Full Name: Mirsad Hadžić
Email: mirsad.hadzic@stu.ibu.edu.ba
Department: Data Science

FIFA Twitter Data Analysis using NLP

1. Abstract

The purpose of this research is to perform sentiment analysis on Twitter data using Natural Language Processing (NLP) techniques, particularly leveraging the NLTK library in Python within a Jupyter notebook environment. The study aims to explore sentiment classification methods, evaluating the emotional tone of tweets and categorizing them as neutral, positive, or negative sentiments, utilizing NLTK's SentimentIntensityAnalyzer. The sample consists of Twitter data with columns like 'Tweet' and 'Sentiment' sourced from a CSV file. The methodology involves tokenizing and processing the text, grading sentiment, counting occurrences of the hashtag #fifa, and analyzing word frequencies. (Dataset. (2022)). (NLTK. (2025).)

In addition to the lexicon-based VADER approach, the study incorporates a transformer-based deep learning model—BERT (Bidirectional Encoder Representations from Transformers)—to enhance sentiment classification accuracy. BERT, pre-trained on large corpora and capable of understanding context and nuanced language, offers a state-of-the-art alternative to traditional models. This inclusion allows a comparative analysis between rule-based and deep learning approaches, highlighting BERT's effectiveness in handling complex tweet structures. (Devlin, J., et al. (2018).) (Medium. (2021).)

Furthermore, the study investigates the impact of removing stopwords and explores the list of eliminated stopwords. The expected results include gaining insights into prevalent sentiments on Twitter regarding a specified topic, frequency of the hashtag #fifa, and a comprehensive understanding of word usage, visually depicted through wordclouds. Possible limitations include inherent subjectivity in sentiment analysis, potential variations in language use, reliance on hashtag frequency as an indicator of topic prevalence, and the effectiveness of stopwords removal, which may be context-dependent. The addition of wordcloud analysis enhances the visual representation of the most frequent words, providing a holistic perspective on the dataset. (Dataset. (2022).) (VADER. (2024).)

2. Introduction

Social media has transformed how people express opinions, with Twitter serving as a dynamic platform for real-time commentary. This research explores tweets containing the hashtag #fifa, applying Natural Language Processing (NLP) techniques in Python—primarily through the NLTK library—to analyze sentiment, word usage patterns, and public engagement. The core objectives are to classify tweet sentiments, visualize dominant keywords using word clouds, and examine word variations using Lancaster and Porter stemmers. (Dataset. (2022).) (VADER. (2024).) (Liu, B. (2012).)

In addition to traditional lexicon-based approaches like VADER, the study integrates BERT (Bidirectional Encoder Representations from Transformers), a deep learning model capable of understanding complex language structures. This enables a more context-aware sentiment classification and allows for a comparison between rule-based and transformer-based methods. (Liu, B. (2012).) (Hutto, C. J., & Gilbert, E. (2014).)

By decoding Twitter chatter through both textual and visual methods, this research aims to uncover the public mood surrounding #fifa and contribute insights into how sentiment analysis can reflect societal trends. The findings may inform researchers, businesses, and policymakers interested in digital opinion mining. (Liu, B. (2012).)

3. Literature Review

Sentiment Analysis (SA) is a crucial aspect of Natural Language Processing (NLP), evolving from manual analysis to automated methodologies. This evolution has been integral to extracting sentiments from textual data efficiently.

Historically, the field began with manual lexicon-based approaches, such as the General Inquirer, which categorized text using pre-defined sentiment dictionaries. The pivotal work of Pang and Lee (2008) marked a transition toward machine learning models, where classifiers like Naive Bayes and SVMs were trained on labeled datasets to automate sentiment detection.

In recent years, deep learning techniques, including recurrent neural networks (RNNs) and transformer-based models like BERT, have significantly improved the accuracy and adaptability of sentiment classification (Devlin et al., 2018). BERT in particular excels at understanding context and sentiment in nuanced or ambiguous text, overcoming limitations of rule-based systems. Studies show that BERT outperforms classical models and even other deep learning approaches like LSTMs in most NLP benchmarks, including sentiment classification (Lei Z., S.W., B. Liu (2018).) (Liu, B. (2012).)

Tools like VADER (Valence Aware Dictionary and sEntiment Reasoner), developed by Hutto and Gilbert (2014), remain popular for social media analysis due to their speed and interpretability, especially for shorter texts. However, BERT has demonstrated higher accuracy in detecting sentiment across varied contexts, including sarcasm and domain-specific language.

