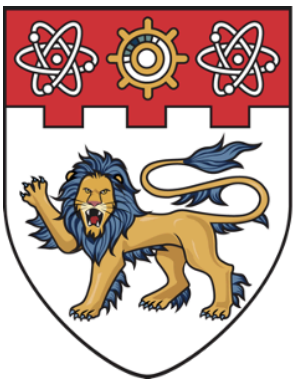


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

School of Computer Science & Engineering

SC4021 - Information Retrieval

Group 24 Project Assignment

Name	Matriculation Number
Ong Zhi Ying, Adrian	U2121883A
Takesawa Saori	U2023120E
Cheong Yong Wen	U2021159L
Kwok Zong Heng	U2021027E
Mandfred Leow Hong Jie	U2122023G
Mao Yiyun	U2022609J

Member Roles & Contribution

Sub-group	Name	Matriculation Number	Contribution
Crawling & Indexing	Ong Zhi Ying, Adrian	U2121883A	Data Crawling Solr Indexing Section 1-3.3 in the report
Indexing	Takesawa Saori	U2023120E	UI design UI implementation
Classification	Cheong Yong Wen	U2021159L	Annotation Vader Classification Textblob Classification
	Kwok Zong Heng	U2021027E	Annotation BERT Classification Innovation
	Mandfred Leow Hong Jie	U2122023G	Innovation
	Mao Yiyun	U2022609J	Classification Test set selection Roberta classification Innovation-majority voting

Table of Contents

1. Introduction

1.1. Background

As part of the commitment to combat climate change, Singapore aims to reduce land transport emissions by transitioning towards electric vehicles (EVs). According to the Land Transport Authority (LTA) vision, the government has set targets to accelerate the nationwide adoption of EVs, aiming for a significant portion, 100% of the vehicle population to be electric-powered by 2040. Given the significant incentives to switch to EVs, members of the public will soon need to decide on the brand of EV to purchase. Individuals may wish to research reviews and public sentiment surrounding popular EV brands and EV Models to make an informed choice that aligns with their preferences. However, sifting through countless reviews across various social platforms can be impractical and time-consuming. Therefore, this highlights a need for an informational retrieval system capable of curating reviews and sentiment text data from different social platforms, deriving sentimental insights efficiently, and allowing consumers to make an informed choice swiftly.

1.2. Objective

This project aims to design and develop an information retrieval system that can not only search and display public users' related comments but also derive deeper insights using Natural Language Processing techniques such as sentiment analysis, subjectivity and sarcasm classifications. The system is aimed to assist Singapore's initiative to transition to EVs by enabling consumers to make well-informed decisions when selecting an EV that aligns with their preferences. Our system will not just focus on the major brands of EVs such as Tesla, but also gather data from general Reddit EV subreddit communities.

2. Web Crawling

2.1. Scope of data

Given the scope of our chosen topic, Reddit will be the best social platform to scrape for data. Reddit is a massive online community and social platform where users can share and discuss almost any topic. It is structured around community groups called subreddits, where each is focused on discussing a specific topic. Therefore, all of our data shall be crawled from different EV-related subreddits. Through our extensive research, we have identified a list of subreddits that meet our specific criteria, such as:

- Active user base, preferably above 1K active members
- Respectful environment and content
- Regular content submission
- Focused on discussions about EVs

The table below shows the overview of the subreddit identified:

Subreddit name	Description	No. of active members (As of 1st April 2024)
r/electricvehicles	General subreddit regarding EV ownerships around the globe, regardless of brands or models.	283K
r/electriccars	General subreddit to electric car enthusiasts around the globe.	13K
r/nzev	General subreddit regarding EV ownership in New Zealand, regardless of brands or models.	7.2K
r/ElectricCarUK	General subreddit regarding EV ownerships in the United Kingdom, regardless of brands or models.	458
r/evcharging/	A discussion forum for EV owners: setting up a charge station at home and finding and using charger networks on the road.	11K

r/teslamotors	The largest Tesla subreddit community. An unofficial forum for Tesla owners and enthusiasts.	2.7M
r/RealTesla	A secondary Tesla subreddit, focused on discussion of Tesla-branded vehicles.	88K
r/BMWi3	Discussion regarding BMW brand I3 EV model.	10K
r/leaf	Discussion regarding the Nissan LEAF EV model.	19K
r/BoltEV	Discussion regarding Chevy Bolt EV, EUV, and Opel Ampera-e models.	30K
r/F150Lightning	Discussion regarding the Ford F-150 Lightning EV model.	15K
r/Taycan	Discussion regarding the Porsche Taycan EV model.	7.4K
r/KiaEV6	Discussion regarding the Kia EV6 EV model.	9.3K

Table 1: *Overview of crawled subreddits*

2.2. Methodology

2.2.1. Overview

Question 1:

a. **How you crawled the corpus (e.g., source, keywords, API, library) and stored it**

We leveraged the **Python Reddit API Wrapper (PRAW)** to provide seamless communication with Reddit API, we managed to crawl **1,229 posts** and **48,194 comments** from the identified subreddits listed in **Table 1** above. Our approach involved retrieving only the **top 100** posts of all time (No time restriction) per subreddit and retrieving all associated comments for those posts. Top posts are defined by their net upvotes (the number of upvotes minus the number of downvotes), the higher the net upvotes, the higher the rank of the post. This methodology was employed to ensure the selection of high-quality posts and at the same time, our information retrieval system does not ignore unpopular and controversial comments. Finally, the crawled data was stored in a structured format (CSV) for further processing and analysis.

The crawling process is implemented in Python, as illustrated by code snippets from **Figure 1-3**.

```python
import praw

# Login to reddit API
reddit = praw.Reddit(
    client_id="xn1EzLWHVbp2ZOuTiNYk-w",
    client_secret="rSRccp3N37yjKuQQWcPxW_SKSXe0SA",
    user_agent="<NanyangTechnologicalUniversity>-<information retrieval project>:<v1> (by /u/CitronResponsible811)",
    username="CitronResponsible811",
    password="Password123"
)
```

Executed at 2024.03.22.14:44:41 in 880ms

Figure 1: Log in to Reddit API

```python
# Start extracting
## Specialized subreddits
specialized_subreddits = [
    # General electric car subreddits
    "electricvehicles",
    "electriccars",
    "nzev",
    "ElectricCarUK",
    "evcharging",

    # Electric Car brands subreddits
    "teslamotors", # Tesla 100% electric car brand
    "RealTesla",
    "BMWi3", # BMW I3 car model electric
    "leaf", # Nissan LEAF EV car model
    "BoltEV", # Chevrolet Bolt EV car model
    "F150Lightning", # Ford F150 Lightning EV car model
    "Taycan", # Porsche Taycan EV car model
    "KiaEV6", # Kia EV6 car model
]

for subreddit_name in specialized_subreddits:
    subreddit = reddit.subreddit(subreddit_name)
    posts = subreddit.top(limit=100, time_filter='all')
    extract_all_post_comments_post_ids(get_mod_list(subreddit_name), posts, subreddit_name)
```

Figure 2: Loop the defined crawling function on all subreddits

```
No: 1, Title: F150Lightning:Cybertruck broke at King of Hammers, PostID: 1ahvzms
No: 2, Title: F150Lightning:No more charging while at work, PostID: 18wq76i
No: 3, Title: F150Lightning:Buyback complete, surrendered it back to ford today, goodbye lightning ;(, PostID: 19elr6g
No: 4, Title: F150Lightning:The worst part of owning an EV is… Electrify America, PostID: 16nxgez
No: 5, Title: F150Lightning:Ford F-150 Lightning breaks monthly sales record, doubling in November, PostID: 188rf2m
No: 6, Title: F150Lightning:Cybertruck and Lightning comparison, PostID: 17ow957
No: 7, Title: F150Lightning:My first EV. Picked up a Lariant, PostID: 1b8t9aa
No: 8, Title: F150Lightning:Who got me beat? 2023 Lariat ER for $49,215., PostID: 1b3irew
No: 9, Title: F150Lightning:Any smoke around this being true?, PostID: 1au5ppz
No: 10, Title: F150Lightning:Ford U.S. EV Sales Surged 81% In February 2024, PostID: 1b6q9yy
```

Figure 3: Crawling progress outputs

2.2.2. Data Quality Measures

Certain data quality measures were put in place to ensure that the crawled data was of high quality. Data quality measures include:

- **Selecting the highest quality posts in each subreddit:** By retrieving only the top 100 posts per subreddit based on upvote ranking, we ensured that the crawled posts do not include low-effort posts such as reposts, spam and memes. This implementation is illustrated in **Figure 2**.
- **Omitting posts and comments generated by Reddit Bot:** Reddit moderators usually utilise scripted bots to police and maintain subreddit rules. These bots will automatically delete any posts or comments that violate the rules of the subreddit, in addition, the bots also issue a warning reply emphasising the rules of the subreddit shown in **Figure 4**. However, these warning comments do not contribute meaningful content to the topic. Therefore are excluded from the crawled dataset **(Figure 5)**.

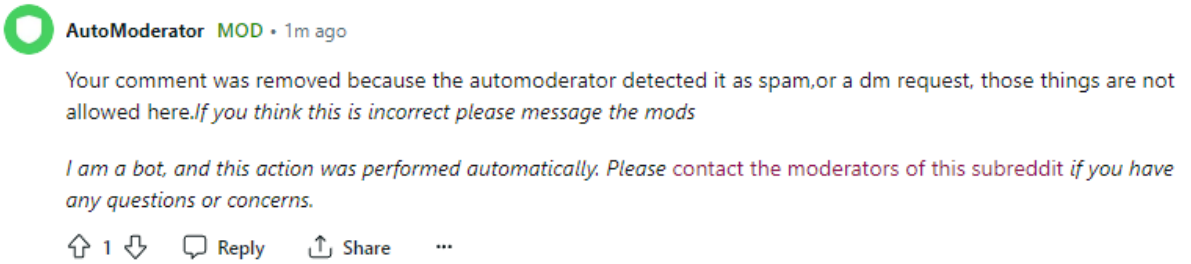


Figure 4: Reddit moderator bot warning

```python
# Get subreddit mod ids
def get_mod_list(subreddit_name):
    mod_names = []
    for moderator in reddit.subreddit(subreddit_name).moderator():
        mod_names.append(moderator.name)

    return mod_names

for comment in post.comments:
    if isinstance(comment, MoreComments):
        # Skip lower level comments
        continue
    # Skip mod comments, usually about rules
    if comment.author and comment.author.name in mod_names:
        continue
    # Skip comments that are deleted
    if comment.body == '[deleted]' or comment.body == '[removed]':
        continue
```

Figure 5: Removal of lower-level, bot, and deleted comments

- **Removal of comments where the content is deleted or removed:** When moderators remove comments and posts violating subreddit rules, an empty placeholder is left where the content is displayed as **'[deleted]'** or **'[removed]'**. This information does not benefit our analysis or contribute to the discussion as the content is lost. Therefore, these are also excluded from the crawled dataset **(Figure 5)**.
- **Exclude lower-level comments (e.g. replies to comments):** Replies to comments typically start to stray away from the topic of the post. As the chain of nested replies continues, the relevance tends to decrease. Therefore, To keep the crawled comments focused on the topic of the main post, replies to any comments are excluded **(Figure 5)**. This decision was made to ensure consistency in the data quality and context preservation of our main topic; EV.
- **Removal of duplicate entries:** Data is kept unique by keeping an array of unique posts and comments identifier (ID). This array of IDs is then used to verify duplicate

posts and comments during crawling **(Figure 6)**. Any duplicated content is then omitted.

```
# Only write unique posts & comments
if post.id not in post_ids:
    post_row = [
        getattr(post, 'author', '[deleted]') or '[deleted]', #
        getattr(post, 'author_flair_text', 'None') or 'None',
        getattr(post, 'clicked', None) or 'None',
```

Figure 6: Exclude duplicated posts and comments

2.2.3. Data Storage

The crawled data is written sequentially and stored in CSV format **(Figure 7)**. From the large number of attributes returned in the API response, we have identified a set of important attributes for both posts and comments as follows:

No	Attribute	Description
1	author	The username of the account that submitted the post/comment
2	text	The textual content of the post/comment
3	created_utc	Timestamp of the post/comment
4	edited	Last edit timestamp
5	id	Content unique identifier
6	permalink	URL to the post/comment
7	upvote	Upvotes minus downvotes, net upvotes
8	subreddit_name	Name of subreddit the content belongs to
9	upvote_ratio	Ratio of upvotes to downvotes
10	num_comments	Number of comments in the post (Left empty if for comments
11	url	URL to the media associated with the post (If applicable)
12	post_id	Foreign key to id of post of the comment

Table 2: *Overview of important attributes for post and comments records*

(Post-specific attributes are coloured in red)

(Comment-specific attributes are coloured in blue)

After the crawling process is finished, each subreddit will generate 2 separate CSV files: one containing the posts data and the other containing the associated comments data **(Figure 7)**.

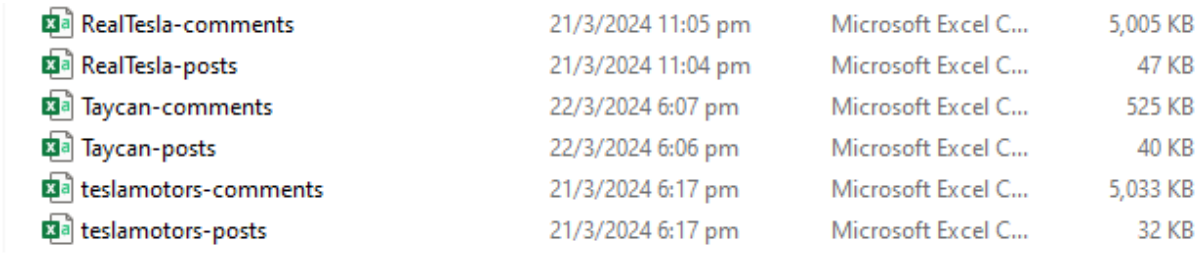


Figure 7: Posts & Comments CSV

To consolidate the data from multiple subreddits, we appended all posts and comments data files together. The resulting dataset contains all information from the crawled posts and comments **(Figure 8)**. To differentiate between a post and a comment, we have added a new column **'type'.** This normalized format facilitates efficient data processing and analysis.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	author	text	created_u	edited	id	num_com	permalink	score	subreddit	upvote_ra	url	type	post_id
2	billbucket	Tesla's Su	2018-01-25T07:13:35		7srdiw	1050	/r/teslam	49182	teslamotc	0.9	https://gf	post	
3	TheMight	Elon Musk	2018-01-30T05:20:51		7tvnx6	394	/r/teslam	45635	teslamotc	0.89	https://i.i	post	
4	BasharK	Tesla vs B	2017-11-20T03:53:56		7e2ze2	2321	/r/teslam	44315	teslamotc	0.8	https://i.r	post	
5	andork28	My grandp	2021-03-22T06:00:09		ma79ze	1002	/r/teslam	40990	teslamotc	0.84	https://i.r	post	

Figure 8: Small snippet of the crawled data

2.2.4. Basic Data Cleaning & Standardization

Before proceeding with any querying or indexing, we applied basic data cleaning and standardization techniques to the consolidated dataset. This step aimed to improve textual data representation and minimise the loss of context. More advanced and complete data cleaning will be performed later in the project as well.

At this stage, the following operations were performed:

- **Removal of non-English records:** Despite Reddit being an English-dominated community, there are still some users who post content in other languages. To reduce the complexity of the system, we have decided to remove any non-English records present in the corpus **(Figure 9)**. We used the *Python langdetect* library to detect non-English comments. Due to URLs, symbols and emojis hindering the prediction accuracy, our team had to double-check manually before removing any records.

```
# Manually check the text and identify the non-english comments
non_english_identified_ids = [241, 11957, 13467, 13490, 13628, 14686, 31678]
# Print the text of these rows
for idx in non_english_identified_ids:
    print(f"Index: {idx}, Text: {merged_df.iloc[idx]['text']}")
# Drop these rows
merged_df = merged_df.drop(non_english_identified_ids)
Executed at 2024.04.04 17:47:14 in 4ms

Index: 241, Text: Louis le campagnard
Index: 11957, Text: 한국차 진짜야
Index: 13467, Text: Visste ikke at de var kommet til norge enda
Index: 13490, Text: Gratulerer! Kult å se dem på veien, rå biler
Index: 13628, Text: Vilken mack är det?
Index: 14686, Text: Tillykke :Denmark:
Index: 31678, Text: Das Qualität gespringen macht frei arbeit
```

Figure 9: Removing non-English comments

- **Emoji Handling:** Emojis are a vital part of determining the sentiment of a piece of text. If emojis are removed, a negative polarity but sarcastic sentence (e.g. With a skull emoji) might be misinterpreted as positive polarity. Therefore, it does not make sense to remove emojis from our corpus. Instead, we decided to convert emojis to their textual representation by using the Python emoji library **(Figure 10)**.

```
comment_row = [
    getattr(comment, 'author', '[deleted]') or '[deleted]',
    emoji.demojize(getattr(comment, 'body', 'None')),
    emoji.demojize(getattr(comment, 'body_html', 'None')),
```

Figure 10: Converting emoji to their textual representation

- **Removal of multiple whitespaces:** Most social media texts are informal, with multiple consecutive whitespaces (e.g. tab, newline). To ensure consistency, any instance of multiple whitespaces will be reduced to a single space character **(Figure 11)**.

