Sentiment Analysis Using Data Mining Techniques

Group 12
Bhanu Siva Kumar Kommana, 811252220, bkomman1@kent.edu
Jaswanth Garikipati, 811227383, jgarikip@kent.edu
K Ravindra Saibaba Veeraraghava Raju Lolabhattu, 811301662, klolabha@kent.edu
Prudhvinath Reddy Katha, 811252707, pkatha@kent.edu
Pavan Kumar Reddy Konduru, 81160758, pkonduru@kent.edu

1. Introduction

Sentiment Analysis also known as opinion mining is a major task in natural language processing (NLP) that focuses on converting normal text into emotions. With the gradual growth in social platforms, understanding sentiment has become increasingly important for various applications such as market research, social media monitoring, and customer feedback analysis. The large amount of user review data is quite useful for the ecommerce websites to know the opinion, attitude, and sentiment towards products, services, and events. Analysing and extracting sentiment from this textual data will be helpful for all the websites, platforms, and products to get an idea of public opinion and to improve their business for better user support.

Sentiment analysis or opinion mining is a subfield of natural language processing (NLP) that analyses, extracts, and defines emotions whether it is positive, negative, and neutral polarities in textual data. sentiment analysis has a wide range of applications across various domains, including but not limited to marketing, customer feedback analysis, brand monitoring, political analysis, and social media analytics. Sentiment analysis is used in so many real time applications to know the customer opinion like., in twitter to understand how people are reacting to a specific feed and to express feeling and their views about crisis and other opinions. Another real-time example of public pulse analysis by using an e-commerce application like amazon. Analysing the product reviews to recommend it to different customers.

2. Project Description

we will be using natural language processing in extracting information from text, and we'll be using this natural language processing to analyse emotions and sentiments of a given text or an essay. If it's a sad essay we can be able to tell whether the emotion present in that essay is sad or not or whatever different emotions are in that text and to analyse different emotions present in an essay or text like sadness happiness jealousy and other emotions and you can also be able to find out the dominant emotion in the text.

Example, if a text is about a happy emotion and you'll be able to tell that the dominant emotion obviously there will be other emotions inside that essay or text, but the dominant emotion will be happiness, so you'll also be able to find out this emotion. you'll be able to plot these emotions on a graph so a particular essay or a big amount of text contains a lot of emotions and plot all these emotions on a graph according to how much these emotions are occurring. To tell whether the whole text is a positive or a negative emotion if a text contains too much of sadness mostly it's a negative emotion and if it contains a lot of good words then it's a happy motion or a positive sentiment.

You can also be able to scrap tweets with a hashtag and find out the public opinion on that hashtag for example you can search for hashtag Donald Trump and find out whether that emotion is associated with a positive or a negative sentiment it doesn't matter like whether it's positive or negative but you'll be able to find out on any given hashtag and also you can even search for hashtag New York and see whether the people are talking about positive things or negative things about New York.

Data Quality: The quality of the data is not up to the mark that source data is not labelled one to overcome these we have used NLTK libraries to get the quality data for the sentimental analysis.

Performance of the model: At first, we have faced some problems with the performance of the data. We have got different outcomes for a different one like when a reaction is happy, we got something like sad reaction for that. For this we have used some keywords as in which we trained the data in such a way that we got the exact reaction of what the word and how the word is being separated.

Workload Distribution: Based on each member expertise we have divided the work equally, Given below is an overview of each work in the project.

- Data Collection and Preprocessing: 2 of us work on collecting different text expressions to train the model. And work on the data correction (preprocessing it)
- Code development: 2 of us will mainly focus on the NLP Emotion algorithm and many other libraries to implement Live Sentiment analysis. And Each of contribute to code development of required in PyCharm.
- Code evaluation: One of us will work on the model execution and see if we there are any bugs to fix and analyse the inputs and outputs.
- Documentation and Presentation: Each member will work to complete the document. One of us will take the lead in organizing everything like document, presentation by summarizing the works we did.

Challenges:

- Data Quality maintenance
- Early-stage performance of the Model
- One main challenge is to understand the context in a document and give accurate sentiment analysis. This is because of limited knowledge on different sentiments, and we are trying to achieve it by giving more attributes like word or contextual embeddings.