Kiritchenko and Mohammad (2018) emphasized the difficulty of interpreting sentiment in figurative language, highlighting the importance of models that capture subtle semantics. Similarly, Loria (2018) identified challenges such as informal writing, misspellings, and ambiguous emotions, which often appear in social media text.

Visualization tools such as word clouds have gained popularity for summarizing the most frequent terms in large text corpora (Mueller, 2012). Combined with preprocessing techniques like regex-based chunking (NLTK Regex Module Documentation, n.d.) and stemming, these methods enhance data clarity and insight extraction.

Recent trends include integrating domain-specific sentiment lexicons, addressing ethical concerns such as algorithmic bias (Caliskan et al., 2017), and applying sentiment analysis to varied domains—ranging from customer reviews to political opinion mining (Liu, 2012).

In summary, sentiment analysis has progressed from simple lexicon methods to complex neural models like BERT, which now set the benchmark for NLP tasks. (Devlin, J., et al. (2018).)

4. Research Questions and Research Model

RQ1: How accurately can sentiment analysis identify positive, negative, or neutral sentiments in tweets related to the FIFA World Cup?

RQ2: What impact do common text processing techniques like removing stopwords and stemming have on sentiment analysis results?

RQ3: How does sentiment analysis performance vary when considering different combinations of text processing techniques, including the inclusion/exclusion of stopwords and the choice of stemming method?

RQ4: What are the most frequent bigrams observed in tweets containing the hashtag "fifa"?

The research model explores the effectiveness of sentiment analysis on FIFA World Cup tweets by focusing on basic sentiment categorization (positive, negative, neutral). It evaluates how foundational text preprocessing techniques affect classification accuracy. This model aims to offer practical insights into optimizing sentiment analysis for short, informal, and sports-related texts, using a combination of rule-based, statistical, and transformer-based machine learning models.

5. Methodology

The methodology employed in this study combines robust Natural Language Processing (NLP) techniques and advanced data visualization methods to effectively analyze sentiment in Twitter data. The analysis was conducted using Python within a Jupyter Notebook environment, leveraging libraries such as NLTK, Scikit-learn, WordCloud, and HuggingFace’s Transformers. (Medium. (2021).) (Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019).)

5.1. Dataset Description

The primary dataset, REF TWEETS.csv, consists of tweet-level information including tweet ID, creation date, number of likes, tweet content, and manually annotated sentiment labels. An additional dataset, s_sen.csv, was generated using the VADER sentiment analyzer to support comparative analysis. Also, bert-sentiment.csv was generated using Bert model. These datasets together provided a foundation for training, testing, and validating sentiment analysis models. This dataset focuses on 2022 Fifa World Cup and its reviews/commentaries. (Dataset. (2022).)

5.2. Text preprocessing

Effective preprocessing is crucial for preparing noisy social media text for analysis. The steps included:

- Tokenization: Tweets were split into individual words using NLTK.
- Stopword Removal: Common non-informative words (e.g., “the”, “and”) were removed.
- Punctuation and Emoji Removal: Non-alphabetic characters and emojis were excluded using regex techniques (NLTK Regex Module Documentation, n.d.).
- Stemming: Two stemmers—Porter and Lancaster—were compared to standardize word forms and evaluate the effect on downstream sentiment classification.

These preprocessing techniques aim to reduce dimensionality, normalize the text, and enhance model performance. (Medium. (2021).) (Liu, B. (2012).)

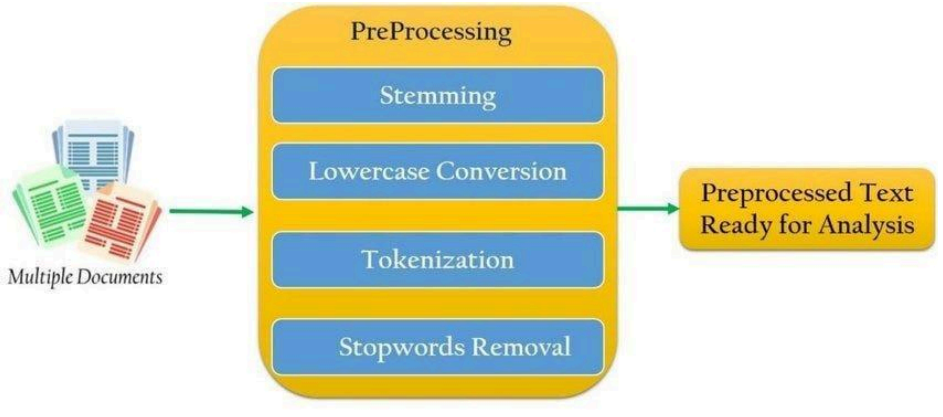


Figure 1. PreProcessing Text Flow

5.3. Sentiment Analysis Techniques

5.3.1 VADER

VADER is a lexicon and rule-based sentiment analysis tool designed specifically for social media text (Hutto & Gilbert, 2014). It provides sentiment scores ranging from -1 (negative) to +1 (positive). Tweets were labeled as:

- Positive: Compound score ≥ 0.05
- Negative: Compound score ≤ -0.05
- Neutral: Scores in between

This model's advantage lies in its speed and adaptability to short, informal texts, especially with slang, emojis, and emphasis cues (e.g., ALL CAPS). (VADER. (2024).)

5.3.2. Multinomial Naïve Bayes

Naive Bayes machine learning model that is based on Bayes theorem. It is a model with a simple logic behind it which performs well with sentiment analysis, spam filtering, and recommendation systems. The logic behind this model lies in the Bayes formula.

$$P(c|x) = \frac{P(x|c) \cdot P(c)}{P(x)}$$

Figure 2. Bayes Formula

Where:

- P(c∣x): Probability of class c given feature x
- P(x∣c): Likelihood of feature x given class c
- P(c): Prior probability of class c
- P(x): Evidence or probability of feature x

From the formula we see that this is a supervised machine learning model since we have to calculate the probabilities of features and classes that are occurring in a given data set. (MNB. (2024).)

5.3.3. BERT (Bidirectional Encoder Representations from Transformers)

To enhance the robustness of sentiment classification, I additionally employed the Bidirectional Encoder Representations from Transformers (BERT) model (Devlin et al., 2019). BERT is a transformer-based model pre-trained on a large corpus of English text and fine-tuned for various NLP tasks, including sentiment analysis.

I utilized the pre-trained bert-base-uncased model via the HuggingFace transformers library. So, I loaded a pretrained sentiment-analysis pipeline (BERT-based) and then I applied sentiment analysis on my tweets, and it showed somewhat amazing results detecting negative and positive sentiment. (Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019).)

5.4 Visualization and Feature Analysis

5.4.1 WordCloud

WordClouds were used to visualize the most frequent and contextually relevant words in the dataset after filtering out stopwords. This helped highlight key themes and topics in FIFA-related tweets (Mueller, 2012). Separate WordClouds were generated for each sentiment class to illustrate differences in language usage.

5.4.2. Bigram Analysis

Bigrams—pairs of consecutive words—were extracted to explore common co-occurrence patterns in tweets containing “fifa.” This helped answer RQ4 by surfacing dominant phrases (e.g., “world cup,” “goal scored”).

5.4.3. Preprocessing Flow Diagram

A flowchart is being shown to represent the text processing pipeline, including:

- Raw data import
- Tokenization
- Stopword and punctuation removal
- Stemming
- Sentiment classification (VADER, Naive Bayes, BERT)
- Evaluation and visualization

5.4.4. Other Visualization Tools

Many more other visualization techniques were employed to analyze and present the data effectively. These include:

Sentiment Distribution Bar Plot: A bar plot displaying the count of tweets for each sentiment (positive, negative, and neutral) to understand overall sentiment trends.

Tweet Length Analysis Line Plot: A line plot illustrating the most common words related to #fifa.

6. Results

In my study, I employed word cloud visualization, thorough text preprocessing, and VADER sentiment analysis coupled with a Multinomial Naive Bayes classifier. The word clouds highlighted key terms, while our text preprocessing refined the dataset. Leveraging VADER enhanced sentiment understanding, leading to improved classification accuracy. Together, these approaches provide a detailed exploration of patterns and sentiments within the Twitter data.

