



WOMEN AND NON-CHRISTIAN CHARACTERS IN ELIZABETHAN PLAYS

An NLP based approach for Sentimental Analysis using NER & RE



B.Tech THESIS PROJECT

Bhavya Kumar Garg (20UCC033)

Rashi Gupta (20UCC088)

Twishpeeka Rathore (20UCC110)



Elizabethan Plays

- Elizabethan plays are plays that were written and performed in England during the reign of **Queen Elizabeth I**, which lasted from **1558 to 1603**. This was a period of great cultural and artistic flourishing in England, and many of the greatest plays in the English language were written during this time.
- The most famous Elizabethan playwright was **William Shakespeare**, who wrote many of his plays during this period. Other notable playwrights of the time include *Christopher Marlowe*, *Ben Jonson*, *Thomas Kyd*, and *John Webster*.
- Many Elizabethan plays were written in blank verse, which is unrhymed iambic pentameter. They often explored themes such as *love*, *power*, *revenge*, and *tragedy*, and many were based on historical events or stories from mythology.

Women in Elizabethan Plays

- In terms of female characters, women were often played by male actors in Elizabethan theatre, as women were not allowed to perform on stage at the time. Female characters were typically portrayed as either **virtuous and chaste**, or as **scheming and manipulative**. However, there were also some notable exceptions, such as Shakespeare's "*Rosalind*" in "*As You Like It*," who was a strong and independent female character who disguised herself as a man in order to navigate the male-dominated world of the play.





Non-Christians in Elizabethan Plays

- Non-Christian characters, particularly Jews and Muslims, were also often depicted in Elizabethan plays, although their portrayal was often negative and stereotypical. For example, Shakespeare's "*The Merchant of Venice*" features the character of **Shylock**, a **Jewish** moneylender who is portrayed as **greedy and vengeful**. Similarly, Marlowe's "*The Jew of Malta*" portrays the **Jewish** character **Barabas** as a **cunning and cruel villain**.

In A Nutshell

- Women and non-Christian characters were often depicted in Elizabethan plays, although their roles and portrayals varied depending on the specific play and playwright. Overall, while Elizabethan plays did include female and non-Christian characters, their portrayal was often **limited and stereotypical**, reflecting the cultural and social attitudes of the time.
- While these references to women are relatively minor in the play, they do contribute to the overall atmosphere and themes of the work.

Our Aim

- So far, humans have been responsible for interpreting these characters, relying on their own understanding of the texts. However, there is significant potential for further analysis of the texts to gain valuable insights into how communities were perceived during the Elizabethan Era. Most conclusions drawn have been based on just a few well-known plays written by a small number of authors, which could be problematic given the large amount of literature yet to be explored.
- Due to the limitations of human analysis, we have been motivated to use an NLP-based approach to perform Sentiment Analysis, in order to uncover deeper insights from the plays.

Sentimental Analysis

Sentiment analysis is a natural language processing technique used to determine the emotional tone of a piece of text, such as a tweet, review, or article. The goal of sentiment analysis is to classify the text as positive, negative, or neutral. This is achieved by analysing various features of the text, such as the words used, the context in which they are used, and the overall structure of the text. Sentiment analysis has a wide range of applications, including market research, customer feedback analysis, and social media monitoring.

NLP for Sentimental Analysis

- **Natural Language Processing (NLP)** has become a popular approach for sentiment analysis in recent years. The goal of sentiment analysis is to automatically determine the sentiment expressed in each text, whether it is positive, negative, or neutral. Here are the general steps for building an NLP-based approach for sentiment analysis:
 - *Data collection: Collect data that includes text data with labeled sentiments, such as product reviews or social media posts.*
 - *Text pre-processing: This step involves cleaning and preparing the text data for analysis. It typically includes removing stop words, punctuation, and special characters, as well as tokenizing the text into individual words.*
 - *Feature extraction: Extract meaningful features from the text data. Some common feature extraction methods include Bag of Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF), and Word Embeddings.*