3. Background

Opinion mining or sentiment analysis can be defined as the undertaking of recognizing, extricating and characterizing sentiments on something. ML approach. This approach includes training and test collection. there are many approaches Support Vector Machines

(SVM), Naive Bayes (NB), and Maximum Entropy (ME). In the second part semantic orientation approach uses unsupervised learning techniques. AI approach – proposed algorithms - Text mining, SVM (Support Vector Machine), Bayesian algorithm. It is a sort of handling of the regular language (NLP) to follow the public state of mind to a specific regulation, strategy, or showcasing. It includes a way that improvement for the assortment and assessment of remarks and conclusions about regulations, strategies, and so on, which are posted via web-based entertainment.

The current procedures for feeling investigation incorporate AI (supervised and unsupervised), and lexical-based approaches. Twitter is the platform to express feeling and their views about crisis and other opinions. Another real-time example of public pulse analysis by using an e-commerce application like amazon. Analysing the product reviews to recommend it to different customers. The authors proposed three popular ML algorithms for review and rating analysis. Elkobaisi, Mohammed R., Fadi Al Machot, and Heinrich C. Mayr stated that the literature on emotion recognition is largely comprehensive, with an emphasis placed on five key areas: models and language; various other ontologies; datasets and systems. M. Wongkar and A. Angdresey conducted a study on sentiment analysis using Twitter data to analyze public opinions on goods, services, political figures, and celebrities. They collected tweets about 2019 Indonesian presidential candidates and used Python libraries for data collection and preprocessing. They used Naïve Bayes Classification to classify sentiments, achieving an accuracy of approximately 80.1%. The study also compared Naïve Bayes, SVM, and K-Nearest Neighbour methods.

Technologies Used:

- VScode
- PyCharm
- NLP Emotion Algorithm
- NLP K library is basically a very famous library for national language processing.
- PostgreSQL /My SQL (Data storage and data retrieval)
- Matplotlib (Python Libraries), object-oriented programming

4. Problem Definition:

Let f be a function, t be a text that exactly given a dominant emotion e from the given set of documents d.

Sentiment analysis is a computer technique for locating and classifying viewpoints in written material, particularly to ascertain the writer's stance on a certain subject, item, etc. whether it be favourable, unfavourable, or neutral. Sentiment analysis with Python's Natural Language Toolkit (NLTK) presents the following challenges:

- **Data Collection:** Compile a collection of opinion-expressing words or tweets.
- **Text Preprocessing:** To clean and prepare the data, make use of NLTK's text processing features. This covers stop-word elimination, stemming, and tokenization.

- **Feature extraction:** Look for the presence of important words or phrases that are frequently suggestive of sentiment by using NLTK.
- **Vectorization:** Utilized the TF-IDF technique to convert text into a format suitable for the LSTM model, ensuring that the input data retained essential textual features for sentiment analysis.
- **Sentiment Classification:** Use the built-in sentiment analysis features of NLTK, such VADER or custom classifiers, to classify each text or tweet according to its sentiment.
- **Model Evaluation:** By contrasting the model's predictions with a manually labelled test set, you may determine how accurate the sentiment analysis using NLTK was.

The objective is to create an NLP model that can reliably and automatically scan input text essay to identify the dominant sentiment in that essay. We can also plot all the emotions in an essay, or a big amount of text contains a lot of emotions and plot all these emotions on a graph according to how much these emotions are occurring. We can also find the sentiments on hashtags in tweets. This kind of model can be quite useful for figuring out what the general population thinks on social media sites.

5. The Proposed Techniques

Overview: Defining a task that uses NLP techniques to categorize textual data into neutral, positive, and negative sentiments using natural language processing (NLP) techniques. This technology is crucial for businesses and organizations to effectively monitor and address client feedback, enhancing their ability to understand and evaluate emotions in textual communication.

Objective: To implement a model to recognize accurate sentiments from the given input text classification. To preprocess and vectorize textual data for model training. To evaluate the model performance and optimize it for higher accuracy.

Scope: Using NLP techniques, focussing on creating a module to generate a plot of emotions present in the input text by following text processing, feature extraction, sentiment classification, and visualization of sentiment trends.

