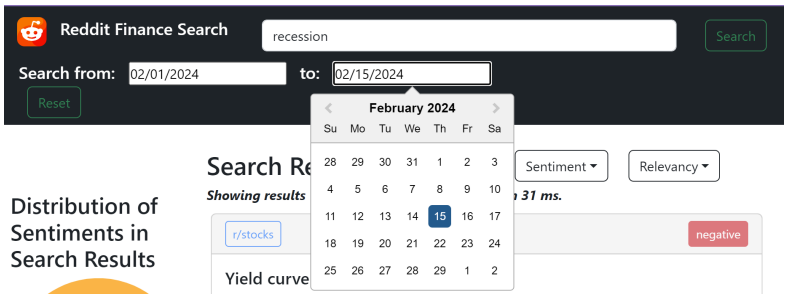


Figure 13: Timeline Search

3.4.2. Word Cloud

It offers a visual data summarization of large amounts of text, allowing the user to identify common themes or keywords without looking through all the searches.

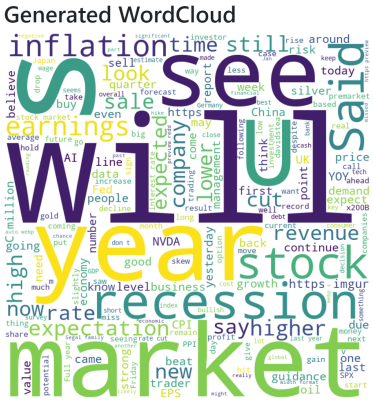


Figure 14: Word Cloud

3.4.3. Interactive Search

It allows real-time feedback from users that helps them gauge the relevance of their search and make adjustments accordingly, leading to a more accurate and effective search result in subsequent searches. When the user selects the upvote or downvote button, “counter” field of the document will be modified and the updated document rankings are displayed.

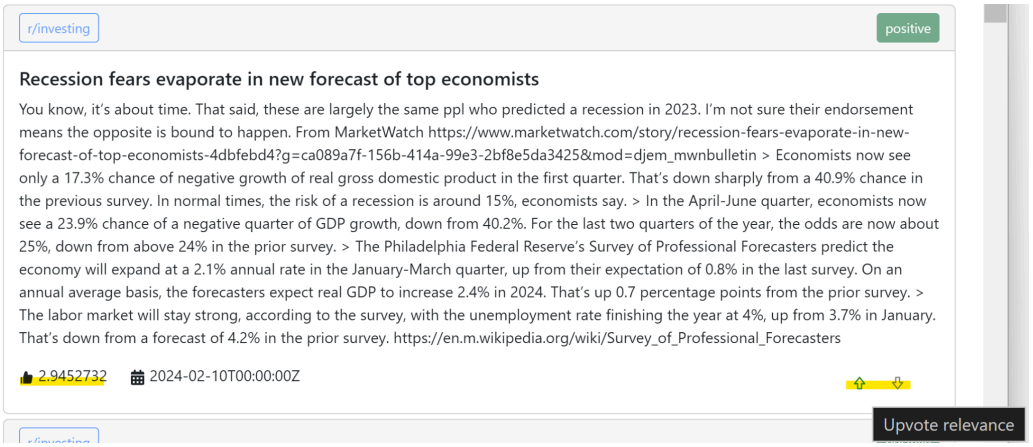


Figure 15: Interactive Search

3.4.4. Spell Check

The search and spell-checking component will provide suggestions to the users if their query contains misspelt words. Since the suggestions provided by Solr are based on the words that were used to create the index, all suggestions are provided in lowercase form.

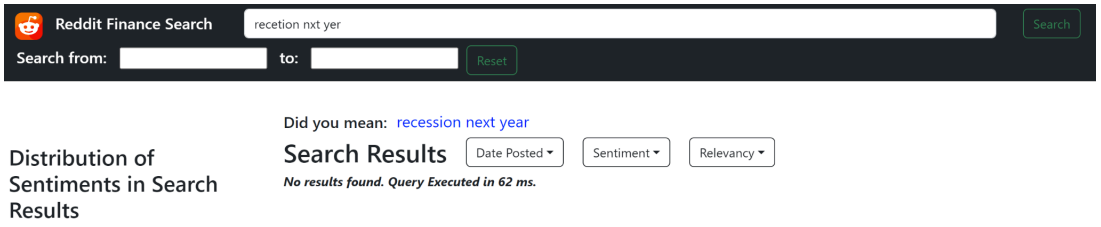


Figure 16: Spell-Check

For the implementation of this feature, we have added the following code snippets to the solrconfig.xml file:

1. Integration of the spell-checking query to the primary search request handler

```
<requestHandler name="/select" class="solr.SearchHandler">
  <lst name="defaults">
    <str name="echoParams">explicit</str>
    <int name="rows">10</int>

    <!-- Spellcheck -->
    <str name="spellcheck.dictionary">default</str>
    <str name="spellcheck">on</str>
    <str name="spellcheck.count">5</str>
    <str name="spellcheck.collate">true</str>
    <str name="spellcheck.maxCollationTries">10</str>
  </lst>

  <arr name="last-components">
    <str>spellcheck</str>
  </arr>
</requestHandler>
```

