NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CE/CZ4034 Information Retrieval

Assignment Report

Group 17

S/N	Name	Matriculation Number	Contributions
1	Chen Xueyao	U1922640G	Crawling, Classification
2	Chua Yixuan	U2021832B	Crawling, Classification
3	Lang Yuezhu	N2203148E	Crawling, Classification
4	Luo Xiaoyang	U2022803B	Indexing, Classification
5	Ng Ching Ting	U1921431C	Indexing, Classification

Table of Contents

1. Introduction

For the CE/CZ4034 Information Retrieval assignment, our team is required to build an information retrieval system that includes sentiment analysis using machine learning. Our team will crawl a text corpus based on a topic of our interest and build a search engine around the data we have crawled. Finally, we perform sentiment analysis on each of the data to provide more insights for the user.

1.1 Selection of Topic for Course Assignment

Our team selected the topic of "2023 Recession" for our project because we feel that it is a relevant and timely topic. The COVID-19 pandemic has caused significant disruptions in the global economy, and many countries are still struggling to recover. The International Monetary Fund (IMF) has predicted that the global economy will grow by 2.9% in 2023, which is lower than the growth rate of 6.0% in 2021 [1]. This prediction suggests that there may be a recession in 2023.

Analysing this data can help businesses adjust their strategies to meet the needs of their customers during a recession. We can identify how people feel about the economy and their financial situation and use this information to determine the general sentiment of the Recession in 2023.

1.2 Problem Statement

The use of social media platforms like Twitter and Reddit has granted us the privilege of sharing our thoughts and opinions on the web without undue fear or stringent limitations. Thus, without the need to conduct individual surveys of every individual worldwide, we can seamlessly glean the diverse perspectives and attitudes of users hailing from disparate backgrounds simply by crawling and analysing their Reddit posts and comments using machine learning. The assimilation and analysis of this vast array of viewpoints can provide invaluable insights into the complex interplay of human emotions and rationality, thereby enabling us to better comprehend the nuanced fabric of societal sentiments.

With this in mind, our team's project objective is to create an information retrieval system that enables the user to enter specific keywords, generating a vast collection of Reddit posts and comments along with their respective classifications. This system would enable the user to gain helpful insights and conduct further analysis on selected features, which would ultimately impact the classification of the Reddit posts and comments. Therefore, we would like to use an information retrieval system to analyse market sentiments on whether there is a recession going on, and hope to gain valuable public insights in a systematic way.

2. Crawling

2.1 Methodology for Crawling of Data

The crawling was done on Reddit's API [2]. This usage of Reddit API is free-of-charge, though it has certain limitations such as maximum rate limit. As the project was done on Python, we used the Praw library [3] to access the Reddit post and comment data.

We crawled for the following information from Reddit:

Field	**Description**
Subreddit	The overall topic that the text falls under
Author	The username of the reddit account that created the post/comment
Created Date	Date the post/comment was created in UTC
Score	The aggregate score of a post/comment (Total upvotes - total downvotes)
Text	The content of the post/comment

In order to see how the recession affects different target groups, we selected subreddits in different industries and countries to collect data from. For countries, we sampled r/singapore, r/india, r/usa, r/china, and r/unitedkingdom. For industries, we sampled r/FinancialCareers, r/cscareerquestions, and r/medicine.

We used the following keywords to search for post and comment data related to the recession: ‘*layoff’, 'recession', 'economic downturn', 'unemployment', 'debt', 'slowdown', 'financial crisis' and 'poverty'.*

While we were crawling the corpus, we noticed that the different subreddits are of different membership sizes ranging from as many as 1.4 million members and as little as 44.9 thousand members. As such, the subreddits contribute unequal amounts of data to our corpus. The distribution of data can be seen from Figure 2.1.1.

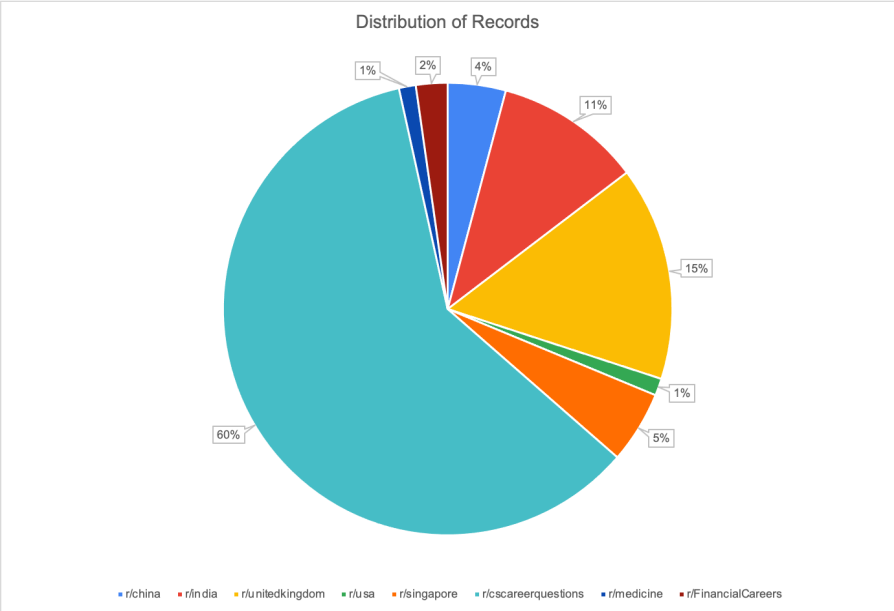


Figure 2.1: Distribution of data based on subreddit

The corpus retrieved is stored in a comma-separated values (.csv) file that combines post and comment data. Each row represents one post or comment.

2.2 Analysis of Crawled Data

Information that users might want to retrieve:

1. How hard the different countries have been hit by the recession. The user might want to compare the impact of the recession across different countries.
2. Users who are involved in the three listed industries. They may be prospective employees considering a migration to the listed industries, hoping to determine the severity of the impact of the recession on that industry. They may also already be part of the industry and may wish to know the future of the industry or how to retain gainful employment in the midst of the recession.

3. Users may be interested in the situation as a whole and may wish to investigate the trends and patterns found in the recession. They may wish to determine the general sentiment that the public have on the recession.

Example Queries:

1. Find from the corpus the general sentiment of Singapore residents towards the recession.
2. Find from the corpus which country is feeling the most positive about the recession.
3. Find from the corpus which industry is feeling the most positive about the recession.
4. Find from the corpus how many people working in the healthcare industry have been laid off.
5. Find from the corpus the unemployment rate in the financial sector.

2.3 Statistics of Crawled Data

Total number of posts and comments : 20,559

Number of words: 955, 882

Number of unique words: 27,806

3. Indexing

3.1 Data Indexing and Querying using Apache Solr

Solr is an open-source enterprise search platform that is popular for full-text search, real-time indexing, database integration and rich document handling [4]. It includes on-the-fly tokenization, spell check and also stop-word removals. Anyone can use the REST API to communicate with Solr and to parse data from Solr to search engines [5].

3.1.1 Data Fields

Field	**Description**
dataset	Dataset the Reddit data belong to - train/test
category	Category the Reddit data belong to - comment/post
subreddit	Subreddit the Reddit data belong to
author	Author of the Reddit data
created_date	Created date of the Reddit data
score	Aggregate score of the Reddit data (Total upvotes - total downvotes)

text_clean	Pre-processed content text of the Reddit data
final_label	Sentiment of the Reddit Data - Positive/Neutral/Negative

3.1.2 Solr Configurations

We did custom configurations to solrconfig.xml files to introduce additional features for our users to have a better search experience.

3.1.2.1 Spell Checking

To enable spell checking [6], we added spell check function and dictionary to the main '/select' query. When there is a spelling error by the user, Solr will filter and suggest top 10 relevant terms that are actually present in our documents.

```
733   <!-- Primary search handler, expected by most clients, examples and UI frameworks -->
734   <requestHandler name="/select" class="solr.SearchHandler">
735     <lst name="defaults">
736       <str name="echoParams">explicit</str>
737       <int name="rows">10</int>
738       <str name="spellcheck.dictionary">default</str>
739       <str name="spellcheck">on</str>
740       <str name="spellcheck.count">10</str>
741     </lst>
742     <arr name="last-components">
743       <str>spellcheck</str>
744     </arr>
745   </requestHandler>
```

Figure 3.1.2.1a: Adding spellcheck

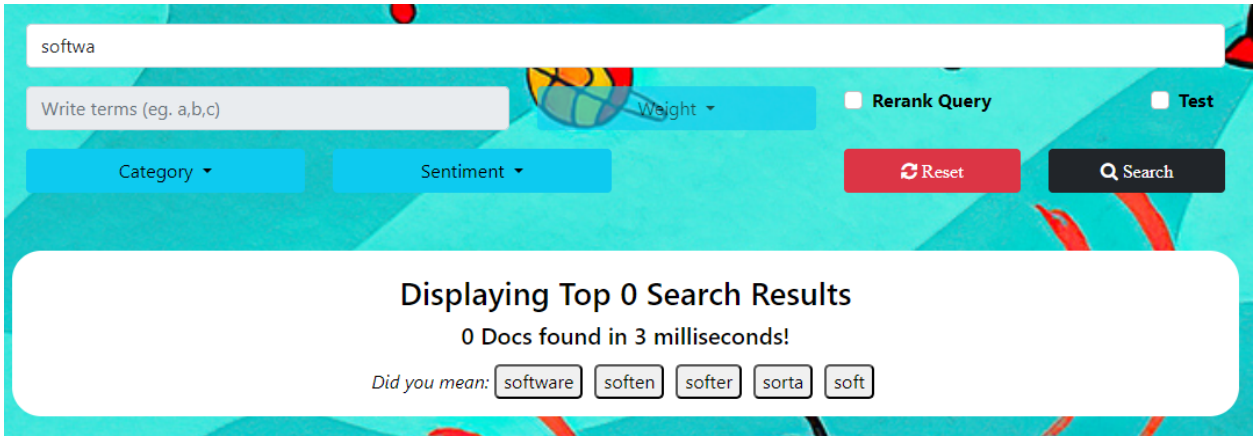


Figure 3.1.2.1b: Suggestions for misspelt term 'software'

