CHAPTER 4

INITIAL RESULTS

4.1 Introductions

This section is describing the implementation and experiment part of this research project based on the objective of the proposed research. The code snippet is provided for each of the experiment's implementation, from data collection, data preprocessing, model implementation to model evaluation metric.

4.2 Data Collection

4.2.1 Crawling Method

In this project, primary data collection was done using web crawling method to crawl discourse data on electric vehicle related topic. Web crawling was performed to collect data from the Reddit by using Python Reddit API Wrapper (PRAW) library to interacted with Reddit's API. In the Reddit scraping pipeline, content related to electric vehicles (EVs) from the r/Malaysia subreddit is extracted.

Figure 4.1 shows the code execution for the web scraping process. It loops through a curated list of EV-related search terms and retrieve a maximum of 50 posts per query. For each submission metadata such as 'submission.title', 'submission.selftext', 'submission.score' and 'submission.created_utc' is extracted. Comments of the post are expanded using 'submission.comments.replace_more(limit=None)' to ensures all nested comments thread are fully resolved. Besides, other relevant attributes such as 'comment.body', 'comment.score', and 'comment.total_awards_received' are written to a structured CSV format using Python built in 'csv.writer'.

In addition, there are strategies employed to ensure efficient and robust data collection as showed below.

(a) Exception handling for HTTP 429 (too many requests):

If it exceeds Reddit's rate limits, the API responds with HTTP 429. It captures this through 'ResponseException' and inspect the 'status_code'. If it facing too many requests, it will enter a cooldown period by waiting for 60 seconds before resuming. This is to ensure compliance and avoiding IP bans or account throttling.

(b) General HTTP and Request Exception Handling:

Using 'requests.exceptions.HTTPError' to caught and handle with logging and fallback sleep behavior. For the pipeline resilient to temporary API and network issues.

(c) Deliberate delays between search queries:

This is to avoid triggering Reddit's rate limiter. A 'time.sleep(60)' is performed after processing each search term to ensures that if there are multiple high-volume queries, the script is below Reddit's request thresholds.

(d) No asynchronous processing is used:

The parameter 'check_for_async = False' is set during the initialization of the PRAW Reddit client. This is to suppress warning related to asynchronous event loops, particularly in Python environments that may already use an asynchronous event loop it the background. This enforces synchronous execution of blocking API calls, ensuring a predictable, sequential flow of data extraction and simplifying the debugging and error handling.

```python
# Reddit API credentials
REDDIT_CLIENT_ID = 'y3h5X9SacEDwC-uL8sQFwQ'
REDDIT_CLIENT_SECRET = 'jEpcfFVGCmjm29a5Z-I4045P1kIT5A'
REDDIT_USER_AGENT = 'EVSentimentAnalysisBot by /u/EzNameGG'

# Initialize Reddit API with async warning suppressed
reddit = praw.Reddit(
    client_id=REDDIT_CLIENT_ID,
    client_secret=REDDIT_CLIENT_SECRET,
    user_agent=REDDIT_USER_AGENT,
    check_for_async=False
)

# Search terms
search_terms = [
    'ev', 'electric vehicle', 'electric car', 'tesla', 'charging station',
    'byd', 'ora good cat', 'EV infrastructure', 'EV subsidy', 'ev experience', 'ev issues'
]
subreddit = reddit.subreddit('Malaysia')

# Prepare CSV file
csv_file = 'ev_sentiment_data_new.csv'
with open(csv_file, 'w', newline='', encoding='utf-8') as f:
    writer = csv.writer(f)
    writer.writerow([
        'post_id', 'title', 'selftext', 'post_score', 'upvote_ratio',
        'num_comments', 'post_created_utc',
        'comment_id', 'comment_body', 'comment_score',
        'comment_awards', 'comment_created_utc'
    ])
    #  Loop through search terms
    for i, term in enumerate(search_terms):
        print(f"\n Searching for: {term}")
        try:
            for submission in subreddit.search(term, limit=50):
                print(f" Processing post: {submission.title[:60]}...")

                try:
                    submission.comments.replace_more(limit=None)
                except Exception as e:
                    print(f"Error expanding comments: {e}")
                    continue

                for comment in submission.comments.list():
                    if not isinstance(comment, Comment):
                        continue

                    writer.writerow([
                        submission.id,
                        submission.title,
                        submission.selftext,
                        submission.score,
                        submission.upvote_ratio,
                        submission.num_comments,
                        submission.created_utc,
                        comment.id,
                        comment.body.replace('\n', ' ').strip(),
                        comment.score,
                        comment.total_awards_received,
                        comment.created_utc
                    ])
        except ResponseException as e:
            if hasattr(e, 'response') and e.response.status_code == 429:
                print(" Hit rate limit (HTTP 429). Sleeping for 60 seconds.")
                time.sleep(60)
            else:
                print(f" Unexpected error: {e}")
                continue
        except requests.exceptions.HTTPError as e:
            print(f" HTTP error: {e}. Sleeping 60 seconds.")
            time.sleep(60)
        except Exception as e:
            print(f" Other error: {e}")
            continue

        #  Sleep between terms to avoid search rate limit
        print("⏸ Waiting 60 seconds before next search term...")
        time.sleep(60)

print(f"\n Done! Data saved to {csv_file}")
```