Figure 17: Spell check query in solrconfig.xml

2. Creation of Spell Check Component

In this component, the IndexBasedSpellChecker and WordBreakSolrSpellChecker are applied to both the title and the body fields. The IndexBasedSpellChecker utilises the existing Solr index to provide suggestions of mispelt words in the search queries. To handle queries with multiple words, the WordBreakSolrSpellChecker was applied to provide suggestions by combining adjacent query terms.

```
<searchComponent name="spellcheck" class="solr.SpellCheckComponent">

  <str name="queryAnalyzerFieldType">text_general</str>
  <!-- Multiple "Spell Checkers" can be declared and used by this
       component
    -->

  <!-- a spellchecker built from a field of the main index -->
    <lst name="spellchecker">
      <str name="classname">solr.IndexBasedSpellChecker</str>
      <!-- field to use -->
      <str name="field">title</str>
      <str name="field">body</str>
      <!-- buildOnCommit|buildOnOptimize -->
      <str name="buildOnCommit">true</str>
      <!-- $solr.solr.home/data/spellchecker-->
      <str name="spellcheckIndexDir">./spellchecker</str>
      <str name="accuracy">0.7</str>
      <float name="thresholdTokenFrequency">.0001</float>
    </lst>

  <lst name="spellchecker">
    <str name="name">wordbreak</str>
    <str name="classname">solr.WordBreakSolrSpellChecker</str>
    <str name="field">title</str>
    <str name="field">body</str>
    <str name="buildOnCommit">true</str>
    <str name="combineWords">true</str>
    <str name="breakWords">true</str>
    <int name="maxChanges">10</int>
  </lst>

</searchComponent>
```

Figure 18: Spell check component in solrconfig.xml

3.4.5. Sentiment Overview

A pie chart is displayed for each query search, offering users an overview of how the sentiments are distributed across Reddit posts. The figure below illustrates the pie chart for the query “recession” for Reddit posts between 1st February 2024 and 15th February 2024. The neutral (orange section), positive (green section), and negative (red section) sentiment posts are almost equally distributed.

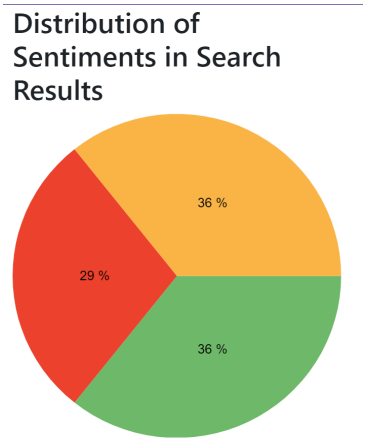


Figure 19: Sentiment Pie-chart

4. Classification

4.1 Literature Review

With respect to the State-of-the-Art (SOTA), we referred to benchmark datasets such as the SUBJ dataset for subjectivity detection, and the IMDb dataset for sentiment analysis. However, we also notice that there exist models that are trained specifically on financial data, such as FinBERT[1], and on social media data as well, such as VADER[2], and therefore choose to adopt them. Therefore, we could not exactly evaluate our methods against the general SOTA methods as our models are more specific and fine-tuned to our domain of financial and social media chatter.

We ultimately opted for a multi-task classification approach, where 2 tasks are jointly tackled with one model[3]. Our models tackle both subjectivity detection and sentiment analysis, giving a prediction on both fronts. This is mainly due to the model's ability to detect neutral-sounding sentences, which could also be assigned a label of neutral subjectivity, as all neutral sentiments must have a neutral subjectivity label.

4.1.1 SUBJ Dataset

From the PapersWithCode repository of benchmarks[4], the State-of-the-Art methods for Subjectivity Detection involve BERT-Base + CLR + LSTM models as implemented by Nandi et. al [5] . These models are benchmarked against the SUBJ dataset, and the improvement in SOTA of this task could be visualised in the plot below in Figure 4. As our model does both subjectivity detection and sentiment analysis, we provide the below benchmarks as a reference, with respect to our reported results later on.

[1] https://huggingface.co/ProsusAI/finbert
[2] https://github.com/cjhutto/vaderSentiment
[3] https://sentic.net/sentiment-and-sarcasm-classification-with-multitask-learning.pdf
[4] https://paperswithcode.com/sota/subjectivity-analysis-on-subj
[5] https://github.com/Ritika2001/Word-Embedding-Models-for-Subjectivity-Analysis