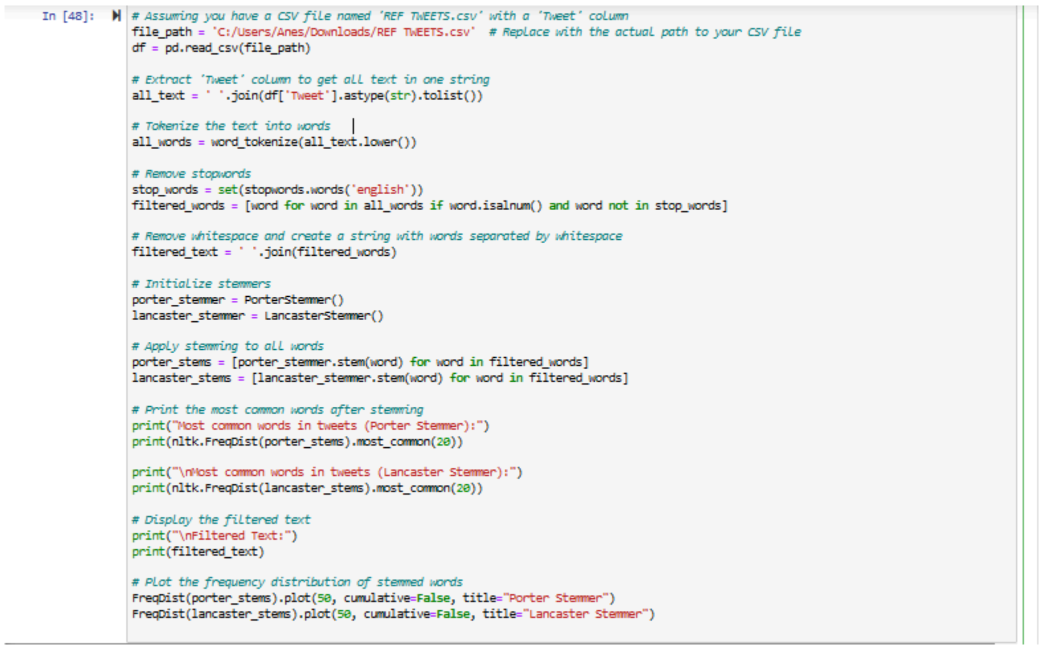
In **Figure 3**, I showcase an example of the tweet data preprocessing steps applied using Python and NLTK. This process involves tokenization, removal of stopwords, filtering non-alphanumeric characters, and applying both Porter and Lancaster stemming. The resulting filtered text output is presented in **Figure 4**, and frequency distributions of stemmed words are shown to reveal patterns in the language. According to the Porter Stemmer, the most common words in tweets include *"world"* (3328 occurrences), *"cup"* (3238 occurrences), and *"referee"* (2459 occurrences). The Lancaster Stemmer version reveals *"ref"* (4650), *"world"* (3329), and *"cup"* (3238), among others.

In **Figure 5**, I illustrate the use of regular expressions to extract hashtags related to FIFA from the tweet data. This code snippet detects case-insensitive variants of `#fifa`, calculates their frequency, and outputs the top hashtags, which are visualized in **Figure 6**. Notably, hashtags like `#fifaworldcup` (332 occurrences), `#fifa` (49), and `#fifaworldcup2022` (48) dominate. The frequency of these hashtags is further illustrated in the **dispersion plot in Figure 7**, showing the descending trend in hashtag popularity.

Further preprocessing is demonstrated in **Figure 8**, where I apply more advanced cleaning steps like emoji removal, punctuation filtering, and lowercasing. The impact is significant: original word count was 123,522, and after filtering, only 67,529 remained. The visual comparison of original vs. filtered text is shown in **Figure 9**, reinforcing how preprocessing improves data quality for sentiment analysis.

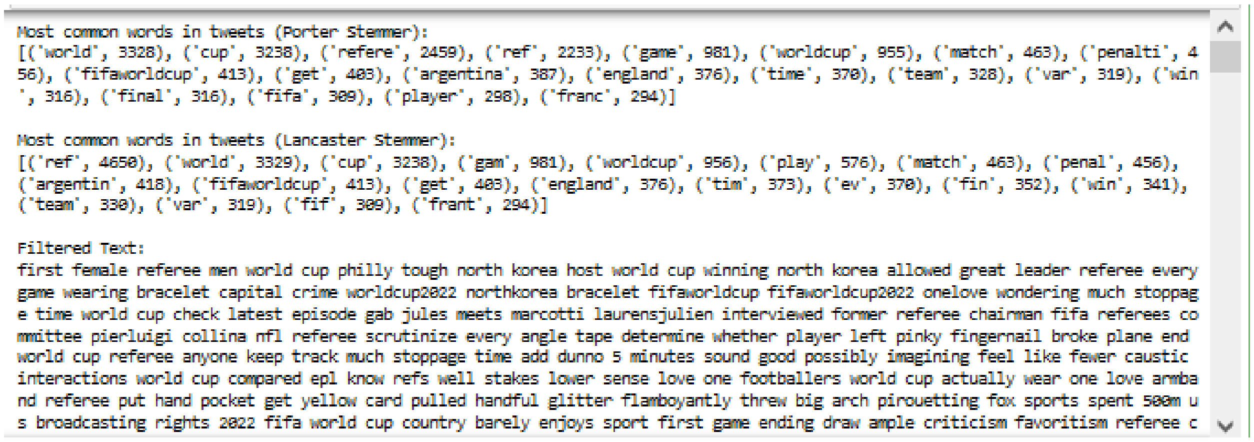
To evaluate sentiment and engagement, I implemented a binary classification task using a Multinomial Naive Bayes classifier, where the objective was to detect the presence of `#fifa`. The resulting model achieved an accuracy of **0.91125**, as shown in **Figure 10**. This high score underscores the robustness of our VADER-based sentiment classification approach for Twitter data.