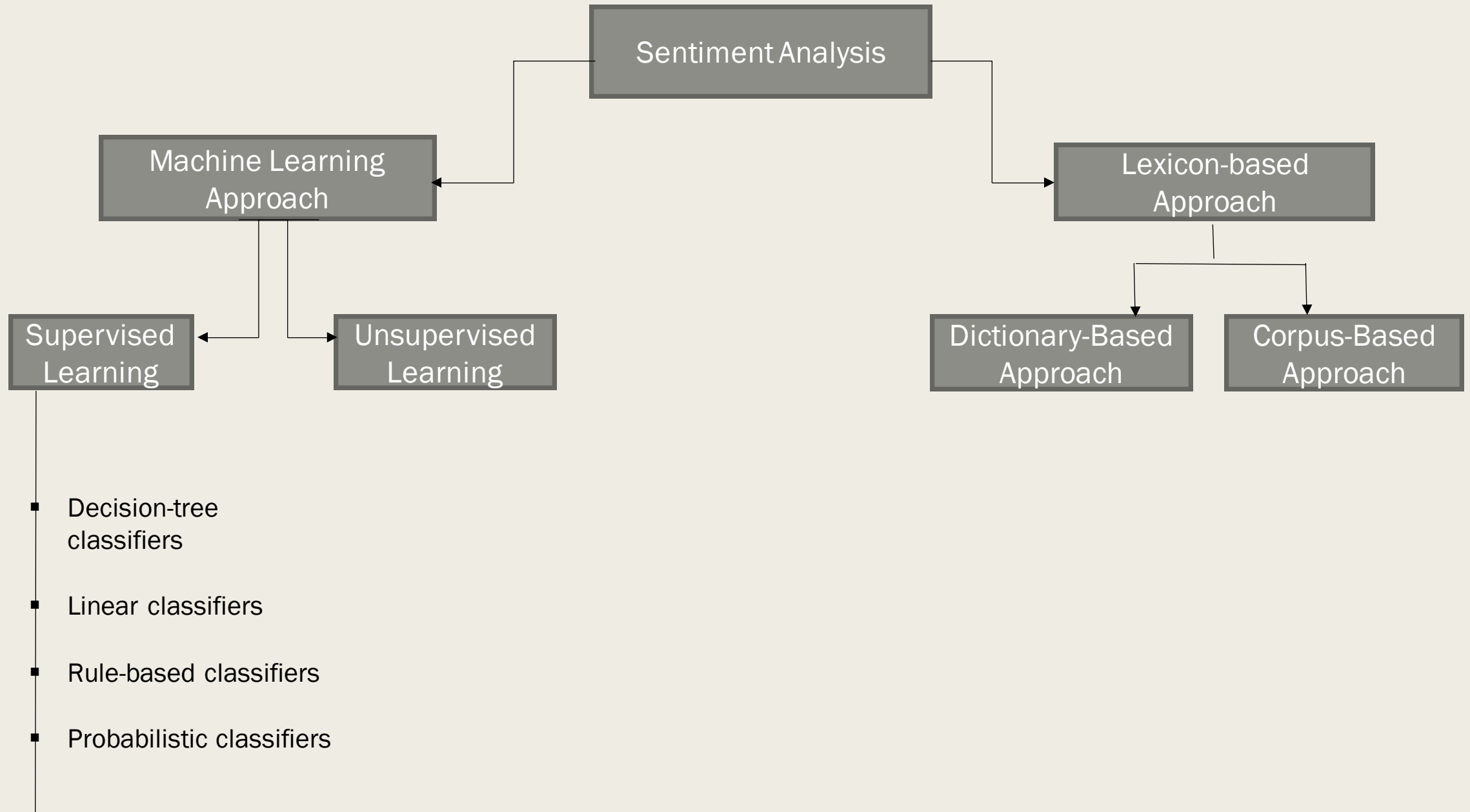
NLP for SA

- *Model selection: Select a suitable machine learning model for sentiment classification. Some common models used for sentiment analysis include Naive Bayes, Support Vector Machines (SVM), and Neural Networks (NNs).*
 - *Training and evaluation: Train the selected model on the labeled data and evaluate its performance on a holdout set of data. Common evaluation metrics for sentiment analysis include accuracy, precision, recall, and F1-score.*
 - *Deployment: Deploy the trained model in a production environment to perform sentiment analysis on new text data.*
- Overall, NLP-based approaches for sentiment analysis can be effective in accurately identifying the sentiment expressed in text data and can be applied to a variety of use cases, including customer feedback analysis, brand monitoring, and social media sentiment analysis.

Limitations of NLP

- While NLP (Natural Language Processing) has made significant advancements in recent years, there are still some limitations to its approach, including:
 - *Ambiguity: Natural language is often ambiguous, and words or phrases can have multiple meanings depending on the context in which they are used. This can make it difficult for NLP models to accurately interpret and understand the meaning of a text.*
 - *Contextual understanding: Understanding the meaning of a text requires contextual understanding, which can be challenging for NLP models. NLP models may not be able to capture the nuances of human language and may struggle with idiomatic expressions, humour, and sarcasm.*
 - *Data availability and quality: NLP models rely on large amounts of high-quality data to be trained effectively. However, obtaining and labelling large amounts of data can be time-consuming and costly.*

- *Bias: NLP models can be biased, which can lead to inaccurate or unfair results. This bias can be introduced through the training data, the algorithms used, or the design of the model itself.*
- *Multilingual support: While NLP has made significant strides in supporting multiple languages there are still many languages for which NLP tools are not available or not as accurate.*
- *Privacy concerns: The use of NLP for text analysis raises concerns around privacy, particularly when it comes to sensitive personal information. The misuse of NLP tools could lead to violations of privacy and other ethical concerns.*



Literature survey

■ Sentiment analysis algorithms and applications: A survey

by Walaa Medhat, Ahmed Hassan and Hoda Korashy, DEC 2014

<https://www.sciencedirect.com/science/article/pii/S2090447914000550>

This research paper tackles a comprehensive overview of the last update in this field. Many recently proposed algorithms' enhancements and various SA applications are investigated and presented briefly in this survey. These articles are categorized according to their contributions in the various SA techniques. The related fields to SA. The main contributions of this paper include the sophisticated categorizations of a large number of recent articles and the illustration of the recent trend of research in the sentiment analysis and its related areas.

Literature survey

■ Sentiment Analysis Using VADER

by Juveriya Mahreen OCT, 2022

[https://www.analyticsvidhya.com/blog/2022/10/sentiment-analysis-using-vader/#:~:text=VADER\(%20Valence%20Aware%20Dictionary%20for,as%20either%20positive%20or%20negative](https://www.analyticsvidhya.com/blog/2022/10/sentiment-analysis-using-vader/#:~:text=VADER(%20Valence%20Aware%20Dictionary%20for,as%20either%20positive%20or%20negative)

The article explains how to use VADER (Valence Aware Dictionary for Sentiment Reasoning) for sentiment analysis. VADER is a rule-based sentiment analysis tool that is specifically designed to analyze social media text, which often includes slang, abbreviations, and other informal language. Overall, the article provides a useful introduction to VADER and demonstrates how it can be used for sentiment analysis. It also highlights the importance of choosing the right sentiment analysis tool based on the specific application and domain, and the need for ongoing evaluation and refinement of sentiment analysis models.

Literature survey

■ **Text2emotion: Python package to detect emotions from textual data**

by Amey Band SEPT 2020

<https://towardsdatascience.com/text2emotion-python-package-to-detect-emotions-from-textual-data-b2e7b7ce1153>

This was published in 'Towards Data Science' helps us discover the emotions in textual messages. Text2Emotion is a Python package that uses machine learning algorithms to detect emotions from textual data. It can be used to analyze the emotional tone of social media posts, customer feedback, or any other form of text data.

The package uses a deep learning model trained on a large corpus of text data to classify text into seven emotion categories: anger, fear, joy, love, sadness, surprise, and neutral. It uses a combination of text preprocessing, feature extraction, and classification techniques to accurately identify emotions in text.

Text2Emotion can be used in a variety of applications, such as sentiment analysis, customer feedback analysis, and social media monitoring. It is easy to use and can be integrated with other Python packages and libraries.

Literature survey

■ Relation Extraction | Natural Language Processing

by Chandresh Maurya MAR, 2021

<https://www.youtube.com/watch?v=t0R3NdT1vGY>

Through this talk we learn about the concepts of Relation Extraction. Relation Extraction is a process of identifying and extracting the connections between different things or entities mentioned in a text. This task involves analyzing the text and identifying the entities involved, such as people, places, or things, and then determining how they are related to each other. The goal of relation extraction is to extract these structured relationships from unstructured text data, which can be valuable for a wide range of applications.