3.1.2.2 REST API Communication

To enable the permission for REST APIs [5], we added an additional block of code in the webapp\WEB-INF file.

```
<filter>
  <filter-name>cross-origin</filter-name>
  <filter-class>org.eclipse.jetty.servlets.CrossOriginFilter</filter-class>
  <init-param>
    <param-name>allowedOrigins</param-name>
    <param-value>*</param-value>
  </init-param>
  <init-param>
    <param-name>allowedMethods</param-name>
    <param-value>GET,POST,OPTIONS,DELETE,PUT,HEAD</param-value>
  </init-param>
  <init-param>
    <param-name>allowedHeaders</param-name>
    <param-value>origin, content-type, accept</param-value>
  </init-param>
</filter>

<filter-mapping>
  <filter-name>cross-origin</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

Figure 3.1.2.2: REST API permission fields

3.1.3 Indexing in Solr

We index the documents in Solr by creating a core in its cloud called ‘reddit’ and upload our data .csv file in raw form via Config API’s ‘/update’ [5].

3.1.4 Querying in Solr

After the data has been indexed successfully into Solr, we can perform different types of search queries to retrieve our desired documents. Our search engine will communicate with Solr through the use of REST APIs. To query for documents, we can use the GET request. Solr then will return all documents relating to the user query in JSON format [7]. An example of how Solr returns it in JSON format:

http://localhost:8983/solr/reddit/select?indent=true&q.op=OR&q=text_clean%3Ajob&useParams=


```
{
  "responseHeader":{
    "zkConnected":true,
    "status":0,
    "QTime":4,
    "params":{
      "q":"text_clean:job",
      "indent":"true",
      "q.op":"OR",
      "useParams":"",
      "_":"1680758729399"}},
  "response":{"numFound":2780,"start":0,"numFoundExact":true,"docs":[
      {
        "dataset":["train"],
        "category":["comment"],
        "subreddit":["r/FinancialCareers"],
        "author":["foodVSfood"],
        "created_date":["10/13/2022 18:14"],
        "score":[1],
        "text":["Finding a job is much easier when you have a job. Treat the job search like you d
```