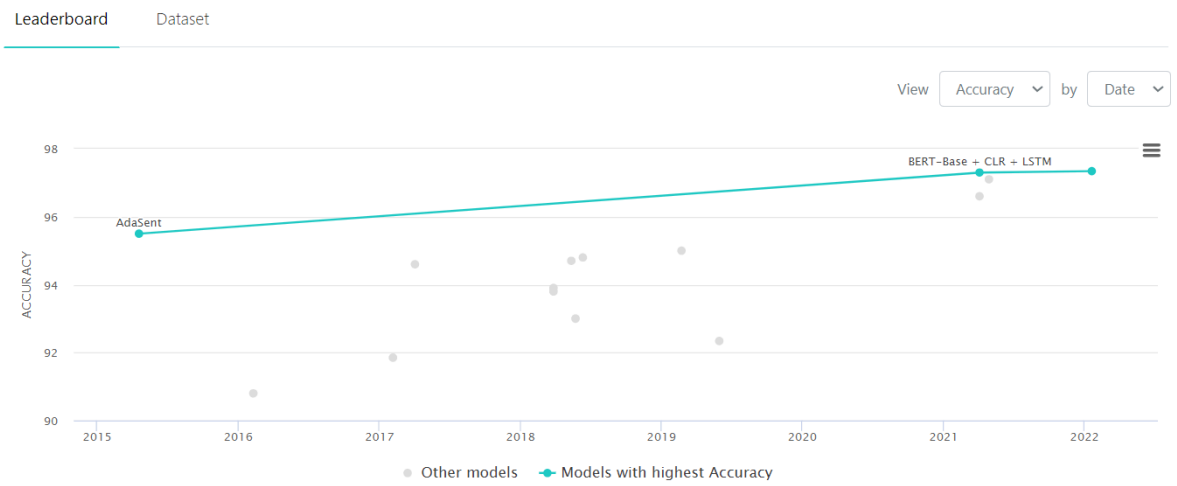


Figure 20: Improvements in the SOTA for Subjectivity Detection on the SUBJ Dataset

4.1.2 IMDb Dataset

For Sentiment Analysis, we use the IMDb benchmark dataset, where Transformer-based models such as BERT and RoBERTa obtain SOTA performance on the task. We would use a similar architecture for one of our models: FinBERT.

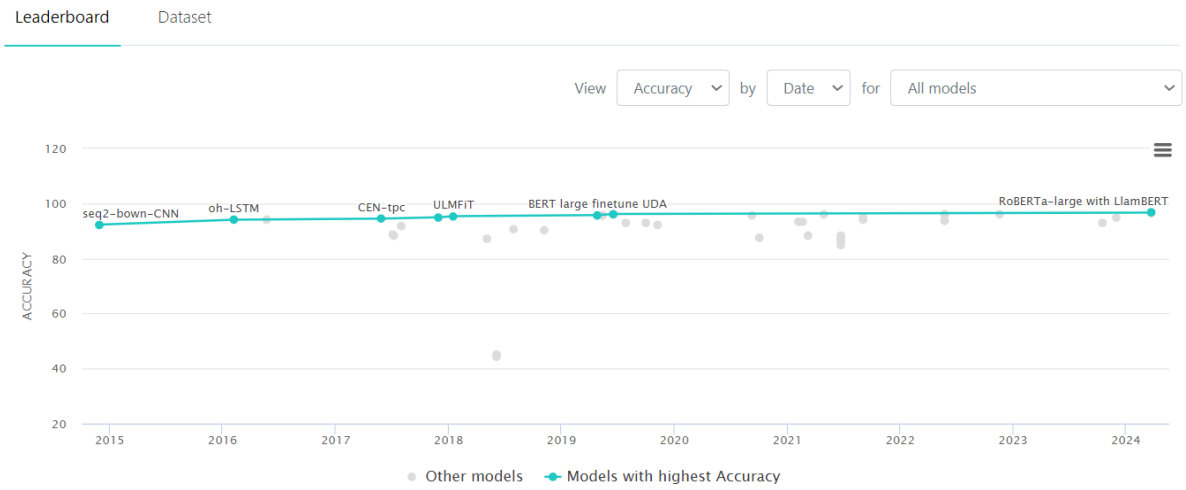


Figure 21: Improvements in the SOTA for Sentiment Analysis on the IMDb Dataset

4.2 Data Preprocessing

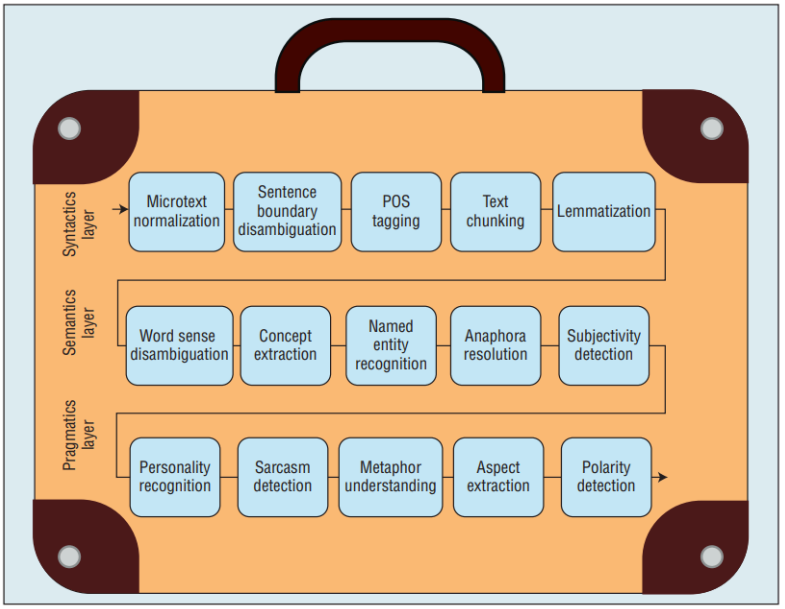


Figure 22: Suitcase of NLP, detailing potential steps in an NLP task

4.2.1 Preprocessing Pipeline

There are many definitions of what could constitute preprocessing of data, but in our case, we are taking it to be the "Syntactics Layer" with a secondary focus on the semantics and pragmatics layer, as shown above in the Suitcase of NLP. We started with processing the raw data into some cleaner form, removing outliers, and making the data more predictable and consistent. This stage, as the name suggests, takes place before any prediction by the NLP model. In our case, as we are handling social media data, we undertook several preprocessing steps, which are detailed in the following sections:

Concat title and body[6] → Remove metadata → Remove duplicates → Case Normalisation → Punctuation & Special Characters Removal → Stopword Removal → Lemmatization[7]

To investigate the effectiveness of these preprocessing techniques, we do an ablation study on several of the above techniques, such as lemmatization and stopword removal.