One of the key challenges associated with relation extraction is determining the correct relationships between entities. This can be especially difficult when dealing with complex or ambiguous text. Additionally, different types of relations may require different extraction techniques, and the quality of the extracted relations can be affected by factors such as the quality of the input data and the specific extraction algorithm used.

Lexicon-based Approach

- The lexicon-based approach to sentiment analysis uses pre-defined dictionaries or lexicons with assigned sentiment scores.
- Lexicons can be manually created or generated using machine learning techniques.
- The approach matches words in the text with the words in the lexicon and aggregates sentiment scores to determine overall sentiment.
- This approach is simple and computationally efficient but may struggle with sarcasm, irony, or words with multiple meanings.
- It requires a comprehensive and accurate lexicon to produce reliable results.
- The lexicon-based approach is often used as a baseline for sentiment analysis models.
- It can be combined with other techniques to improve accuracy.

The Tempest

- "The Tempest" is a play written by William Shakespeare, believed to have been written around 1610–1611. The play is set on a remote island, where the exiled Duke of Milan, Prospero, uses his magic powers to manipulate the events that take place on the island.
- The play is notable for its exploration of themes such as the abuse of power, the nature of forgiveness, and the relationship between art and illusion. It also features a diverse range of characters, including the sprite Ariel, the monstrous Caliban, and the shipwrecked nobles Alonso, Antonio, and Ferdinand.



SA on “The Tempest”

- Approach used: Lexicon Based
- Libraries used:
 - *NLTK*
 - *VADER*
 - *Text2Emotion*
 - *Source used for eBook: <https://www.gutenberg.org/files/23042/23042-0.txt>*

NLTK

1. Natural Language Toolkit
2. Python library that provides tools and resources for working with human language data.
3. In this case, used to tokenize the text into sentences.

VADER

1. Valence Aware Dictionary and sentiment Reasoner
2. Dictionary-Based Approach
3. Contains over 7,500 lexical features, including words, phrases, and emoticons, each with an associated positive, negative, and neutral sentiment score.
4. Also considers other factors such as punctuation, capitalization, and degree modifiers (such as "very" or "extremely").

Text2emotion

1. Text2emotion is the python package developed with the clear intension to find the appropriate emotions embedded in the text data.
2. Processes any textual data, recognizes the emotion embedded in it, and provides the output in the form of a dictionary.
3. Well suited with 5 basic emotion categories such as Happy, Angry, Sad, Surprise and Fear.

```
✓ 18s !pip install text2emotion
!pip uninstall emoji
!pip install emoji==0.6.0

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting text2emotion
  Downloading text2emotion-0.0.5-py3-none-any.whl (57 kB)
    57.0/57.8 kB 2.2 MB/s eta 0:00:00
Requirement already satisfied: nltk in /usr/local/lib/python3.9/dist-packages (from text2emotion) (3.8.1)
Collecting emoji>=0.6.0
  Downloading emoji-2.2.0.tar.gz (240 kB)
    240.9/240.9 kB 8.5 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: click in /usr/local/lib/python3.9/dist-packages (from nltk->text2emotion) (8.1.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.9/dist-packages (from nltk->text2emotion) (4.65.0)
Requirement already satisfied: joblib in /usr/local/lib/python3.9/dist-packages (from nltk->text2emotion) (1.2.0)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.9/dist-packages (from nltk->text2emotion) (2022.10.31)
Building wheels for collected packages: emoji
  Building wheel for emoji (setup.py) ... done
  Created wheel for emoji: filename=emoji-2.2.0-py3-none-any.whl size=234926 sha256=cfb1d321d3f90a46677fd607d024375ac56e2ffec7db0f54663e590af29e38e
  Stored in directory: /root/.cache/pip/wheels/9a/b8/0f/f580817231cbf59f6ade9fd132ff60ada1de9f7dc85521f857
Successfully built emoji
Installing collected packages: emoji, text2emotion
Successfully installed emoji-2.2.0 text2emotion-0.0.5
Found existing installation: emoji 2.2.0
Uninstalling emoji-2.2.0:
  Would remove:
    /usr/local/lib/python3.9/dist-packages/emoji-2.2.0.dist-info/*
    /usr/local/lib/python3.9/dist-packages/emoji/*
Proceed (Y/n)? y
  Successfully uninstalled emoji-2.2.0
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting emoji==0.6.0
  Downloading emoji-0.6.0.tar.gz (51 kB)
    51.0/51.0 kB 3.1 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: emoji
  Building wheel for emoji (setup.py) ... done
  Created wheel for emoji: filename=emoji-0.6.0-py3-none-any.whl size=49732 sha256=69ea70cdf638213e25e58b296fc39d67741273ebec0dfaf244f94dbf7fbd04a1
  Stored in directory: /root/.cache/pip/wheels/70/2a/7f/1a0012c86b1061c6ee2ed9568b1f830f857a51e8e416452af2
Successfully built emoji
Installing collected packages: emoji
Successfully installed emoji-0.6.0
```