```
  "text_clean":["finding a job is much easier when you have a job treat the job search like
  "label_1":["POSITIVE"],
  "label_2":["POSITIVE"],
  "label_3":["NEUTRAL"],
  "final_label":["POSITIVE"],
  "text_clean_stopword":["finding job much easier job treat job search like dont job right"]
  "id":"0eb1e522-299a-4dd5-ba41-10ea5a5cf6c2",
  "_version_":1762076364452659200},
{
  "dataset":["train"],
  "category":["comment"],
  "subreddit":["r/cscareerquestions"],
  "author":["Countrydeepfrypie"],
  "created_date":["12/1/2022 2:58"],
  "score":[3],
  "text":["200 job applications in one night, 5 second interviews, 3 job offers, one job suc
  "text_clean":["job applications in one night second interviews job offers one job successf
  "label_1":["POSITIVE"],
  "label_2":["POSITIVE"],
  "label_3":["NEUTRAL"],
  "final_label":["POSITIVE"],
  "text_clean_stopword":["job applications one night second interviews job offers one job su
  "id":"ed8d7982-c90e-449b-9889-5f69665c9a34",
  "_version_":1762076362651205633},
{
  "dataset":["train"],
  "category":["comment"],
  "subreddit":["r/cscareerquestions"],
  "author":["awesome_budo"],
  "created_date":["5/5/2021 20:49"],
  "score":[1],
  "text":["This is technically my third software engineering job. The first job lasted for 1
  "text_clean":["this is technically my third software engineering job the first job lasted
  "label_1":["NEUTRAL"],
  "label_2":["POSITIVE"],
  "label_3":["NEUTRAL"],
  "final_label":["POSITIVE"],
  "text_clean_stopword":["technically third software engineering job first job lasted months
  "id":"2d4f0c41-8424-4db0-9af8-6877726f89c4",
  "_version_":1762076367876259842},
```


Figure 3.1.4: Query “job” results

3.2 User Interface (UI)

The frontend UI for performing interactive queries is built using HTML, CSS, Javascript programming languages with Bootstrap5 [8], amCharts [9], AnyChart [10] third-party libraries. Aside from basic querying, the main aim of our UI is to also provide the best user interactivity and experience.

3.2.1 Search Interface

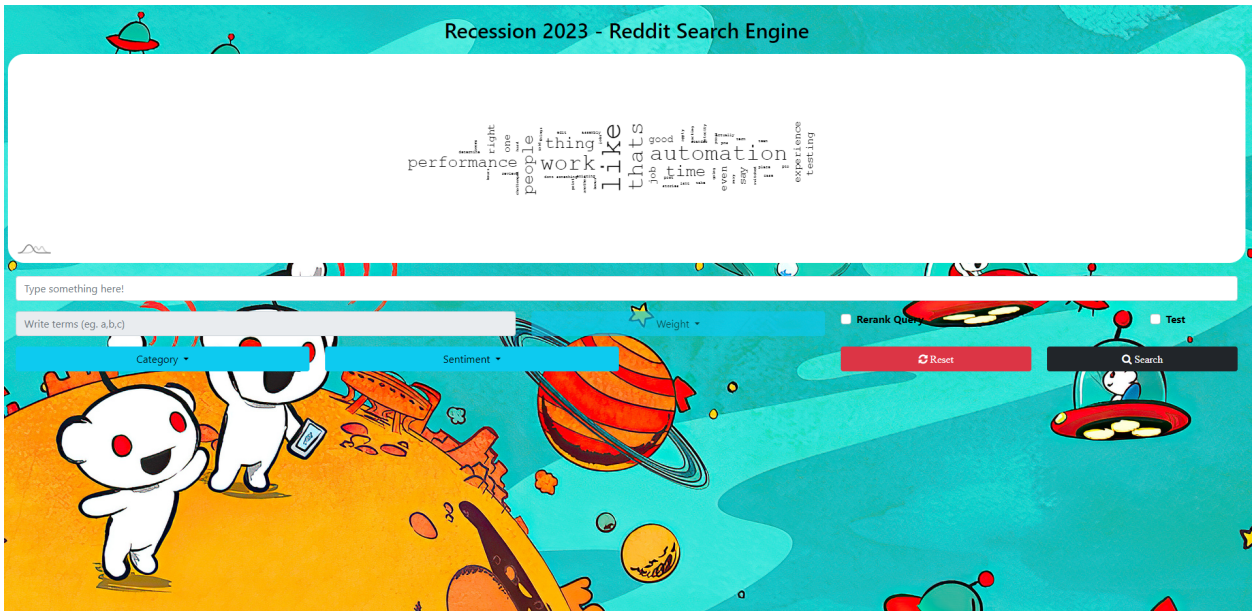


Figure 3.2.1: Search interface of the app showing the search bar

Figure 3.2.1 depicts the search interface of our app, users can type in their query in the search bar. He/She then can click on the ‘Search’ button to get relevant information

from Solr based on their search input.

Above the search bar is a word cloud displaying the most common terms found in our Reddit corpus. Bigger words in the word cloud emphasises higher count of the term. These could be inspiration for users to formulate their queries.

3.2.2 Advanced Search

3.2.2.1 Categorization

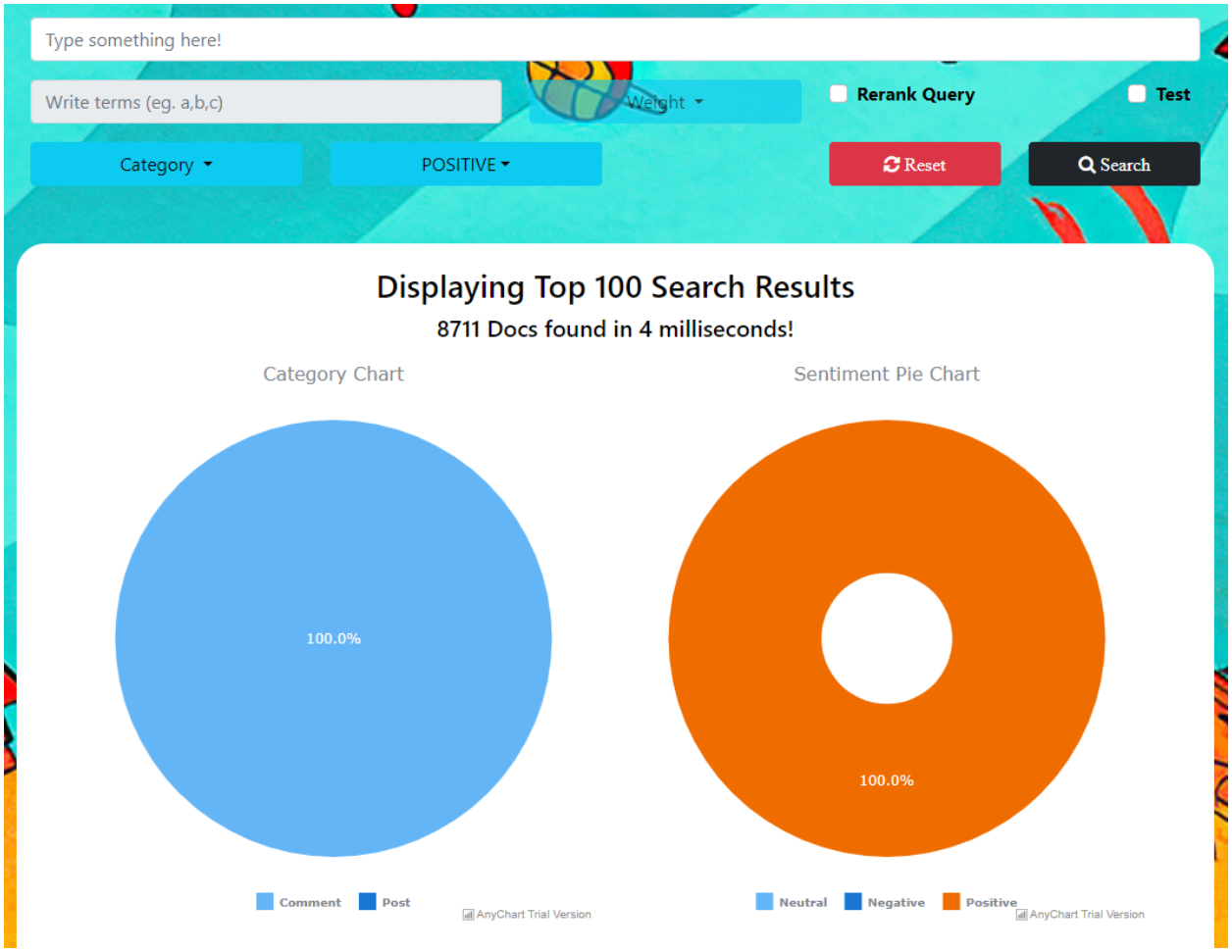


Figure 3.2.2.1: Advanced search options for positive sentiment Reddit data

Advanced search options in specific column values allow users to get a more targeted result.

Our search engine has 2 column filters for Reddit data:

Filter	Options
Category	Comment - return Reddit comments Post - return Reddit posts
Sentiment	Positive - return Reddit data with ‘Positive’ label Neutral - return Reddit data with ‘Neutral’ label Negative - return Reddit data with ‘Negative’ label

3.2.2.2 Re-rank Query

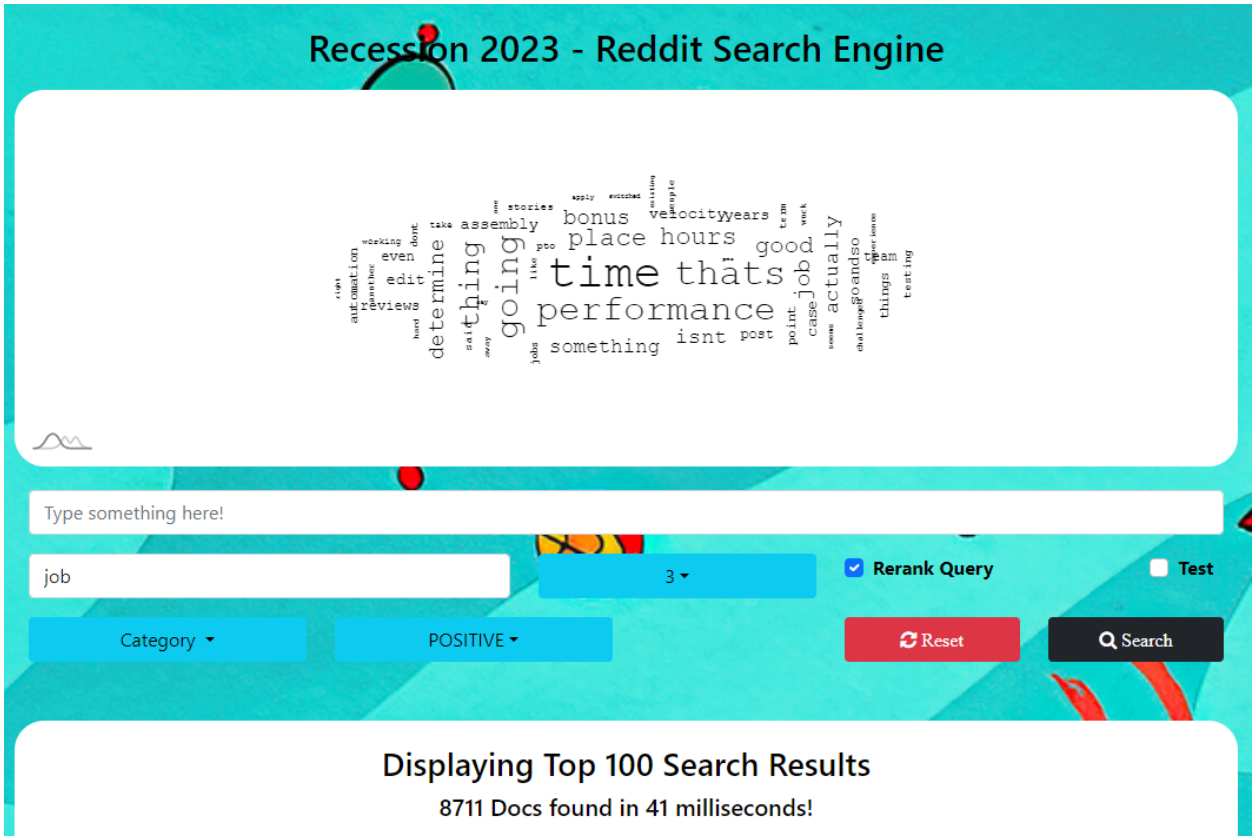
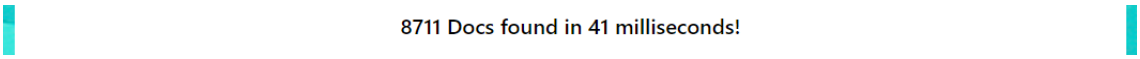


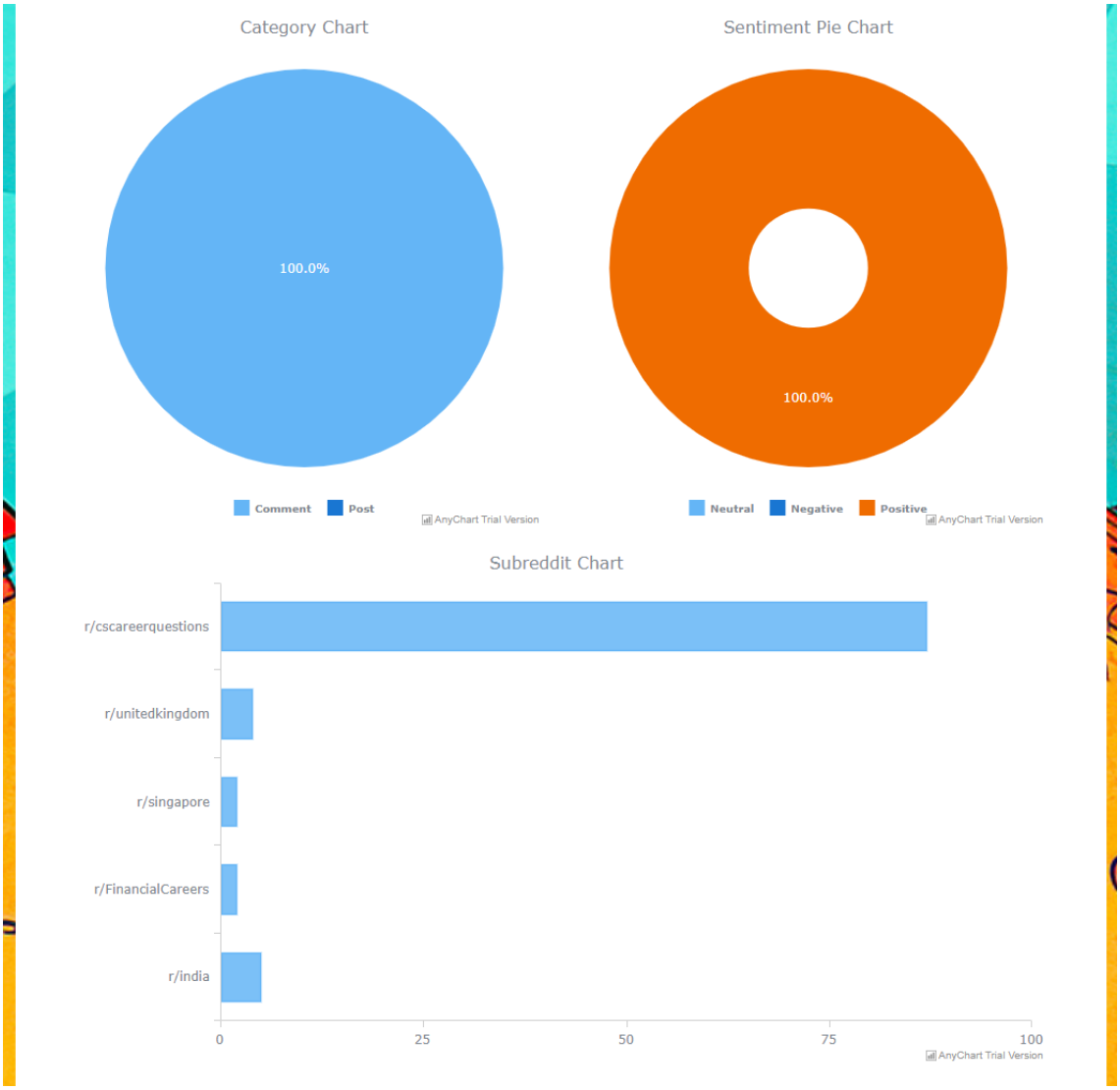
Figure 3.2.2.2: Re-ranked weights for search results with term ‘job’

In advanced search, we also allow users to re-rank their query by specifying keywords and their desired weights. For example, we query “job” and we can re-rank the keywords like “layoff”, “job hunting” with higher weights (value > 1) such that for the top 1000 documents initially, Solr re-computes the TF-IDF scores to return even more relevant results base on the user’s wishes [11].