Figure 11: Removing multiple whitespaces

- **Domain-specific microtext and abbreviation translation:** In informal social media settings, particularly on platforms like Reddit, users frequently employ domain-specific abbreviations, acronyms, lingo and microtext. E.g. “IIRC” for “If I Recall Correctly”. To improve the readability and context, we decided to translate these domain-specific terms to their complete forms using a curated mapping dictionary of common Reddit abbreviations and lingos **(Figure 12)**. The full mapping dictionary can be found in the **Appendix section**. Our mapping dictionary contains:
    - Domain-specific Reddit lingo (OP -> Original Poster)
    - General Internet Micro-texts (JK -> Just Kidding)

```
        'JK': 'Just kidding',
        'IDC': 'I dont care',
        'FTW': 'For the win',
        'LMAO': 'Laughing my ass off',
        'LMFAO': 'Laughing my fucking ass off',
        'BFF': 'Best friend forever',
        'MFW': 'My face when',
        'TFW': 'That feeling when',
        'G2G': 'Got to go',
        'MSG': 'Message',
    }

    # Standardize all cases in mapper
    abbr_mapper =  {key.lower(): val for key, val in abbr_mapper.items()}

    # Regex to match every key
    # r'\b' -> Matches from start to finish
    # (?:) -> Capturing group
    # '|'.join(mapper) -> Loop and form pattern of all keys
    pattern = r'\b(?:' + r'|'.join(re.escape(abbr) for abbr in [k.lower() for k in abbr_mapper.keys()]) + r')\b'
    # Compile and replace
    pattern = re.compile(pattern, re.IGNORECASE)
    # Count
    num_substitutions = len(re.findall(pattern, text))
    replaced_text = pattern.sub(lambda match: abbr_mapper[match.group(0).lower()], text)

    return replaced_text, num_substitutions


# Apply to text field of df
merged_df['text'], counts = zip(*merged_df['text'].apply(replace_abbr))
print(f"Number of changes: {sum(counts)}")
Executed at 2024.03.23 16:01:59 in 7s 200ms

Number of changes: 2424
```

Figure 12: Snippet of microtext translation implementation

2.3. Possible Use Cases

Question 1:

b. What kind of information users might like to retrieve from your crawled corpus (i.e., applications), with sample queries

With the purpose of our information retrieval system in mind, we assume that users will be using our system to perform extensive research on public sentiment and opinions of EV brands and features (e.g. Fast charging). Ultimately, our system will provide insights for users to make informed decisions regarding their EV purchases.

Some potential queries might include:

- “Best Electric Car for going long distances?”

- “Is Tesla EV worth it?”
- “Pros and cons of driving an EV?”

Obviously, our system will not return a direct answer to these queries. Instead, it will retrieve the most relevant posts and comments from Reddit, performing analysis and generating insights, facilitating efficient exploration of user-generated content.

2.4. Statistics summary of corpus

. **Question 1:**

c. The number of records, words, and types (i.e., unique words) in the corpus

Subreddit	Posts crawled	Comments crawled
r/electricvehicles	100	6,882
r/electriccars	99	2,303
r/nzev	83	1,446
r/ElectricCarUK	55	73
r/evcharging/	96	1,339
r/teslamotors	100	11,788
r/RealTesla	100	10,786
r/BMWi3	100	1,268
r/leaf	99	1,515
r/BoltEV	98	4,439
r/F150Lightning	99	3,419
r/Taycan	100	1,334
r/KiaEV6	100	1,602

Table 3: *Statistics per crawled subreddit*

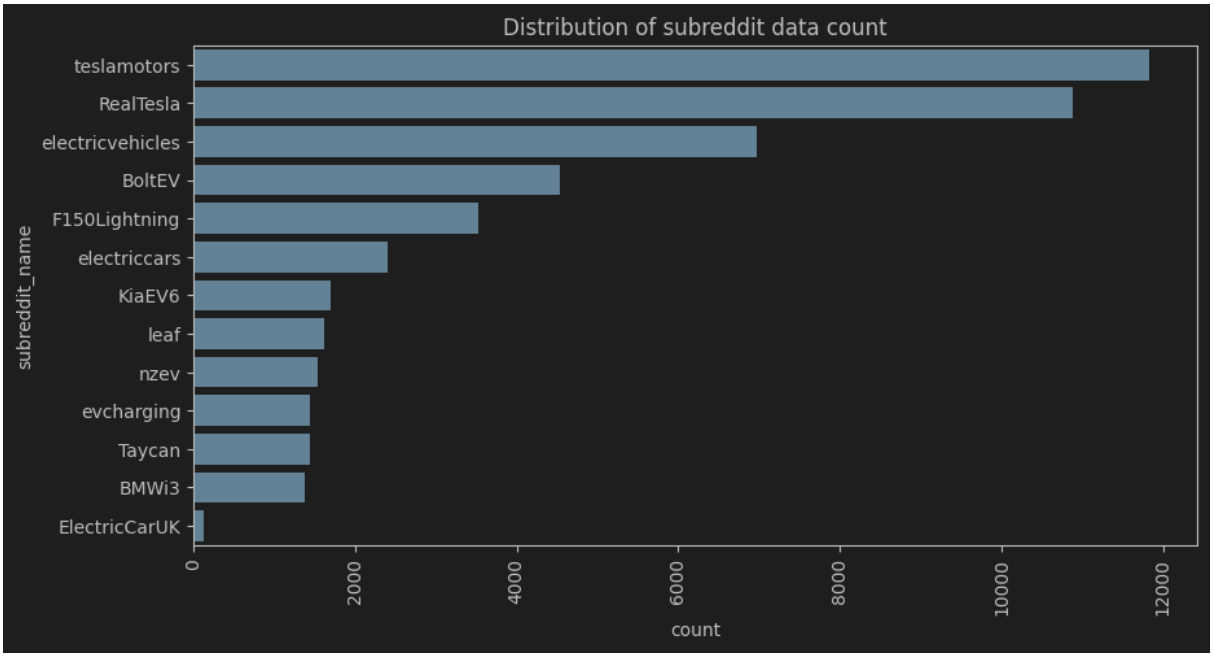


Figure 13: Crawled data distribution

Statistics	No.
The number of subreddits crawled	13
Number of crawled posts	1,229
Number of crawled comments	48,194
Total number of records in the corpus	49,423
Average record text length	23.83
Average upvote of all records	66.49
Median upvotes of all records	2.0
Total number of tokens in the corpus	1,176,272
Total number of unique tokens in the corpus	74,055
Total number of tokens in the corpus (Removed stopwords)	662,004
Total number of unique tokens in the corpus (Removed stopwords)	73,896
Total number of tokens in the corpus (Stemmed)	1,388,027
Total number of unique tokens in the corpus (Stemmed)	30,773

Table 4: *Statistics of combined corpus*

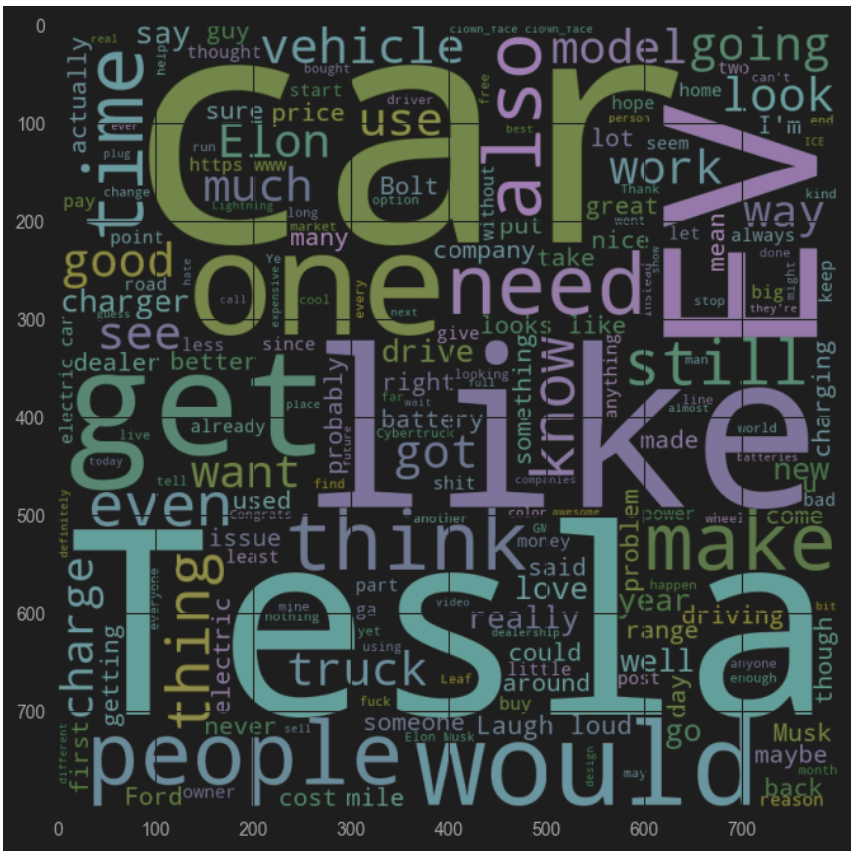


Figure 14: Wordcloud of top words in the corpus

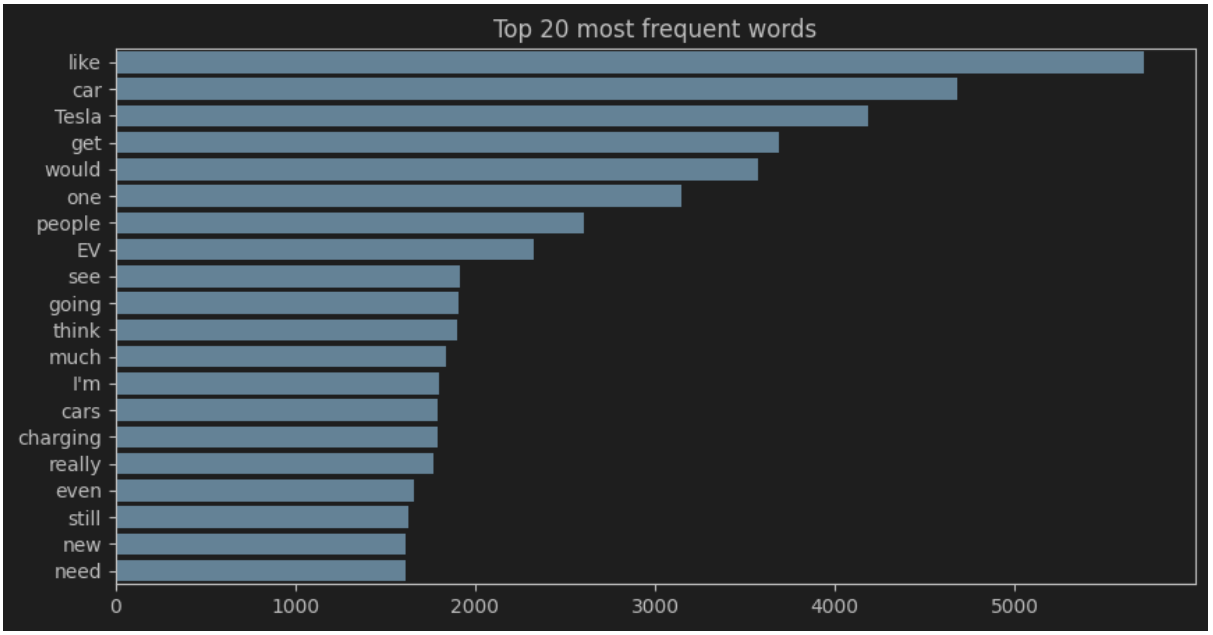


Figure 15: Barchart of top words in the corpus

3. Indexing

3.1. Approach

For our information retrieval system, Apache Solr was used as the backend system for supporting storage, indexing and querying purposes. Apache Solr is an open-sourced, Java-based search platform built on top of Apache Lucene. Apache Solr uses an inverted index data structure to efficiently query and retrieve data. In addition, it also provides a RESTful API interface, allowing for easy integration with any programming language.

3.2. Indexing Process (Indexing Innovations)

Question 3:

a. **Explore some innovations for enhancing the indexing and ranking. Explain why they are important to solve specific problems, illustrated with examples. You can list anything that has helped improving your system from the first version to the last one, plus queries that did not work earlier but now work because of the improvements you made.**

3.2.1. Defining Custom Field Type

After setting up Apache Solr and creating a core, we must first define our field type named **'text_reddit'**. Since we have already done some light data pre-processing to our corpus in the **2.2.4. Basic Data Cleaning & Standardization** stage, it does not make sense to perform the same data pre-processing operations again if we were to use pre-defined filters in field types such as **'text_general'.** Thus, defining our field type allows us to tailor and customize the text processing pipeline by applying appropriate filters and tokenizers, and optimizing the text processing pipeline and indexing process.

```
<fieldType name="text_reddit" class="solr.TextField" positionIncrementGap="100" multiValued="true">
  <analyzer type="index">
    <tokenizer class="solr.WhitespaceTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" words="stopwords.txt" ignoreCase="true"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.SnowballPorterFilterFactory" language="English"/>
    <filter class="solr.NGramFilterFactory" minGramSize="2" maxGramSize="20"/>
  </analyzer>
  <analyzer type="query">
    <tokenizer class="solr.WhitespaceTokenizerFactory"/>
    <filter class="solr.StopFilterFactory" words="stopwords.txt" ignoreCase="true"/>
    <filter class="solr.SynonymGraphFilterFactory" ignoreCase="true" expand="true" synonyms="synonyms.txt"/>
    <filter class="solr.LowerCaseFilterFactory"/>
    <filter class="solr.SnowballPorterFilterFactory" language="English"/>
  </analyzer>
</fieldType>
```

Figure 16: Defined custom field type in managed-schema.xml

Our field type **'text_reddit'** contains the following filters:

- **WhitespaceTokenizerFactory:** Splits sentences into tokens.
- **StopFilterFactory:** Removes common stop words, we have our own custom set of stopwords.
- **LowerCaseFilterFactory:** Normalizes the text to lowercase.
- **SnowballPortalFilterFactory:** Performs stemming, reducing words to their base forms.
- **NGramFilterFactory (For index only):** Generates n-grams during indexing, and improves partial matching capabilities.
- **SynonymGraphFilterFactory (For query only):** Expands queries with synonyms, enhancing the coverage of search queries. We have our own custom set of synonyms.

3.2.2. Defining stopwords & synonyms

The next step is to populate our own set of stopwords and synonyms for our use case. For stopwords, we have used Python NLTK's pre-defined stopwords list combined with Apache Solr's pre-defined stopwords list, forming a more exhaustive stopwords list.

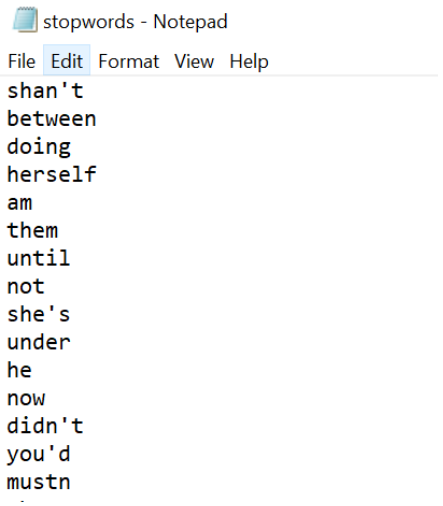


Figure 17: Stopwords.txt

To accommodate the user's lazy habits, we predict that most users will only type model names in the search query (E.g. "I3" -> referring to BMW I3 EV Model). This search query does not match documents related to BMW, because it does not have the word "BMW" in it. Therefore, we decided to perform short-form brand conversion (e.g., "I3" -> "BMW I3 EV Model") by mapping model names to their respective brands using synonyms.

```
BMW => BMW I3
I3 => BMW I3
leaf => Nissan Leaf
Nissan => Nissan Leaf
Bolt => Chevrolet Bolt
Chevy => Chevrolet Bolt
BoltEV => Chevrolet Bolt
F150Lightning => Ford F150Lightning
F150 => Ford F150Lightning
Ford => Ford F150Lightning
Lightning => Ford F150Lightning
Taycan => Porsche Taycan
Porsche => Porsche Taycan
EV6 => Kia EV6
Kia => Kia EV6
```

Figure 18: Synonyms.txt

We can see each filter in effect in the analysis tab in the Solr Graphical User Interface (GUI).