[6] As some posts have empty bodies but meaningful titles
[7] Lemmatization is preferred compared to Stemming as it considers the context of the word

4.2.2 Stopword Identification and Removal

From the current literature[8], it is noted that the general strategy of determining stopwords involves looking at the occurrence frequencies and hand-filtering for their semantic content. We therefore adopt a similar approach.

We plot the frequency of each word, to identify stopwords, and also to verify the concurrence with Zipf's Law, which is shown below.

Zipf's Law: $$\text{word frequency} \propto \frac{1}{\text{word rank}}.$$

To identify stopwords, we take the top 30 (~5%) most frequently occurring words, and remove them accordingly from the corpus. The figure below, as well as Table X, shows the resulting stopword list. Figure 23 also illustrates that our dataset follows Zipf's Law, with an exponential decrease in the word frequencies of the most commonly occurring words. Due to our dataset having a large size of around 19k unique words, we take the most commonly occurring 500 words instead, to illustrate Zipf's Law. The stopword list is manually filtered, to remove meaningful words such as 'but', which contains crucial sentiment information.

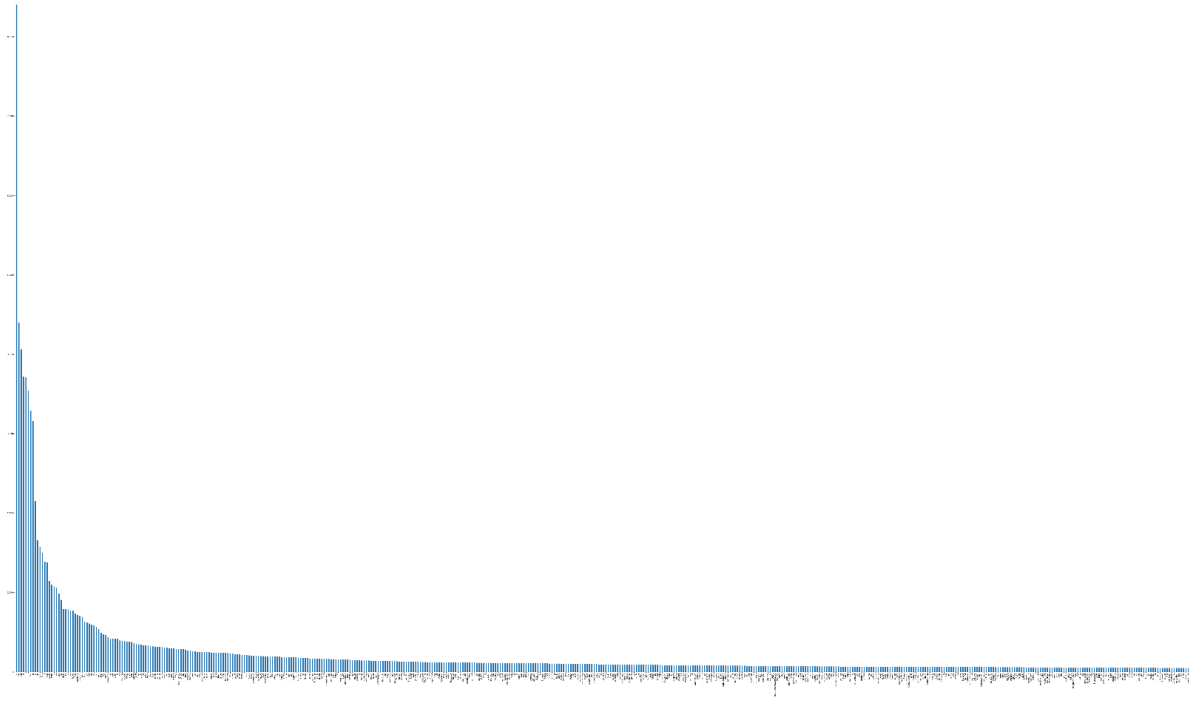


Figure 23: Plot of 500 commonly occurring words against their frequency

8 https://nlp.stanford.edu/IR-book/html/htmledition/dropping-common-terms-stop-words-1.html#:~:text=The%20general%20strategy%20for%20determining,documents%20being%20indexed%2C%20as%20a

Table 4: List of 30 Stopwords

Word	Frequency
the	84061
to	43967
and	40593
of	37220
	37072
a	35404
for	32916
in	31577
is	21502
on	16589
here	15696
this	14958
i	13848
that	13787
stock	11420
with	10987
it	10733
as	10583
are	9845
at	9063
click	7932
its	7927
be	7914
from	7731
have	7723
you	7316
earnings	7157
market	7005
has	6910

4.2.3 Lemmatization

We chose to do Lemmatization instead of Stemming, as the former considers the context of the words and therefore provides better and more accurate reduction to their base forms. We use the WordNetLemmatizer from the nltk.stem library in Python.