Figure 4.1 Code Snippet of Reddit Web Scraping

4.2.2 System Design and Flow of Web Scraping

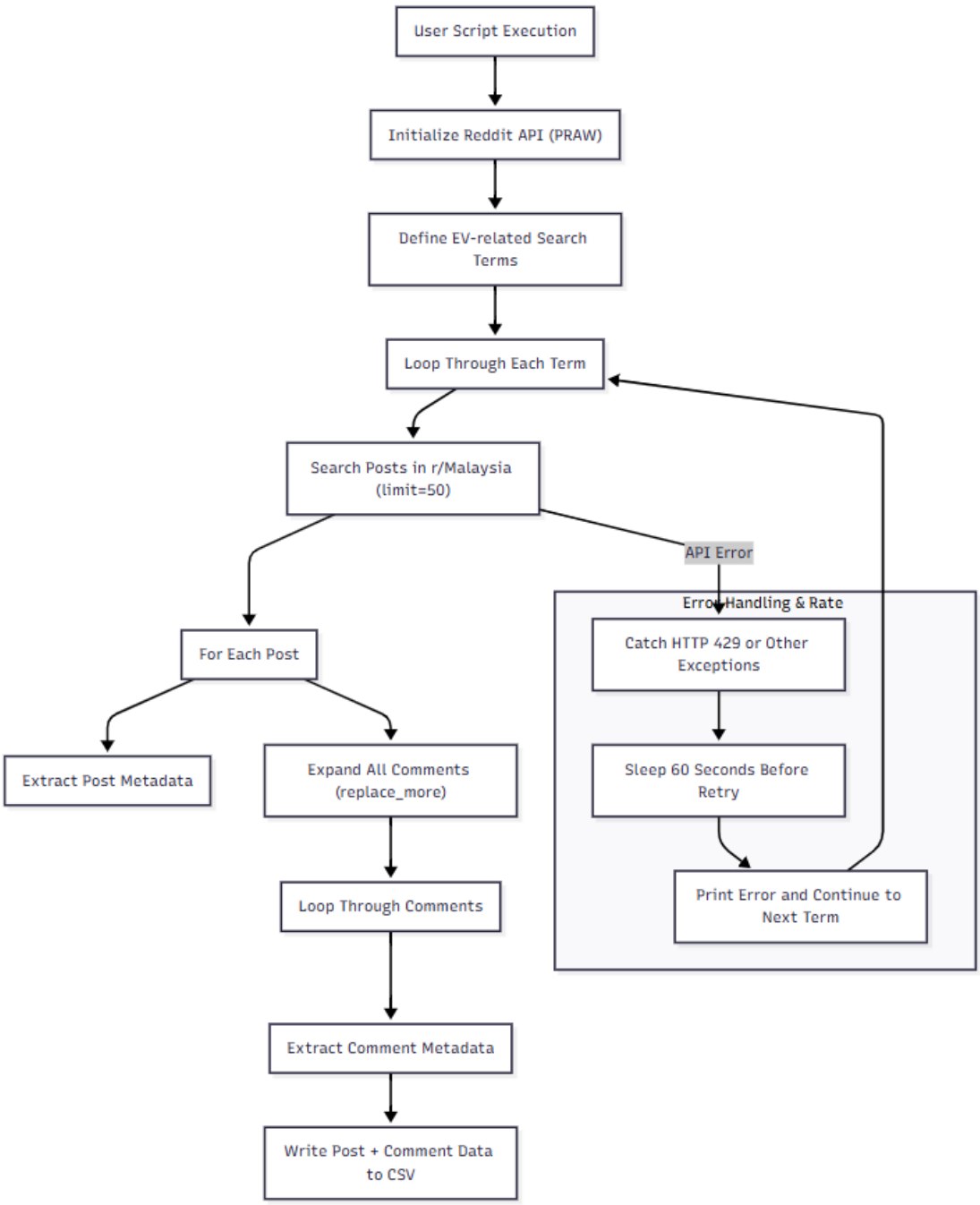


Figure 4.2 Reddit web scraping system design and flow

4.2.3 Number of records collected

The crawler successfully collected 27078 samples with 12 samples as shown in Figure 4.4.

```
import pandas as pd
dir = '/content/drive/MyDrive/Colab_Notebooks/Research_data/ev_sentiment_data_new.csv'
ev_data = pd.read_csv(dir)
ev_data
```

Figure 4.3 Snippet of importing the collected data

	post_id	title	selftext	post_score	upvote_ratio	num_comments	post_created_utc	comment_id	comment_body	comment_score	comment_awards	comment_created_utc
0	113tako	What are your thoughts on EV owners using publ...	NaN	792	0.97	365	1.676562e+09	j8se6pf	EV owners when they find an unattended electri...	540	0	1.676566e+09
1	113tako	What are your thoughts on EV owners using publ...	NaN	792	0.97	365	1.676562e+09	j8twbf7	Apparently some electric cars have battery cap...	212	0	1.676587e+09
2	113tako	What are your thoughts on EV owners using publ...	NaN	792	0.97	365	1.676562e+09	j8uhyyh	Imagine if I set up a bitcoin mining rig here....	129	0	1.676596e+09
3	113tako	What are your thoughts on EV owners using publ...	NaN	792	0.97	365	1.676562e+09	j8s4nfc	That's gonna be one really slow charge. Probab...	385	0	1.676563e+09
4	113tako	What are your thoughts on EV owners using publ...	NaN	792	0.97	365	1.676562e+09	j8sn4cg	It's amazing how many people here feel that it...	250	0	1.676570e+09
...	...	...	...	...	...	...	...	...	...	...	...	...
27073	17pmdxp	New structure to lower taxes for EVs, says Loke	NaN	4	0.83	5	1.699332e+09	k86uwol	Yesssssssssss make EV affordable	3	0	1.699346e+09
27074	17pmdxp	New structure to lower taxes for EVs, says Loke	NaN	4	0.83	5	1.699332e+09	k88l2nx	I just had to spent loads just to buy batterie...	1	0	1.699378e+09
27075	17pmdxp	New structure to lower taxes for EVs, says Loke	NaN	4	0.83	5	1.699332e+09	k8mhejl8	I highly doubt it will be affordable for anyon...	1	0	1.699607e+09
27076	17pmdxp	New structure to lower taxes for EVs, says Loke	NaN	4	0.83	5	1.699332e+09	k8b6goz	Hi there, I doubt your blackouts followed by p...	0	0	1.699414e+09
27077	17pmdxp	New structure to lower taxes for EVs, says Loke	NaN	4	0.83	5	1.699332e+09	k8bi4wf	nope, breakers were not tripped and other hous...	1	0	1.699420e+09