Document: *"The i3 is still the coolest car I've ever owned. LOVE IT SO MUCH."*

Query: *"BMW cars"*

Filter	Changes made
WhitespaceTokenizerFactory	Process both document and query into tokens
StopFilterFactory	Document: Removed words: “the”, “it”, “so”
LowerCaseFilterFactory	Converts all characters to lowercase
SnowballPortalFilterFactory	Query: Cars -> car
NGramFilterFactory	Document: Coolest -> co | coo | cool | coole | cooles | coolest … and so on
SynonymGraphFilterFactory	Query: BMW -> BMW I3

Table 5: Analysis of implemented filters

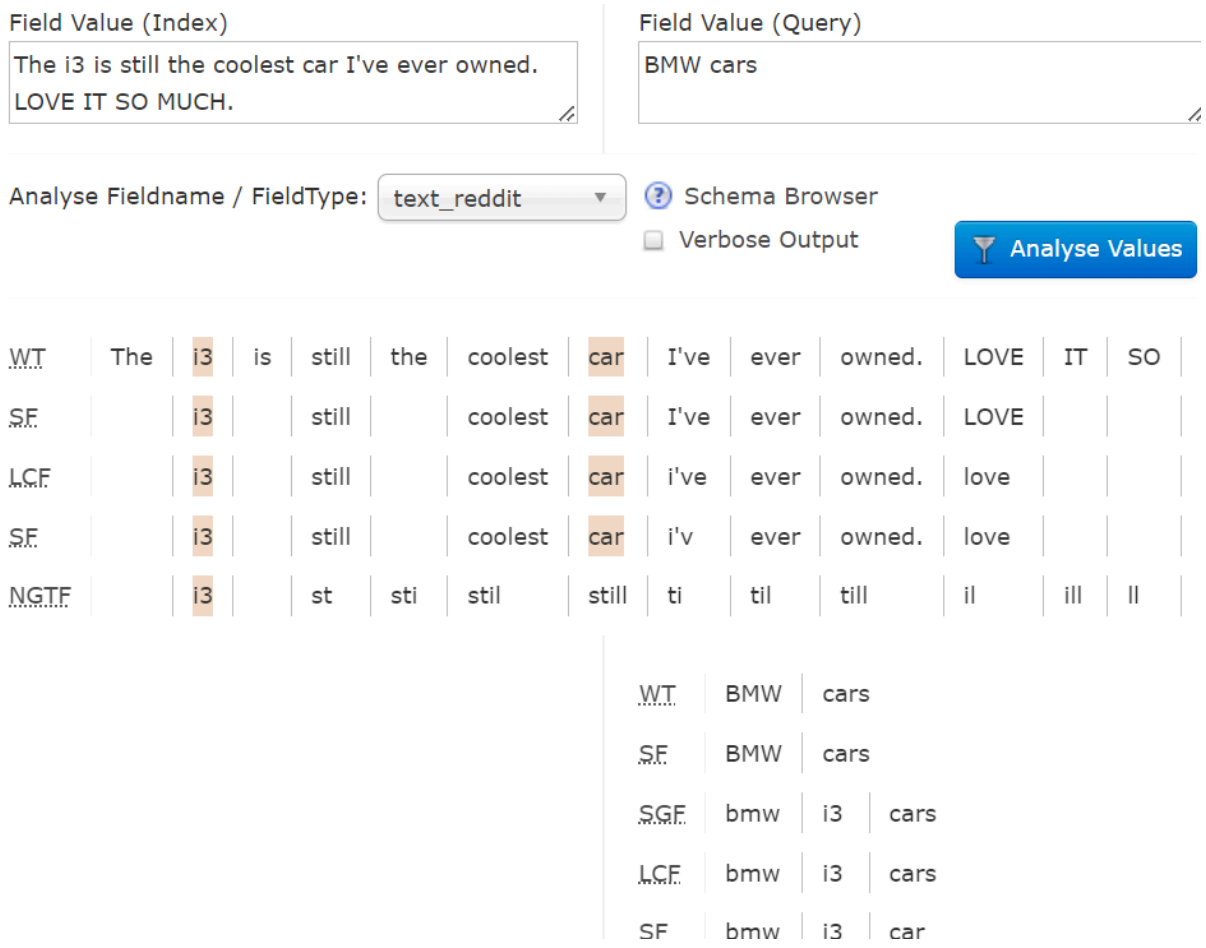


Figure 19: Analysis of implemented filters

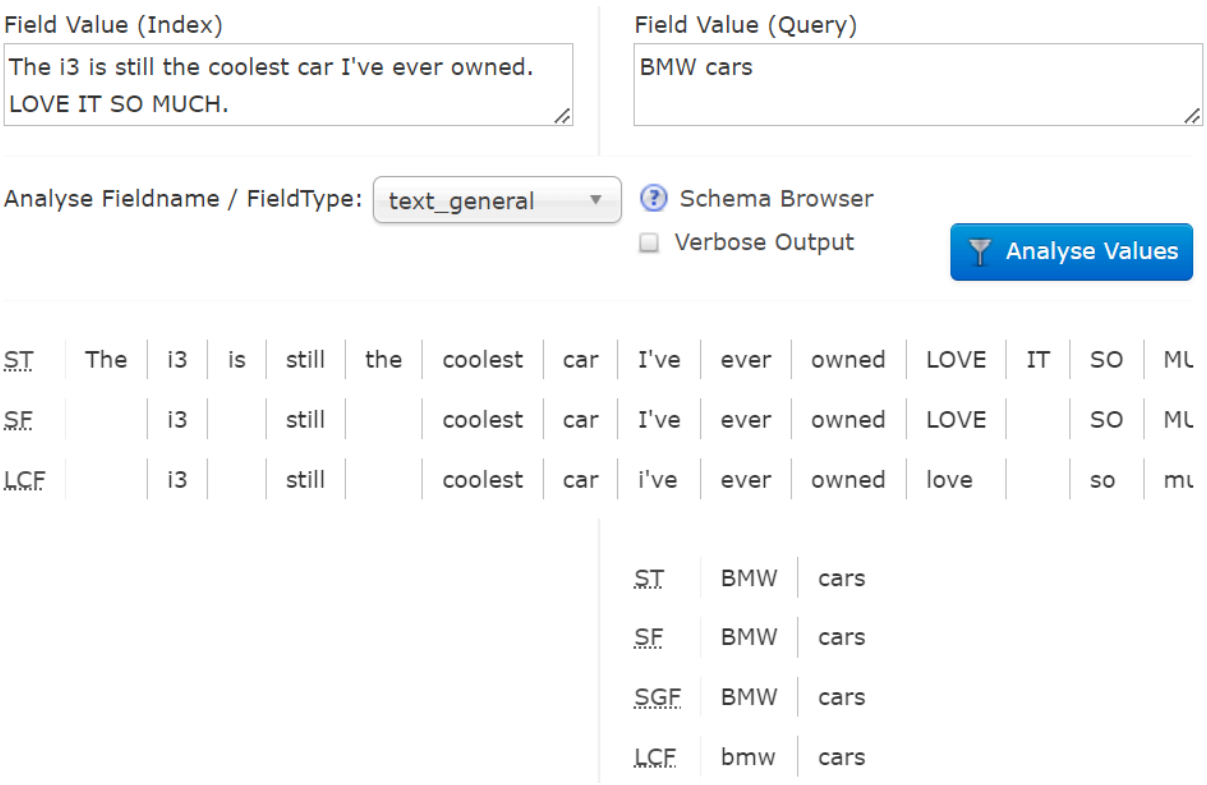


Figure 20: Query fails without custom filters

3.2.3. Defining Custom Schema

After setting up our field type **'text_reddit',** we can go ahead and define a custom schema tailored to our dataset. Although Apache Solr provides a 'Schemaless mode' which automatically detects the schema for your data, this approach may lead to issues with inconsistencies when trying to determine the correct field type. This is because automatic field type detection only considers the first field, and thus can sometimes misinterpret fields. For example, if the first row in a field is only a number and the second row contains a string, Schemaless mode would incorrectly classify the entire field as an integer type based solely on the first value.

Our defined schema specifications are as follows:

```
<field name="author" type="string" multiValued="false" indexed="true" required="true" stored="true"/>
<field name="created_utc" type="pdates" multiValued="false" indexed="true" stored="true"/>
<field name="edited" type="pdates" indexed="true" stored="true"/>
<field name="id" type="string" multiValued="false" indexed="true" required="true" stored="true"/>
<field name="num_comments" type="pdoubles" indexed="true" stored="true"/>
<field name="permalink" type="string" stored="true"/>
<field name="post_id" type="string" stored="true"/>
<field name="subreddit_name" type="string" required="true" stored="true"/>
<field name="text" type="text_reddit" multiValued="true" indexed="true" required="true" stored="true"/>
<field name="text_spellcheck" type="text_general" multiValued="true" indexed="true" stored="false"/>
<field name="type" type="string" required="true" stored="true"/>
<field name="upvote" type="plongs" indexed="true" stored="true"/>
<field name="upvote_ratio" type="pdoubles" indexed="true" stored="true"/>
<field name="url" type="string" stored="true"/>
```

Figure 18: Defined schema in managed-schema.xml

After defining our custom field type and schema, and populating stopwords.txt & synonym.txt, we can then ingest the data either by using Solr GUI or by HTTP POST request. When all is done, we are ready to query our system.

3.2.4. Implementing spellchecker

We have also implemented spellchecking for user queries if users misspelt words leading to queries that yield less than a certain threshold of results. This helps to improve the user's overall search experience. To implement spellchecking, we have to add a new field into our schema **(Figure 18)**. This field is **'text_spellcheck**" and it is used only for spellchecking. Therefore, the field will undergo filters defined in "text_general" which includes the removal of special characters which will be essential in spellchecking. This is because we do not want noise-filled words such as "Tesla!" or "Tesla?" to pollute the frequency of suggested corrections for the word "Tesla". Our spellchecker is defined below:

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

Figure 19: Defined spellchecker in solrconfig.xml

3.3. Querying Apache Solr

As mentioned earlier in the section, Solr can be queried using RESTful requests. We will be using Postman to simulate sending and receiving requests.

3.3.1. REST API Query

The table below shows the queries that we have performed and **Figure 20-22** shows the results of those queries.

Query for	Query type	Request URL
Documents	Single keyword filter (Text = “*Tesla*”)	GET http://localhost:8983/solr/search_reddit/select?q=text:*Tesla*
	Multiple filters (Text = “*Tesla*” AND timeframe = 2017-01-01T00:00:00Z TO 2018-01-31T23:59:59Z AND rows=10)	GET http://localhost:8983/solr/search_reddit/select?q=text:*Tesla* AND created_utc:[2017-01-01T00:00:00Z TO 2018-01-31T23:59:59Z]&row="5"
Spellcheck suggestions	Mispelt the word “Tesla” as “Tesle”	GET http://localhost:8983/solr/search_reddit/spell?indent=true&spellcheck.q=Tesle&spellcheck=true&spellcheck.collate=true