Installing necessary libraries.


```
# First we import all the necessary libraries
# nltk is used for tokenizing, while vader is being used for performing SA
import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('vader_lexicon')
from nltk.sentiment import SentimentIntensityAnalyzer

# Here we load our book "The Tempest"
with open('the_tempest.txt', 'r', encoding='utf8') as f:
    text = f.read()

# Initializing the sentiment analyzer
sia = SentimentIntensityAnalyzer()

# Tokenizing the text into sentences
sentences = nltk.sent_tokenize(text)

# Finding the sentiment of each sentence
results = []
for sentence in sentences:
    score = sia.polarity_scores(sentence)
    results.append(score)

# We print the overall sentiment scores of all the sentences
compound_scores = [score['compound'] for score in results]
overall_score = sum(compound_scores) / len(compound_scores)
print(f"Overall Sentiment Score: {overall_score:.2f}")

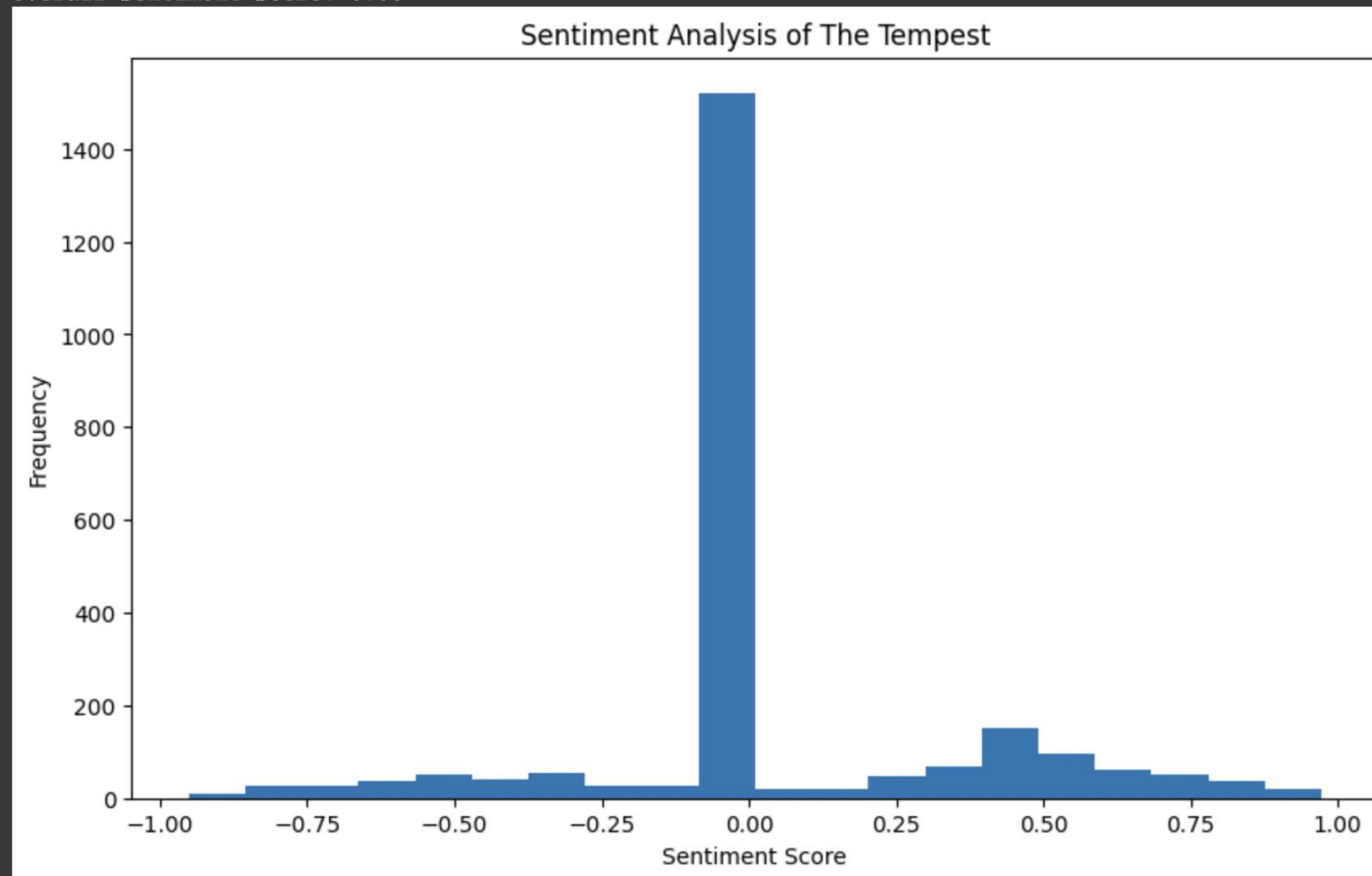
# Plotting the sentiment scores on a graph
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(10, 6))
ax.hist(compound_scores, bins=20)
ax.set_title('Sentiment Analysis of The Tempest')
ax.set_xlabel('Sentiment Score')
ax.set_ylabel('Frequency')
plt.show()
```

Python code to
perform SA on
the complete
book.

Results

```
[nltk_data] Downloading package punkt to /root/nltk_data...  
[nltk_data]   Unzipping tokenizers/punkt.zip.  
[nltk_data] Downloading package averaged_perceptron_tagger to  
[nltk_data]   /root/nltk_data...  
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.  
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...  
[nltk_data]   Package vader_lexicon is already up-to-date!  
Overall Sentiment Score: 0.06
```



```
import requests
from bs4 import BeautifulSoup
import nltk
nltk.download('punkt')
nltk.download('vader_lexicon')
from nltk.sentiment.vader import
SentimentIntensityAnalyzer
import text2emotion as te

# Download the text of "The Tempest"
url = "https://www.gutenberg.org/files/23042/23042-0.txt"
response = requests.get(url)
text = response.content.decode('utf-8')

# Use NLTK to tokenize the text into sentences
sentences = nltk.sent_tokenize(text)

# Filter out sentences spoken by the character

character_sentences = [s for s in sentences if '_Pros._'
in s]

# Character name
character_name = "PROSPERO"
```

Python code to
perform SA on
PROSPERO

```
# Initialize emotion counts
emotions = {
    "Happy": 0,
    "Sad": 0,
    "Angry": 0,
    "Surprise": 0,
    "Fear": 0
}
# Iterate through the sentences and count emotions for
the character
for sentence in sentences:
    if character_name.lower() in sentence.lower():
        sentence_emotions = te.get_emotion(sentence)
        for emotion, value in sentence_emotions.items():
            emotions[emotion] += value

# Print the emotion counts for the character
print("\nEmotions for", character_name, ":",
      emotions, "\n")

# Initialize the SentimentIntensityAnalyzer
sia = SentimentIntensityAnalyzer()
```