27078 rows × 12 columns

Figure 4.4 Snippet of collected data dataset frame

4.3 Data Preprocessing

After web scraping, the raw CSV file is extracted from the google drive data storage to undergoes data preprocessing process. The data preprocessing process is as below.

(a) Data Transformation

Transformed the 'post_created_utc' and 'comment_created_utc' timestamps column to readable Malaysia time (UTC +8) using code as shown in Figure 4.5 to achieve result as shown in Figure 4.6. As the previous 'post_created_utc' and 'comment_created_utc' are in Unix timestamps format.

```
import pytz

# Define Malaysia timezone
malaysia_tz = pytz.timezone('Asia/Kuala_Lumpur')

# Convert and localize timestamps to Malaysia time
ev_data['post_created_utc'] = pd.to_datetime(ev_data['post_created_utc'], unit='s').dt.tz_localize('UTC').dt.tz_convert(malaysia_tz)
ev_data['comment_created_utc'] = pd.to_datetime(ev_data['comment_created_utc'], unit='s').dt.tz_localize('UTC').dt.tz_convert(malaysia_tz)

# Save to Google Drive
cleaned_path = '/content/drive/MyDrive/Colab_Notebooks/Research_data/ev_sentiment_data_cleaned.csv'
ev_data.to_csv(cleaned_path, index=False, encoding='utf-8')

print(f"Cleaned data saved to: {cleaned_path}")
```

Figure 4.5 Code Snippet for data transformation

post_created_utc	comment_created_utc
2023-09-25 10:25:34+08:00	2023-09-26 07:37:41+08:00
2023-09-25 10:25:34+08:00	2023-09-25 10:26:04+08:00
2023-09-25 10:25:34+08:00	2023-09-25 11:02:29+08:00
2025-05-11 18:34:04+08:00	2025-05-11 21:15:19+08:00
2025-05-11 18:34:04+08:00	2025-05-12 06:17:10+08:00

Figure 4.5 Snippet for data transformation

(b) Data Cleaning: Handling Duplicates

Checking of the duplicated rows in the dataset is conducted as some of the posts appeared many times because they were associated with more than one tags or categorizations. Removed the duplicates so that each post title would only appear as a single post to avoid redundant data.

Figure 4.6 shows the code snippet for checking the duplicated data. It shows in Figure 4.7 that's there is 462 duplicated data. Then, 'cleaned_ev_data = cleaned_ev_data.drop_duplicates()' is used to remove the duplicated data as shown in figure 4.7. After removed duplicated data, only 26606 samples left.

```
# check duplicate
duplicate_rows = cleaned_ev_data.duplicated().sum()
print(f"Number of fully duplicated rows: {duplicate_rows}")

# Preview some duplicate rows
if duplicate_rows > 0:
    print("\nSample duplicate rows:")
    display(cleaned_ev_data[cleaned_ev_data.duplicated()].head())
```

Figure 4.6 Code Snippet checking duplicates

```
Number of fully duplicated rows: 462

Sample duplicate rows:
```

	post_id	title	selftext	post_score	upvote_ratio	num_comments	post_created_utc	comment_id	comment_body	comment_score	comment_awards	comment_created_utc
5153	16rfzdp	A must watch before buying a car! Fuel Subsidy...	NaN	0	0.40	5	2023-09-25 10:25:34+08:00	k27kvvo	Tldw; buy electric car or keep my 18year old h...	2	0	2023-09-26 07:37:41+08:00
5154	16rfzdp	A must watch before buying a car! Fuel Subsidy...	NaN	0	0.40	5	2023-09-25 10:25:34+08:00	k2319s8	The video have english subtitles. If you are ...	2	0	2023-09-25 10:26:04+08:00
5155	16rfzdp	A must watch before buying a car! Fuel Subsidy...	NaN	0	0.40	5	2023-09-25 10:25:34+08:00	k236094	Ya but thought China got plenty of models in t...	1	0	2023-09-25 11:02:29+08:00
5286	1kjy0uu	If someone usually drive 280km everyday for 7 ...	this is a mathematical question and also if an...	87	0.98	98	2025-05-11 18:34:04+08:00	mrqt21w	It depends on the EV you're driving. Tesla ten...	7	0	2025-05-11 21:15:19+08:00
5291	1kjy0uu	If someone usually drive 280km everyday for 7 ...	this is a mathematical question and also if an...	87	0.98	98	2025-05-11 18:34:04+08:00	mrtjc81	RM 280km per day for 7 days and petrol expense...	3	0	2025-05-12 06:17:10+08:00

Figure 4.7 Snippet for duplicated data found

```
cleaned_ev_data = cleaned_ev_data.drop_duplicates()
cleaned_ev_data
```

Figure 4.8 Code Snippet for remove duplicate

26606 rows × 12 columns

Figure 4.9 Snippet of data after removed duplicate

(c) Data Cleaning: Handling Missing data

Before performing methods for handling missing values or text data in the columns, an inspection of missing values in which column is conducted first using 'isnull().sum()' as show in Figure 4.10.

```
cleaned_ev_data.isnull().sum()
```

Figure 4.10 Code Snippet for checking missing values

Based on Figure 4.11, since only the column 'selftext' has missing values and this project is planning doing sentiment analysis on comments, this column is non-essential. Hence, no missing value handling is needed.