3.2.3 Result Interface

After clicking on the ‘Search’ button, a table of top 100 Reddit data (comments and/or posts) is returned, ranked from the most relevant to least relevant based on their TF-IDF scores. The order of results visualizations are as follows:

No.	Visualization
1	Total number of display results 
2	Total number of documents related to search term & Time taken to query Solr for results 
3	Charts are easily interpretable forms of search results. Pie Chart: show proportion of comments vs posts Doughnut Chart: show proportion of different sentiments Bar Chart: show distribution of subreddit origin



4 Data Card:

r/china (*@SuspiciousStable9649*)

workers can see the ceiling due to market maturations hu said and government policies are now not that favourable to big internet its just not very stable now government policy is more favourable to what we call hardcore emerging technical industries like ai cloud computing biotech and other infrastructure yuwan is a dumbass edit apologies yuwan is correct the policies are dumbass edit no yuwan is a dumbass for calling it maturations

Category: comment **Date Created:** 5/14/2022 3:53

Reddit Score: 1 **Sentiment:** NEGATIVE

TF Value: 1 **IDF Value:** 1 **TF-IDF Value:** 1

NB	SVM	XGBoost	Stacked	BERT
NEGATIVE	NEGATIVE	POSITIVE	NEGATIVE	NEUTRAL

Showing

- Subreddit
- Author
- Category
- Date Created
- Reddit Score
- Sentiment (original)
- TF Value
- IDF Value
- TF-IDF Value
- Sentiment (by each classification model)

Color	Sentiment
Green	Positive
Yellow	Neutral
Red	Negative

3.3 Test Query Results and Performance

Id	Query Term	Number of Results	Top Found Result	Query Time (ms)
1	challenged	3	r/cscareerquestions (*@CoderDispose*) pto is seen as similar to a bonus right and once youre given the bonus they cant take it away how can they actively say you have hours of pto right now and when you quit they say actually weve taken those hours back this seems like the sort of thing that happens but isnt heavily challenged so the decision isnt out on it but maybe im just crazy edit i saw another post explaining this and it seems like it has been challenged and the law is just shit lol **Category:** comment **Date Created:** 1/20/2023 20:21 **Reddit Score:** 1 **Sentiment:** POSITIVE **TF Value:** 0.4994734 **IDF Value:** 8.678291 **TF-IDF Value:** 4.3345757	1
2	job	2780	r/FinancialCareers (*@foodVSfood*) finding a job is much easier when you have a job treat the job search like you dont have a job right now **Category:** comment **Date Created:** 10/13/2022 18:14 **Reddit Score:** 1 **Sentiment:** POSITIVE **TF Value:** 0.8429179 **IDF Value:** 2.000668 **TF-IDF Value:** 1.686399	19
3	team	472	r/cscareerquestions (*@kennyroach*) i mean team size company size company and team culture how the pms operate etc **Category:** comment **Date Created:** 5/5/2021 23:40 **Reddit Score:** 1 **Sentiment:** NEGATIVE **TF Value:** 0.77205426 **IDF Value:** 3.7730162 **TF-IDF Value:** 2.9129732	9
4	bonus	124	r/FinancialCareers (*@sefarrell*) start looking asap bonus move try to leverage your severance as the bonus it is into a sign on bonus **Category:** comment **Date Created:** 10/13/2022 0:58 **Reddit Score:** 1 **Sentiment:** POSITIVE **TF Value:** 0.8135986 **IDF Value:** 5.1067486 **TF-IDF Value:** 4.1548433	12
5	reviews	57	r/cscareerquestions (*@audaciousmonk*) no code reviews bad company **Category:** comment **Date Created:** 7/24/2021 17:07 **Reddit Score:** 2 **Sentiment:** NEGATIVE **TF Value:** 0.71590644 **IDF Value:** 5.879269 **TF-IDF Value:** 4.209007	6

3.4 Innovations

3.4.1 Enhanced Search

3.4.1.1 Word Cloud

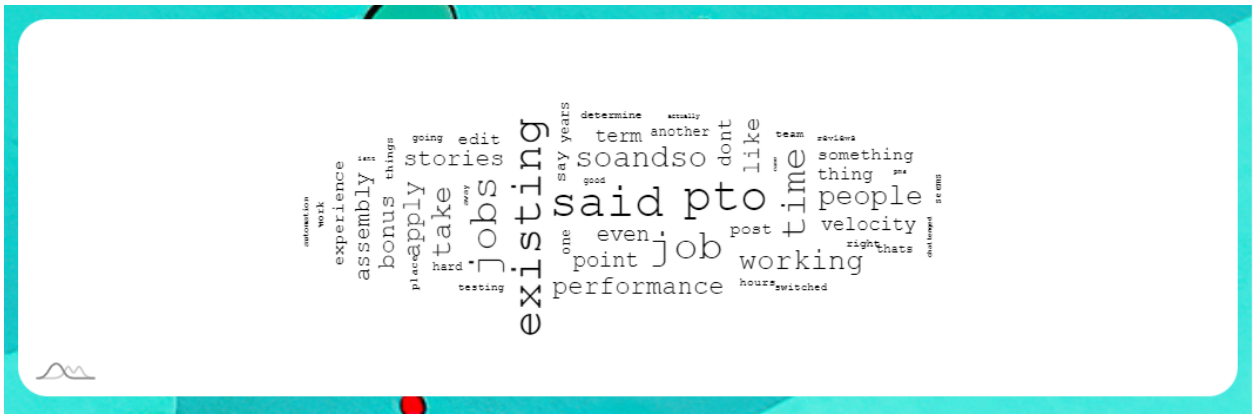