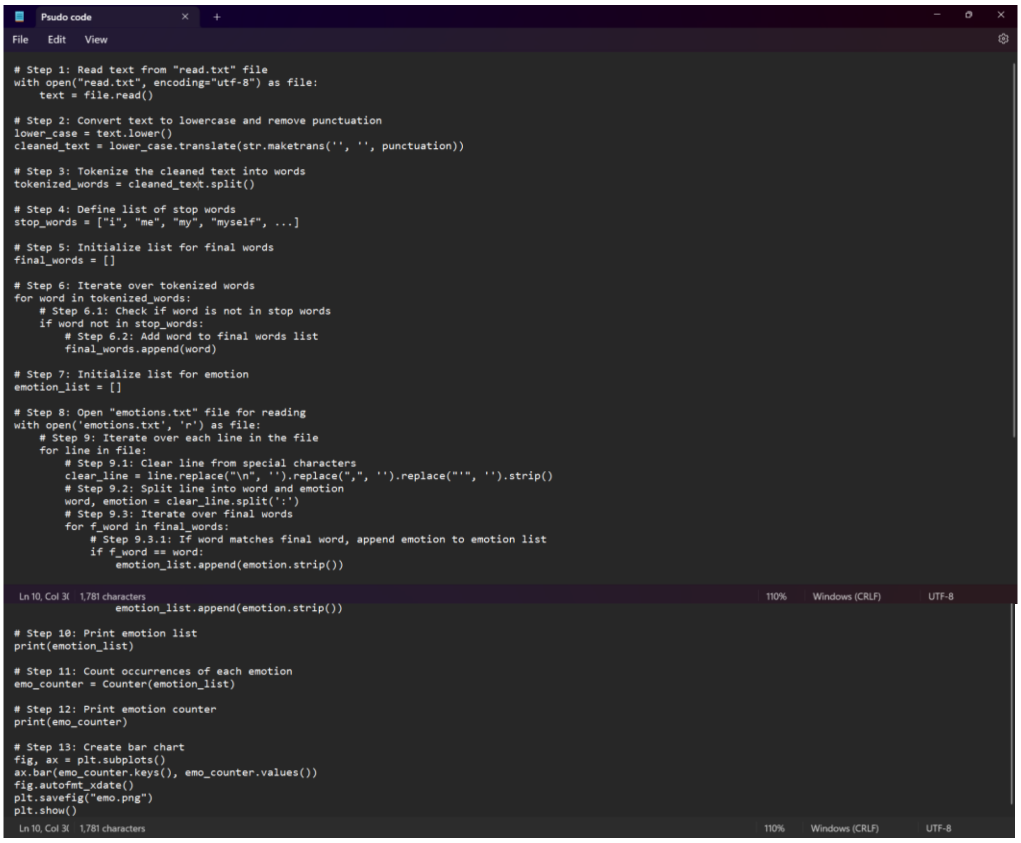
Major techniques

Text Preprocessing: The project used Python libraries such as Pandas and NLTK for cleaning and preparing the data. This involved removing special characters, tokenizing text, and converting sentences into lower case to maintain consistency. In this method, from the read file the text will be pre-processed by first splitting the data into tokens and then the removing the stop words. In the data filtering out the common words that are in the stop word list. Reducing the words into their root form.

Feature extraction: In this method, from the token it will extract the features needed from the given input. Using Bag of words approach, from the tokenized words it will calculate the frequencies of each word and that is used for computer vision. From all the tokenized words based on their frequencies and importance it will weigh the words using term frequency inverse document frequency.

Sentiment Classification: Machine learning and deep learning modules are used for recognizing sentiments from the given input text. Naïve bayes and support vector machines classifiers to predict the sentiment labels.