	0
post_id	0
title	0
selftext	8633
post_score	0
upvote_ratio	0
num_comments	0
post_created_utc	0
comment_id	0
comment_body	0
comment_score	0
comment_awards	0
comment_created_utc	0

Figure 4.11 Snippet of Inspect Missing Values

(d) Data Cleaning: Remove Text Noise

Focus on cleaning the comments stores in the 'comment_body' column only, as this is the text that will be used for sentiment analysis implementation. Before data cleaning the unnecessary text noise in the comments is inspect first before removing it as shown in Figure 4.12. To view whether there is non-alphabetical character, containing digits, containing multiple spaces and URLs. Figure 4.13 shows parts of the unnecessary contains found in the comment texts. This unnecessary text noise need to be remove to reduce the dimension of the data.

```
import re

def inspect_text_noise(text):

  full_text = ' '.join(text.astype(str))

  non_alpha = sorted(set(re.findall(r'[^a-zA-Z\s]', full_text)))
  has_digits = any(char.isdigit() for char in full_text)
  has_multiple_spaces = bool(re.search(r'\s{2,}', full_text))
  urls = re.findall(r'http[s]?://\S+', full_text)


  print("Non-alphabetic characters:", non_alpha)
  print("Contains digits:", has_digits)
  print("Contains multiple spaces:", has_multiple_spaces)
  print("Sample URLs found:", urls[:5])


inspect_text_noise(cleaned_ev_data['comment_body'])
```

Figure 4.12 Code Snippet of Inspect Text Noise

```
Non-alphabetic characters: ['\x05', '!', '"', '#', '$', '%', '&', "'", '(', ')', '*', '+', ',', '-', '.', '/', '0', '1', '2', '3', '4', '5',
Contains digits: True
Contains multiple spaces: True
Sample URLs found: ['https://preview.redd.it/lin8sp7ogmia1.jpeg?width=256&format=pjpg&auto=webp&s=82949a8b74c6197d07c63a9d644fb0b51bffd7f2',
```

Figure 4.13 Snippet of Founded Text Noise

The steps used to clean the data as shown in Figure 4.14 are removing the URL, hashtags, digit, keeping only English letter, remove multiple space, remove space at beginning and end of string and lowercase the text. Figure 4.15 shows the snippet of the text does not containing any text noise after text cleaning.

```python
def clean_text(text):

  text = re.sub(r'https?://\S+|www\.\S+', '', text) #remove url
  text = re.sub(r'@\w+', '', text) # remove mentions
  text = re.sub(r'#\w+', '', text) # remove hashtags
  text = re.sub(r'\d+', '', text) # remove digit
  text = re.sub(r'[^a-zA-Z\s]', '', text) # keep only english letter
  text = re.sub(r'\s+', ' ', text) #remove multiple spaces
  text = text.strip() #remove space at beginning and end of string
  text = text.lower()

  return text

cleaned_ev_data['cleaned_text'] = cleaned_ev_data['comment_body'].astype(str).apply(clean_text)
```

Figure 4.14 Code Snippet for Text Cleaning

```
Non-alphabetic characters: []
Contains digits: False
Sample URLs found: []
```

Figure 4.15 Snippet for After Text Cleaning

(e) Data Transformation: Data Normalization.

Next is data normalization to convert all the text into standardized format to reduce complexity and variability for text classification tasks later. Text Normalization such as removing stop word and word lemmatization is conducted as shown in Figure 4.15. the output of normalized text is shown in Figure 4.16 in new ‘mormalized_text’ column.

```python
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

def normalize_text(text):

  words = text.split()
  stop_words = set(stopwords.words('english'))
  words = [word for word in words if word not in stop_words]  # Remove stopwords
  lemmatizer = WordNetLemmatizer()
  words = [lemmatizer.lemmatize(word) for word in words]      # Lemmatize

  return ' '.join(words)


cleaned_ev_data['normalized_text'] = cleaned_ev_data['cleaned_text'].astype(str).apply(normalize_text)
```