Table 6: *Queries tested*

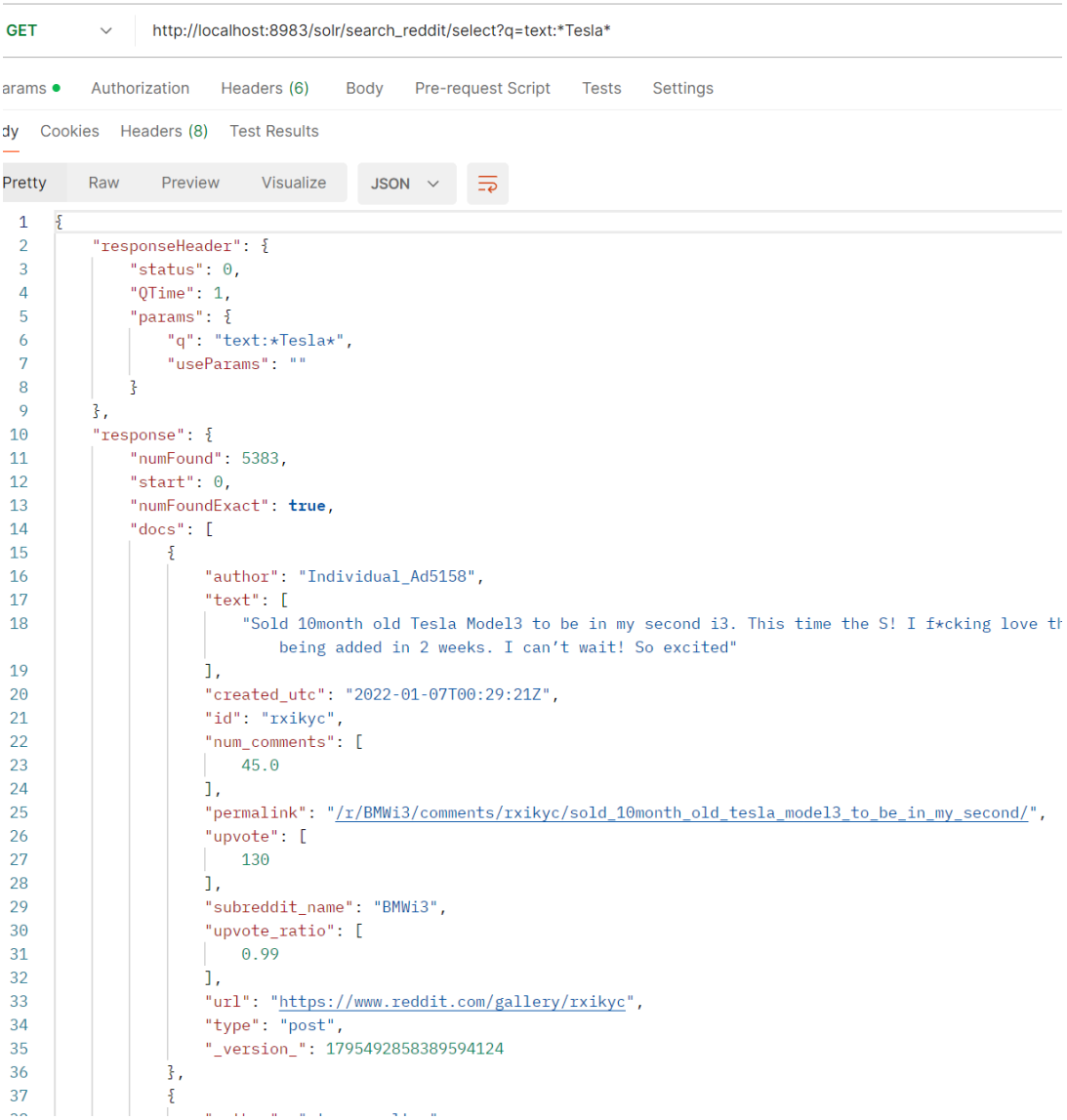


Figure 20: Single keyword filter query search results

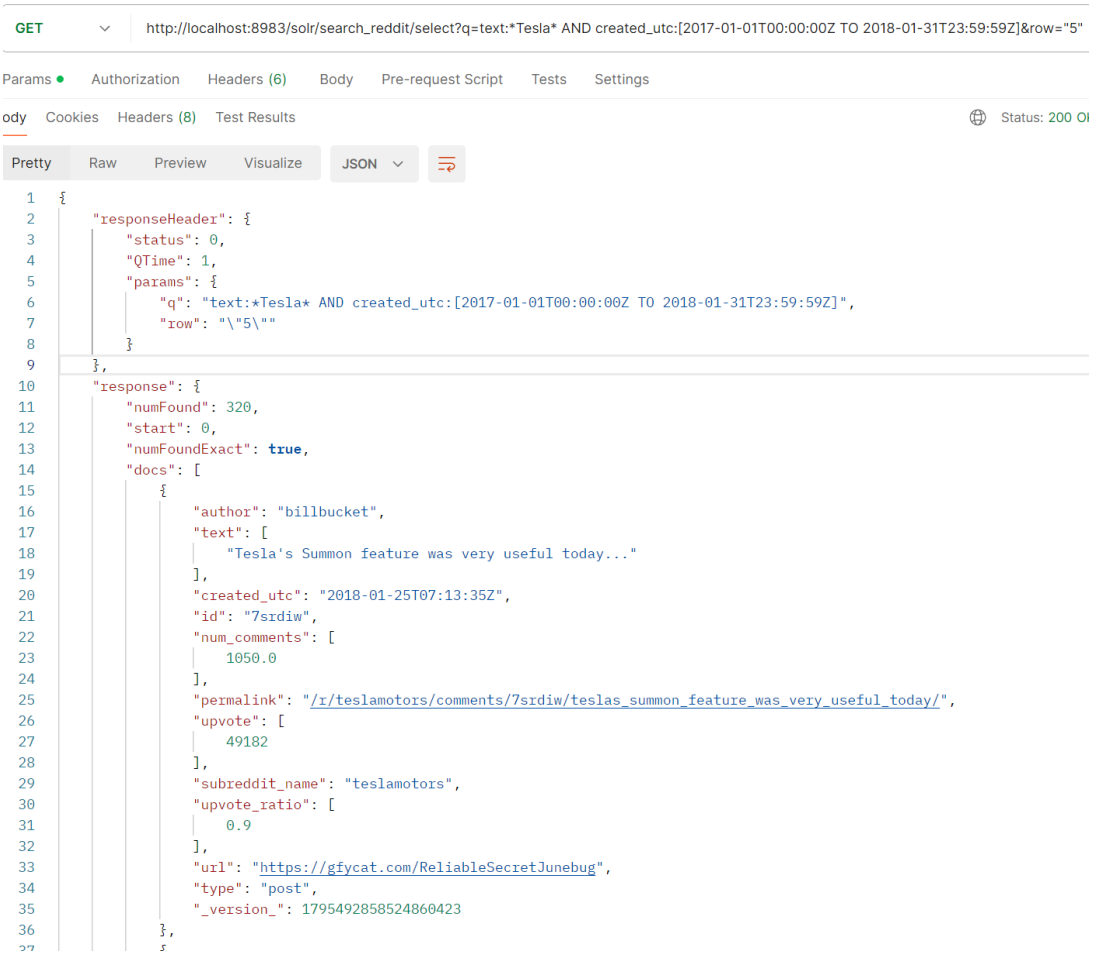


Figure 21: Multiple filters query search results

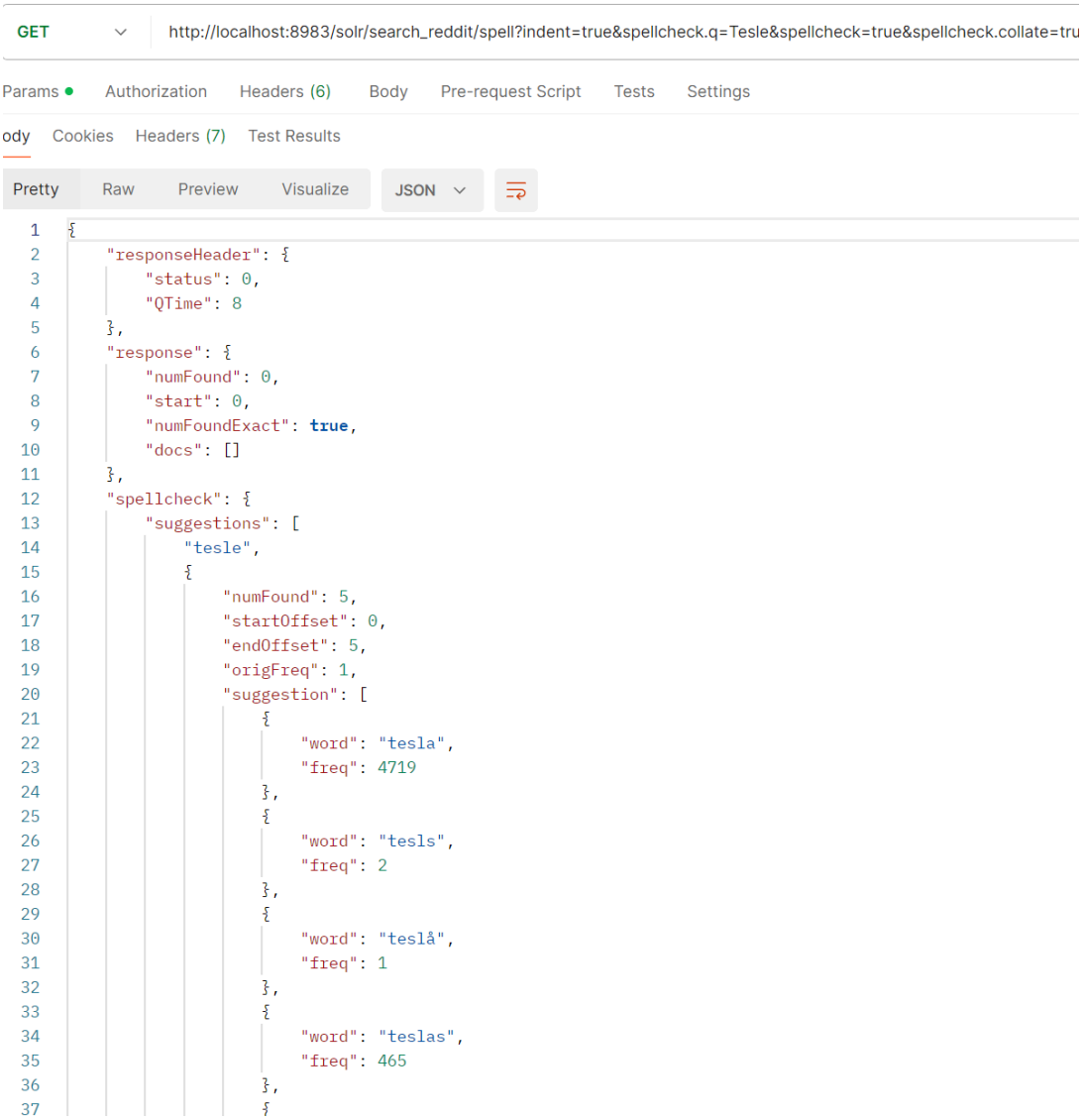


Figure 22: Spellcheck suggestions API results

3.4. Integration into User Interface (UI)

3.4.1. UI Preview & Features

3.4.1.1 Basic features

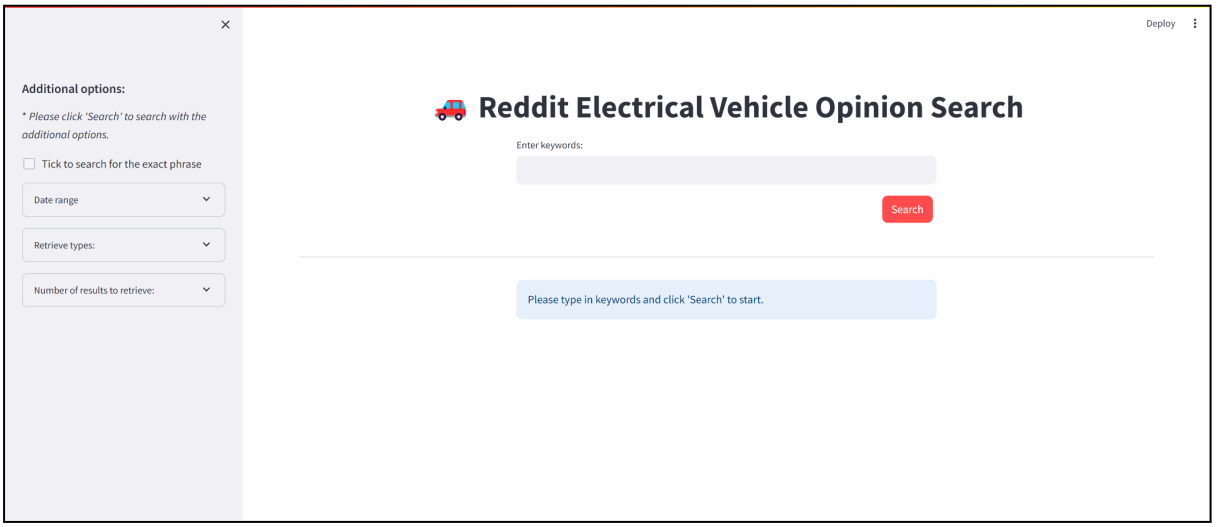


Figure 23: Screenshot of User Interface (UI)

The User Interface (UI) has text input to enter keywords and a search button to search for text in Reddit with the keywords. On the side, there is a sidebar that users can use to apply search filters.

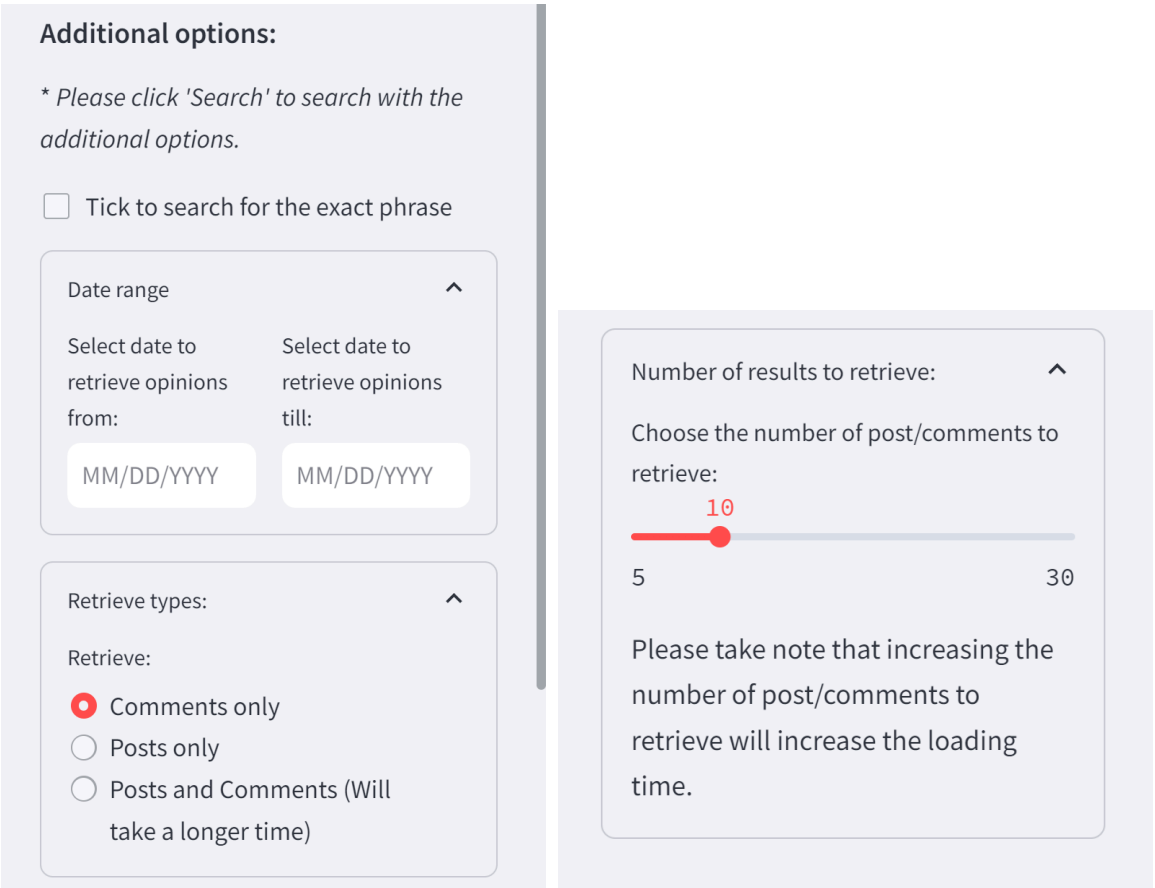


Figure 24: Additional Options in the sidebar of the UI, with the dropdowns expanded

The dropdown containing the filters is minimized at first launch to minimize information overload to the user.

In the Additional Options, users select these filters:

1. Checkbox: When ticked, the exact phrase (if the keyword is a multi-word query), will be searched. By default, this will be unchecked.
2. Date range: Users can specify dates to filter results that were created in that date range. The range can be set blank (i.e. get all results with the keyword), or either one of the dates can be filled up (i.e. get results from the starting date or before the ending date), or both dates can be filled up (i.e. get results only in that date range, with end date inclusive). By default, the date range will be left blank.

   These date filters are provided with a user-friendly UI to select the date, as seen in the figure below:

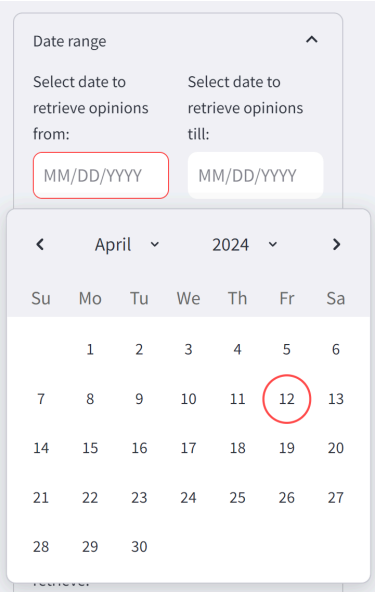


Figure 24: Calendar UI when selecting a date range

3. Retrieve types: Reddit consists of a post and their comment. Users can either choose to retrieve comments or post, or both posts and their comments.
4. Number of results to retrieve: Users can drag the slider to choose how many results to get. By default, the value is set to 10.

3.4.1.2 “Opinions on Reddit” tab

When a keyword is typed and the search button is pressed, a loading spinner is shown below to indicate to the user that the application is running to retrieve the results.

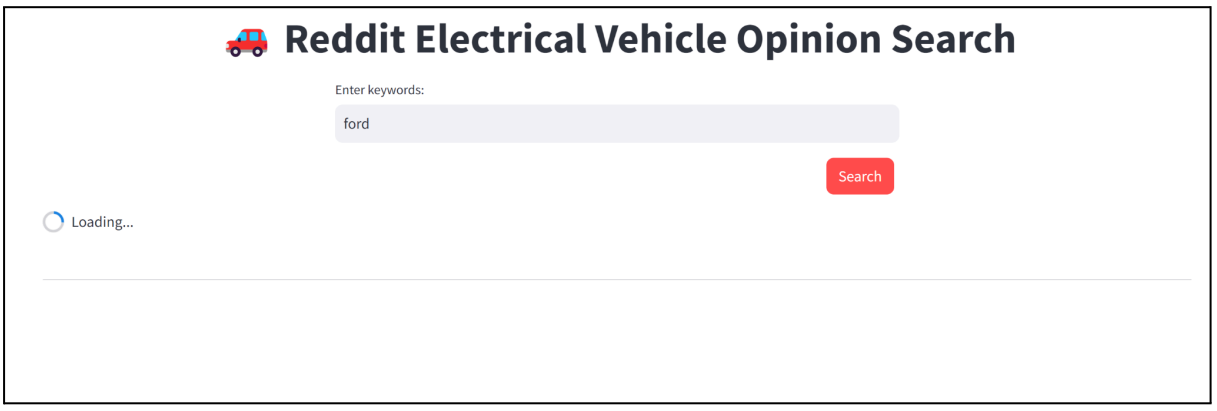


Figure 25: Loading spinner when retrieving results

When results are retrieved, there will be two tabs shown: “Opinions on Reddit” and “Text Analysis”. By default, the user will be directed to the “Opinions on Reddit” tab.

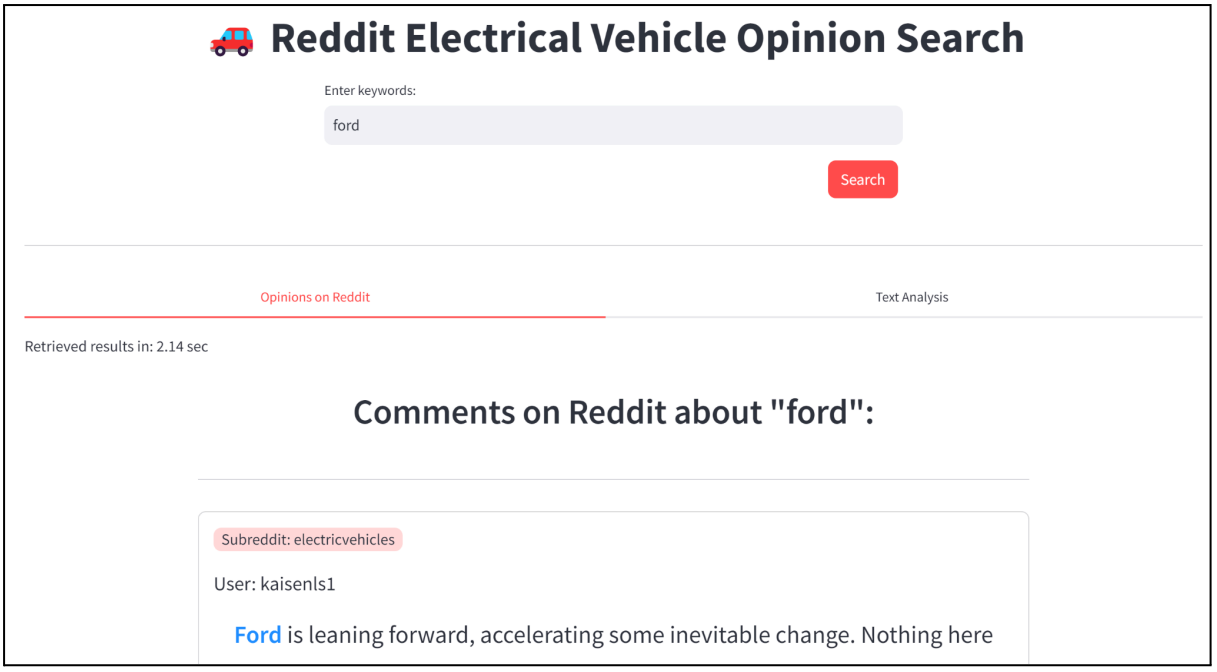


Figure 26: Top part of UI when queried

Under the “Opinions on Reddit” tab, the query time is first displayed, followed by a header indicating to the user what they have searched with.

Results for “Comments only” and “Posts only” will be displayed in the following manner:

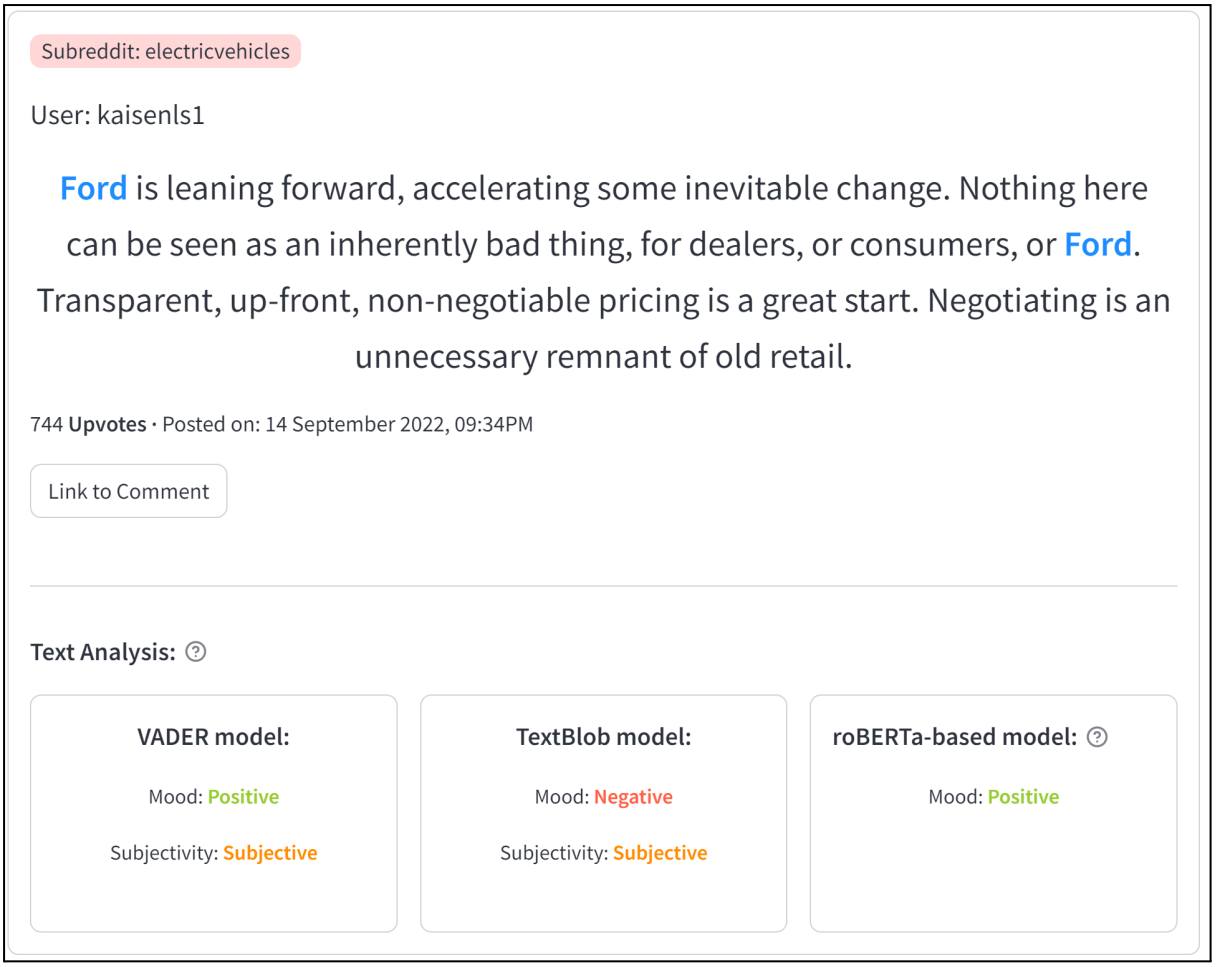


Figure 27: Example of a display of a result for “Comments only” and “Posts only”

A result is enclosed in a box, with the following contents:

1. Name of the subreddit the post/comment originated from
2. Username of the post/comment creator
3. The text content of the post/comment. The query word will be bold and coloured, similar to Google’s search result.
4. The number of upvotes (similar to likes on Twitter, Meta, etc.)
5. Date posted
6. A button to the permalink of the post/comment
7. Prediction results of the text content with VADER, TextBlob and roBERTa-based model. Mood refers to the sentiment of the text. The word ‘Mood’ is used as its meaning will be easily understood by non-technical users, rather than using the word ‘sentiment’. Subjectivity is whether the text content is subjective or objective.

a. A tooltip is displayed to explain the content of “Text Analysis”. It is to help users, especially non-technical users, understand why there are two models.