4.2.4 Word Sense Disambiguation

Moving on from the syntactic layer to the semantics layer, we thought this was a prudent step in better understanding our data. Since information retrieval techniques anchor heavily on grammatical understanding, the context and meaning of sentences are as important in order to decipher semantics to better classify data. Since Reddit data often has jargon, complex and nuanced nomenclature, we thought word sense disambiguation would be apt in filtering out useful data from useless and provide more context for manually labelling the evaluation set. In our exploration of word sense disambiguation (WSD), we utilised the Natural Language Toolkit (NLTK) and its implementation of the Lesk algorithm, a classic approach in natural language processing. The Lesk algorithm operates by analysing the context in which a target word appears within a sentence and matches it to the definition that shares the most overlapping words from a predefined lexical database, specifically WordNet in this case.

We crafted a Python function that applies the Lesk algorithm to each text entry, determining the most probable sense of a predefined target word based on its usage in the text. This function tokenizes each post, identifies the context of the target word, and returns the most likely sense from WordNet, enriching the DataFrame with a new column detailing these senses, as can be seen in the image below. The most frequent words and their definitions were also noted, with repeated disambiguations of financial keywords like stocks, derivatives, futures, SPY, etc.

	url	num_comments	body	created	cleaned_text	tokens	disambiguated_sense
	https://i.redd.it/8xp7l2atpskc1.jpeg	3	Lately I've seen big drops in Capacity Utiliza...	25/2/2024	lately seen big drops capacity utilization eur...	[lately, seen, big, drops, capacity, utilizati...	a supply of something available for future use
	://www.reddit.com/r/wallstreetbets/commen...	28	Make sure you're in the [WSB Discord] (https://...	25/2/2024	make sure wsb discord https check earnings thr...	[make, sure, wsb, discord, https, check, earni...	a supply of something available for future use
	https://i.redd.it/symyx8lniskc1.jpeg	20	I'm 18, convinced my mom to give me a grand to...	25/2/2024	convinced mom give grand trade split profit pu...	[convinced, mom, give, grand, trade, split, pr...	a former instrument of punishment consisting o...

Figure 24: Dataframe columns

Although this approach provides an automated way to disambiguate word meanings in a large dataset, it relies heavily on the quality of context and the extent of overlap with dictionary definitions, which might not capture all nuances, and was only useful in the manual labelling of the evaluation dataset.

4.2.5 Concept Extraction

Concept extraction followed a similar implementation with the purpose of preliminary data analysis. While slightly insightful, it observed the same issues of not being able to interpret contexts precisely which led to the next section.

4.2.6 Sarcasm Detection

By far the most common format of posts on popular financial subreddits are memes and satirical text, commonly seen in our extracted data from subreddits like r/wallstreetbets and XRP. Sarcasm, a common root for a lot of financial posts, was identified to be of great importance for feature extraction in order to better increase our classification score and performance of the NLP models.

In our approach to detect sarcasm in the cleaned data, we employed a rudimentary approach to initially identify potentially sarcastic texts, and using this subset we manually labelled texts that were deemed to be sarcastic or not sarcastic. This evaluation set was then used to finetune a pre-trained model to label our original dataset with sarcasm, to be used as features for classification. We employed the TextBlob library to perform sentiment analysis as a rudimentary proxy for detecting sarcasm. The method involved assessing the sentiment polarity of each text entry in the 'cleaned_text' column of a DataFrame, where texts exhibiting extremely positive (greater than 0.8) or extremely negative (less than -0.8) sentiment were labelled as "Potential sarcasm." This assumption was based on the premise that excessively strong sentiments in contexts that might not warrant such extremes could indicate sarcasm. Each entry was processed through a custom Python function detect_sarcasm, which appended a 'sarcasm' column to the DataFrame with labels distinguishing between "Potential sarcasm" and "Not sarcasm." Subsequently, entries identified as "Potential sarcasm" were extracted into a separate DataFrame as can be seen in the snapshot below:

num_comments	body	created	cleaned_text	tokens	disambiguated_sense	sarcasm
25	I've got some good losses saved on here... but...	21/2/2024	got good losses saved need want greatest close...	[got, good, losses, saved, need, want, greates...	a supply of something available for future use	Potential sarcasm
16	Equillium dropped 20 percent today. No news, n...	15/2/2024	equillium dropped percent today news nada hung...	[equillium, dropped, percent, today, news, nad...	a supply of something available for future use	Potential sarcasm
20	Hoping for the best with the one folks . ✌	19/12/2023	hoping best one folks	[hoping, best, one, folks]	the descendants of one individual	Potential sarcasm
3	What are the best Bitcoin/Crypto documentaries...	25/2/2024	best documentaries watch	[best, documentaries, watch]	a supply of something available for future use	Potential sarcasm

Figure 25: Dataframe with "sarcasm" column

Once all the potentially sarcastic data points have been identified, we manually labelled around 1150 points to be sarcastic or not sarcastic, on which we fine-tuned BERT-mini. Since most of the data points were neutral in nature as a lot of bots plague the subreddits with financial updates rather than opinions, sarcastic posts were quite sparse in nature with our manual labelling only finding about 415 data points to be actually sarcastic. This was incorporated and appended to the original data as a feature but was found to not be that useful for training the classification model as only 3.5% of the total dataset was sarcastic in nature, thus not being a good feature for the model to train on.