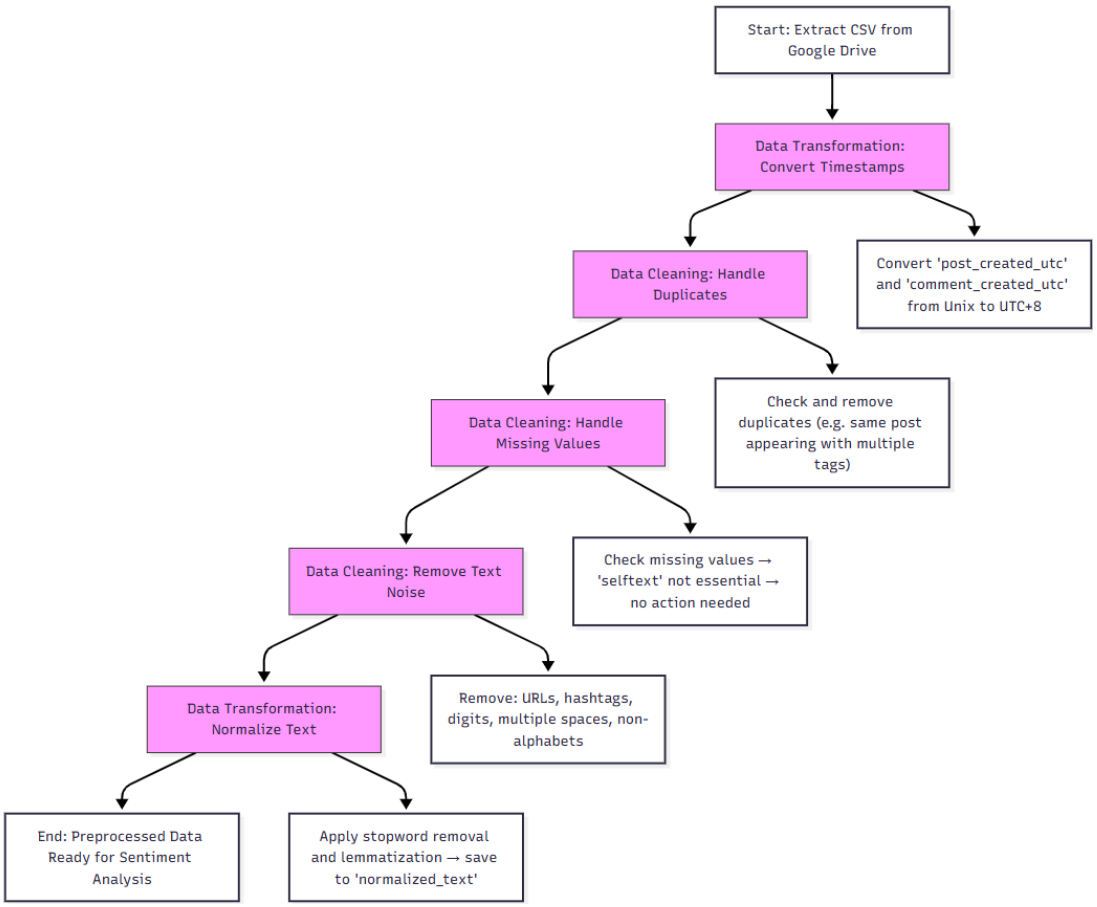
Figure 4.15 Code Snippet for Text Normalisation

	normalized_text
0	ev owner find unattended electrical outlet
1	apparently electric car battery capacity kwh p...
2	imagine set bitcoin mining rig sudden tapping ...
3	thats gonna one really slow charge probably ta...
4	amazing many people feel totally ok charge car...
...	...
27063	yesssssssssssss make ev affordable
27064	spent load buy battery ups overloadings due ex...
27065	highly doubt affordable anyone outside buy use...
27066	hi doubt blackout followed power surge arent f...
27067	nope breaker tripped house experiencing issue

26606 rows × 1 columns

Figure 4.15 Snippet of Normalized Text

4.3.1 Data Preprocessing Flowchart



4.4 Exploratory Data Analysis (EDA)

After data preprocessing, an EDA is conducted to view the structured of the data in order to have overview and understanding of the underlying data. Figure 4.16 shows that many posts have very low scores (minimum is 0), while a few have exceptionally high scores (maximum is 1,478). The median score is 28, suggesting that most posts have moderate engagement.

```
<class 'pandas.core.frame.DataFrame'>
Index: 26606 entries, 0 to 27067
Data columns (total 14 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   post_id              26606 non-null  object
 1   title                26606 non-null  object
 2   selftext             18197 non-null  object
 3   post_score           26606 non-null  int64
 4   upvote_ratio         26606 non-null  float64
 5   num_comments         26606 non-null  int64
 6   post_created_utc     26606 non-null  object
 7   comment_id           26606 non-null  object
 8   comment_body         26606 non-null  object
 9   comment_score        26606 non-null  int64
 10  comment_awards       26606 non-null  int64
 11  comment_created_utc  26606 non-null  object
 12  cleaned_text         26606 non-null  object
 13  normalized_text      26606 non-null  object
dtypes: float64(1), int64(4), object(9)
memory usage: 4.1+ MB
None
```

	post_score	upvote_ratio	num_comments	comment_score	comment_awards
count	26606.000000	26606.000000	26606.000000	26606.000000	26606.0
mean	228.345148	0.903821	305.486845	5.617868	0.0
std	350.902110	0.097813	226.680608	19.530234	0.0
min	0.000000	0.310000	1.000000	-59.000000	0.0
25%	14.000000	0.880000	97.000000	1.000000	0.0
50%	28.000000	0.910000	305.000000	2.000000	0.0
75%	353.000000	0.970000	507.000000	5.000000	0.0
max	1478.000000	1.000000	800.000000	869.000000	0.0

Figure 4.16 Snippet of Statistical data description

Figure 4.17 shows the frequency of posts by year. There is rapid growth of comments from 2019 to 2022, and a subsequent decline from 2023 to 2025