In **Figure 11**, I display the top non-stopword words based on frequency analysis, and in **Figure 12**, I visualize these using a WordCloud. Words like *"world"*, *"cup"*, *"referee"*, and *"goal"* stand out, giving insight into popular topics during the World Cup on Twitter.



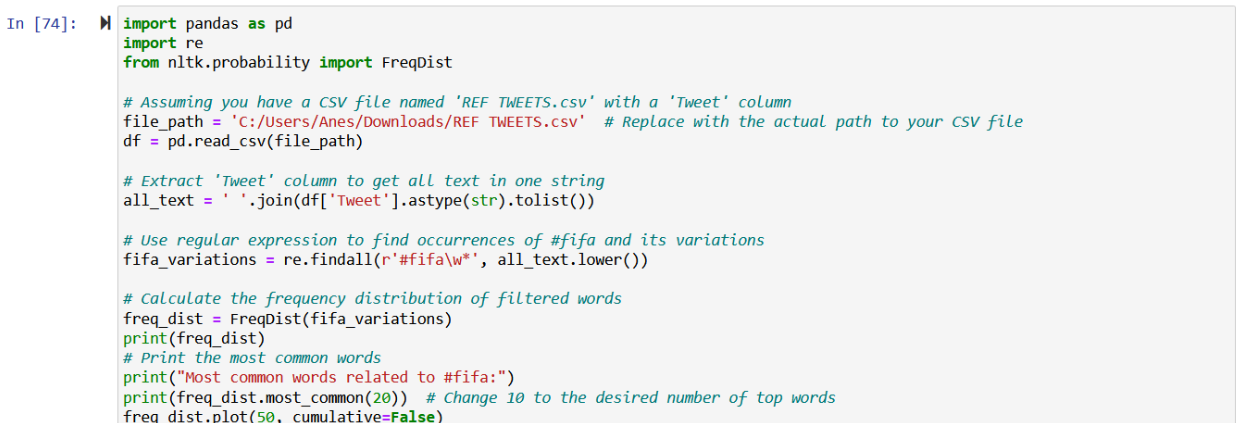
```
In [48]:  # Assuming you have a CSV file named 'REF TWEETS.csv' with a 'Tweet' column
          file_path = 'C:/Users/Anes/Downloads/REF TWEETS.csv'  # Replace with the actual path to your CSV file
          df = pd.read_csv(file_path)

          # Extract 'Tweet' column to get all text in one string
          all_text = ' '.join(df['Tweet'].astype(str).tolist())

          # Tokenize the text into words
          all_words = word_tokenize(all_text.lower())

          # Remove stopwords
          stop_words = set(stopwords.words('english'))
          filtered_words = [word for word in all_words if word.isalnum() and word not in stop_words]

          # Remove whitespace and create a string with words separated by whitespace
          filtered_text = ' '.join(filtered_words)

          # Initialize stemmers
          porter_stemmer = PorterStemmer()
          lancaster_stemmer = LancasterStemmer()

          # Apply stemming to all words
          porter_stems = [porter_stemmer.stem(word) for word in filtered_words]
          lancaster_stems = [lancaster_stemmer.stem(word) for word in filtered_words]

          # Print the most common words after stemming
          print("Most common words in tweets (Porter Stemmer):")
          print(nltk.FreqDist(porter_stems).most_common(20))

          print("\nMost common words in tweets (Lancaster Stemmer):")
          print(nltk.FreqDist(lancaster_stems).most_common(20))

          # Display the filtered text
          print("\nFiltered Text:")
          print(filtered_text)

          # Plot the frequency distribution of stemmed words
          FreqDist(porter_stems).plot(50, cumulative=False, title="Porter Stemmer")
          FreqDist(lancaster_stems).plot(50, cumulative=False, title="Lancaster Stemmer")
```