Python code to
perform SA on
PROSPERO

```
# Analyze sentiment of each sentence spoken by the character
results = []
for sentence in character_sentences:
    score = sia.polarity_scores(sentence)
    results.append(score)

# Print the compound sentiment score for each sentence spoken
by the character
print("Sentiment analysis for the character:")
for i, score in enumerate(results):
    print(f"Sentence {i+1}: {score['compound']:.2f}")

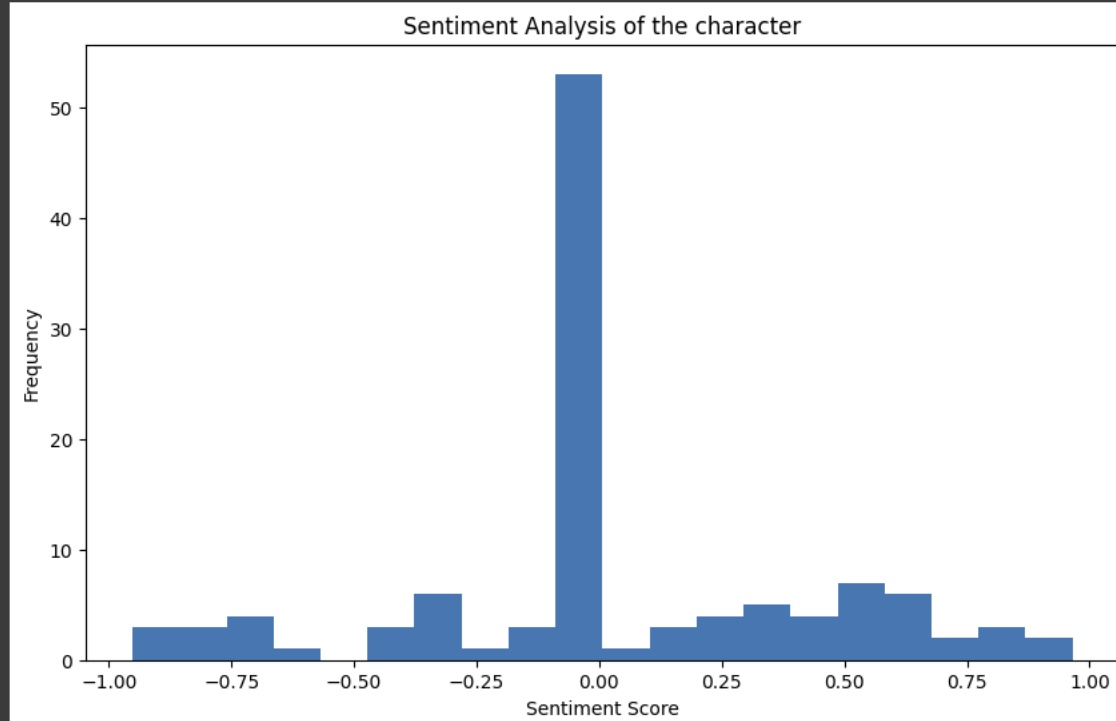
# We print the overall sentiment scores of all the sentences
compound_scores = [score['compound'] for score in results]
overall_score = sum(compound_scores) / len(compound_scores)
print(f"Overall Sentiment Score: {overall_score:.2f}")

# Visualize the sentiment scores
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(10, 6))
ax.hist(compound_scores, bins=20)
ax.set_title('Sentiment Analysis of the character')
ax.set_xlabel('Sentiment Score')
ax.set_ylabel('Frequency')
plt.show()
```

Python code to
perform SA on
PROSPERO


```
Sentence 111: 0.00  
Sentence 112: 0.00  
Sentence 113: -0.71  
Sentence 114: 0.56  
Overall Sentiment Score: 0.05
```



Results on PROSPERO

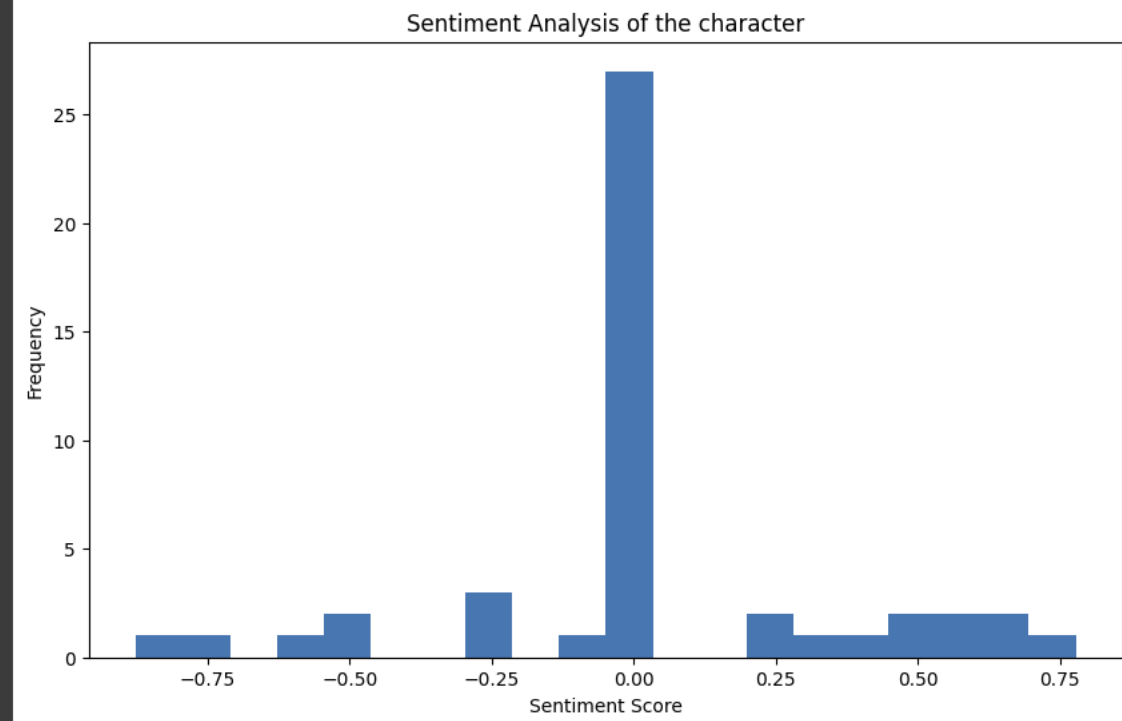
```
[nltk_data] Downloading package punkt to /root/nltk_data...  
[nltk_data] Package punkt is already up-to-date!  
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...  
[nltk_data] Package vader_lexicon is already up-to-date!
```

```
Emotions for PROSPERO : {'Happy': 2.7900000000000005, 'Sad': 6.000000000000001, 'Angry': 2.3900000000000006, 'Surprise': 6.3500000000000005, 'Fear': 9.47}
```

```
Sentiment analysis for the character:
```

```
Sentence 1: -0.65  
Sentence 2: -0.69  
Sentence 3: 0.00  
Sentence 4: -0.42  
Sentence 5: 0.00  
Sentence 6: 0.00
```

```
Sentence 45: 0.00  
Sentence 46: 0.00  
Sentence 47: 0.27  
Overall Sentiment Score: 0.03
```