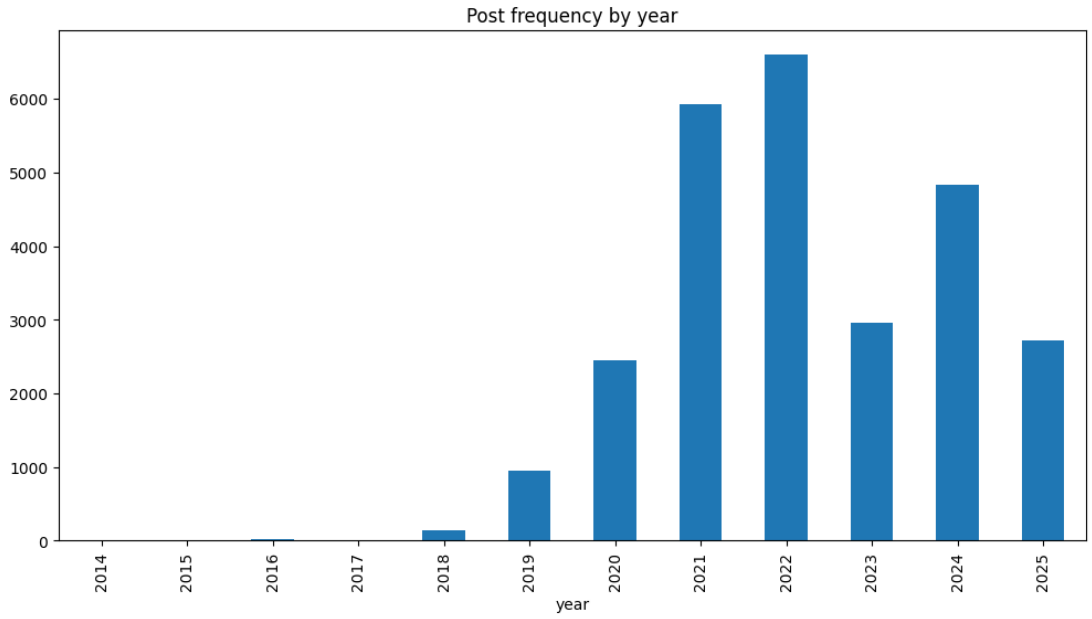


Figure 4.17 Snippet of Post Frequency by Years

Figure 4.18, 4.19, 4.20 shows that most users tend to write very short comments, possibly indicating brief interactions or quick responses. Extremely long comments are rare. The distribution of comment lengths is highly right-skewed. Most comments are very short, while a small number of comments are extremely long. The short comments are majority falling below 131 characters.

```
 Comment Length Stats:
count    26606.000000
mean       111.504999
std        166.462445
min          0.000000
25%         27.000000
50%         63.000000
75%        131.000000
max       4957.000000
Name: normalized_text, dtype: float64
```