Figure 3.4.1.1: Wordcloud

The word cloud is a visualization that tells the user what are the most used terms in our documents. It also measures what is the general view and outlook on ‘Recession 2023’. More significant words appear larger in the word cloud. Users can then view and understand the economic situation at a glance and potentially gain valuable insights without even starting to search.

3.4.1.2 Visualization of data via charting

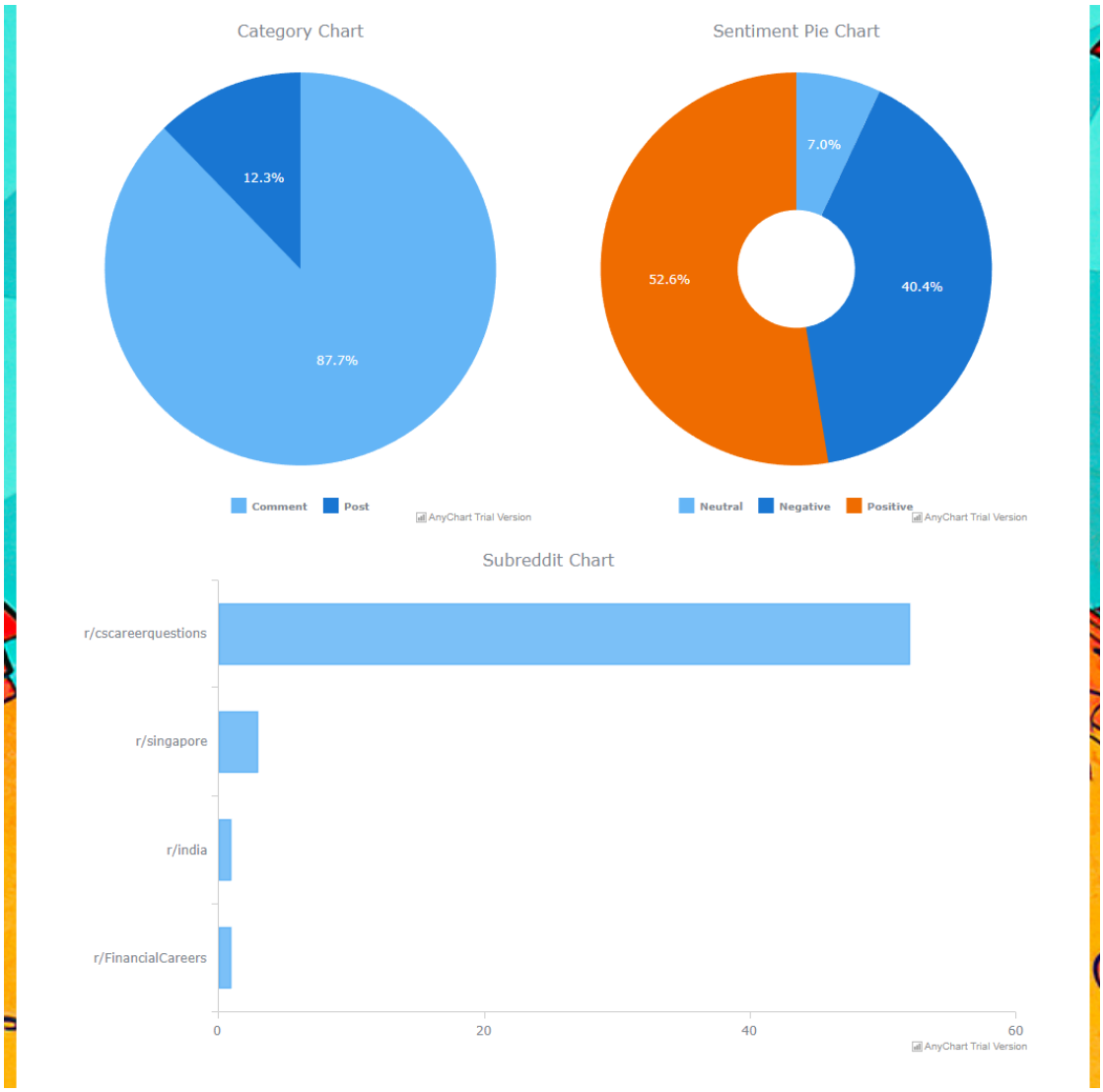


Figure 3.4.1.2: Pie Chart, Doughnut Chart & Bar Chart

The multiple use of charts is to enhance user experience as chart drawings are often easy to digest and interpret. We can visualize through charts and learn that our results are maybe skewed to a certain variable such as being a comment/post, having positive/neutral/negative sentiment, predominantly coming from certain subreddit's

users. This then allows users to gauge and/or fine-tune their queries to get results of their expectations.

3.4.2 Interactive Search

3.4.2.1 Re-Rank Query

We have the option that allows users to re-rank query during their search. By adding weights to specific keywords, the search results might be different due to the newly computed TF-IDF scores at Solr. This could potentially generate a more user-directed and user-relevant search result as users' feedback are gathered at this optional input.

Figure 3.4.2.1 (before): Top 3 results for no re-ranked query

Figure 3.4.2.1 (after): Top 3 results for re-ranked query of term 'job' given weight of 3

3.4.3 Multifaceted Search

3.4.3.1 Categorization of Results

From Figure 3.2.2.1, we have the option that allows users to filter values in different columns. Based on the selected advanced search inputs, the results returned will strictly follow that particular criterion. Users can select comments or posts for the category. For the sentiment, users can select from Positive, Neutral, or Negative. By allowing users to set certain column search values, it provides them with streamlined searches that suit their needs. Moreover, the filtered results statistics will be visualized in pie chart, doughnut chart, bar chart for their perusal.

4. Classification

In this section, we will describe the classification techniques used to perform on each of the reddit comments and posts to conduct sentiment analysis and detect subjectivity, polarity within each data. Subjectivity will first differentiate neutral vs opinionated and within opinionated, polarity will classify into positive vs negative.

4.1 Approaches to Classification Problem

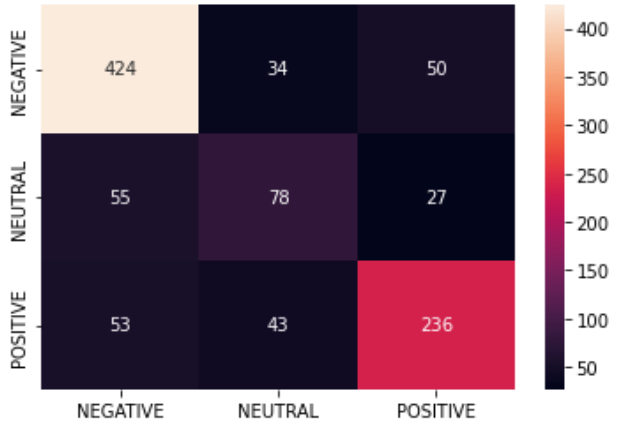
To solve the Natural Language Processing (NLP) classification problem, our group decided to conduct both traditional Machine Learning (ML) and Deep Neural Networks (DNN) methods to compare.

4.1.1 Bidirectional Encoder Representations from Transformers (BERT)

BERT is a state-of-the-art language model designed to pre-train deep bidirectional representations from unlabeled text by considering both left and right contexts in all layers [12]. Composed by DNN, it is best trained on large unlabelled data to achieve great results without needing substantial task-specific model architecture modifications. BERT is able to model complex relationships between words and phrases through language inference [12]. The transformer architecture encodes the text in a bidirectional manner, capturing the context of each word in relation to its surrounding words.

4.1.1.1 Details

	precision	recall	f1-score	support
NEGATIVE	0.80	0.83	0.82	508
NEUTRAL	0.50	0.49	0.50	160
POSITIVE	0.75	0.71	0.73	332
accuracy			0.74	1000
macro avg	0.68	0.68	0.68	1000
weighted avg	0.74	0.74	0.74	1000