Results on ARIEL

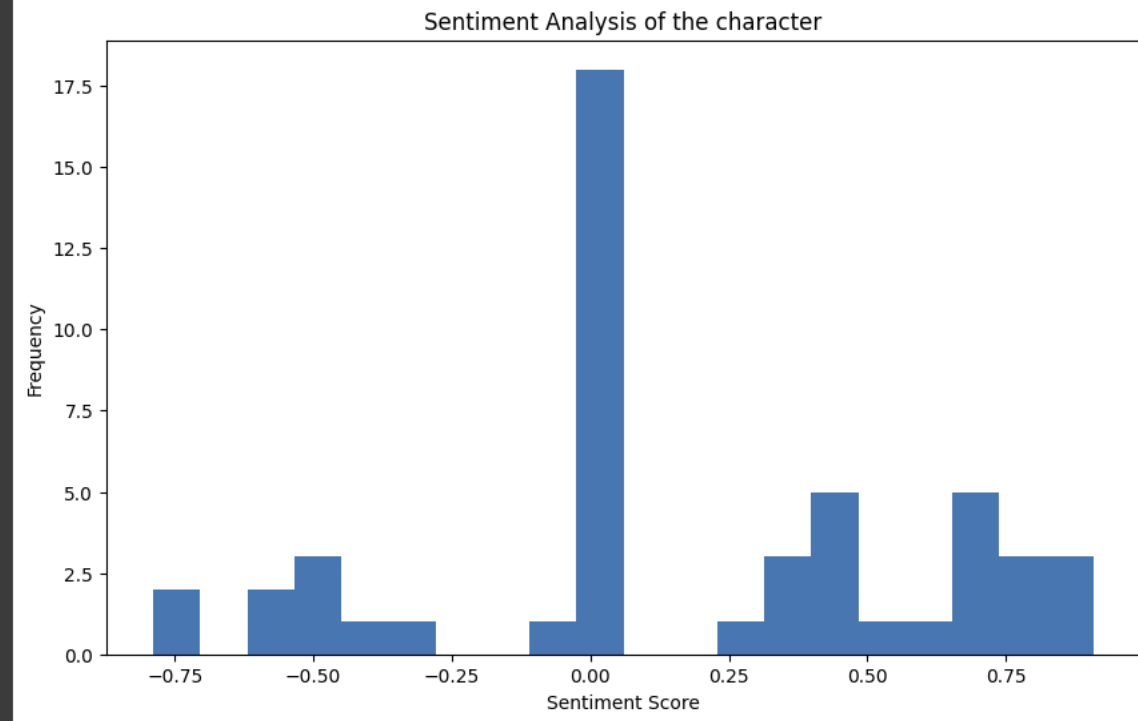
```
[nltk_data] Downloading package punkt to /root/nltk_data...  
[nltk_data] Package punkt is already up-to-date!  
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...  
[nltk_data] Package vader_lexicon is already up-to-date!
```

```
Emotions for ARIEL : {'Happy': 8.370000000000001, 'Sad': 7.489999999999999, 'Angry': 0.0, 'Surprise': 5.52, 'Fear': 11.6}
```

```
Sentiment analysis for the character:
```

```
Sentence 1: 0.69  
Sentence 2: 0.00  
Sentence 3: -0.88  
Sentence 4: 0.00  
Sentence 5: 0.00  
Sentence 6: 0.78
```


Sentence 48: 0.74
Sentence 49: 0.75
Sentence 50: 0.00
Overall Sentiment Score: 0.16



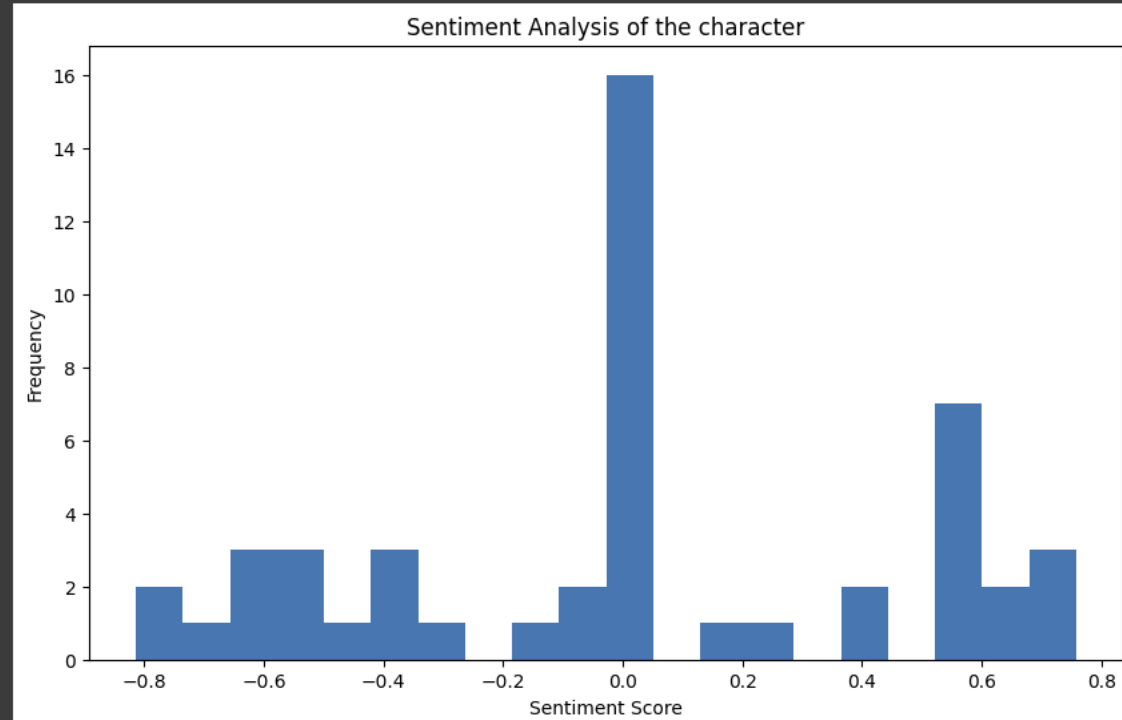
Results on MIRANDA

Emotions for MIRANDA : {'Happy': 0.2, 'Sad': 1.4700000000000002, 'Angry': 2.5300000000000002, 'Surprise': 1.03, 'Fear': 1.77}