Vectorization and indexing: Converting the given text into numerical vectors needed to train the machine learning module. Using word embedding techniques to know the semantic meaning of the words. Giving indexes for the tokens to retrieve the tokens and to analyse. Using inverted indexing techniques to words for quick access of tokens and to know the frequencies.



```
# Step 1: Read text from "read.txt" file
with open("read.txt", encoding="utf-8") as file:
    text = file.read()

# Step 2: Convert text to lowercase and remove punctuation
lower_case = text.lower()
cleaned_text = lower_case.translate(str.maketrans('', '', punctuation))

# Step 3: Tokenize the cleaned text into words
tokenized_words = cleaned_text.split()

# Step 4: Define list of stop words
stop_words = ["i", "me", "my", "myself", ...]

# Step 5: Initialize list for final words
final_words = []

# Step 6: Iterate over tokenized words
for word in tokenized_words:
    # Step 6.1: Check if word is not in stop words
    if word not in stop_words:
        # Step 6.2: Add word to final words list
        final_words.append(word)

# Step 7: Initialize list for emotion
emotion_list = []

# Step 8: Open "emotions.txt" file for reading
with open('emotions.txt', 'r') as file:
    # Step 9: Iterate over each line in the file
    for line in file:
        # Step 9.1: Clear line from special characters
        clear_line = line.replace("\n", '').replace(",", '').replace("'", '').strip()
        # Step 9.2: Split line into word and emotion
        word, emotion = clear_line.split(':')
        # Step 9.3: Iterate over final words
        for f_word in final_words:
            # Step 9.3.1: If word matches final word, append emotion to emotion list
            if f_word == word:
                emotion_list.append(emotion.strip())
```

```
                emotion_list.append(emotion.strip())

# Step 10: Print emotion list
print(emotion_list)

# Step 11: Count occurrences of each emotion
emo_counter = Counter(emotion_list)

# Step 12: Print emotion counter
print(emo_counter)

# Step 13: Create bar chart
fig, ax = plt.subplots()
ax.bar(emo_counter.keys(), emo_counter.values())
fig.autofmt_xdate()
plt.savefig("emo.png")
plt.show()
```

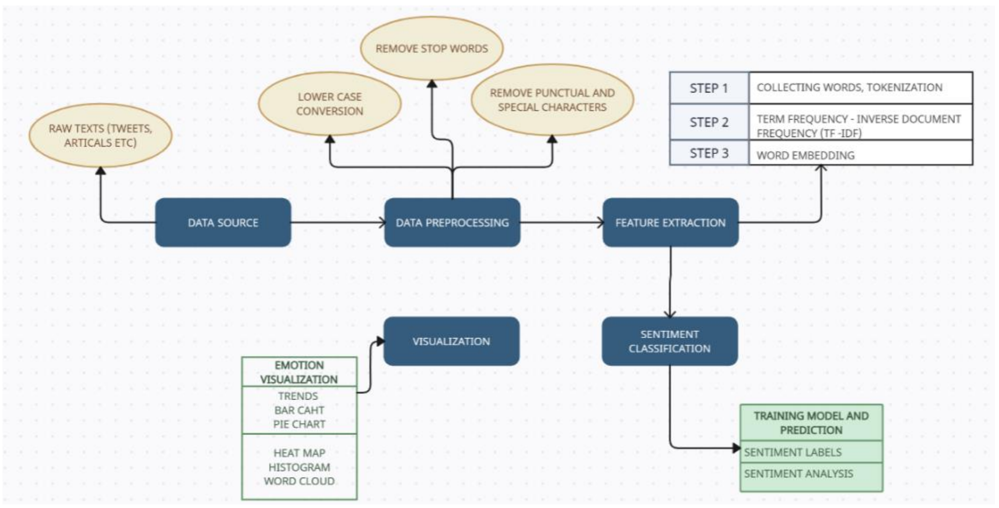
6. System Architecture

System architecture gives a brief step by step idea of what we are trying to do in the sentiment analysis.

1. Data Source & Preprocessing: The data source shall be the starting point of a sentiment analysis process, which consists in raw text information obtained from different sources like Twitter, product reviews, news articles or any content based on textual elements. To extract the sentiment information in an efficient way, these raw text data should be composed of typically unaggregated texts that must undergo processing and analysis. For example, tweets containing opinions or expressions of opinion on specific topics are considered when analysing Twitter data. To clean and prepare raw text data for sentiment analysis, data preparation is an essential step. This process involves several steps, including converting all text to lowercase to ensure consistency, removing punctuation marks and special characters, eliminating common stop words that do not carry significant sentiment information, and stemming words

to their base form for normalization. The text data are standardized and ready for further analysis because of this preprocessing.