4.2.7 Review on Preprocessing:

While doing preprocessing, we noted some points, which are worth a mention in this report.

Firstly, we notice a lot of repeats/duplicates in messages online. These are due to many automated welcome/greeting messages within the Reddit system. We remove 109 duplicate values from the CSV file. An example is shown below:

- E.g. *Good Saturday morning to all of you here on r/EarningsWhispers. I hope everyone on this sub made out pretty nicely in the market this past week, and are ready for the new holiday-shortened trading week ahead.*

Furthermore, we also notice a lot of variability in length. Some posts are very short (1 sentence, only a title asking for advice, for example), while others are very long (a multi-paragraph summary on the week's crypto news for example). While we did not normalise or act on this information, we believe that predictions on longer data would naturally be more accurate as they have more words to work with.

We also notice that most data is neutral/news content. And the dataset may not be balanced as a result. However, efforts to balance the dataset by truncating neutral data points may reduce the size of the dataset significantly. It is also rather hard to classify a news summary of the week as positive or negative, as it is just a summary, and in some cases may contain *both* positive and negative news. Furthermore, there's also a subtle difference between news reporting on negative events (e.g. shares of a major bank tumbling) and comments on said event, while we manually classify the former to be "neutral", the latter could be said to be of "negative" sentiment.

4.2.8 The Need for Preprocessing

To answer Q4.2, we believe that whether there is a need for preprocessing of data ultimately depends on 2 factors:

1) The task/objective at hand, and
2) The model being used to predict

The current Reddit corpus has many words of informal tone and language (acronyms, emoticons, etc). While it is standard NLP practice to "clean" and normalise such words, it is more nuanced than that:

- Informal words (eg capitalisations) may hold key features in predicting the sentiment of the text (more specifically, the valence of the text, ie how strongly the emotions being conveyed are)
- Emoticons are, by definition, as straightforward and obvious an expression of sentiments as there is, and therefore should not be removed, if the model is able to parse and recognise them.

- Ultimately, it depends on the model. For example, as VADER was tuned on raw social media text, it probably does not require preprocessing. But other more traditional and multi-purpose NLP models such as FinBERT, preprocessing steps would probably be required.

4.3 Creation of Evaluation Set

A random 10% of all the data is extracted and manually labelled. It consists of 1003 rows, and is labelled according to 2 aspects:

- Subjectivity (neutral or opinionated)
- Sentiment (positive or negative)
    - Neutral only applies when subjectivity is neutral

Due to time and manpower constraints, each row is only labelled once by an annotator. Ideally, it would be good to have at least 80% inter-annotator agreement, but we opt for 3 labellers, each labelling half 500, 300, and 200 rows respectively.

	data	subjectivity	sentiment
2	Stock Up 400%! Pampa Metals $PMMCF Options New Gold - Copper Project in Argentina	opinio…	positive
3	Ethereum (ETH) Approaches $4000 In Multi-Week High, Mark Cuban Sees Strong Upside	neutral	neutral
4	After a 141% Surge, Golden Inu's Appears To Be Heading to a New ATH	neutral	neutral
5	(10/18) Tuesday's Pre-Market Stock Movers & News #Good morning traders and investors of the r/EarningsWhispers sub! Welcome to Tuesday! Here are y	opinio…	positive
6	Miner hosting. We accept any miner above 28th If you have old miners lying around with no place to host. Let us know. We are based in Malaysia and we h	neutral	neutral
7	Plug Power (NASDAQ: PLUG) Shares Soar on $1.6 Billion Loan News	opinio…	positive
8	Sony cuts PlayStation 5 sales forecast to 21 million units after posting record revenue Sony cut its sales forecast for its flagship PlayStation 5 console on W	neutral	neutral
9	01/19/24 [Join XRPLounge Discord] - discord.com/invite/XRPLounge-621074040169431052 # XRPLounge Discord	opinio…	positive
10	Ethereum Film Featuring Vitalik Buterin Raises Near $2M in ETH	neutral	neutral
11	Sam Altman plans to tap TSMC to rival Nvidia with his own AI chip (seeking partners and funds) More good news for TSMC, who already makes Nvidia's A	opinio…	negative
12	I'm confused about T-bills Hi,	opinio…	positive
13	What happened to the extra 55 Billion coins? Hi, ive been into the 'project' for a while and when it had been operating for a while, there were a few things th	opinio…	negative

Figure 26: A snippet of manually labelled data

Link to eval_set:

4.4 Model Setup

We adopt a **multi-task classification approach**, where 2 tasks are jointly tackled with one model. For this task, we implemented 2 models, namely VADER and FinBERT. As both

models have multi-class prediction for sentiment (Positive, Negative, or Neutral), we interpret the Neutral class as Neutral subjectivity as well, therefore tackling both subjectivity detection and sentiment analysis using one model.

For each model, we adopt different data processing pipelines. As VADER is trained on raw social media text, we choose to do minimal preprocessing, only removing duplicates and metadata, and simply feed VADER that data. For FinBERT, we thoroughly preprocess the data, with lemmatization and stopword removal as detailed above, and do an ablation study on the preprocessing methods as well. The inference pipelines are summarised below:

VADER pipeline:

Raw Data → Metadata and Duplicate Removal → Prediction with VADER

FinBERT pipeline:

Raw Data → Metadata and Duplicate Removal → Case Normalisation → Punctuation & Special Characters Removal → Stopword Removal → Lemmatization → Prediction with FinBERT

4.5 VADER Results

VADER outputs results in the format: `{'neg': 0.13, 'neu': 0.482, 'pos': 0.388, 'compound': 0.9337}`, where the compound score is a 'normalised, weighted composite score' [9], where -1 represents most negative sentiment and +1 is most positive sentiment. For our project, as we are only concerned about the final sentiment label and corresponding probability, we ignore the composite score, and we retrieve the model's final prediction with an argmax() over other scores.