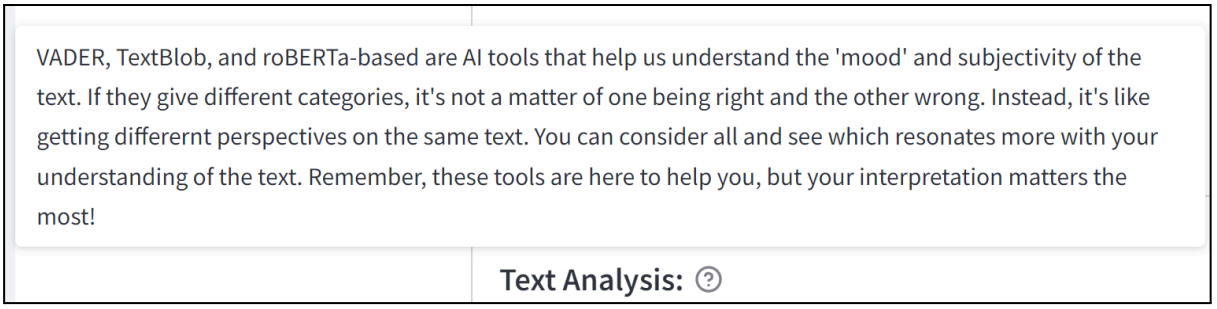


Figure 28: Text analysis tooltip content when hovered over with a mouse

b. A tooltip is also available for the roBERTa-based model, explaining that the subjectivity result for this model not being displayed is not a technical issue, but it is simply not available.

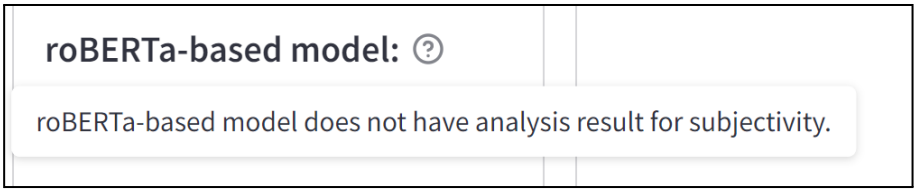


Figure 29: roBERTa-based model tooltip content when hovered over with a mouse

Results for “Both Posts and Comments” will appear differently.

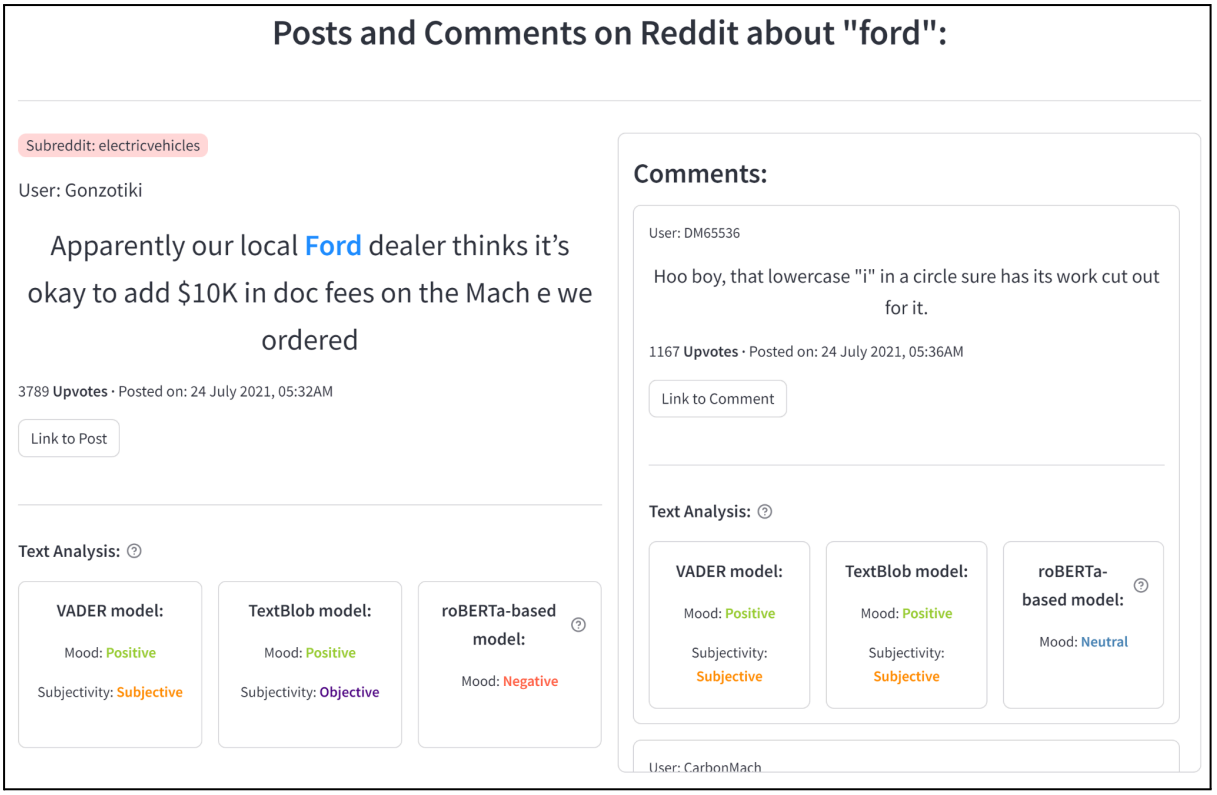


Figure 30: Example of a display of a result for "Both Posts and Comments"

Similar to the results for "Comments only" and "Posts only", there will be a heading and each text content will be displayed in the same format. However, there will be two parts to the results: the post (on the left), and the top comments from the post (on the right). The comments are enclosed in a box, where it can be scrolled to see more comments within the box.

3.4.1.3 "Text Analysis" tab

After the query results have been retrieved, there is also another tab called "Text Analysis". In this tab, users can see the data visualisation and the results based on the two radios shown below.

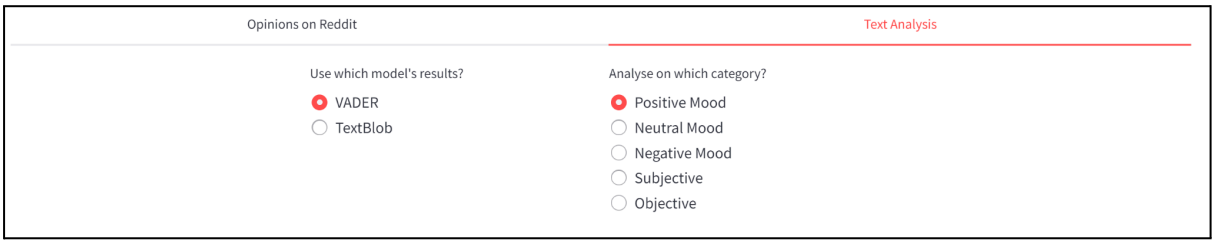


Figure 31: Two radio components in the Text Analysis tab

Users can select the model and the category they want to analyse. By default, “VADER” and “Positive Mood” are selected.

Below the radio components, there is a pie chart and a wordcloud. The pie chart shows the percentage of the sentiment or subjectivity with the selected model. The wordcloud is generated using tokens retrieved from the text labelled with the specified category (in the radio) and the model (specified in the radio as well). Below are examples of the visualisations with “TextBlob model” and “Positive mood”, followed by “TextBlob model” and “Subjective”.

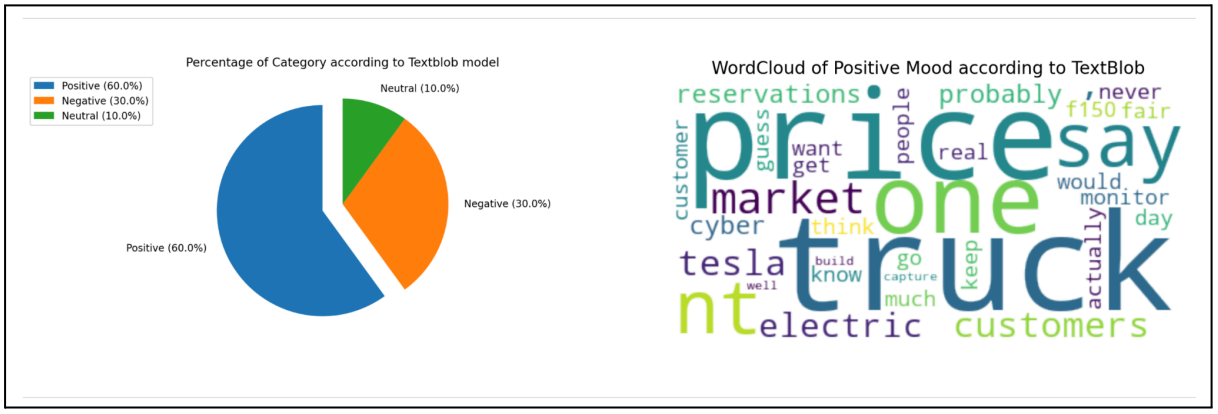


Figure 32: Pie chart and Wordcloud with the query “ford”, with “TextBlob model” and “Positive mood” selection

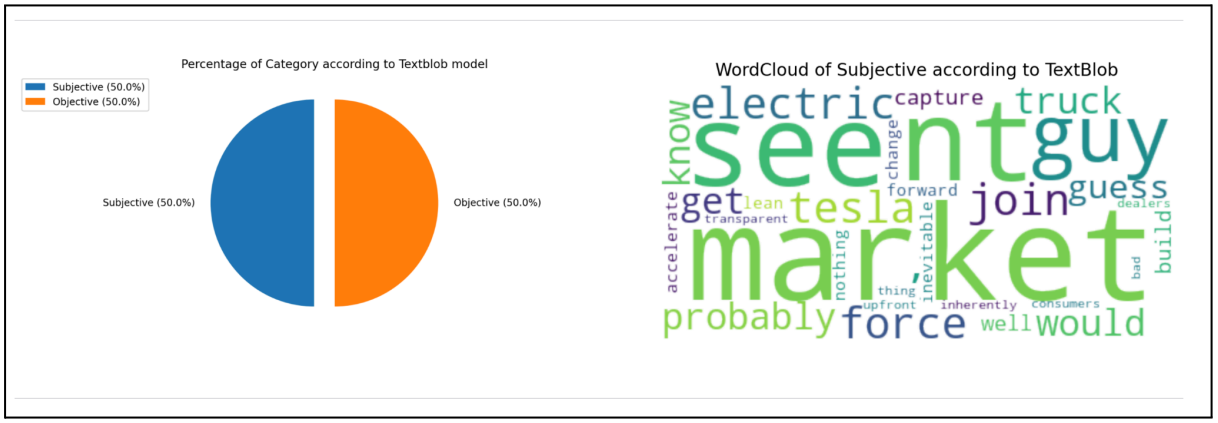


Figure 33: Pie chart and Wordcloud with query “ford”, with “TextBlob model” and “Subjective” selection

Under the visualisations, the text results with selected categories based on the selected model (specified by the two radios) are displayed. It will be displayed the same as the results in the “Opinions on Reddit” tab when querying with “Comments only” and “Posts only” (refer to Section 3.4.1.1).

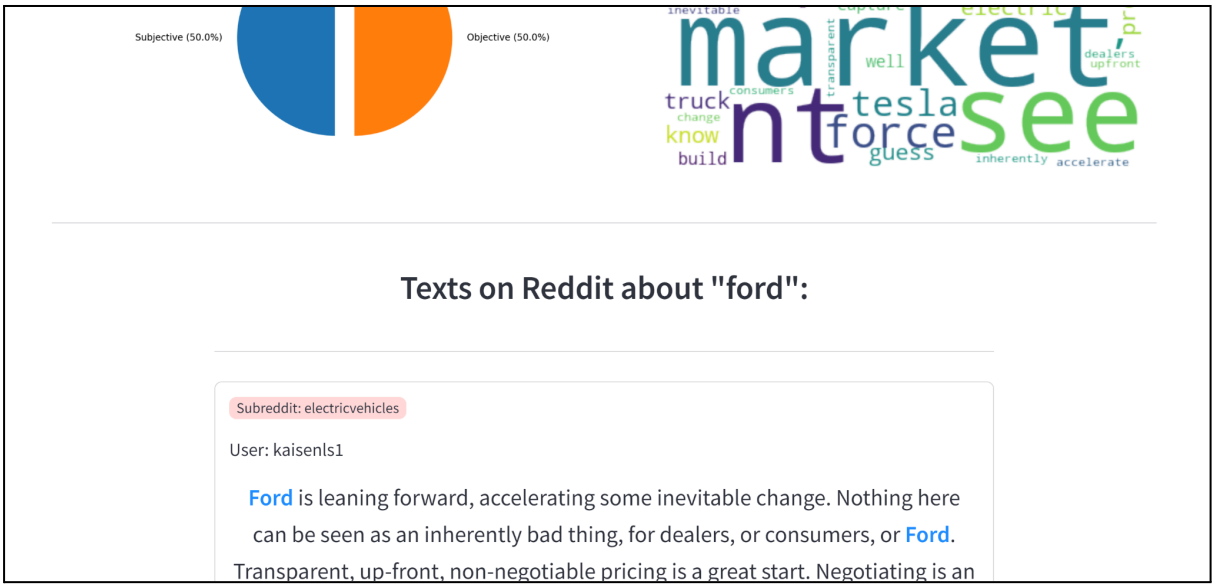


Figure 34: Text results shown below data visualisations, with query “ford”, with “TextBlob model” and “Subjective” selection

3.4.2. UI Performance Analysis

The queries below are tested with the default filters:

- Retrieve types: Comments only
- Number of results to retrieve: 10

Query	Time taken (Secs)	Top Result

“Is Tesla worth it?”	**2.07**	Subreddit: RealTesla User: Greedy_Event4662 Its worth 0 its probably the number 1 scam of all times. Tesla was on the verge of bankruptcy and the fsd and robotaxi lie made elona the wealthiest man on earth, on paper, at least. So its not exagerrated. I knew when they said it comes equiped with hardware that its a pure scam, i work in software development and know whats hype, whats not. Fsd without the software is nothing, without govt approval its even moreso nothing, its plain banned in eu, even ap is. The wording was carefully chosen and deliberate. Fsd can happen, in a reguöated environment maybe, but elona is absolutelly a bonafide scammer. 6 **Upvotes** · Posted on: 27 July 2023, 05:28AM Link to Comment
“tesla resale value”	**2.17**	Subreddit: RealTesla User: biddilybong If you buy a Tesla you are at the mercy of the whims of a narcissistic douchebag. He doesn’t care about you or your safety or your resale value. He will constantly fuck you over if you let him. Get ready for all the subscriptions coming. 284 **Upvotes** · Posted on: 03 September 2023, 10:21AM Link to Comment
“ford”	**2.14**	Subreddit: electricvehicles User: kaisenls1 Ford is leaning forward, accelerating some inevitable change. Nothing here can be seen as an inherently bad thing, for dealers, or consumers, or Ford. Transparent, up-front, non-negotiable pricing is a great start. Negotiating is an unnecessary remnant of old retail. 744 **Upvotes** · Posted on: 14 September 2022, 09:34PM Link to Comment
“should you buy electric cars”	**2.10**	Subreddit: BoltEV User: Move-forward321 Honestly, there should be no other car than a plug in hybrid. Both power plants have their down falls, environmentally and practically. Our infrastructure in no way shape or form support power consumption to that many cars. Plus what it takes to make a large battery like that you have to mine so much material, what you gained in CO2 driving it, you lost making it. I recommend you research how lithium batteries are created. On the practical side a pure electric car has allot of downfalls on its own, terrible range, cold weather kills this exponentially. Infrastructure is not set up for charging station’s and properly trained techs are not up to speed yet. It was reckless and’s irresponsible for this admin to demand all these electric car demands without having a plan to support them. Electric cars shine and out perform their ICE brothers with in town driving and’s short commutes. Now for the ICE

		it has its downfalls as well, maintenance cost to up keep them going, having to buy fuel and terrible in town driving and short commutes. It's not only bad on efficiency but it's hard on them because they don't get up to normal temperatures with the proper amount of thermal cycles to alleviate moisture, which then causes issues with rot and drivability issues. Together though these two compliment each others downfall's and makes a very good reliable, efficient car. That's why I own a Volt:) 1 **Upvotes** · Posted on: 01 February 2024, 09:05PM Link to Comment
"what is the best brand"	**2.09**	Subreddit: RealTesla User: Roguewave1 In 1989 I was in the the market for a new car. I wanted an upscale sedan from what I had previously been able to afford, and I was considering this new car Toyota was bringing to market in its new brand, Lexus. I had owned Toyotas before and respected the company, but, Lexus was totally new. As the car neared market lots of reviews and speculation came forth. The car looked like it had what I was after, but… Finally, I read a review from some Mercedes engineers who got their hands on one of the first Lexus cars on the ground and tore it down because it was designed to be their competition. After dismantling the car completely bolt by bolt, they issued their opinion, which was that Toyota was "trying to buy the market and there was no way in hell they could produce such a car for $39,500." Their equivalent Mercedes was somewhere in excess of $60,000 at the time, as I recall. I took their conclusion to heart and allowed Toyota to buy me in, so I purchased two in the fall of 1989. Turned out they were both terrific buys, and were practically bullet-proof first rattle out of the box for a new car company. How the hell did they do that! I got a couple of friends to follow suit, and theirs were great too. All needed the tires swapped out early on because Lexus missed on that, but did it under warranty, and all had the air conditioner fail just @ the 50k mile warranty period also, which Lexus covered too. But, thank you Merc engineers for your guidance. Thinking back, one deciding factor besides the big one of price difference was that the Lexus had drink holders and Mercs with their Germanic austerity refused to incorporate at that time. I bought a string of Lexus LS models with a few BMW's, Jag's, Lincoln's, etc. thrown in until going Tesla. Never did buy a Mercedes though. Tesla is the best car I have ever driven, although my Ford diesel Excursion was a favorite for its purpose. 2 **Upvotes** · Posted on: 20 September 2023, 02:00AM Link to Comment

3.4.2. Innovations for Enhancing Search

Question 3:

b. Explore some innovations for enhancing the indexing and ranking. Explain why they are important to solve specific problems, illustrated with examples.