```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 reddit (InputLayer)            [(None,)]            0           []

 preprocessing (KerasLayer)     {'input_type_ids':   0           ['reddit[0][0]']
                                (None, 128),
                                 'input_word_ids':
                                (None, 128),
                                 'input_mask': (Non
                                e, 128)}

 BERT_encoder (KerasLayer)      {'encoder_outputs':  28763649    ['preprocessing[0][0]',
                                 [(None, 128, 512),               'preprocessing[0][1]',
                                 (None, 128, 512),                'preprocessing[0][2]']
                                 (None, 128, 512),
                                 (None, 128, 512)],
                                 'pooled_output': (
                                None, 512),
                                 'sequence_output':
                                 (None, 128, 512),
                                 'default': (None,
                                512)}

 dropout (Dropout)              (None, 512)          0           ['BERT_encoder[0][5]']

 classifier (Dense)             (None, 3)            1539        ['dropout[0][0]']

==================================================================================================
Total params: 28,765,188
Trainable params: 28,765,187
Non-trainable params: 1
__________________________________________________________________________________________________
```

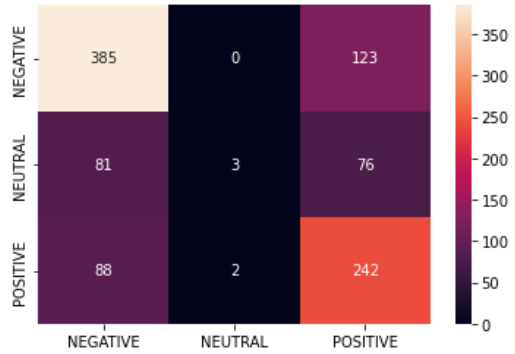
4.1.2 Naives Bayes (NB)

NB was once a popular algorithm due to its simplicity, efficiency, and relatively good accuracy. It is a probabilistic classifier centered around Bayes Theorem, $P(A|B)$ = $P(B|A)P(A)$ / $P(B)$ [13]. Assuming that the words are independent of each other, it calculates the probability of each class for a given input text based on the frequency of each word in text. Despite its naive assumption, it performs well in high-dimensional, sparse data, which are common for NLP tasks.

For our NB classifier, we further tune the hyperparameter: Alpha which controls the laplace smoothing. Laplace smoothing helps the model avoid the problem of zero probability [14] when it encounters words that are present in the test set but not in the training set. Having an optimal value of Alpha therefore makes the model more robust.

4.1.2.1 Details

	precision	recall	f1-score	support
NEGATIVE	0.69	0.76	0.73	508
NEUTRAL	0.60	0.02	0.04	160
POSITIVE	0.55	0.73	0.63	332
accuracy			0.63	1000
macro avg	0.61	0.50	0.46	1000
weighted avg	0.63	0.63	0.58	1000



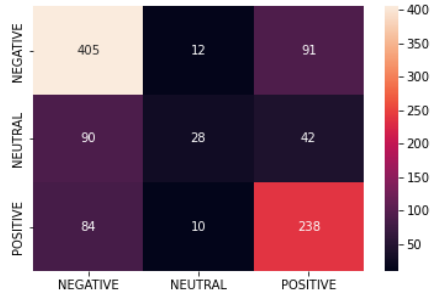
4.1.3 Support Vector Machines (SVM)

SVM is a supervised learning method commonly used for classification, regression and outlier detection. For classification, SVM separates different target classes in a hyperplane in n-dimensional space. It finds the optimal hyperplane that separates the different classes of data points and draws that hyperplane using mathematical functions called “Kernels” [15].

For our SVM classifier, we tuned the hyperparameters: C, Gamma and Kernel. C is a hyperparameter which enforces a penalty on the error term, somewhat justifying how much error tolerance we give to our SVM model [16]. Gamma defines the influence of a single training sample, controlling the shape of the decision boundary [16]. A small gamma value creates a smoother boundary while a large gamma value creates a more complex boundary. Having optimal C, Gamma value and choice of Kernel pushes for the best model outcome.

4.1.3.1 Details

	precision	recall	f1-score	support
NEGATIVE	0.70	0.80	0.75	508
NEUTRAL	0.56	0.17	0.27	160
POSITIVE	0.64	0.72	0.68	332
accuracy			0.67	1000
macro avg	0.63	0.56	0.56	1000
weighted avg	0.66	0.67	0.65	1000



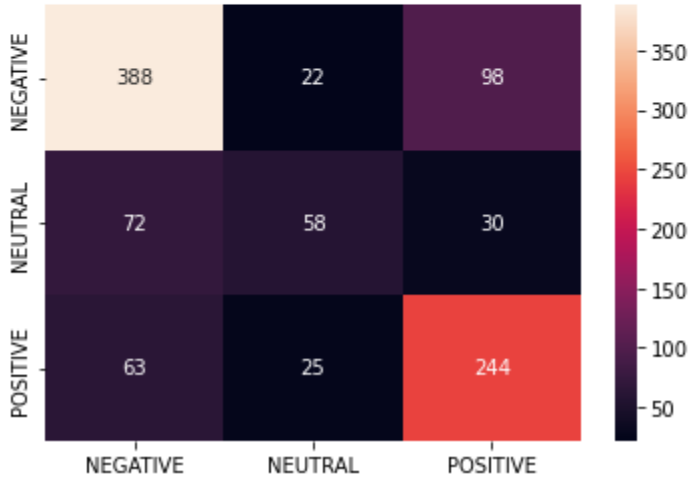
4.1.4 XGBoost

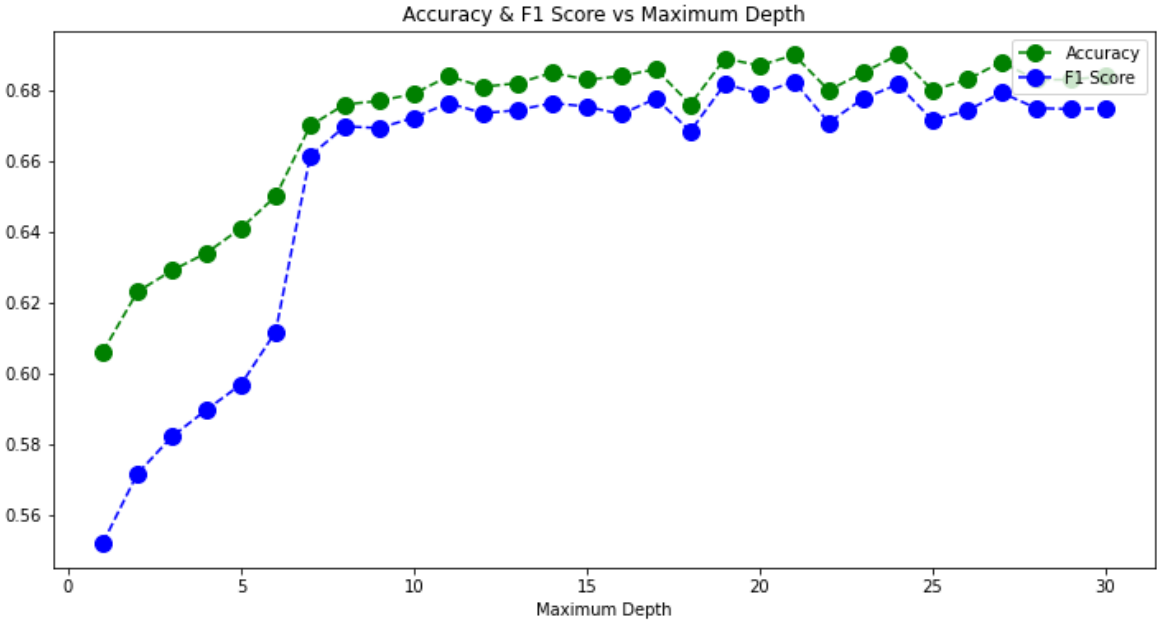
XGBoost is a library for distributed gradient-boosted decision trees. A Gradient Boosting Decision Tree (GBDT) is a decision tree with ensemble learning for classification and regression. XGBoost builds an ensemble of decision trees sequentially to correct the mistakes of the previous ones and learn from complex non-linear relationships between input features and output labels [17]. Gradient boosting is an approach to then create subsequent models to predict the prior models. Altogether, XGBoost produces a final decision with minimized loss and maximized accuracy.

For our XGBoost classifier, we tune the max_depth parameter that is used in the ensemble of decision trees [18]. The maximum depth of the decision tree is the maximum distance between the root and any leaf. If the tree is too deep, it might be prone to overfitting the training data and will not generalize well on the test set. On the other hand, if the tree is too shallow, it will underfit the data.

4.1.4.1 Details

	precision	recall	f1-score	support
NEGATIVE	0.74	0.76	0.75	508
NEUTRAL	0.55	0.36	0.44	160
POSITIVE	0.66	0.73	0.69	332
accuracy			0.69	1000
macro avg	0.65	0.62	0.63	1000
weighted avg	0.68	0.69	0.68	1000





4.2 Classification Task

4.2.1 Preprocessing Text

Preprocessing of data is necessary to filter out any “noisy” data that would affect model performance.

Our data source Reddit is a social media platform where the users can express themselves freely in their own languages and expressions. As such, there might be micro texts that are unintelligible to models. These include emojis (:), :( ) , hyperlinks (www, https), hashtags (#), shortened word forms (@ = at). Hence, we have come up with several preprocessing functions to take into account these variables shown in table below.

Preprocessing Function	Use
re.sub(r'@', 'at ', text)	Replace '@' with 'at'
re.sub(r'[&+]', 'and ', text)	Replace '&' with 'and'
re.sub(r'https?:\/\/.*[\r\n]*', '', text)	Remove hyperlinks
re.sub(r'[^\w\s]', '', text)	Remove punctuations
re.sub(r'[0-9]', '', text)	Remove numbers
text.replace('\n', ' ').encode("utf-8").decode("utf-8")	Replace new line
re.sub(r'\s+', ' ', text)	Remove additional space
text.lower().strip()	Lowercase all words

The aforementioned preprocessing seeks to standardize and reduce ambiguity of words in our dataset.

4.2.2 Building Datasets

4.2.2.1 Training Dataset

Excluding the data that we are labeling for our evaluation dataset, we used 3 different models to label our huge corpus in unsupervised learning. The models are VADER, TextBlob and TweetNLP HuggingFace transformer.

VADER and TextBlob are Python libraries widely used for sentiment analysis. Both use a lexicon-based method which maps words and sentiment, to derive the sentiment of a sentence using the aggregate of the sentiment of each term. VADER is more focused on identifying social media content sentiments [19]. TweetNLP is a Twitter roBERTa-base model trained on approximately 58 million tweets and fine tuned for analysis with TweetEval benchmark [20]. We justify the usage of TweetNLP as both Twitter and Reddit are social media platforms.

Final predicted label for our training data is based on a scoring system: Positive = 1, Neutral = 0, Negative = -1. The addition of scores in all outputs by the 3 models returns the score of our final label. This method ensures a holistic evaluation of our training dataset.

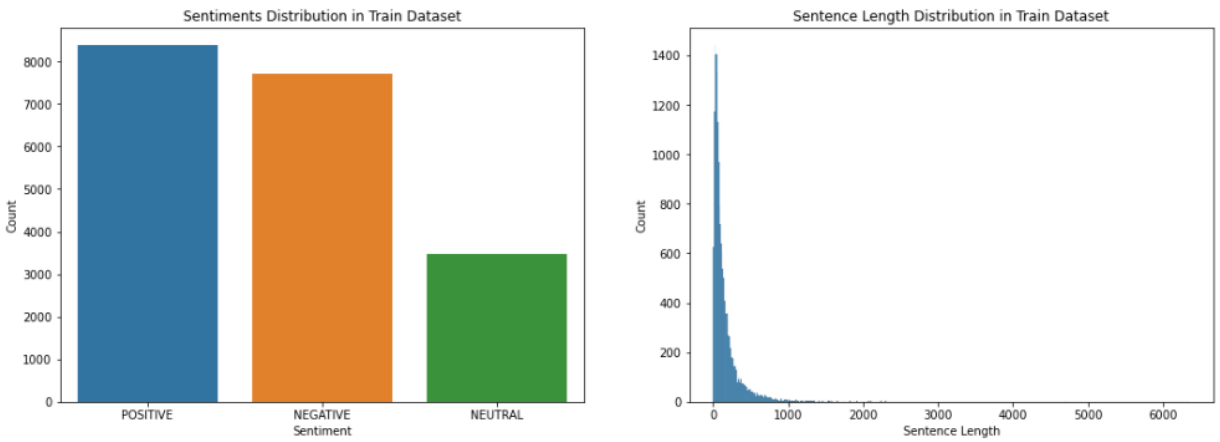


Figure 4.4.2.1: Distributions of train dataset

4.2.2.2 Test Dataset

We manually labelled 1000 text data for our test dataset. In particular, the first 600 for comment and the first 400 records for post as there are more records of comment than post.

We first label the polarity of the selected reddit comments/posts. From the labelled polarity, we then determine the subjectivity of the comments/posts. Neutral records are labelled to be objective while non-neutral records are labelled to be subjective. Non-neutral records are thereafter classified according to their sentiments, Positive or Negative.

For labelling, our team had 3 members to give their reviews on a single reddit comment/post. This is to ensure that the manually labelled dataset is not biased towards a single annotator's point of view and that the test dataset is labelled as accurately as possible. We gathered together for the task and came to an inter-annotator agreement of 90.5%. For the records that were labelled differently, we

took the majority vote for the final value of subjectivity and polarity with other members of the group.