2. Feature Extraction: Feature extraction converts pre-processed text input into numerical feature representations suitable for sentiment analysis. One prominent approach is the Bag of terms (BoW), which depicts the text as a collection of distinct terms and their frequency across the page. Another technique is Term Frequency-Inverse text Frequency (TF-IDF), which considers a word's frequency in the text as well as its inverse frequency across all documents. These numerical representations capture key aspects of the text data for sentiment analysis.
3. Sentiment Classification: It is the process of classifying opinion in each text as positive, negative, or neutral emotion. Using machine learning modules like NLP techniques to recognize emotions in given text. Techniques like Naive Bayes, Support Vector Machines (SVM), Logistic Regression, or Neural Networks are used in this module.
4. Visualizations: Using visualisations is essential for understanding and conveying results from sentiment analysis. Data is presented in a graphic format to illustrate sentiment patterns and trends. A bar chart displaying the distribution of different emotions in an analysed text is an example. Throughout the text, the word cloud represents their visual prominence, with larger words representing higher frequencies. Sensational analysts can use these visualizations to better understand and extract insights from the results of their analyses.



7. Experimental Evaluation

Experimental settings: Some of the setting that helped us to get the project without failure are as below.

Source of data: The emotion analysis has taken from the source named read.txt, which is plain text.

Preprocessing: To get the exact analysis of the text we give as input the text has converted into lower case if the text is in upper case or block letters and by removing the punctuation and the stop words that we considered in the code that has been filtered in the text we have given as input in the read.txt file we have.

Descriptions of real/synthetic data sets: As there are no big data sets, we have used the read.txt but the text file has characteristics as it provides real data of emotions and language and genre.

Competitors (baseline method, or existing techniques to compare with): Comparing code with the base line method and techniques we have, we can have a consideration in the following ways.

Frequency based method: We can use the creation of base line by getting the frequency of each word in text and emotion that based on most appearing word we have in the text we provide. This algorithm doesn't involve NLP techniques.

Lexicon Sentiment Analysis of text: To get the sentimental score associated with each word in the text we give that match against a lexicon for getting the score. Some of the lexicon methods are like a FINN or SentiWordNet.

When compared to the existing and comitative technologies NLP gives a vast way of getting the analysis of the words give in text format in other words sentimental analysis of the text.

Parameter settings:

Tokenization: We have used word_tokenize() function from NLTK for text tokenization, and parameter English is passed for language identification and rule applicability.

Lemmatization: WordNetLemmatizer() function we have used for lemmatization in words we give.

Sentiment Analysis: SentimentIntensityAnalyzer() function we have used this for analysis of the words we give from the VADER lexicon.

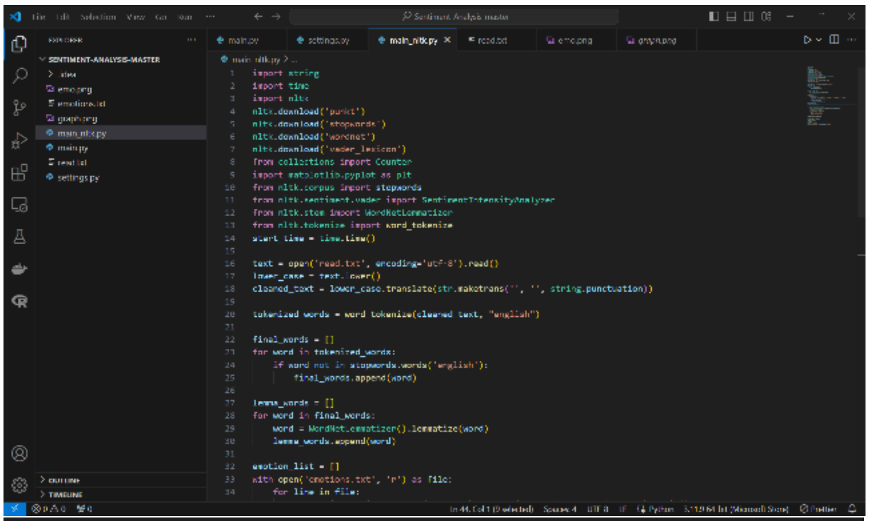
Evaluation Measure:

Accuracy of analysis- By utilizing sentiment analysis, the code categorizes overall text sentiment as positive or negative and assigns it to either side. By comparing sentiment analysis results with ground truth labels, one can determine precise metrics like precision, recall, and F-measure.