You can list anything that has helped improve your system from the first version to the last one, plus queries that did not work earlier but now work because of the improvements you made.

3.4.2.1 Problems faced in the initial stages

Initially, the search engine only showed results for “Both Posts and Comments”. This was because comments especially might be difficult to understand without the context of the post the comment was commented on. However, this was taking too long to display.

The following is the pseudocode for querying with “Both Posts and Comments”:

- Get results for 30 posts with the keywords
- For each post,
    - Get the top 10 comments (top comments with the highest number of upvotes), containing keywords

As each querying roughly takes about 2+ seconds, running the above code will take roughly about (1 query for posts + 30 query for comments for each post) * 2 sec = 62 sec = 1 min 2 sec. This is a very long waiting time for the user, making the user experience bad.

Moreover, the querying method was not efficient. Previously, the querying was done by the following pseudocode:

- For a multi-word query, query with an exact phrase.
    - if the number of results is not at least 30,
        - query with AND operator (search for text with all words, but does not have to be together)
    - if the total number of results (summing up the exact and the AND operator) is still not at least 30,
        - query with OR operator (search for text with either of the words)
- return all results

Doing this query method will take at worst more than 6 sec to complete (2+ sec for each query * 3 queries). With the previously mentioned problem of the long waiting time, a single search (multi-word search) will take (1 query for posts + 30 query for comments for each post) * 6 sec = 186 sec = 3+ min. This waiting time is extremely long and not efficient.

3.4.2.2 Improvements

Instead of showing results with “Posts and Comments” only, we have added an option to search results with “Comments only”, “Posts only”, and “Posts and Comments”. By adding these options, users can search for results more efficiently. If the users wish to search for both, they can select the corresponding selection. However, as it will take more time to query, a note is added beside the option (“Will take a longer time”), to let the user know that this option will take a longer time and that the application did not have any technical issues which resulted to the long querying time. Additionally, “Comments only” was set as default as it was faster than “Posts and Comments”, speeding up the time for default query time. Moreover, posts act like a heading in a news article, hence naturally, posts have fewer words than comments, leading to lesser results for posts compared to comments. Therefore, “Comments only” was chosen as default instead of “Posts and Comments” and “Posts only”.

Next, the query method for comments in “Posts and Comments” has been changed. Instead of retrieving the top comments with the keywords, the top 10 comments with the highest number of upvotes are retrieved. Since comments (especially the ones with a high number of upvotes) are related to the topic of the post, we have deemed that retrieving the top comments of the post (with the keywords) is considered retrieving comments related to the keyword. Additionally, the default number of results retrieved (across all modes: “Posts and Comments”, “Posts only”, “Comments only”) is set to 10. This is because many users of the Google search engine usually only look at the first 10 results at most, and by decreasing the default number of results from 30 to 10, the querying time is greatly reduced, enhancing efficiency. If the user wants more results, they can manually change with the additional

options. There will be a note indicating it will take longer time, hence the user will understand that the application did not have technical difficulties while waiting for the longer querying time.

Therefore, the pseudocode is now changed to the following:

- Get results for 10 posts with the keywords
- For each post,
    - Get the top 10 comments with the highest upvotes

Finally, an option to search for exact matching has been added for the user. This was implemented to reduce the default querying time and to make the default search requirements more lenient to get more results. With this, the pseudocode has been simplified to the following:

- if the user selected exact matching,
    - query for 10 results with exact phrases (search text with exactly the same phrase)
- else,
    - query for 10 results with AND operator (search for text with all words, but does not have to be together)

The OR operator was removed as we deemed irrelevant to the search. Since the OR operator only searches for any one of the words in the keywords and does not score relevance higher when all words are present, it does not make sense to include search results for the OR operator. For example, if the keyword is “electrical vehicle”, including results with the keyword “electrical” might not be relevant as it may be completely unrelated to vehicles (it can be electrical tools, etc.).

Therefore, the longest query time will be (1 query for posts + 10 queries for comments for each post) * 2+ sec = 22+ sec. The calculations for the applied settings are: “Posts and Comments” and 10 results (the default number of results to retrieve). Compared to the

longest query time before the improvements (3+ min), this is a great improvement in efficiency.

3.4.2.3 Other innovations

Spellchecking is implemented when fewer results are retrieved than the supposed number of results to retrieve (can be specified by the user).

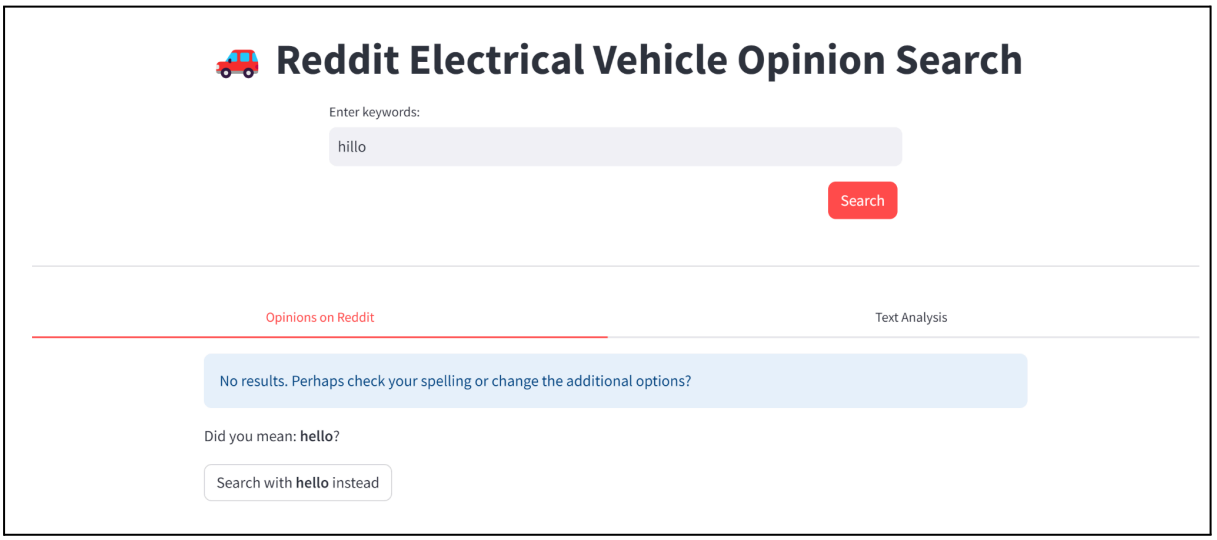


Figure 35: Screenshot of application with spellchecking when few or no results are retrieved

When the button with the suggestions is clicked, the application will automatically search for results based on the suggested keyword.

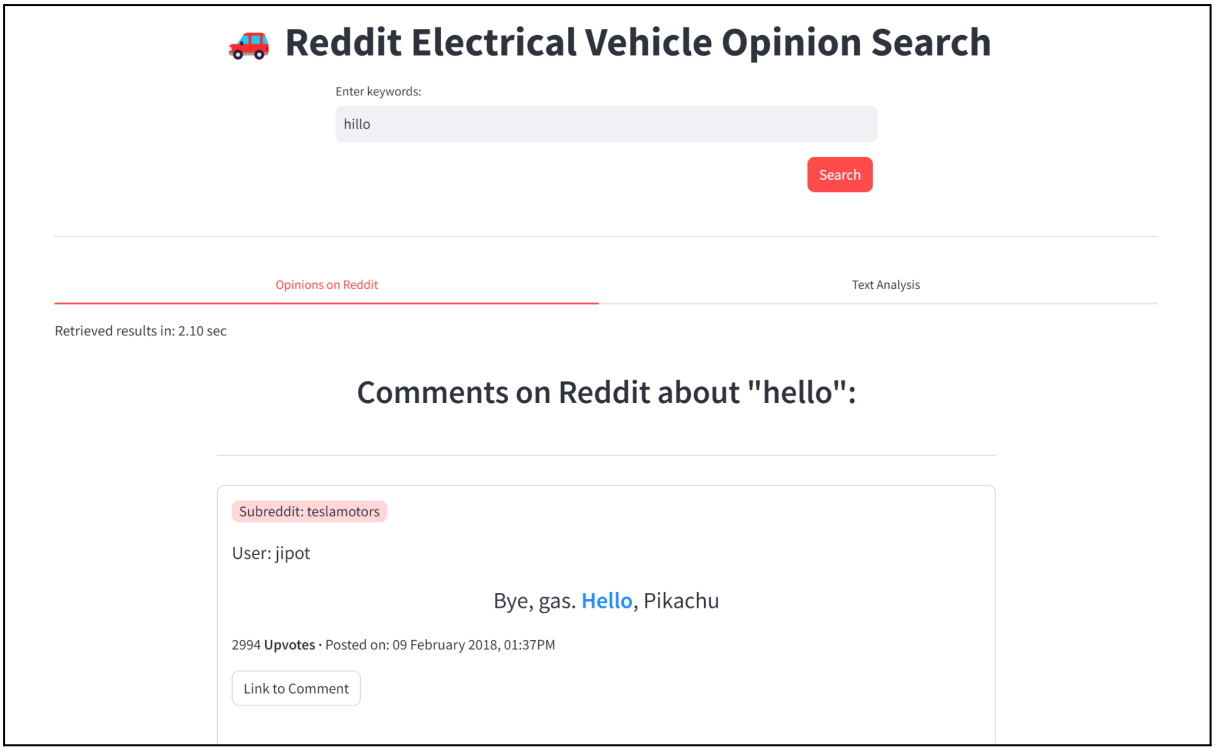


Figure 36: Screenshot of application after applying the suggested keyword search

4. Classification

4.1. Classification Approaches

4.1.1. Motivate the choice of your classification approach in relation with the state-of-the-art

Our team researched several classification approaches to classify the sentiment and the subjectivity of the Reddit posts and comments. All approaches, VADER (Valence Aware Dictionary and sEntiment Reasoner) combined with TextBlob, Bert and roBERTa based models, represent prominent methodologies within the field.

1. VADER

VADER is part of the NLTK (Natural Language Toolkit) library in Python. VADER within NLTK provides a pre-trained sentiment analysis model specifically designed for social media texts.

It has a vast list of words with each tagged with a polarity score representing its sentiment (positive, negative, or neutral). Also, VADER is good at handling nuances such as emoticons, slang, and capitalization commonly found in informal texts. Its rule-based approach results in quick sentiment analysis without having the need for extensive training data.

2. Textblob

After working on VADER, we realised that VADER does not give a numerical representation for subjectivity detection. So, along with VADER, we included Textblob. TextBlob is a simple and intuitive NLP library in Python that offers a range of text processing functionalities, including part-of-speech(POS) tagging, noun phrase extraction, and sentiment analysis. Its sentiment analysis module provides a polarity score for a given text, indicating whether the sentiment expressed is positive, negative, or neutral. We chose it for its ease of use and fast implementation.

3. BERT

BERT (Bidirectional Encoder Representations from Transformers) is a transformer model developed by Google that revolutionised language processing tasks. Unlike other models that process text sequentially, BERT can capture bidirectional contextual information from a given text. BERT considers the full context of a word by looking at both the words that come before and after it, making it effective in various NLP tasks such as sentimental analysis. Hence, this approach will also be useful in sentimental analysis of Reddit comments due to BERT's ability to capture complex contextual relationships within texts. Through a pre-trained model, the model can effectively differentiate between positive, negative and neutral sentiments when we provide it with Reddit comments.

4. Twitter-roBERTa-base

RoBERTa, an evolution of the BERT model designed for natural language processing tasks, is adept at understanding and generating text. The model introduces modifications to the pre-training process, such as removing the next sentence prediction (NSP) objective, using dynamic masking during pre-training, and training on a larger corpus of text for a longer

duration. These modifications aim to improve the model's ability to understand and generate text by exposing it to a more diverse range of language patterns and contexts. The Twitter-roBERTa-base model is a publicly available checkpoint for Sentiment Analysis available on Hugging Face, which is particularly well-suited for our sentiment analysis task. Firstly, it has been trained on an extensive dataset of around 124 million tweets spanning from January 2018 to December 2021. This vast corpus ensures that the model captures the diverse language patterns and expressions present in Twitter data, which is a similar domain to the Reddit data we are using for this project. Moreover, the model has undergone fine-tuning on the TweetEval benchmark, specifically tailored for sentiment analysis tasks. This additional refinement ensures that the model performs optimally in sentiment analysis.

5. roberta-large-mnli

The RoBERTa-large-mnli model, while primarily fine-tuned on the Multi-Genre Natural Language Inference (MNLI) corpus, can also be a suitable choice for our sentiment analysis task. The MNLI corpus, upon which the model is fine-tuned, encompasses a broad range of genres in both spoken and written text. This diversity in text genres ensures that the model is exposed to various linguistic styles. As the Reddit data we use is generated by users with diverse language habits, fine-tuning the MNLI corpus may be beneficial for our task. The RoBERTa-large-mnli model inherits the robust pre-training from the RoBERTa large model, which has been shown to capture rich semantic representations of text. These representations can encode nuanced linguistic features relevant to sentiment, such as sentiment-bearing words, negations, and context dependencies.

4.2. Preprocessing

The preprocessing steps typically involve several stages aimed at cleaning and refining the text. Firstly, we remove unnecessary columns from the dataset, retaining only those relevant to our analysis, such as the text of the comments and possibly metadata like the author and score. Next, we identify and remove comments made by bots or automated accounts, as

these may not reflect genuine sentiment. Additionally, we address any extraneous spaces within the text by systematically removing them, ensuring uniformity in text formatting. Furthermore, we replace common abbreviations and slang terms with their full forms to standardise the language across the dataset. Finally, we may choose to filter the comments based on their word count, selecting only those within a specified range to focus on comments of adequate length for sentiment analysis. By performing these preprocessing steps, we prepare the data for sentiment analysis, enhancing the accuracy and reliability of our subsequent analysis.

```
def preprocess(df, min, max):
    print("Number of rows in original dataset: ", len(df))
    df = drop_useless_columns(df)
    df = drop_bot_comment(df)
    df = remove_df_extra_spaces(df)
    df = df_replace_abbr(df)
    df = select_comments(df, min, max)
    return df
```

4.3. Performance

To evaluate the performance of the classification approaches we have built an evaluation dataset to compare with the classification results.

4.3.1. Evaluation Dataset

The evaluation dataset is built upon the dataset we crawled from the subreddit on Bolt EV, which is an electric vehicle (EV) produced by General Motors (GM) under the Chevrolet brand. The dataset has 3879 rows after preprocessing. To select a balanced evaluation dataset from this subreddit, we first use Twitter-roBERTa-base model to predict all entries. The whole Bolt EV dataset has the below predicted sentiment distribution:

```
Percentage of each unique value:
neutral     39.572055
negative    37.535447
positive    22.892498
Name: label, dtype: float64
```

We then select the popular comments based on the number of upvotes the comments have, aiming to achieve a more balanced predicted sentiment distribution, which implies a more balanced ground truth sentiment distribution. After setting a threshold of 4 for predicted negative and neutral rows and a threshold of 3 for predicted positive rows, the resulting predicted sentiment distribution for the popular comments is as below:

```
Percentage of each unique value:
neutral     35.284281
negative    33.612040
positive    31.103679
Name: label, dtype: float64
```

The distribution is quite balanced and the resulting dataset contains 1196 records. The manual labelling was done by two different annotators and between the annotators 1068 records were in agreement. As two annotators are labelling the same item, we used Cohen's Kappa which is a statistical measure that is commonly used to assess the level of agreement between two annotators who are labelling the same items.

```
kappa = cohen_kappa_score(merged_df['annotator 1'], merged_df['annotator 2'])

print("Cohen's Kappa:", kappa)
```

```
Cohen's Kappa: 0.8391246687137581
```

When doing the annotation, we also asked the annotators to label the comments they think are not relevant or out of context, so that performing sentiment analysis on these comments is impossible even for human annotators. After removing the rows that both annotators believe should be removed, we are left with 1034 rows as the evaluation dataset. The ground truth label distribution shown below:

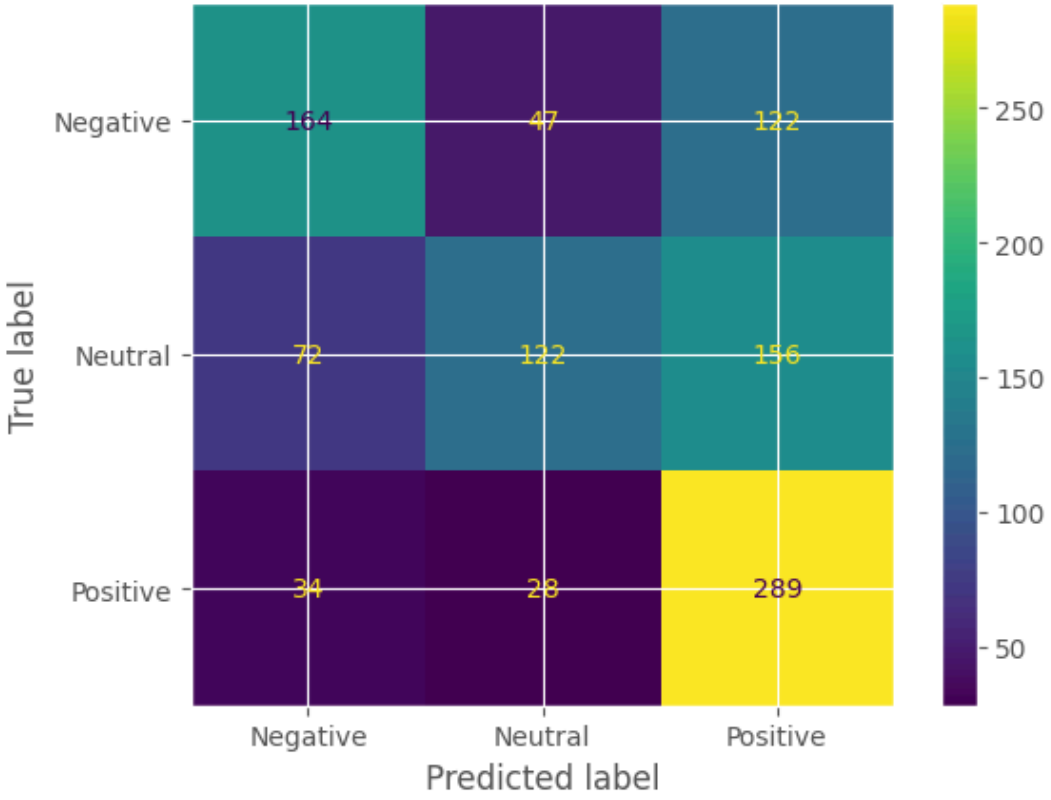
```
Percentage of each unique value:
negative    34.332689
neutral     34.042553
positive    31.624758
Name: label, dtype: float64
```

We can see the ground truth distribution of the evaluation dataset is balanced.