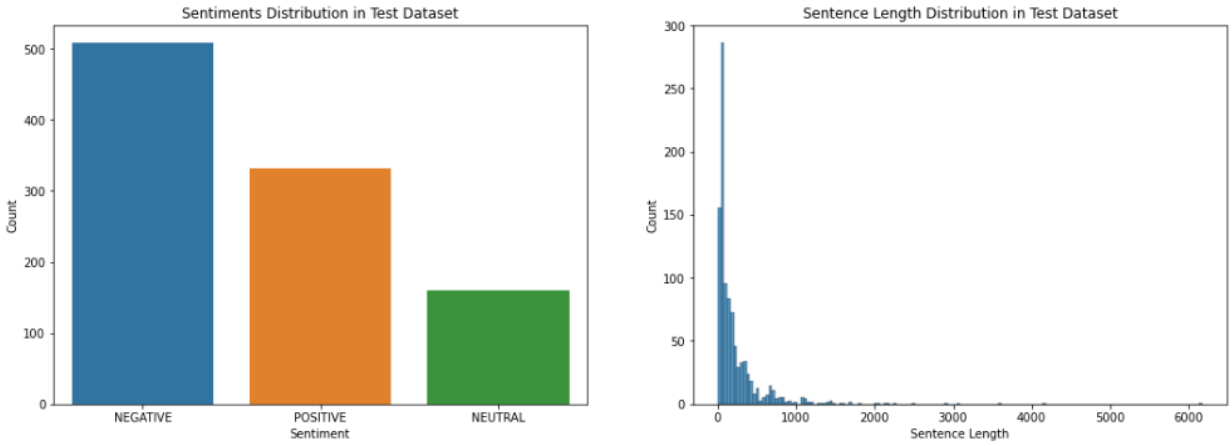


Figure 4.4.2.2: Distributions of test dataset

4.2.3 Text Normalization

We want to reduce noises generated by a single word in multiple forms and increase efficiency in model training.

We execute in the following sequence: Tokenization → Removal of Stopwords → Stemming → Lemmatization. After the process is complete, the dataset vocabulary is downsized to only contain relevant words in its simplest forms.

4.2.3.1 Stemming

Process that stems or removes the last few characters from a word without considering its context [21]. For example, for the word "wanted", ‘-ed’ is removed to form the stem "want". This method might form words that are not necessarily found in the dictionary.

4.2.3.2 Lemmatization

Process that considers context and converts word to its meaningful base form [21]. Unlike Stemming, Lemmatization requires a corpus, and the base form will be in proper English words.

4.2.4 Feature Scaling

For common machine learning approaches, models are only able to interpret numerical data. Therefore, we vectorize the normalized data before fitting into the model for training. Term Frequency-Inverse Document Frequency (TF-IDF) is one of the many vectorizers that helps to convert input data of words and sentences into vectors of real numbers.

4.2.4.1 TF-IDF Vectorizer

The choice for TF-IDF vectorizer is its ability to convert text into numerical features based on the frequency of occurrence of each word in a document, and how often it appears in the entire corpus [22]. The resultant matrix has its row representing a document and column representing a word, with a value indicating its importance in the document. The advantage of using TF-IDF vectorizer is that it gives more weight to rare words that could be more informative than common words.

4.2.5 Model Training

After finalizing our dataset, we will attempt to perform training with various classification

models mentioned in Chapter 4.1.

4.2.6 Evaluation Metrics

The evaluation metrics used are time, precision, recall, F1 Score and accuracy of each model.

4.2.6.1 Time

Time is the time taken for a model to generate a prediction.

4.2.6.2 Precision

Precision is the number of True Positives (TP) divided by the total number of TP and False Positives (TP/TP+FP) [23]. It measures the proportion of TP out of all Positives and is useful when we want to minimise FP.

4.2.6.3 Recall

Recall is the number of TP divided by the total number of TP and False Negatives (FN) [23]. It measures the proportion of TP out of all actual Positive instances and is important when we want to minimise the chance of missing Positive cases.

4.2.6.4 F1 Score

F1 Score is a harmonic mean of both precision and recall. The formula used is (2 * Precision * Recall) / (Precision + Recall) [23]. It is useful when we want to balance the tradeoff between precision and recall.

4.2.6.5 Accuracy

Accuracy is calculated by dividing the number of correct predictions by the total number

of predictions [23]. It is a good metric to judge model performance but does not provide information on FP and FN.

4.2.7 Results

Subjectivity & Polarity Classification				
Model	Best hyperparameter	Wall Time	F1 Score (%)	Accuracy (%)
BERT	Optimizer = Adam Learning Rate = 0.00005 Epsilon = 0.00000001 Clipnorm = 1 EarlyStopping = 3 epoch	2hr 41min 10s	73.6	73.8
Naives Bayes	Alpha = 1	11.3ms	58.2	63.0
SVM	C = 1 Gamma = 1 Kernel = RBF	2.37s	64.6	67.1
XGBoost	Max_depth = 21	2min 57s	68.2	69.0

(Detailed results can be found under individual models in Chapter 4.1)

As shown, the BERT model outperforms other traditional ML models in terms of F1

score and accuracy. For BERT, we have fine tuned the Adam optimizer, implemented EarlyStopping callbacks, prioritised val-accuracy in the classification. Although the other traditional ML models' accuracies are all lower, DNN like BERT has its limitations in terms of time taken and model complexity.

We believe that with a bigger dataset, these differences in performance will be further amplified, especially when DNN like BERT is made to perform well with more data available.

4.2.8 Classification UI

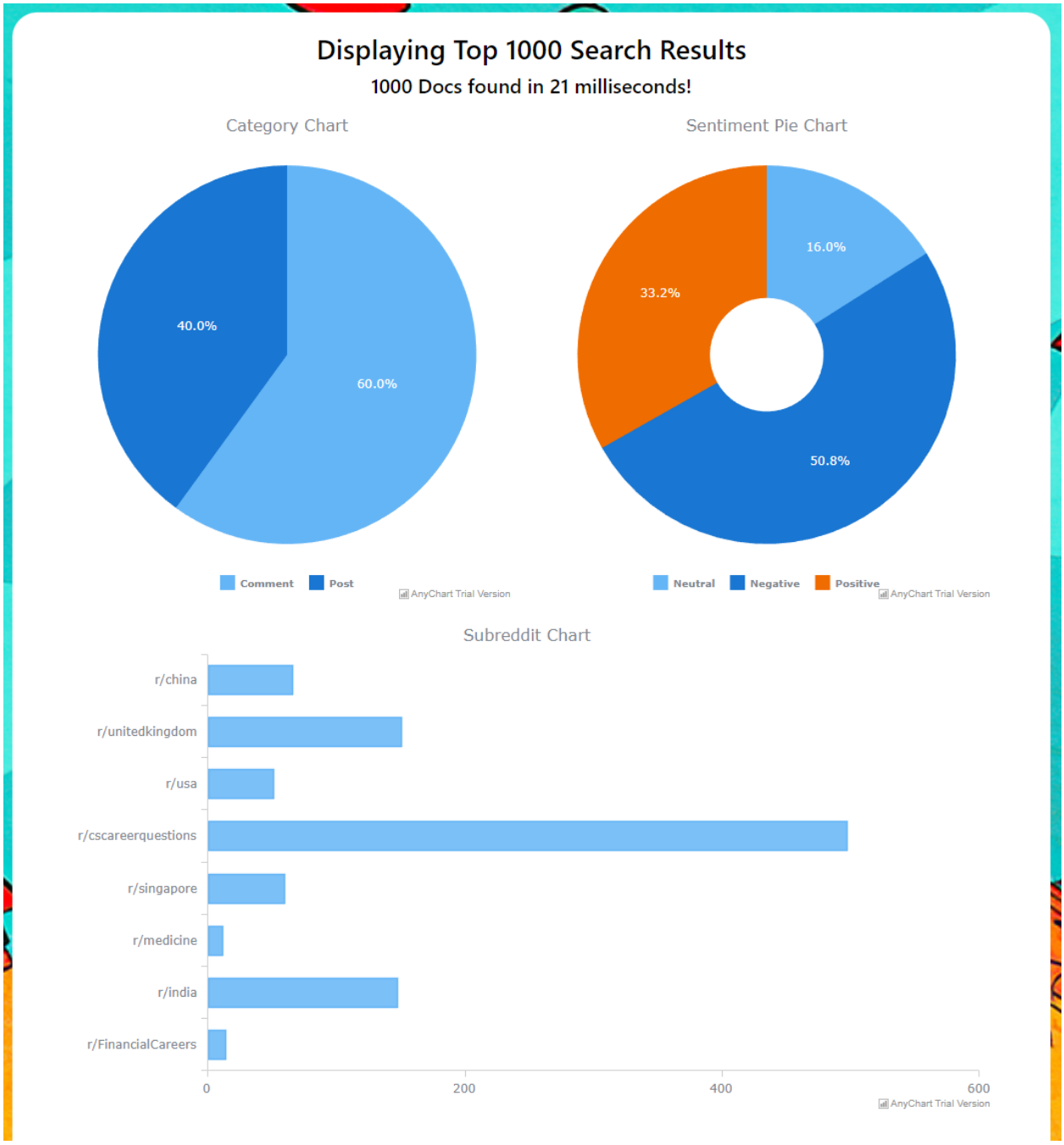


Figure 4.2.8a: Charts displaying the proportion of classification results

As seen in Figure 4.2.8a, we have the category, sentiment and subreddit distribution of the test dataset in the form of pie chart, doughnut chart and bar chart respectively. After each user's search, he/she is able to get a summary of the Reddit data retrieved in relation to his search query, whether most of them are opinionated, or which subreddit majority of the data come from.

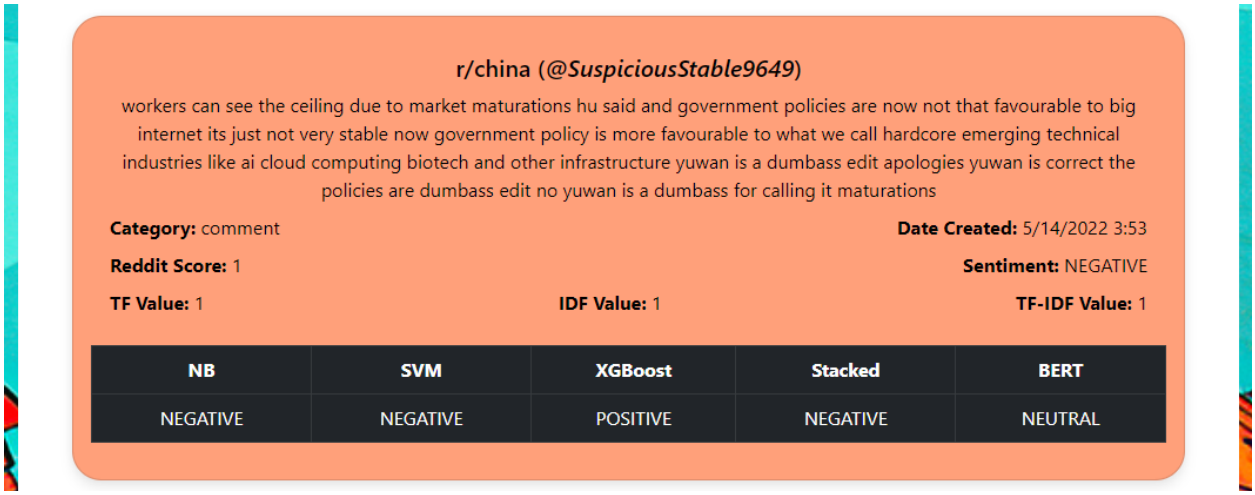


Figure 4.2.8b: Result card showing classification results by each model

We display the outputs of our different model results in the Result Card as shown in Figure 4.2.8b. This is to give users a more informed interpretation of the text found in the card. Furthermore, the Result Card is colour-coded based on actual sentiment for better categorization and user experience.

4.3 Innovations

4.3.1 Ensemble Classifier

Ensemble learning is an approach to using multiple models, which are called base estimators, in the prediction process [24]. A stacked classifier combines the outputs of the baseline models and lets a final meta classifier learn and make a concluding decision [25]. The ensemble utilises the strengths of each individual model and seeks to improve model performance.

For our stacking classifier, we used Naive Bayes, SVM and XGBoost as the baseline classifiers and Logistic Regression for our final output model. We chose this configuration to aggregate top traditional ML models to compare against state-of-the-art DNN models in terms of model results.

Subjectivity & Polarity Classification			
Model	Wall Time	F1 Score (%)	Accuracy (%)
StackedClassifier	21min 21s	66.7	68.2

In this case, we observed a slight decrease in accuracy and f1-score comparing the results of the XGBoost model. We can understand that stacking may or may not improve classification accuracy as classifiers are imperfect and might not be able to

exploit the information new features prove. One potential consideration could be that new features share information with existing ones. Future works could include more classifiers from traditional machine learning techniques as well as deep learning ones.

4.3.2 Named Entity Recognition (NER)

We decided to use NER as one of our innovations to identify named entities such as country names and organisation names [26]. We used the Library SPACY, which allows us to train our own NER model as well as have a pre-trained model. After analysing the results of the pre-trained model, *en_core_web_sm*, we noticed that the model was not able to recognise many of the named entities that are present in the corpus. Therefore, we decided to modify the pre-trained model by training our own model, adding in some common country names and organisation names such as United States, Russia, China, United Nations, Europe Union, etc.

We then trained the named entities with each of our classification models, and realised that after using named entities, the training results did not improve. Figure 4.3.2 shows the comparison of the weighted average accuracy of the data before and after using named recognition for all three models.

	Precision		Recall		F1 Score		Support	
	Before	After	Before	After	Before	After	Before	After
Naive Bayes	0.63	0.48	0.63	0.54	0.58	0.50	1000	625
SVM	0.66	0.48	0.67	0.53	0.65	0.50	1000	625
XGBoost	0.68	0.49	0.69	0.59	0.68	0.52	1000	625

Figure 4.3.2: Comparison of Model Performance With and Without NER

As such, we choose to not incorporate NER into our classification process.

4.3.3 GoEmotion

One of the innovations that we tried was to use a HuggingFace transformer to enhance sentiment analysis using GoEmotions. We first used the bert-base-uncased model to tokenize the text data before using arpanghoshal's EmoRoBERTa model [27] to classify the tokenized text into a total of 28 emotions: *admiration, amusement, anger, annoyance, approval, caring, confusion, curiosity, desire, disappointment, disapproval, disgust, embarrassment, excitement, fear, gratitude, grief, joy, love, nervousness, optimism, pride, realisation, relief, remorse, sadness, surprise* and *neutral.*

These emotions were then classified as positive, neutral and negative according to the mapping shown in Figure 4.3.3a.