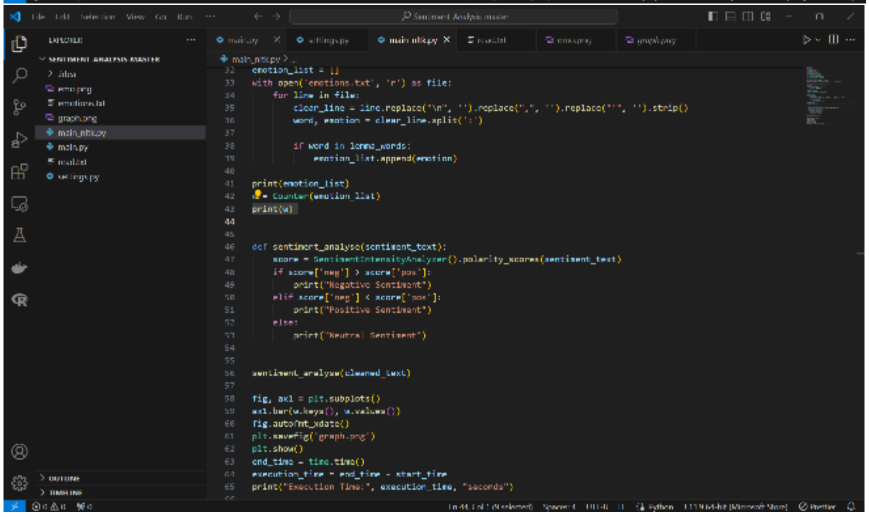
Execution CPU time: The execution CPU time can be got through time.time() function which is much lesser that most of the existing code we have.

I/O Cost: Input and output overhead are determined by the input file's size, as well a combination of the number of read/write operations executed by this code.

Screenshots:

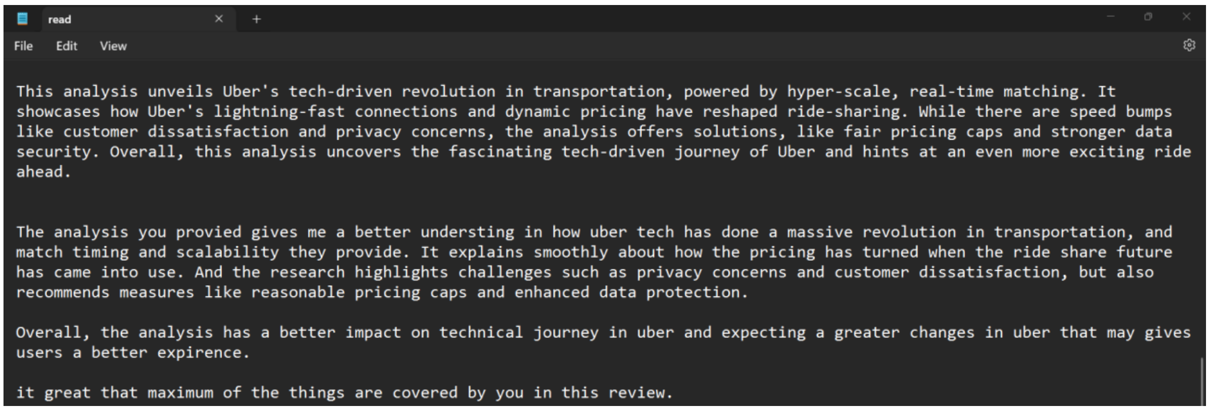


```python
import string
import time
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('vader_lexicon')
from collections import Counter
import matplotlib.pyplot as plt
from nltk.corpus import stopwords
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
start_time = time.time()

text = open('read.txt', encoding='utf-8').read()
lower_case = text.lower()
cleaned_text = lower_case.translate(str.maketrans('', '', string.punctuation))

tokenized_words = word_tokenize(cleaned_text, "english")

final_words = []
for word in tokenized_words:
    if word not in stopwords.words('english'):
        final_words.append(word)

lemma_words = []
for word in final_words:
    word = WordNetLemmatizer().lemmatize(word)
    lemma_words.append(word)

emotion_list = []
with open('emotions.txt', 'r') as file:
    for line in file:
```