Sentiment analysis for the character:

Sentence 1: 0.56
Sentence 2: -0.48
Sentence 3: 0.00
Sentence 4: -0.42
Sentence 5: 0.34
Sentence 6: 0.71
Sentence 7: 0.00
Sentence 8: 0.00
Sentence 9: 0.46
Sentence 10: 0.00
Sentence 11: 0.00

```
Sentence 46: -0.33  
Sentence 47: 0.57  
Sentence 48: -0.60  
Sentence 49: 0.71  
Overall Sentiment Score: 0.02
```



Results on CALIBAN

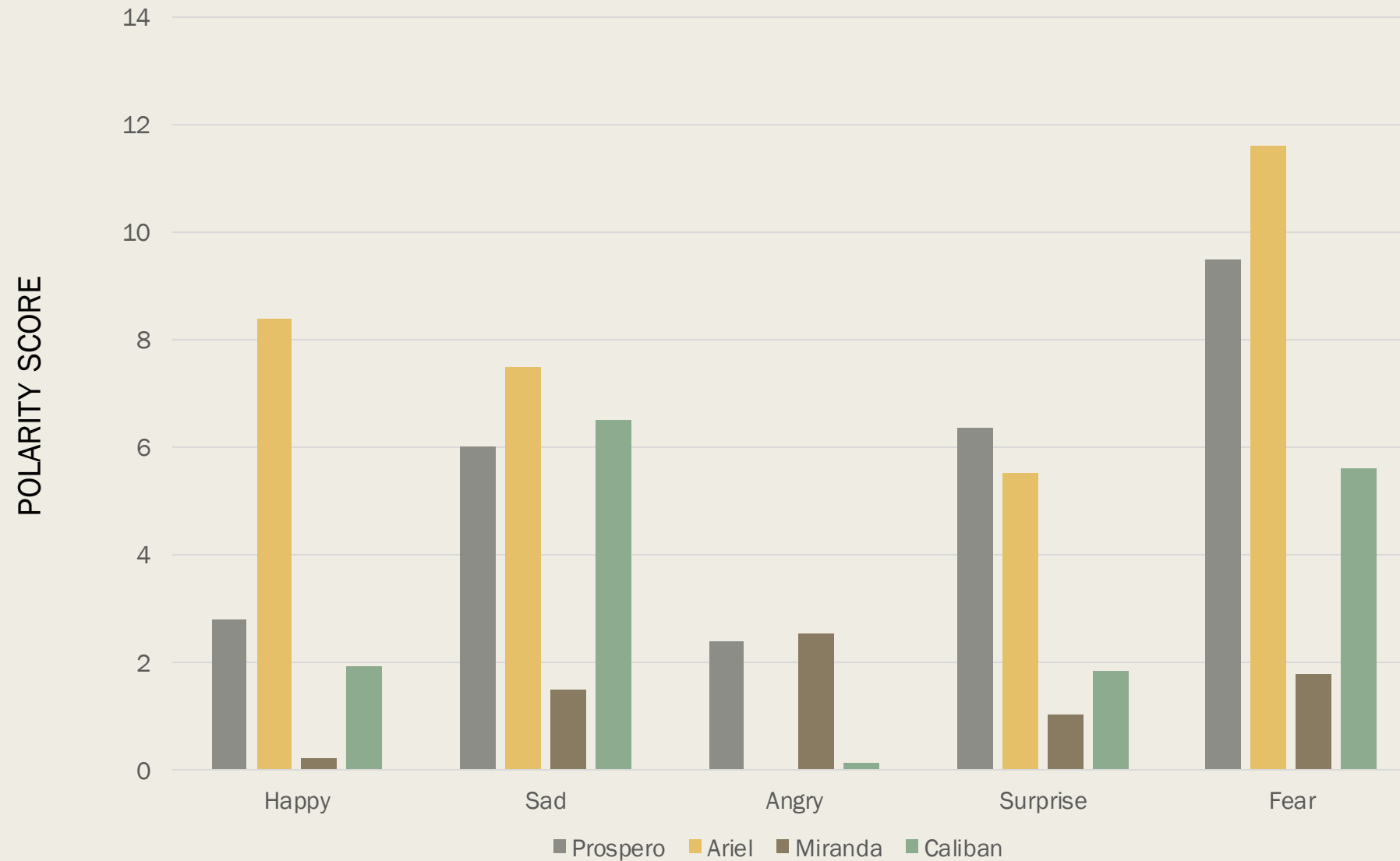
```
[nltk_data] Downloading package punkt to /root/nltk_data...  
[nltk_data] Package punkt is already up-to-date!  
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...  
[nltk_data] Package vader_lexicon is already up-to-date!
```

```
Emotions for Caliban : {'Happy': 1.9100000000000001, 'Sad': 6.5, 'Angry': 0.12, 'Surprise': 1.83, 'Fear': 5.61}
```

```
Sentiment analysis for the character:
```

```
Sentence 1: 0.00  
Sentence 2: -0.70  
Sentence 3: 0.00  
Sentence 4: 0.00  
Sentence 5: -0.15  
Sentence 6: 0.03
```

Conclusion – ED Results



Machine Learning Approach

- The machine learning approach for sentiment analysis involves training a model to predict sentiment based on labelled examples.
- Relevant features are selected from the text, such as the frequency of certain words or phrases.
- These features are used to train a machine learning algorithm, such as a decision tree, Naive Bayes, or neural network.
- The trained model can then predict the sentiment of new, unlabelled text.
- This approach can handle more complex language usage, such as sarcasm and irony, and can adapt to new contexts or domains.
- However, it requires a large amount of labelled data for training and can be computationally intensive.
- The machine learning approach is widely used in sentiment analysis applications, such as social media monitoring, customer feedback analysis, and brand reputation management.

Named-Entity-Recognition (NER)

- Named entity recognition (NER) is a natural language processing (NLP) technique that involves identifying and categorizing named entities in text. Named entities are specific objects, people, places, organizations, and other entities that have a unique name or identifier. NER is used to automatically identify and extract these entities from unstructured text data.
- There are several tools and libraries available for named entity recognition (NER) that can be used for natural language processing tasks. Some of the commonly used tools for NER include SpaCy and NLTK.