```
emotion_labels = {
    'admiration': 'POSITIVE',
    'amusement': 'POSITIVE',
    'approval': 'POSITIVE',
    'caring': 'POSITIVE',
    'desire': 'POSITIVE',
    'excitement': 'POSITIVE',
    'gratitude': 'POSITIVE',
    'joy': 'POSITIVE',
    'love': 'POSITIVE',
    'optimism': 'POSITIVE',
    'pride': 'POSITIVE',
    'relief': 'POSITIVE',
    'anger': 'NEGATIVE',
    'annoyance': 'NEGATIVE',
    'disappointment': 'NEGATIVE',
    'disapproval': 'NEGATIVE',
    'disgust': 'NEGATIVE',
    'embarrassment': 'NEGATIVE',
    'fear': 'NEGATIVE',
    'grief': 'NEGATIVE',
    'nervousness': 'NEGATIVE',
    'remorse': 'NEGATIVE',
    'sadness': 'NEGATIVE',
    'neutral': 'NEUTRAL',
}
```

Figure 4.3.3a: Labelling each emotion into POSITIVE/NEUTRAL/NEGATIVE

One limitation we faced was that the bert-base-uncased tokenization model could only accommodate 512 characters, which some of our post/comment data exceeded. As a result, we had to truncate the data, which would lead to a loss in the amount of data sampled and may also result in misinterpretations of emotion.

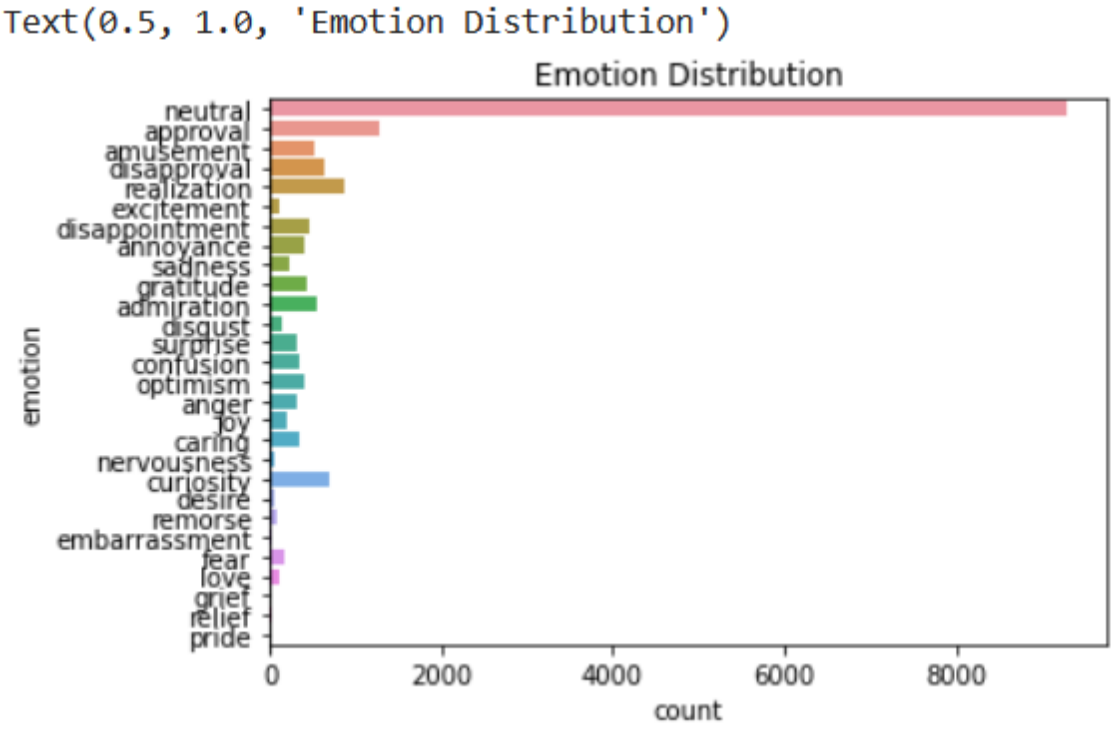


Figure 4.3.3b: Emotion Distribution

We visualised the emotion classification into a bar chart, as seen in Figure 4.3.3b. The majority of the texts have been classified as “neutral”, perhaps due to the prevalence of un-opinionated articles (such as news articles, journal excerpts, etc.) in the corpus.

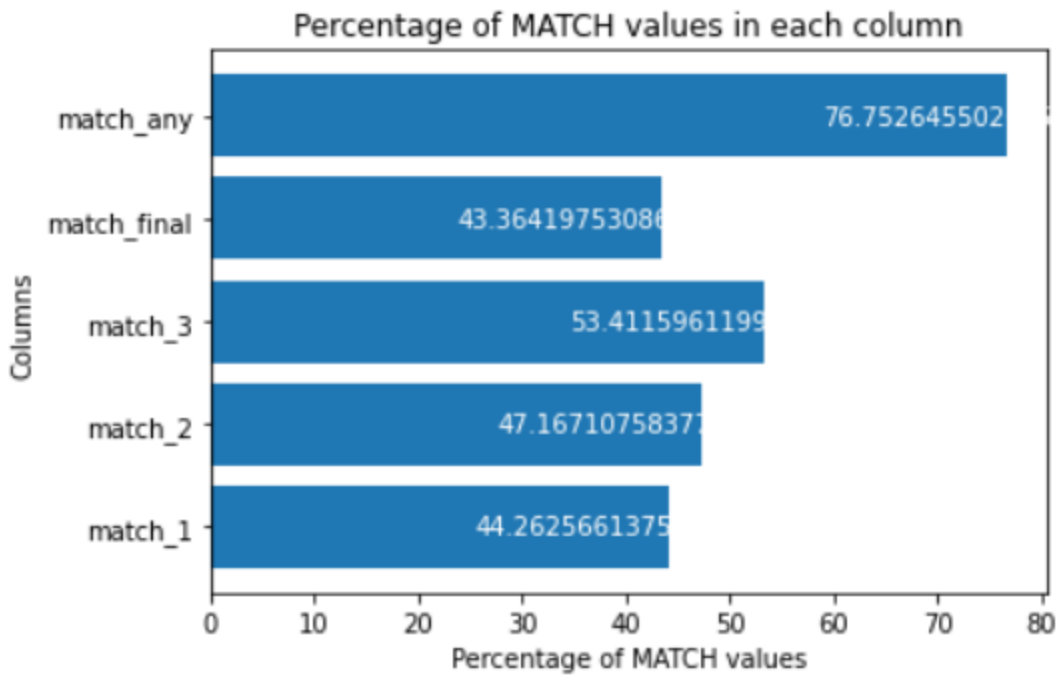


Figure 4.3.3c: Percentage of Match values

Figure 4.3.3c shows the percentage of “matches" between the emotion detected by the EmoRoBERTa model and other labelled data. The results for match_1, match_2, and match_3 have been obtained by comparing EmoRoBERTa’s emotion label with the sentiment analysis label from VADER, TextBlob, and TweetNLP respectively. We can see that the EmoRoBERTa’s emotion label corresponds most closely to TweetNLP’s sentiment analysis label.

5. Conclusion

In this project, we crawled and analysed various subreddits including subreddits from different countries, as well as subreddits containing posts with certain keywords relating to the recession, to create a text corpus. We manually labelled 1000 text data for our test dataset and used Naive Bayes, SVM and XGBoost to classify them.

From the results obtained, the BERT model outperforms other traditional ML models in terms of F1 score and accuracy. From the Ensemble Classifier, we notice a slight decrease in accuracy and F1 score compared to the results of the XGBoost model. Using NER, we realised the training results did not improve, and hence we did not incorporate it. Finally, we use the HuggingFace transformer to enhance sentiment analysis using GoEmotions, and found that the majority of the texts have been classified as “neutral”, perhaps due to the prevalence of un-opinionated articles in the corpus.

6. Submission Links

YouTube Link:

https://youtu.be/BgXiV5fzDKQ

Google Drive Link: (Dataset + Source Code for Crawling + Indexing + Classification + Innovations)

https://drive.google.com/drive/folders/1Oc7XEALFd8YRg7s5OXPczZBVNmB59_nF?usp=sharing

7. References

1. “Global inflation will fall in 2023 and 2024 amid subpar economic growth,” IMF, 30-Jan-2023. [Online]. Available: https://www.imf.org/en/Publications/WEO/Issues/2023/01/31/world-economic-outlook-update-january-2023. [Accessed: 01-Apr-2023].
2. “reddit.com: api documentation”, Reddit. [Online]. Available: https://www.reddit.com/dev/api/. [Accessed: 06-Apr-2023].
3. “PRAW: The Python Reddit API Wrapper”, PRAW. [Online]. Available: https://praw.readthedocs.io/en/stable/. [Accessed: 06-Apr-2023].
4. “Apache Solr”, Apache. [Online]. Available: https://solr.apache.org/. [Accessed: 06-Apr-2023].
5. “Config API - Apache Solr Reference Guide”, Apache. [Online]. Available: https://solr.apache.org/guide/solr/latest/configuration-guide/config-api.html. [Accessed 06-Apr-2023].
6. “Spell Checking - Apache Solr Reference Guide”, Apache. [Online]. Available: https://solr.apache.org/guide/solr/latest/query-guide/spell-checking.html. [Accessed 06-Apr-2023].
7. “JSON Request API - Apache Solr Reference Guide”, Apache. [Online]. Available: https://solr.apache.org/guide/solr/latest/query-guide/json-request-api.html. [Accessed 06-Apr-2023].

8. “Bootstrap v5.0: Getting Started”, Bootstrap. [Online]. Available: https://getbootstrap.com/docs/5.0/getting-started/introduction/. [Accessed: 06-Apr-2023].
9. “amCharts: Changing Data of Word Cloud”, amCharts. [Online]. Available: https://www.amcharts.com/demos/changing-data-word-cloud/. [Accessed: 06-Apr-2023].
10. “AnyChart: Quick Start”, AnyChart. [Online]. Available: https://docs.anychart.com/Quick_Start/Quick_Start. [Accessed: 06-Apr-2023].
11. “Learning to Rank - Apache Solr Reference Guide”, Apache. [Online]. Available: https://solr.apache.org/guide/solr/latest/query-guide/learning-to-rank.html. [Accessed 06-Apr-2023].
12. J Devlin, M.W. Chang, K. Lee and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. [Online]. Available: https://arxiv.org/pdf/1810.04805.pdf. [Accessed 06-Apr-2023].
13. B Vijaykumar, Vikramkumar and Trilochan, “Bayes and Naive-Bayes Classifier”. [Online]. Available: https://arxiv.org/ftp/arxiv/papers/1404/1404.0933.pdf. [Accessed 06-Apr-2023].
14. V. Jayaswal, “Laplace smoothing in Naïve Bayes algorithm,” *Medium*, Nov. 22, 2020. [Online]. Available: https://towardsdatascience.com/laplace-smoothing-in-na%C3%AFve-bayes-algorithm-9c237a8bdece. [Accessed: Apr. 06, 2023].
15. T. Evgeniou and M. Pontil, “Workshop on Support Vector Machines: Theory and

Applications," *Conference Paper*, Sept., 2001. [Accessed: Apr. 06, 2023].

16. A. GrabNGoInfo, "Support Vector Machine (SVM) Hyperparameter Tuning In Python," *Medium*, May. 7, 2022. [Online]. Available: https://medium.com/grabngoinfo/support-vector-machine-svm-hyperparameter-tuning-in-python-a65586289bcb. [Accessed: Apr. 06, 2023].
17. T Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System". [Online]. Available: https://arxiv.org/pdf/1603.02754.pdf. [Accessed 06-Apr-2023].
18. F. Revert, "Fine-tuning XGBoost in Python like a boss," *Medium*, Aug. 10, 2018. [Online]. Available: https://towardsdatascience.com/fine-tuning-xgboost-in-python-like-a-boss-b4543ed8b1e. [Accessed: Apr. 06, 2023].
19. A. GrabNGoInfo, "TextBlob vs. VADER for Sentiment Analysis Using Python," *Medium*, Feb. 20, 2022. [Online]. Available: https://pub.towardsai.net/textblob-vs-vader-for-sentiment-analysis-using-python-76883d40f9ae. [Accessed: Apr. 06, 2023].
20. "cardiffnlp/twitter-roberta-base-sentiment · Hugging Face," Huggingface.co, Jan. 25, 2023. [Online]. Available: https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment. [Accessed: Apr. 06, 2023].
21. "Stemming and lemmatization," *IR book*. [Online]. Available: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html. [Accessed: Apr. 06, 2023].

22. M. Chaudhary, “TF-IDF Vectorizer scikit-learn,” *Medium*, Apr. 24, 2020. [Online]. Available: https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a. [Accessed: Apr. 06, 2023].
23. H. N B, “Confusion Matrix, Accuracy, Precision, Recall, F1 Score,” *Medium*, Dec. 11, 2019. [Online]. Available: https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd. [Accessed: Apr. 06, 2023].
24. A. Mohammed and R. Kora, “A comprehensive review on ensemble deep learning: Opportunities and challenges,” *Journal of King Saud University - Computer and Information Sciences*, Jan. 19, 2023. [Accessed: Apr. 06, 2023].
25. S. N. Alexandropoulos, C. K. Aridas, S. B. Kotsiantis and M. N. Vrahatis, “Stacking Strong Ensembles of Classifiers,” *Publication*, May., 2019. [Accessed: Apr. 06, 2023].
26. C Marshall, “What is named entity recognition (NER) and how can I use it?,” *Medium*, Dec. 18, 2019. [Online]. Available: https://medium.com/mysuperai/what-is-named-entity-recognition-ner-and-how-can-i-use-it-2b68cf6f545d. [Accessed: Apr. 06, 2023].
27. T Kim and P Vossen, “EmoBERTa: Speaker-Aware Emotion Recognition in Conversation with RoBERTa”. [Online]. Available: https://arxiv.org/pdf/2108.12009.pdf. [Accessed 06-Apr-2023].