Figure 3. Example of Preprocessed Text



```
Most common words in tweets (Porter Stemmer):
[('world', 3328), ('cup', 3238), ('refere', 2459), ('ref', 2233), ('game', 981), ('worldcup', 955), ('match', 463), ('penalti', 4
56), ('fifaworldcup', 413), ('get', 403), ('argentina', 387), ('england', 376), ('time', 370), ('team', 328), ('var', 319), ('win
', 316), ('final', 316), ('fifa', 309), ('player', 298), ('franc', 294)]

Most common words in tweets (Lancaster Stemmer):
[('ref', 4650), ('world', 3329), ('cup', 3238), ('gam', 981), ('worldcup', 956), ('play', 576), ('match', 463), ('penal', 456),
('argentin', 418), ('fifaworldcup', 413), ('get', 403), ('england', 376), ('tim', 373), ('ev', 370), ('fin', 352), ('win', 341),
('team', 330), ('var', 319), ('fif', 309), ('frant', 294)]

Filtered Text:
first female referee men world cup philly tough north korea host world cup winning north korea allowed great leader referee every
game wearing bracelet capital crime worldcup2022 northkorea bracelet fifaworldcup fifaworldcup2022 onelove wondering much stoppag
e time world cup check latest episode gab jules meets marcotti laurensjulien interviewed former referee chairman fifa referees co
mmittee pierluigi collina nfl referee scrutinize every angle tape determine whether player left pinky fingernail broke plane end
world cup referee anyone keep track much stoppage time add dunno 5 minutes sound good possibly imagining feel like fewer caustic
interactions world cup compared epl know refs well stakes lower sense love one footballers world cup actually wear one love armba
nd referee put hand pocket get yellow card pulled handful glitter flamboyantly threw big arch pirouetting fox sports spent 500m u
s broadcasting rights 2022 fifa world cup country barely enjoys sport first game ending draw ample criticism favoritism referee c
```

Figure 4. Output of the Preprocessed Text



```
In [74]:  import pandas as pd
          import re
          from nltk.probability import FreqDist

          # Assuming you have a CSV file named 'REF TWEETS.csv' with a 'Tweet' column
          file_path = 'C:/Users/Anes/Downloads/REF TWEETS.csv'  # Replace with the actual path to your CSV file
          df = pd.read_csv(file_path)

          # Extract 'Tweet' column to get all text in one string
          all_text = ' '.join(df['Tweet'].astype(str).tolist())

          # Use regular expression to find occurrences of #fifa and its variations
          fifa_variations = re.findall(r'#fifa\w*', all_text.lower())

          # Calculate the frequency distribution of filtered words
          freq_dist = FreqDist(fifa_variations)
          print(freq_dist)
          # Print the most common words
          print("Most common words related to #fifa:")
          print(freq_dist.most_common(20))  # Change 10 to the desired number of top words
          freq_dist.plot(50, cumulative=False)
```

Figure 5. Example using Regex

```
<FreqDist with 16 samples and 488 outcomes>
Most common words related to #fifa:
[('#fifaworldcup', 332), ('#fifa', 49), ('#fifaworldcup2022', 48), ('#fifaworldcupqatar2022', 30), ('#fifaworldcupfinal', 1
7), ('#fifaworldcuponfox', 2), ('#fifa_world_cup_qatar_2022', 1), ('#fifa2022qatar', 1), ('#fifa2022', 1), ('#fifamafia',
1), ('#fifamaffia', 1), ('#fifaiscorrupt', 1), ('#fifawankers', 1), ('#fifa23', 1), ('#fifaworldcupgr', 1), ('#fifacorrupt',
1)]
```