Relation-Extraction (RE)

- Relation extraction is a natural language processing (NLP) task that involves identifying and extracting semantic relationships between entities in text. The goal of relation extraction is to automatically identify and classify the relationships between entities, such as people, organizations, and locations, that are mentioned in the text.
- There are several techniques used for relation extraction, including rule-based systems, machine learning models, and deep learning models. Rule-based systems use handcrafted rules to identify and classify relationships, while machine learning models use algorithms to learn patterns in the data and automatically identify relationships. Deep learning models, such as neural networks, have been shown to be effective in relation extraction tasks, as they can learn complex patterns in the text data.

Plan of Action

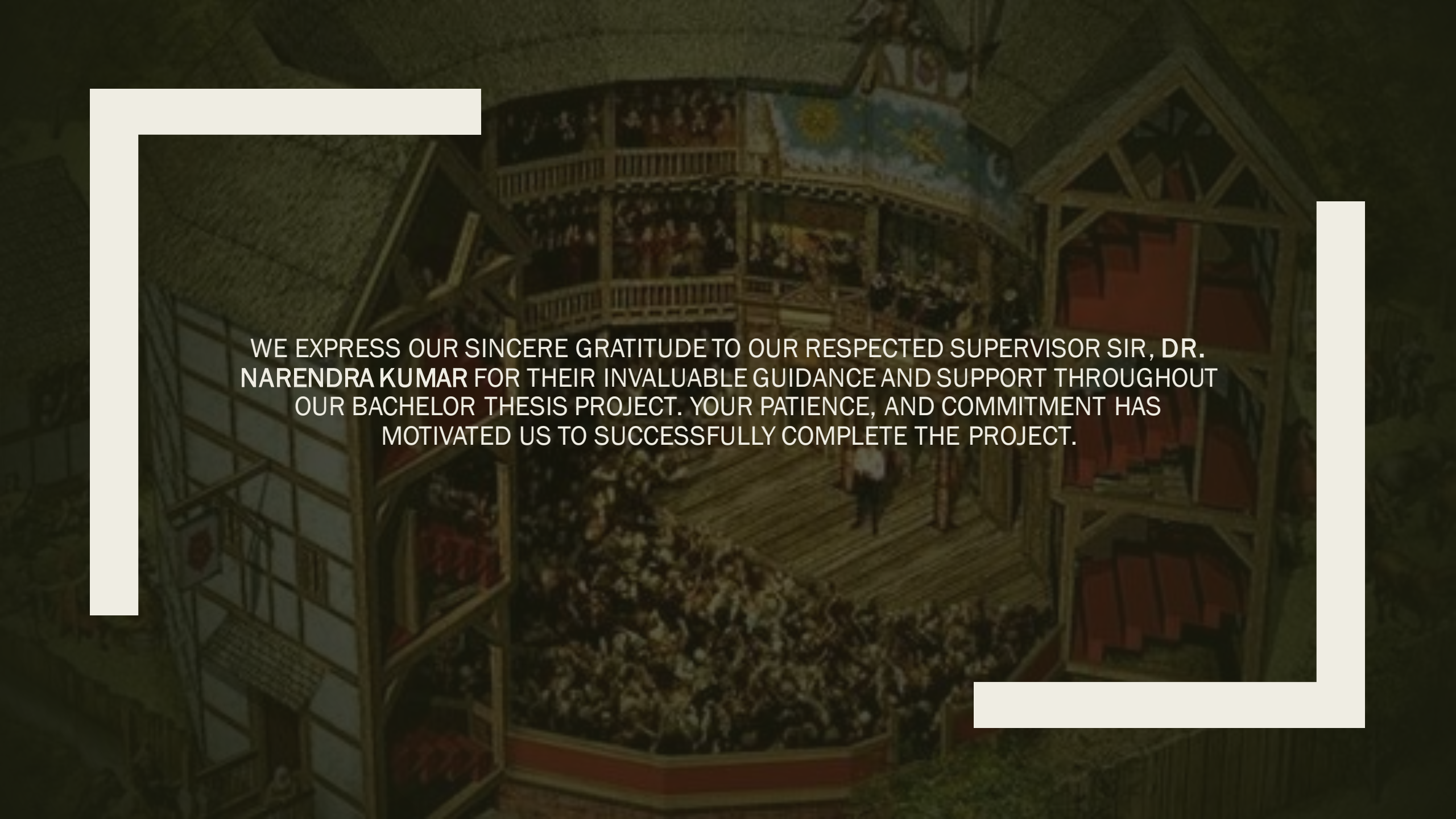
1. Identifying and classifying named entities.
2. Extracting Relations between the entities.
 - I. *Creating our own KB*
 - II. *Using our KB, find all the relations in the corpus*
3. Perform sentimental analysis.

Problems in RE

- Relation extraction is complicated and tedious to work with. There are some problems with it
 - *We need an existing knowledge base to start with , we cannot start it from scratch.*
 - *Multiple relations exists.*
 - *There is a dependency on the previous text.*

References

- 1) NLTK documentation : <https://www.nltk.org>
- 2) Vader (Geeksforgeeks) : <https://www.geeksforgeeks.org/python-sentiment-analysis-using-vader/>
- 3) Text2Emotion (Python documentation) : <https://pypi.org/project/text2emotion/>
- 4) NER (Github) : https://github.com/raaga500/YTshared/blob/master/V3_NamedEntityReco_3.ipynb
- 5) RE: http://nlpprogress.com/english/relationship_extraction.html



WE EXPRESS OUR SINCERE GRATITUDE TO OUR RESPECTED SUPERVISOR SIR, DR. NARENDRA KUMAR FOR THEIR INVALUABLE GUIDANCE AND SUPPORT THROUGHOUT OUR BACHELOR THESIS PROJECT. YOUR PATIENCE, AND COMMITMENT HAS MOTIVATED US TO SUCCESSFULLY COMPLETE THE PROJECT.

Plan of Action - Part II

1. Topic summary
 1. What and why we are doing
 2. Role of SA in our project
2. Recap of previous work
3. Motivation for further research – Improving accuracy
4. Area's to improve
 1. Dependency of emotions on multiple entities
 2. Language barrier
5. RE – Solving dependency issue
6. Limitations of RE – Beyond Scope
7. Multiple translations – Solving language issue



Recap