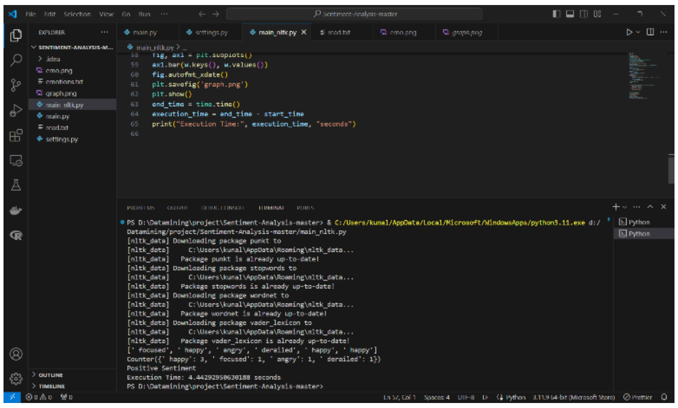
```python
emotion_list = []
with open('emotions.txt', 'r') as file:
    for line in file:
        clear_line = line.replace("\n", '').replace(",", '').replace("'", '').strip()
        word, emotion = clear_line.split(':')

        if word in lemma_words:
            emotion_list.append(emotion)

print(emotion_list)
w = Counter(emotion_list)
print(w)


def sentiment_analyse(sentiment_text):
    score = SentimentIntensityAnalyzer().polarity_scores(sentiment_text)
    if score['neg'] > score['pos']:
        print("Negative Sentiment")
    elif score['neg'] < score['pos']:
        print("Positive Sentiment")
    else:
        print("Neutral Sentiment")


sentiment_analyse(cleaned_text)

fig, ax1 = plt.subplots()
ax1.bar(w.keys(), w.values())
fig.autofmt_xdate()
plt.savefig('graph.png')
plt.show()
end_time = time.time()
execution_time = end_time - start_time
print("Execution Time:", execution_time, "seconds")
```

Input File:



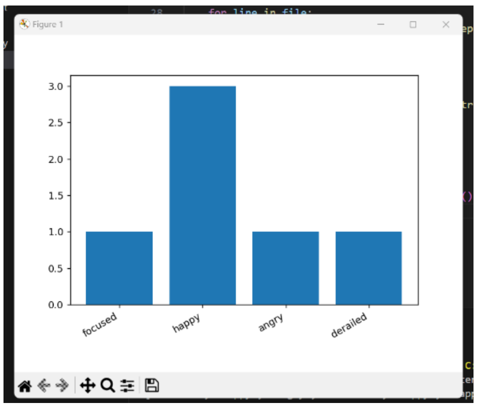
```
This analysis unveils Uber's tech-driven revolution in transportation, powered by hyper-scale, real-time matching. It
showcases how Uber's lightning-fast connections and dynamic pricing have reshaped ride-sharing. While there are speed bumps
like customer dissatisfaction and privacy concerns, the analysis offers solutions, like fair pricing caps and stronger data
security. Overall, this analysis uncovers the fascinating tech-driven journey of Uber and hints at an even more exciting ride
ahead.


The analysis you provied gives me a better understing in how uber tech has done a massive revolution in transportation, and
match timing and scalability they provide. It explains smoothly about how the pricing has turned when the ride share future
has came into use. And the research highlights challenges such as privacy concerns and customer dissatisfaction, but also
recommends measures like reasonable pricing caps and enhanced data protection.

Overall, the analysis has a better impact on technical journey in uber and expecting a greater changes in uber that may gives
users a better exprience.

it great that maximum of the things are covered by you in this review.
```

Output:



```
fig, ax1 = plt.subplots()
ax1.bar(w.keys(), w.values())
fig.autofmt_xdate()
plt.savefig('graph.png')
plt.show()
end_time = time.time()
execution_time = end_time - start_time
print("Execution Time:", execution_time, "seconds")
```

```
PS D:\Datamining\project\Sentiment-Analysis-master> & C:/Users/kunal/AppData/Local/Microsoft/WindowsApps/python3.11.exe d:/
Datamining/project/Sentiment-Analysis-master/main_nltk.py
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\kunal\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\kunal\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\kunal\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\kunal\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
[' focused', ' happy', ' angry', ' derailed', ' happy', ' happy']
Counter({' happy': 3, ' focused': 1, ' angry': 1, ' derailed': 1})
Positive Sentiment
Execution Time: 4.44292950630188 seconds
PS D:\Datamining\project\Sentiment-Analysis-master>
```



Plot that gives information that what type of emotions are given in the paragraph. (X- Frequency at what level does the emotion are in a paragraph, Y- Emotions present in the word text we have given as input)