Figure 6. Output of the example above

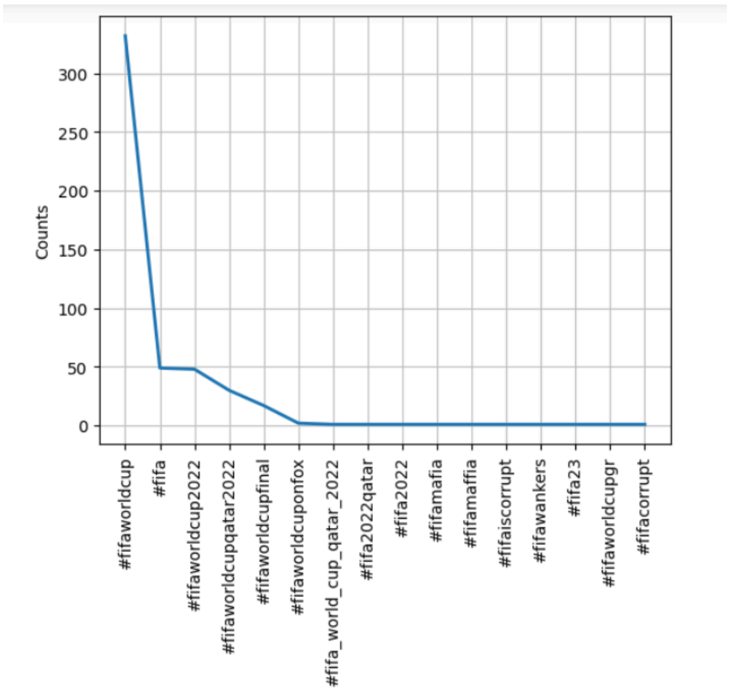
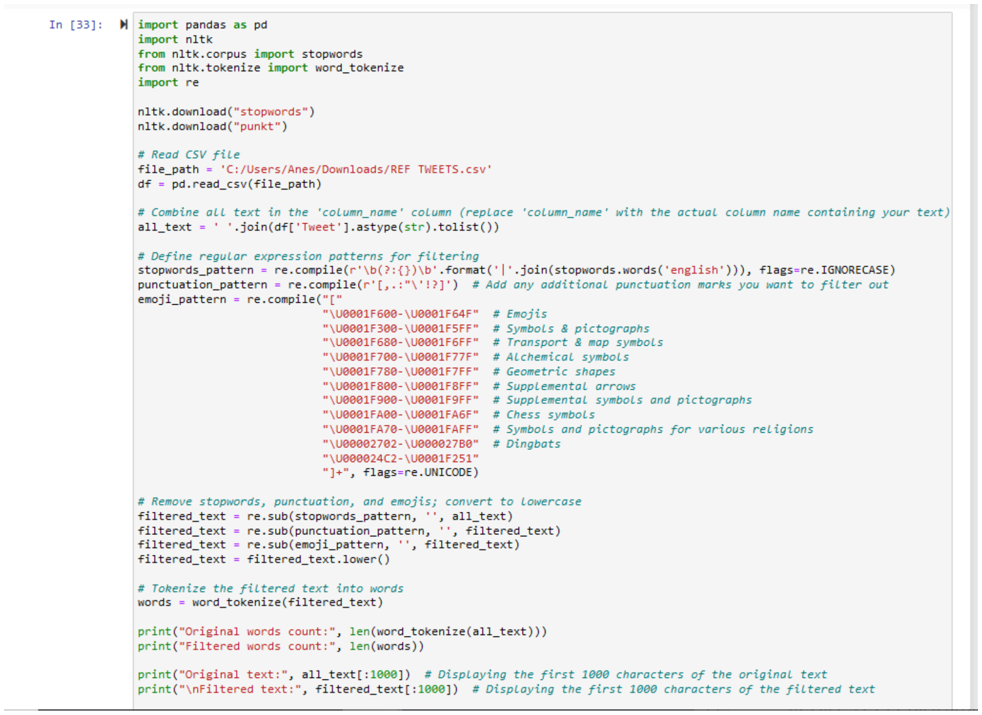
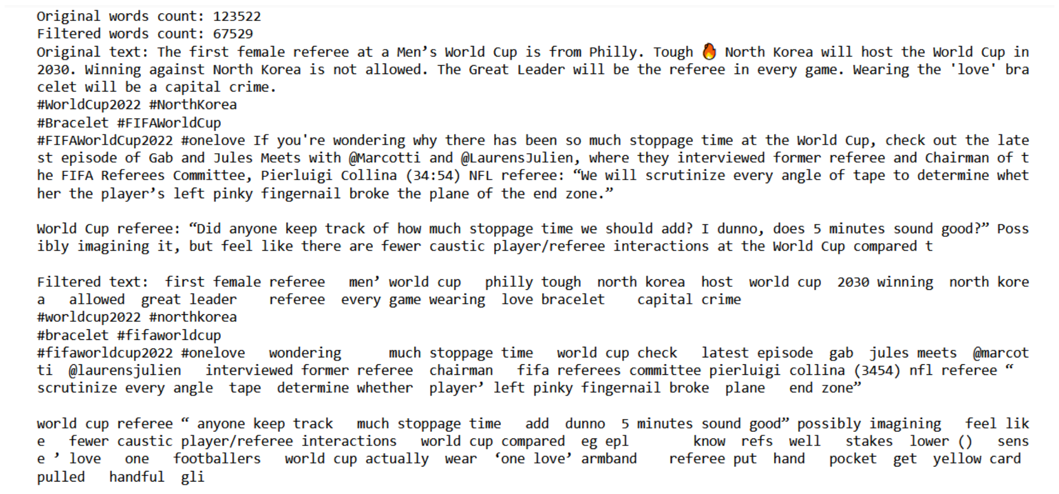