4.3.2. Evaluation Metrics

Precision, recall, and F1 scores are fundamental evaluation metrics used to assess the performance of classification models. The confusion matrix is a tabular representation that visualises the model's predictions against actual class labels, showing true positives, true negatives, false positives, and false negatives. This matrix serves as a foundation for calculating precision, recall, and F1 score. Additionally, the classification report provides a comprehensive summary of the model's performance, including precision, recall, F1 score, and support (the number of occurrences of each class), facilitating a detailed analysis of its predictive capabilities across different classes.

1. VADER

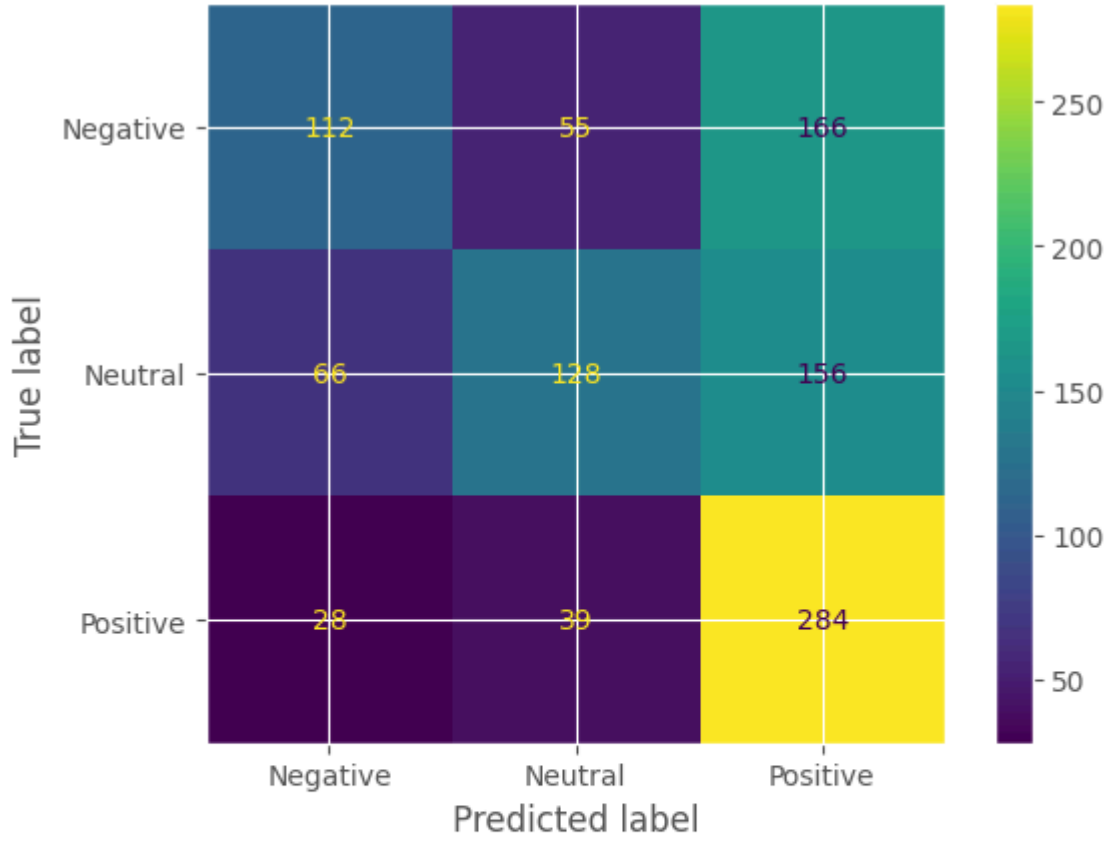


	precision	recall	f1-score	support
negative	0.61	0.49	0.54	333
neutral	0.62	0.35	0.45	350
positive	0.51	0.82	0.63	351
accuracy			0.56	1034
macro avg	0.58	0.55	0.54	1034
weighted avg	0.58	0.56	0.54	1034

From the classification report, we get an accuracy score of 0.56, showing that the classification model correctly predicts the class labels for approximately 56% of the instances in the dataset. “Neutral” has the highest precision(0.62) and “positive” has the highest recall(0.82) and f1-score(0.63) . This model has a lower rate of false positives when predicting 'neutral' instances, and is also effective at minimising false negatives when predicting 'positive' instances, meaning it rarely misses instances that are actually positive (Ground truth).

```
Negative has a accuracy of 49.249249249249246 %
Neutral has a accuracy of 34.85714285714286 %
Positive has a accuracy of 82.33618233618233 %
```

2. Textblob

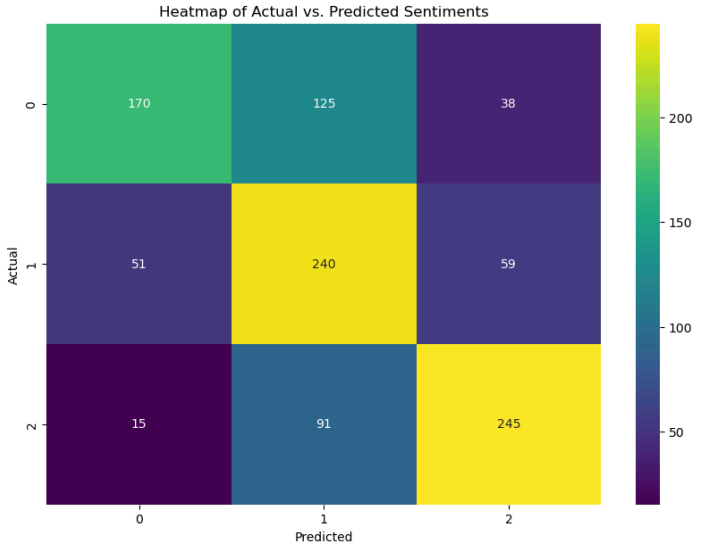


	precision	recall	f1-score	support
negative	0.54	0.34	0.42	333
neutral	0.58	0.37	0.45	350
positive	0.47	0.81	0.59	351
accuracy			0.51	1034
macro avg	0.53	0.50	0.49	1034
weighted avg	0.53	0.51	0.49	1034

From the classification report, we get an accuracy score of 0.51, showing that the classification model correctly predicts the class labels for approximately 51% of the instances in the dataset. “Neutral” has the highest precision(0.58) and “positive” has the highest recall(0.81) and f1-score(0.59) .Evaluation of this model is similar to VADER above, however the accuracy is slightly lower than VADER.

```
Negative has a accuracy of 33.633633633633636 %
Neutral has a accuracy of 36.57142857142857 %
Positive has a accuracy of 80.91168091168092 %
```

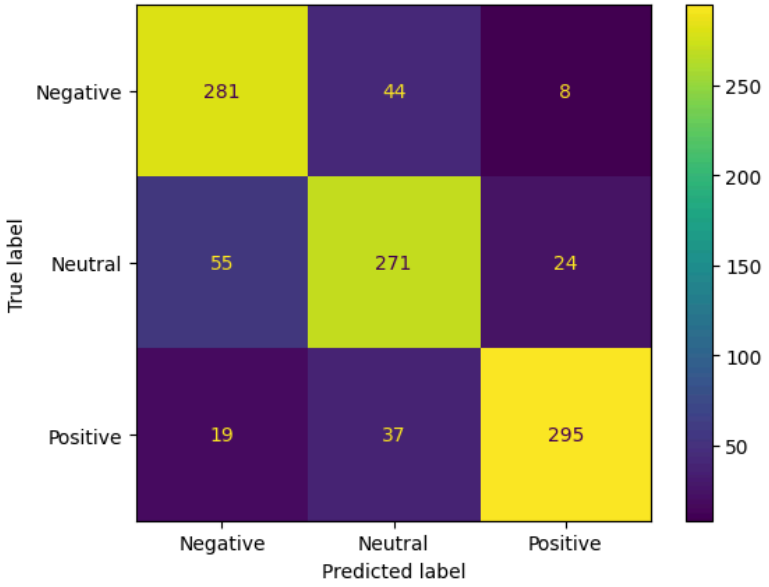
3. BERT



	precision	recall	f1-score	support
negative	0.72	0.51	0.60	333
neutral	0.53	0.69	0.60	350
positive	0.72	0.70	0.71	351
accuracy			0.63	1034
macro avg	0.65	0.63	0.63	1034
weighted avg	0.65	0.63	0.63	1034

From the classification report, this model was able to achieve an accuracy of 0.63. Hence, this model was able to correctly predict the class label for about 63% of the instances in the dataset. While “Positive” and “Negative” class labels have high precision at 0.72, it is worth noting that “Positive” class labels have high recall (0.70) and high f1-score (0.71) among the other class labels. Evaluation for this model seems to be slightly better compared to VADER or TextBlob but predicting for “Neutral” class labels seems to be worse.

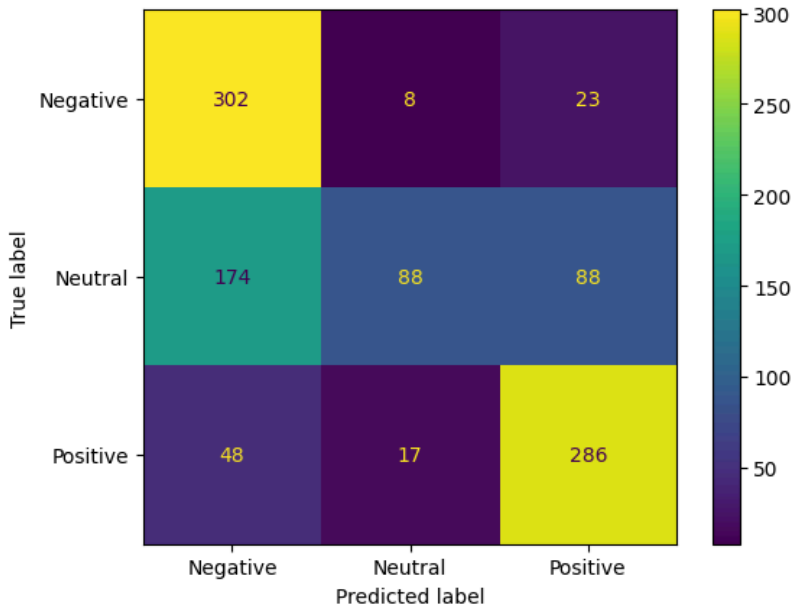
4. Twitter-roBERTa-base



	precision	recall	f1-score	support
negative	0.79	0.84	0.82	333
neutral	0.77	0.77	0.77	350
positive	0.90	0.84	0.87	351
accuracy			0.82	1034
macro avg	0.82	0.82	0.82	1034
weighted avg	0.82	0.82	0.82	1034

For the Twitter-roBERTa-base model, the f1 scores for all three sentiments are around 0.8, which is higher than the previous three models. It shows that the model can reliably analyse sentiment in social media text content.

5. RoBERTa-mnli



	precision	recall	f1-score	support
negative	0.58	0.91	0.70	333
neutral	0.78	0.25	0.38	350
positive	0.72	0.81	0.76	351
accuracy			0.65	1034
macro avg	0.69	0.66	0.62	1034
weighted avg	0.69	0.65	0.62	1034

The RoBERTa-mnli model also reaches high f1 scores in negative and positive comments compared to the first three models. However, its performance on the neutral comments is very unsatisfactory. This may be due to the fact that the RoBERTa-mnli model is not tuned to be a sentiment analysis model. Instead, it is a pre-trained text classification model where users input their candidate labels. Thus, the model may not have a very good understanding of what "neutral" means under the task of sentiment analysis.

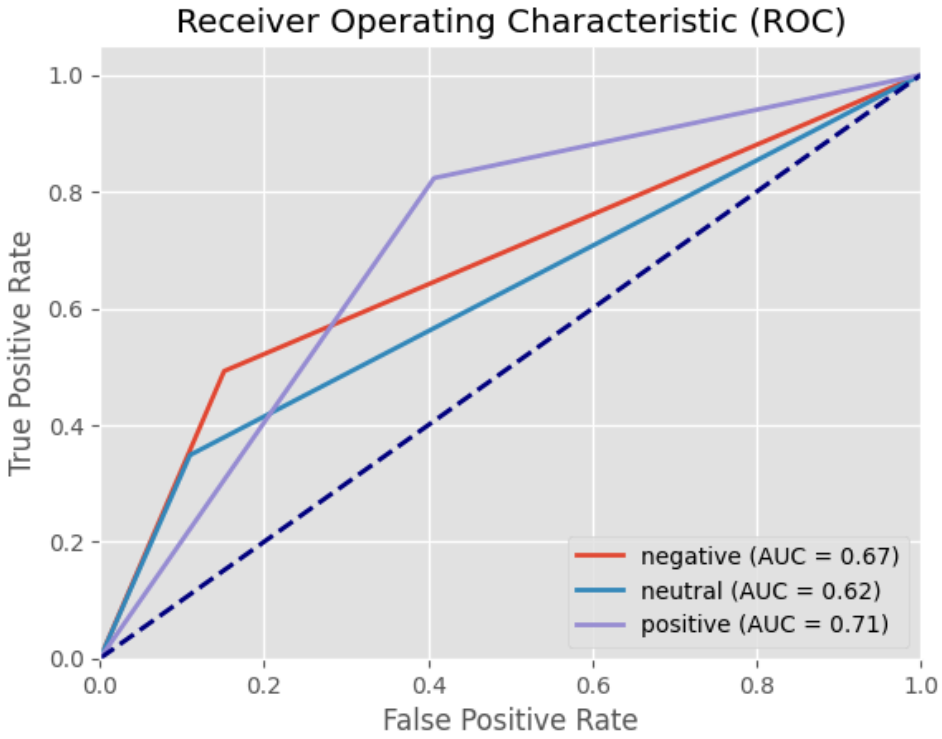
4.3.3. Random Accuracy Test

Random accuracy testing involves establishing a baseline level of accuracy by randomly guessing the sentiment of the text data.

1. VADER

   Plotting the ROC curve, the diagonal line from (0,0) to (1,1) represents random

guessing. A classifier that performs no better than random chance would follow this line.



As for all three labels, they have an AUC value above 0.5, which shows that the model has some discriminatory power, with a higher AUC indicating better discrimination ability. A model with an AUC closer to 1 is better at distinguishing between positive and negative instances.

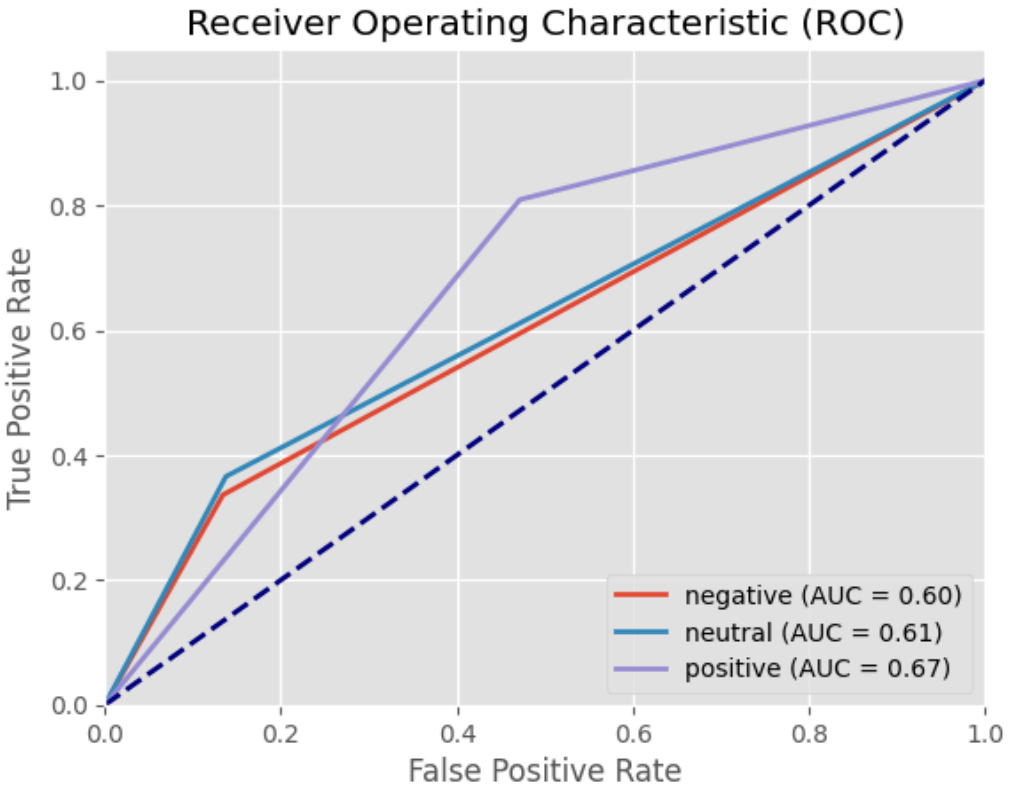
The team also compared the predicted labels with random sampling to test random accuracy.

```
for i in range(0,10):
    random_rows = merged_df.sample(n=100)
    vaderaccuracy = accuracy_score(random_rows["annotator_combined"], random_rows["vader_sentiment"])
    tbaccuracy = accuracy_score(random_rows["annotator_combined"], random_rows["textblob_sentiment"])
    print("Random Sampling Accuracy For Vader:" , vaderaccuracy)
    print("Random Sampling Accuracy For Textblob:" , tbaccuracy)
    print('')
```

```
Random Sampling Accuracy For Vader: 0.56
Random Sampling Accuracy For Textblob: 0.52

Random Sampling Accuracy For Vader: 0.65
Random Sampling Accuracy For Textblob: 0.58

Random Sampling Accuracy For Vader: 0.56
Random Sampling Accuracy For Textblob: 0.56

Random Sampling Accuracy For Vader: 0.58
Random Sampling Accuracy For Textblob: 0.54

Random Sampling Accuracy For Vader: 0.57
Random Sampling Accuracy For Textblob: 0.52

Random Sampling Accuracy For Vader: 0.67
Random Sampling Accuracy For Textblob: 0.58

Random Sampling Accuracy For Vader: 0.52
Random Sampling Accuracy For Textblob: 0.58

Random Sampling Accuracy For Vader: 0.59
Random Sampling Accuracy For Textblob: 0.5

Random Sampling Accuracy For Vader: 0.59
Random Sampling Accuracy For Textblob: 0.49

Random Sampling Accuracy For Vader: 0.56
Random Sampling Accuracy For Textblob: 0.48
```