On our crawled dataset of 9905 samples, on a CPU, it took 44.6 s. This showcases the scalability of VADER, and it makes sense as VADER is a rule-based model implementing several syntactic and grammatical rules.

Table 5: Results of Eval Set on VADER

[9] https://github.com/cjhutto/vaderSentiment?tab=readme-ov-file#about-the-scoring

	Precision	Recall	F1 Score	Support
Negative	0	0	0	192
Neutral	0.56	1	0.72	560
Positive	0.67	0.02	0.03	251
Accuracy	0.56			1003
Weighted Average	0.48	0.56	0.41	1003

VADER F1 Score: 0.560

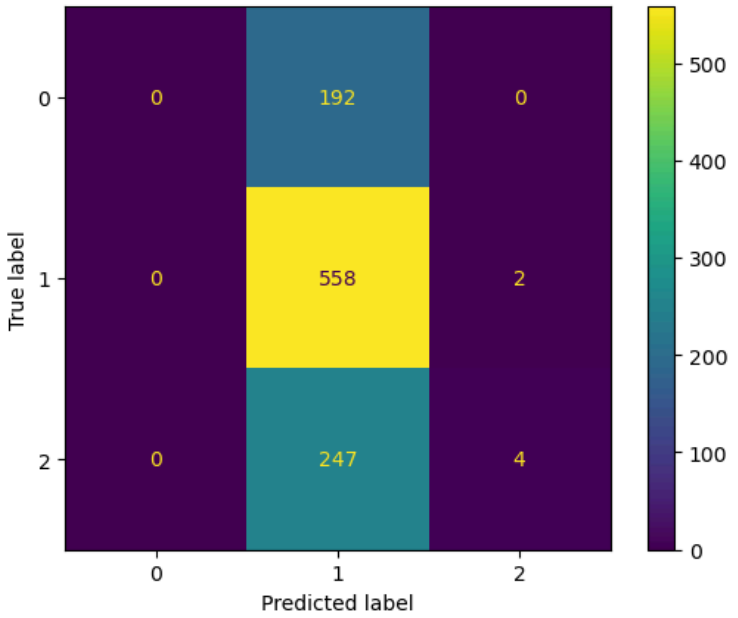


Figure 27: Confusion Matrix of VADER's performance on evaluation set

Some comments on results:

- Definitely not ideal
- Maybe due to the fact that annotations are subjective, the vast majority of data points in the evaluation set can indeed be construed to be neutral
- Very fast results and therefore scalable to large datasets

4.6 FinBERT Results

FinBERT outputs results in the format: `{'label': 'negative', 'score': 0.9513100385665894}` for each sample. We take the results as it is.

On our crawled dataset of 9905 samples, on a CPU, the inference took 40 mins 9s. This speaks to one of the disadvantages of using Deep Learning-based models, which is that they are much more computationally intensive and require significant resources (e.g. GPUs) to scale effectively. Therefore, FinBERT is much less scalable.

Table 6: Results of Eval Set on FinBERT

	Precision	**Recall**	**F1 Score**	**Support**
Negative	0.41	0.20	0.27	192
Neutral	0.57	0.82	0.67	560
Positive	0.36	0.16	0.22	251
Accuracy	0.54			1003
Weighted Average	0.49	0.54	0.48	1003

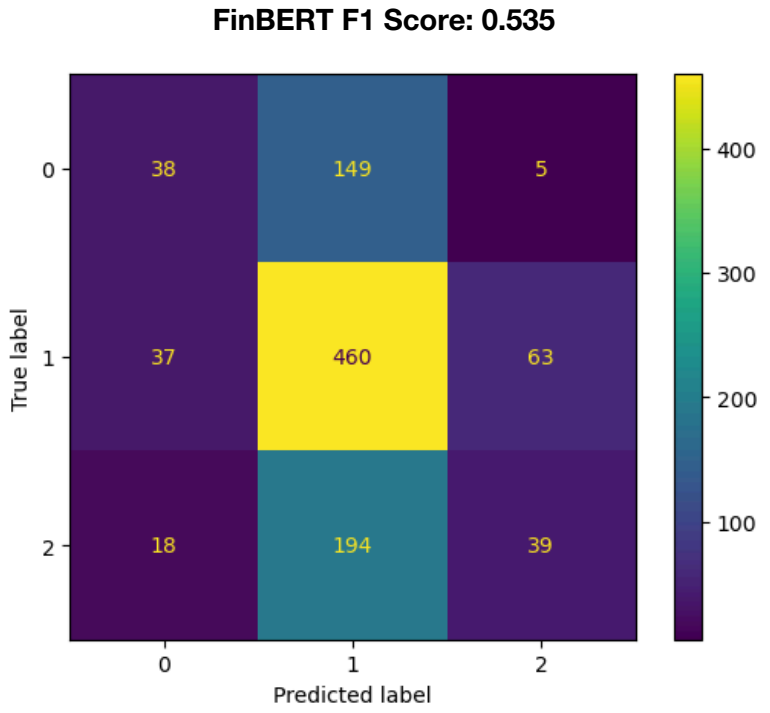


Figure 28: Confusion Matrix of FinBERT's performance on evaluation set

In conclusion, VADER performs marginally better, in comparison to FinBERT.