Figure 4.18 Comments Length Statistic

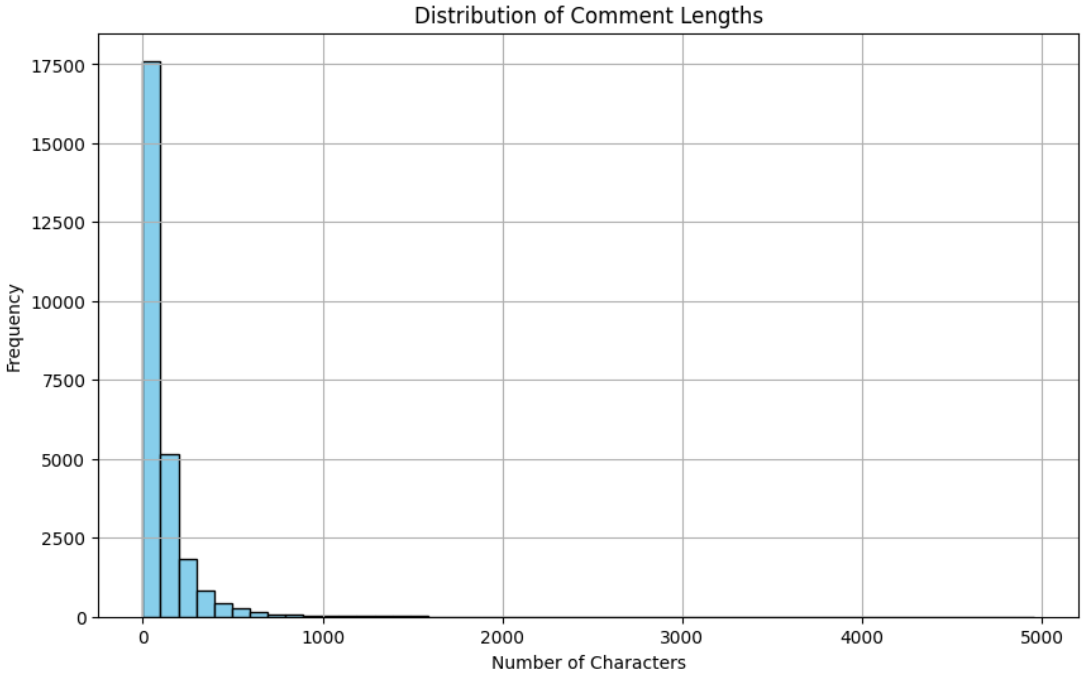


Figure 4.19 Distribution of Comment Lengths

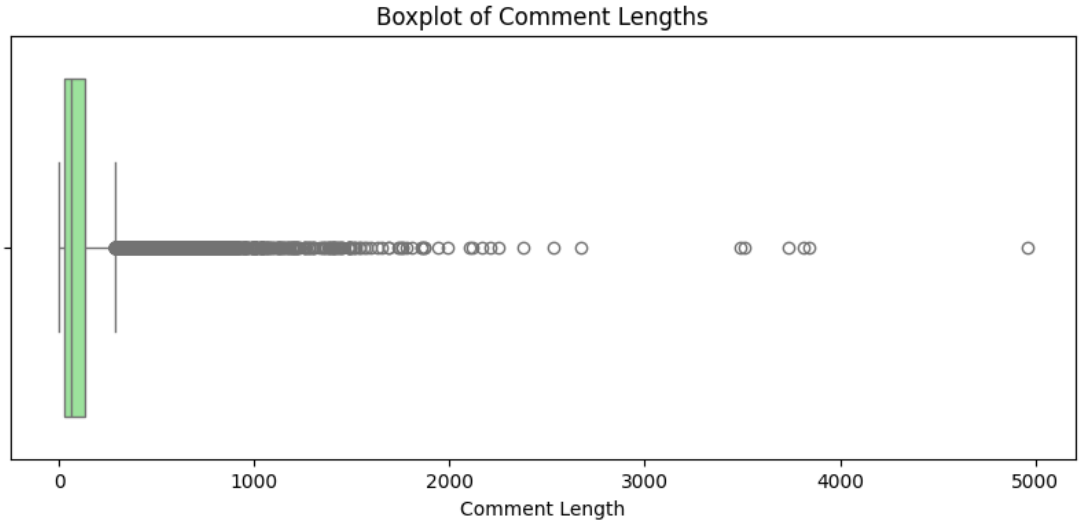


Figure 4.20 Boxplot of Comment Lengths

Both visualizations from Figure 4.21 and Figure 4.22 confirm that the primary topic of discussion is electric vehicles. Positive sentiments are reflected by words like good, like, and work. Negative sentiments or challenges are indicated by words such as problem, issue, expensive, and hard. Discussions likely revolve around the benefits and drawbacks of EVs, including technological aspects, cost, and government policies.