8. Future Work

We will focus on improving the performance of sentiment analysis models, for example developing models with BERT or GPT architecture, by applying modern machine learning and deep learning techniques. To check data quality issues, we can try to develop automatic error detection and correction methods for all types of data (labelled & unlabelled) in a view to effectively use unsupervised learning and data enhancement techniques. Along with that, we can plan to develop a customized sentiment analysis for the different domains and integrate domain specialized knowledge bases. Real-time sentiment analysis is another area of interest, where we may build scalable systems capable of processing, streaming data in real-time and integrate them with interactive UI for enhanced user experience. Furthermore, we will explore multimodal sentiment analysis, incorporating information from text, images, and audio sources, and develop standardized evaluation metrics and benchmarks to assess model performance and generalization capabilities effectively.

9. References

[1]. *Yousaf et al., "Emotion Recognition by Textual Tweets Classification Using Voting Classifier (LR-SGD)," in IEEE Access, vol. 9, pp. 6286-6295, 2021, doi: 10.1109/ACCESS.2020.3047831. Emotion Recognition by Textual Tweets Classification Using Voting Classifier (LR-SGD)*

[2]. *Saberi, Bilal, and Saidah Saad. "Sentiment analysis or opinion mining: A review." Int. J. Adv. Sci. Eng. Inf. Technol 7.5 (2017): 1660-1666.Microsoft Word - 7. Bilal Saberi - 2137-Final (core.ac.uk)*

[3]. *Wongkar, Meylan, and Apriandy Angdresey. "Sentiment analysis using Naive Bayes Algorithm of the data crawler: Twitter." 2019 Fourth International Conference on Informatics and Computing (ICIC). IEEE, 2019.*

[4]. *Zucco, Chiara, et al. "Sentiment analysis for mining texts and social networks data: Methods and tools." Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 10.1 (2020): e1333.Sentiment analysis for mining texts and social networks data: Methods and tools - Zucco - 2020 - WIREs Data Mining and Knowledge Discovery - Wiley Online Library*

[5]. *Saxena, Akrati, Harita Reddy, and Pratishtha Saxena. "Introduction to sentiment analysis covering basics, tools, evaluation metrics, challenges, and applications." Principles of Social Networking: The New Horizon and Emerging Challenges (2022): 249- 277.Introduction to Sentiment Analysis Covering Basics, Tools, Evaluation Metrics, Challenges, and Applications*

[6]. *Liu, Bing. Sentiment analysis and opinion mining. Springer Nature, 2022.Sentiment Analysis and Opinion Mining - Bing Liu - Google Books*

[7]. *Balaji, Penubaka, D. Haritha, and O. Nagaraju. "An overview on opinion mining techniques and sentiment analysis." International Journal of Pure and Applied Mathematics 118.19 (2018): 61-69.An overview on opinion mining techniques and sentiment analysis.*

[8]. *Păvăloaia, Vasile-Daniel, et al. "Opinion mining on social media data: sentiment analysis of user preferences." Sustainability 11.16 (2019): 4459.Sustainability | Free Full-Text | Opinion Mining on Social Media Data: Sentiment Analysis of User Preferences (mdpi.com)*

[9]. *Piryani, Rajesh, D. Madhavi, and Vivek Kumar Singh. "Analytical mapping of opinion mining and sentiment analysis research during 2000–2015." Information Processing & Management 53.1 (2017): 122- 150.Analytical mapping of opinion mining and sentiment analysis research during 2000–2015 - ScienceDirect*

[10]. *Gómez, Lucía Martín, and María Navarro Cáceres. "Applying data mining for sentiment analysis in music." Trends in Cyber-Physical Multi-Agent Systems. The PAAMS Collection-15th International Conference, PAAMS 2017 15. Springer International Publishing, 2018.Applying Data Mining for Sentiment Analysis in Music | SpringerLink*

[11]. *Shaik, Thanveer, et al. "Sentiment analysis and opinion mining on educational data: A survey." Natural Language Processing Journal 2 (2023): 100003.Sentiment analysis and opinion mining on educational data: A survey - ScienceDirect*

[12]. *Riaz, Sumbal, et al. "Opinion mining on large scale data using sentiment analysis and k-means clustering." Cluster Computing 22 (2019): 7149- 7164.Opinion mining on large scale data using sentiment analysis and k-means clustering | Cluster Computing (springer.com)*