4.7 FinBERT Fine-Tuning

The VADER (Valence Aware Dictionary and Sentiment Reasoner), is a lexicon and rule-based sentiment analysis tool specifically attuned to sentiments expressed in social media. Typically, models like VADER and TextBlob are not trained or fine-tuned on new datasets because they aren't machine learning models, but rather a set of hardcoded rules combined with sentiment lexicons.

As such, even though VADER was seen to perform better, we used FinBERT to finetune the model on our manually labelled data, to see if the performance would increase.

Necessary libraries were imported and the FinBERT pre-trained model and its corresponding tokenizer were used to preprocess the text data, ensuring that each input string was tokenized and encoded according to the model's requirements. This preprocessing step included padding and truncating the text to a uniform length, facilitating batch processing during training.

Subsequently, the dataset was divided into training and validation subsets with a 70/30 train/test split. The training process involved 10 epochs where each batch of data was fed through the model, and the network weights were adjusted via back propagation based on the computed loss and accuracy metrics. To optimise our model, we utilised the AdamW optimizer, known for its effectiveness in adjusting learning rates and stabilising the training process in deep learning tasks. The code can be found in the *finetuned.ipynb* file, and the accuracy for the FinBERT model after fine-tuning was seen to be **0.72,** which was the highest and was used for generating the subjectivity and polarity scores for the original dataset, which was used for indexing.

4.8 Ablation of Preprocessing Techniques

We also conducted an ablation study over preprocessing techniques for VADER. We investigated all combinations of the following preprocessing steps:

- Special Characters Removal

- Stopword Removal
- Lemmatization

Notation: Y means Yes, N means No

Special Characters Removal	Stopword Removal	Lemmatization	VADER Accuracy
N	N	N	0.5623
Y	N	N	0.5623
N	Y	N	0.5623
N	N	Y	0.5623
Y	Y	N	0.5623
N	Y	Y	0.5623
Y	N	Y	0.5623
Y	Y	Y	0.5623

Table 7: Results of Ablation Study on Preprocessing Techniques

Interestingly, the same results occur, where VADER predicts everything the vast majority of data to be neutral. Some data that VADER predicts as non-negative is shown below in Table 8. As a result, all ablation tests result in the same accuracy. This is interesting as it suggests that preprocessing has little to no effect on the outcome of rule-based models.

data	Sentiment groundtruth	Vader prediction	vader_prob
Perfect timing Best day trading so far	positive	pos	0.612
Read Good information	positive	pos	0.592
First yolo	positive	pos	0.677
Stay Irrational My Friends	positive	pos	0.413
Interesting thoughts	neutral	pos	0.73
Gas fees like	negative	pos	0.556
SMCI great Woke up nice account	positive	pos	0.633

Genesis Granted Approval Sell 1 6 Billion GBTC Shares	neutral	pos	0.549
4k woot	positive	pos	0.737

Table 8: Sample of data that VADER predicts as non-neutral

Overall, preprocessing the data with syntactic and semantics layers in mind proved to provide the most effective improvements on the barebones pre-trained models. While still not the best, the validation accuracies improved to 0.72 with the fine-tuned FinBERT model after all the preprocessing steps in the pipeline as compared to 0.51 on the validation set before. While attempts at concept extraction and sarcasm detection were to better contextualise the financial data and its comments, due to the neutral nature of the majority of the posts these features were seen to provide only marginal improvements. This finding is also in line with research online on existing attempts of classifying Reddit data using sentiment analysis, as models on Hugging Face have also encountered a vast majority of the data to be neutral in nature

(such as here: https://huggingface.co/datasets/SocialGrep/the-reddit-irl-dataset).

5. Conclusion

In conclusion, our team has successfully built a Reddit financial data search engine where users can search for financial posts from prominent subreddits. This will help them gauge the investor sentiment related to various topics and also allow them to extract detailed analyses of topics posted by other investors.

While sentiment classification remains a challenging problem (especially when a corpus of Reddit posts is concerned), our team was able to achieve a classification accuracy of 72% on the validation dataset. This was a remarkable improvement over the 51% initial accuracy that was achieved before the implementation of our preprocessing pipeline and implementation of the FinBERT model. We hope that this will allow the users of our search engine to reliably retrieve information about various financial topics and serve as a tool for furthering their financial analysis. Although we were able to achieve 72% accuracy, future works can explore other preprocessing techniques and models to further build on our work. Sentiment classification on Reddit posts related to finance can be particularly challenging as the datasets usually contain a lot of neutral stances. Extracting a larger corpus of data and building a better-quality dataset will help improve the accuracy and can be explored in future works.

Overall, this project has allowed us to explore the various steps involved in building a functional search engine, as well as the preprocessing and model selection/fine-tuning steps necessary for good classification results. Through extensive research and development, we hope that our search engine can serve our users' needs and provide them with accurate and fast retrieval of information related to the financial topics and associated sentiments that they may be interested in.