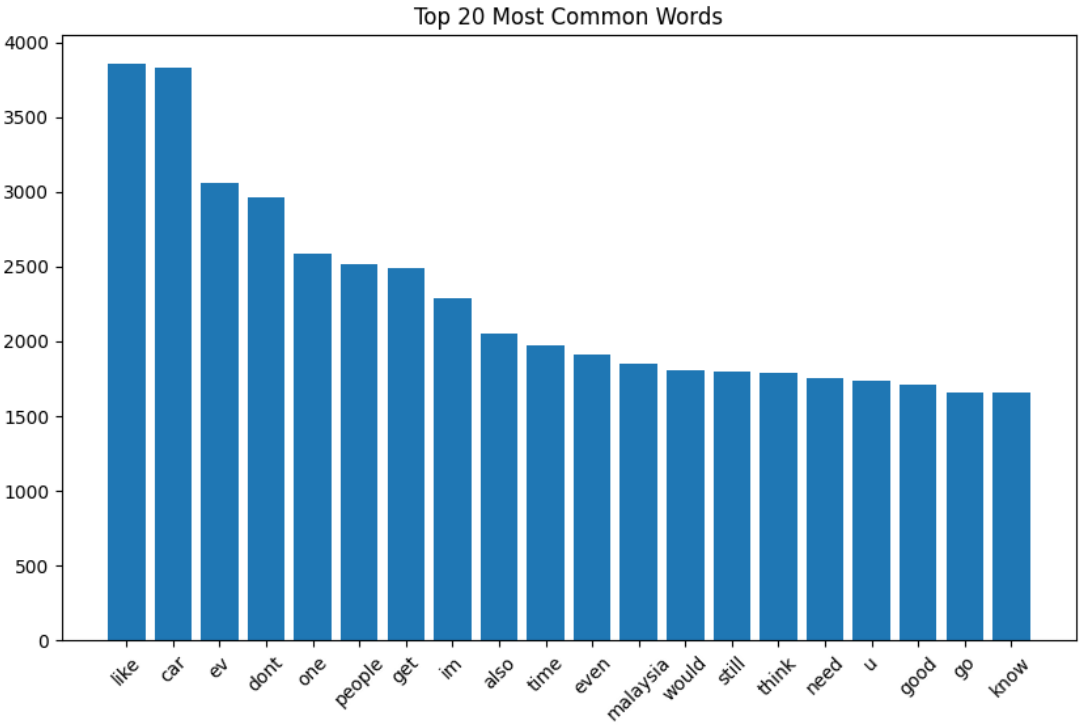


Figure 4.21 Bar Chart of Top 20 Words

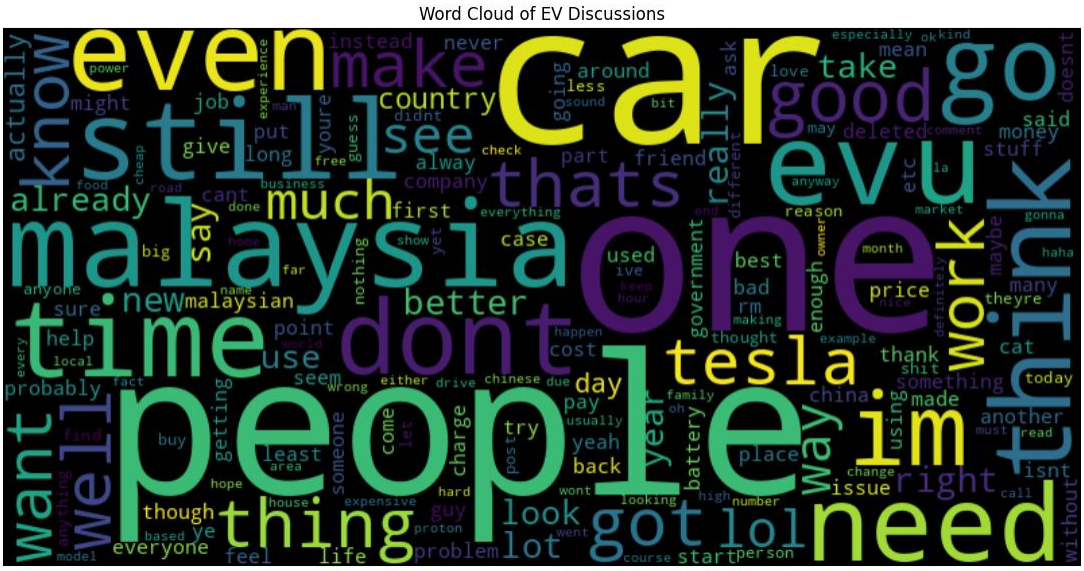


Figure 4.22 Word Cloud of the Text

4.5 Data Vectorization

Next is the features extraction phase. The features is conducted using BERT embeddings. BERT is used to extract machine readable data from the comments text. Figure 4.23 shows the implementation of BERT embedding.

```
from transformers import BertTokenizer, BertModel
import torch

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased')
model.eval()

def get_bert_embedding(text):
    with torch.no_grad():
        inputs = tokenizer(text, return_tensors='pt', truncation=True, padding=True, max_length=512)
        outputs = model(**inputs)
        return outputs.last_hidden_state.mean(dim=1).squeeze().numpy()

X_bert = np.vstack(cleaned_ev_data['normalized_text'].apply(get_bert_embedding))
```

Figure 4.23 Code Snippet of BERT Embedding

4.6 Sentiment Labelling

Valence Aware Dictionary and Sentiment Reasoner (Vader) is used for the automatic sentiment labelling for the normalized text comment data before model training. Vader used predefined sentiment lexicon which is a list of English words and

expression had a valance score to indicated how positive or negative is the particular word. Figure 4.24 shows the code implementation for VADER labeling. Each row in the 'normalized_text' column is classified based on the compound score computed by the VADER. A compound score of more than and equal to 0.05 is positive sentiment, while a score of less than and equal to -0.05 is negative sentiment and score between it is neutral sentiment.

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

analyzer = SentimentIntensityAnalyzer()

def get_sentiment(text):
    score = analyzer.polarity_scores(text)['compound']
    if score >= 0.05:
        return 'positive'
    elif score <= -0.05:
        return 'negative'
    else:
        return 'neutral'

cleaned_ev_data['sentiment'] = cleaned_ev_data['normalized_text'].apply(get_sentiment)
```

Figure 4.24 Code Snippet of Vader Sentiment Labelling

Figure 4.25 and Figure 4.26 shows the results oof automatic labelling using VADER. The bar graph visualisation shows that there is imbalance class distribution for the labelled sentiment with most of the text sentiment are classified as positive.

	sentiment	normalized_text
0	neutral	ev owner find unattended electrical outlet
1	negative	apparently electric car battery capacity kwh p...
2	negative	imagine set bitcoin mining rig sudden tapping ...
3	neutral	thats gonna one really slow charge probably ta...
4	positive	amazing many people feel totally ok charge car...

Figure 4.25 Output of Sentiment labelled for Comment Text

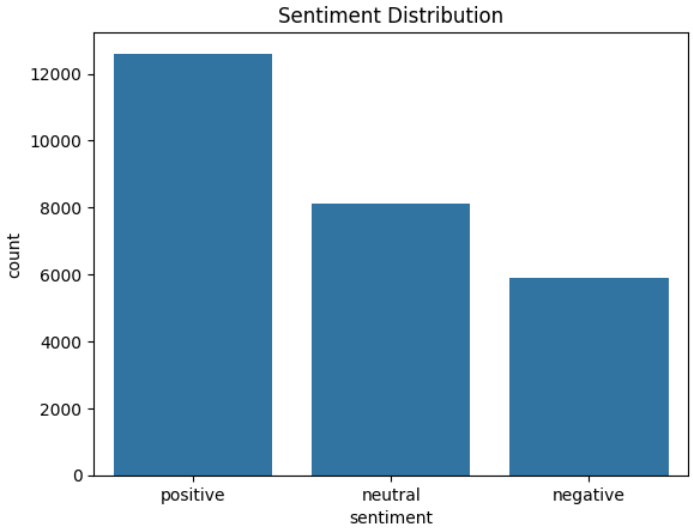


Figure 4.26 Bar Chart for Sentiment Distribution

4.7 Model Training

The labelled data is then train test split for the BERT-based model for training. The model used were Bert-based uncased model and distilled BERT-based model for comparison which model can performed better in sentiment classification of the comments text.

4.8 Initial Results

The expected results for the BERT based model would be as shown in Figure 4.27 and Figure 4.28. the classification reports will shows the accuracy of how well the model can predict the sentiment label.

	precision	recall	f1-score	support
negative	0.92	0.91	0.91	350
neutral	0.85	0.87	0.86	300
positive	0.90	0.89	0.89	350
accuracy			0.89	1000
macro avg	0.89	0.89	0.89	1000
weighted avg	0.89	0.89	0.89	1000

Figure 4.27 Classification Report for Trained Model

normalized_text	predicted_sentiment
Charging stations in KL are still hard to find.	negative
Finally bought a BYD! So quiet and smooth to drive!	positive
Heard mixed reviews about EV batteries lasting long term.	neutral

Figure 4.28 Predicted Sentiment Label of the BERT model

4.9 Conclusion

The implementation of the project flows is presented in detailed in this chapter. The code flow of each implementation stage is shown starting from data collection to results of the model evaluation.