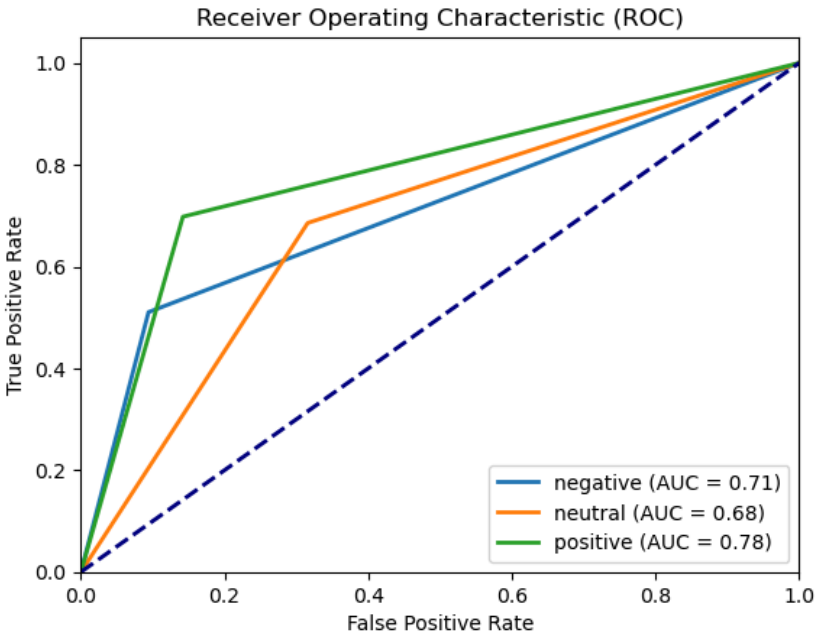
By sampling 100 data for testing 10 times, we can see that the general trend is that Vader performs slightly better than Textblob.

2. Textblob



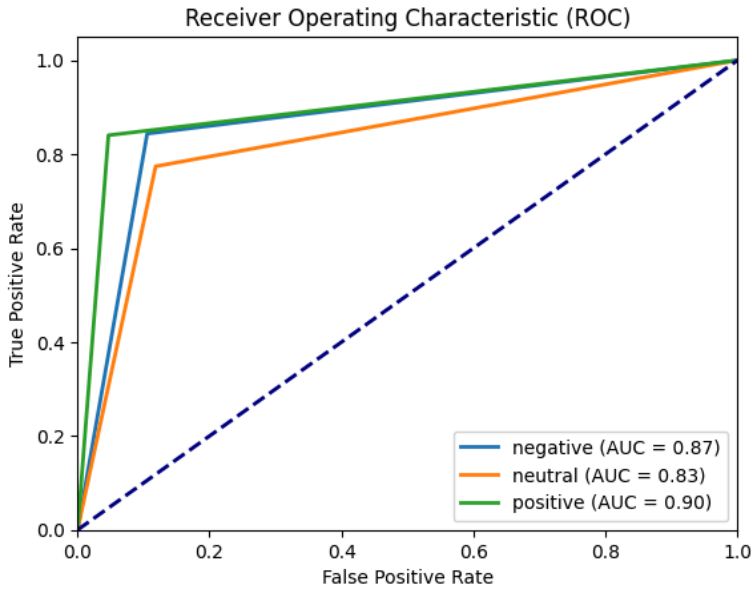
Albeit having AUC values above 0.5, it is lower as compared to VADER which shows that it performs worse than VADER for classifying sentiments. However, with all 3 graphs being above the dotted line, it still shows that the Textblob model performs better than random guessing.

3. BERT

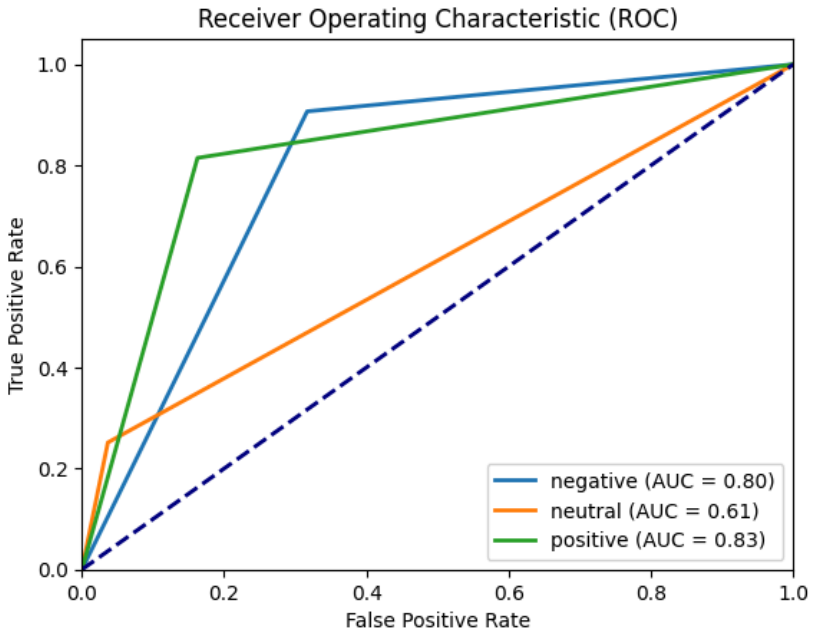


With this ROC curve, we observed that the 3 class labels (negative, neutral and positive) are all above the dotted line which represents the ROC curve of a completely random classifier (AUC of 0.5). Hence, the classifier performs better than random guessing for all the classes. Furthermore, the shapes of the different curves would suggest that at different thresholds, it could optimise the true positive rates for each class while minimising the false positive rate. Compared to Vader and Textblob, it seems that the BERT model can discriminate between the classes better.

4. Twitter-roBERTa-base



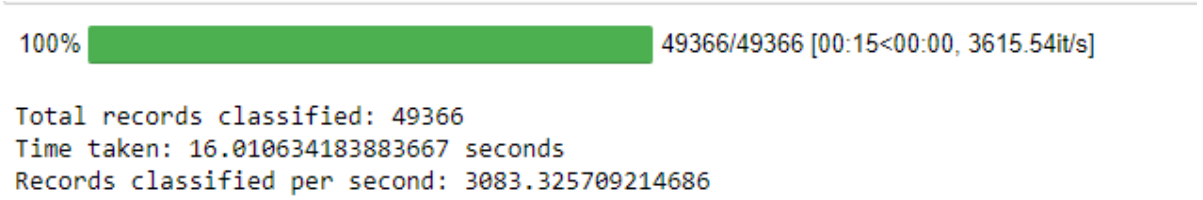
5. RoBERTa-mnli



We can see that for the 2 roBERTa based models, the AUC values are high, which means their performances are much better than random guesses.

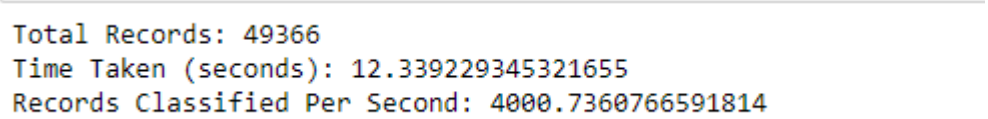
4.3.4. Performance Metrics

1. VADER



The speed of the VADER model is ~3083 records per second which is fast but relatively slower compared to textblob although its accuracy is slightly better.VADER's lexicon-based approach allows for easy customization and adaptation to different domains or languages by adding or modifying sentiment lexicons.

2. Textblob



The speed of the textblob model is ~4000 records per second which is relatively fast.TextBlob is highly extensible, allowing users to easily add custom models, corpora, or word lists to enhance its functionality. This flexibility enables researchers and developers to adapt TextBlob to their specific use cases and domain-specific requirements.

3. Twitter-roBERTa-base

To process 41968 rows (~ the number of rows in the entire Solr system), evaluation using the Twitter-roBERTa-base model checkpoint took 94 minutes 12 seconds. Records classified per second are 7.425, which is way less compared to Vader and Textblob, which can be attributed to their varying levels of complexity and resource requirements. RoBERTa, being a deep learning model with a large number of parameters and sophisticated architecture, demands substantial computational resources for efficient evaluation. It involves tokenization, encoding, and multi-layer

computation for each input text, contributing to its slower processing time. In contrast, Vader and TextBlob rely on rule-based approaches and predefined lexicons, resulting in faster evaluation. As RoBERTa offers state-of-the-art performance, the trade-off between its performance and evaluation speed is inevitable.

4. RoBERTa-mnli

   To process 1034 rows, evaluation using the RoBERTa-mnli model checkpoint took 28 minutes 32 seconds. Records classified per second is 0.604. Thus, with higher accuracy and quicker evaluation, Twitter-roBERTa-base model is more suitable for RoBERTa-mnli model regarding predicting large amounts of data.

5. BERT

   To process 49359 rows, the speed of it is about 91.405 records per second. Even though it is relatively slower as it has a large number of parameters and demands quite an amount of substantial computation resources for evaluation, it still fair well compared to Vader and Textblob

4.4 Classification innovations

To further enhance the performance and robustness of our model, we explored various classification innovations that can aim to improve the accuracy, precision, and recall of our model, ultimately delivering better results from text data.

We explored the following classification innovations:

1. Sarcasm detection
2. Ensemble learning

4.4.1 Sarcasm detection

As the data is from Reddit, one of the significant challenges we face is accurately interpreting sarcasm. Sarcasm can distort the actual sentiment behind a comment,

potentially leading to misunderstandings or incorrect assessments of public opinion. To address this challenge, we decided to try a pre-trained sarcasm detection model to classify comments as either sarcastic or non-sarcastic.

As we previously had two annotators labelling, we combined the annotations into a single true label using an inclusive approach: if either annotator found the text sarcastic, we labelled it as sarcastic. This method ensures that our true labels are more inclusive of sarcastic instances.

The pre-trained sarcasm detection model was then used to classify each comment in the dataset. The model is Google's T5 base fine-tuned on Twitter Sarcasm Dataset for Sequence classification (as text generation) downstream task.

We stored the model's predictions in a new column called "predicted label." The result is shown below:

```
Accuracy: 0.5219611848825332
Precision: 0.11042944785276074
Recall: 0.6206896551724138
F1-score: 0.1875

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       892
           1       1.00      1.00      1.00        87

    accuracy                           1.00       979
   macro avg       1.00      1.00      1.00       979
weighted avg       1.00      1.00      1.00       979
```

The sarcasm detection model's performance was measured using important metrics such as accuracy, precision, recall, and F1-score. While the model demonstrates a moderate ability to recognize sarcastic comments (high recall), its high rate of false positives (low precision) indicates that it may be unreliable for accurately classifying sarcasm in comments.

4.4.2.1 Stacked Ensemble + Training of Models

In this innovation, we touched on training a BERT model using the annotator data from before. The dataset was then split into training and test sets with a ratio of 75/25%, ensuring that most of the data was used for training and a portion of the data would be used for model evaluation. The BERT model that we used was pre-trained ("bert-base-uncased"). We used this model to fine-tune our dataset to ensure the model was able to recognize sentiments expressed in our data. The model may be able to learn from training data and be able to make predictions whether the comments are negative, neutral or positive. Upon training, we test the remaining 25% of the data that has not been seen before to evaluate the results.

We also noticed that with this small (1k) dataset, the dataset can be easily overfitted even after fine-tuning. We have set the learning rate to be 0.0001, batch size = 8 and epoch = 5. The results of the following prediction of the test results can be seen in the figure below.

	precision	recall	f1-score	support
negative	0.83	0.58	0.68	83
neutral	0.65	0.77	0.70	94
positive	0.74	0.81	0.77	79
accuracy			0.72	256
macro avg	0.74	0.72	0.72	256
weighted avg	0.73	0.72	0.72	256

In addition, we also try to use LogisticRegression and RandomForest together with BERT to see if the results may improve or become worse. The result is shown below.

	precision	recall	f1-score	support
negative	0.76	0.63	0.69	83
neutral	0.65	0.73	0.69	94
positive	0.78	0.81	0.80	79
accuracy			0.72	256
macro avg	0.73	0.72	0.72	256
weighted avg	0.73	0.72	0.72	256

Stacking Ensemble seems to do worse compared to just using the BERT model. However, it could also be due to the model used in the stacking ensemble. Perhaps, with different models used, it could provide a better outcome. Furthermore, it is also worth noting that with this small dataset of 1000+ rows, it may not be optimal to train a model well. As 75% of the data is used for training, about 750+ of data may not be optimal to capture detailed features. Moreover, even if the result may seem better than some of the classification models, we must also note that it is based on a small test dataset (250). Hence, we may need to re-evaluate once we have more annotated data.

4.4.2.2 Majority Voting

In ensemble learning, majority voting is a method where multiple individual models or classifiers are combined to make predictions. Each model independently predicts the outcome for a given input, and the final prediction is determined by the most commonly predicted class among all the models. Theoretically, this approach helps improve overall prediction accuracy by leveraging the collective wisdom of multiple models.
We experimented with doing majority voting on the 3 models we mentioned above. Below are the results for the individual models and the 3-model majority voting ensemble model.

1. Vader

```
Accuracy: 0.5560928433268859
Precision: 0.5560928433268859
Recall: 0.5560928433268859
F1 Score: 0.5560928433268859
```

2. Bert

```
Accuracy: 0.6334622823984526
Precision: 0.6334622823984526
Recall: 0.6334622823984526
F1 Score: 0.6334622823984526
```

3. RoBERTa-mnli

```
Accuracy: 0.6537717601547389
Precision: 0.6937598452290512
Recall: 0.6537717601547389
F1 Score: 0.6537717601547389
```

4. 3-model majority voting ensemble model

```
Accuracy: 0.6760154738878144
Precision: 0.6760154738878144
Recall: 0.6760154738878144
F1 Score: 0.6760154738878144
```

We can see that the 3-model majority voting ensemble model outperforms all three models individually. Since the three models are from different categories, the diversity among the ensemble models is essential for improving performance. Combining diverse models helps mitigate individual weaknesses and enhances overall robustness.

We did not involve the Twitter-roBERTa-based model in majority voting, because in the voting process, each model carries equal weight. Thus, since the accuracy of Twitter-roBERTa-based is much higher than the other 3 models but it only carries the same weightage with others, the performance of the majority voting model will still be worse than the performance of Twitter-roBERTa-based model. If including the Twitter-roBERTa-based model, we will not be able to tell whether doing majority voting on several similar performance models can improve the results.

5. Appendix

5.1. Full abbreviation & micro-text mapping dictionary

Abbreviation	Mapped Terms
Reddit specific slangs & lingo	
Alt	Alternative Reddit account
AMA	Ask me anything
AMAA	Ask me almost anything
Benned	Banned
Brony	Male fan of My Little Pony
Cakeday	Birthday
Circlejerk	Elitist group
DAE	Does anyone else
Ent	Pot smoker
ETA	Edited to add
F7U12	FU
[FIXED]	Remix of an original post
FTA	From the article
FTFY	Fixed That For You
Hivemind	Collective
IAMA	I Am A
IMO	In My Opinion
IMHO	In my honest opinion
IIRC	If i recall correctly
ITT	In this thread
Karma	Reddit score

Karmawhore	Desperate for reddit points
Meta-sub	Subreddits talking about Reddit
Meta-subreddits	Subreddits talking about Reddit
MIC	More in comments
Mod	Moderator
MRA	Mens rights activist
Neckbeard	Dirty reddit user
Ninjaedit	sneaky edit
Novelty account	joke account
NSFW	Not safe for work
NSFL	Not safe for life
OP	Original Poster
Orangered	Unread messages
Power user	User with high reddit score
Pun thread	Chain of punny comments
Reddiquette	Rules of reddit
RES	Reddit enhancement suite
Shadow-ban	Silent ban
Shitpost	Trash post
Sockpuppet	Alternate reddit account
SJW	Social Justice Warrior
SRD	Subreddit drama
SRS	Shit reddit says
Sub	Subreddit
TIL	Today I learned
TL;DR	Too Long Didnt read
TLDR	Too Long Didnt read
WIP	Work in progress
X-post	Crosspost

Xpost	Crosspost
wh[o]+sh	Dont get the joke
General internet text slangs & lingo	
LOL	Laugh out loud
TTYL	Talk to you later
ASAP	As soon as possible
FYI	For your information
JK	Just kidding
IDC	I dont care
FTW	For the win
LMAO	Laughing my ass off
BFF	Best friend forever
MFW	My face when
TFW	That feeling when
G2G	Got to go
MSG	Message

6. Submission Links

GitHub Repository Link	https://github.com/ao9000/SC4021-Project
YouTube presentation link	https://youtu.be/vNFm7AD4Iqw?si=XBmpZCtOoCiI31Al
Google Drive link of crawled text data, queries and their results, evaluation dataset, automatic classification results	https://drive.google.com/drive/folders/17FkZQAbmMELhEWYIU_q3TACwpkdKc1m8
Google Drive link to all your source codes and libraries, with a README file that explains how to compile and run the source codes	https://drive.google.com/file/d/1dxKPCGL-LxmZTaST_f6lQIj9Uw6eptu5/view?usp=sharing