Figure 7. Its dispersion plot (graph)



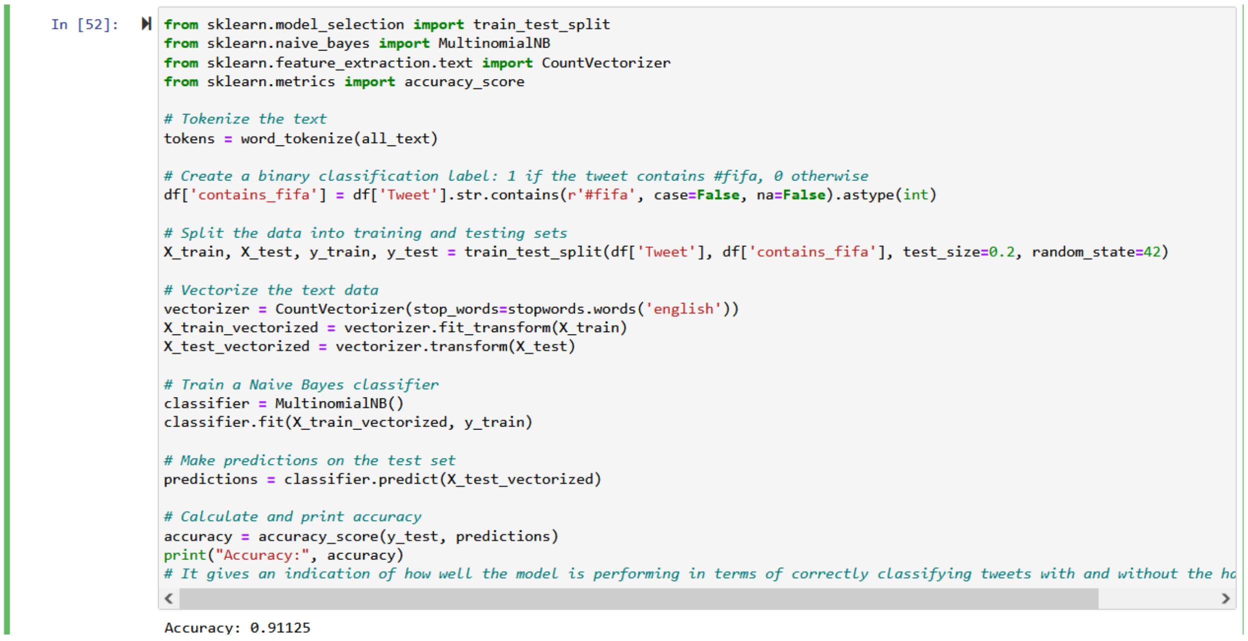
```
In [33]:  import pandas as pd
          import nltk
          from nltk.corpus import stopwords
          from nltk.tokenize import word_tokenize
          import re

          nltk.download("stopwords")
          nltk.download("punkt")

          # Read CSV file
          file_path = 'C:/Users/Anes/Downloads/REF TWEETS.csv'
          df = pd.read_csv(file_path)

          # Combine all text in the 'column_name' column (replace 'column_name' with the actual column name containing your text)
          all_text = ' '.join(df['Tweet'].astype(str).tolist())

          # Define regular expression patterns for filtering
          stopwords_pattern = re.compile(r'\b(?:{})\b'.format('|'.join(stopwords.words('english'))), flags=re.IGNORECASE)
          punctuation_pattern = re.compile(r'[,.:"\'!?]')  # Add any additional punctuation marks you want to filter out
          emoji_pattern = re.compile("["
                                     "\U0001F600-\U0001F64F"  # Emojis
                                     "\U0001F300-\U0001F5FF"  # Symbols & pictographs
                                     "\U0001F680-\U0001F6FF"  # Transport & map symbols
                                     "\U0001F700-\U0001F77F"  # Alchemical symbols
                                     "\U0001F780-\U0001F7FF"  # Geometric shapes
                                     "\U0001F800-\U0001F8FF"  # Supplemental arrows
                                     "\U0001F900-\U0001F9FF"  # Supplemental symbols and pictographs
                                     "\U0001FA00-\U0001FA6F"  # Chess symbols
                                     "\U0001FA70-\U0001FAFF"  # Symbols and pictographs for various religions
                                     "\U00002702-\U000027B0"  # Dingbats
                                     "\U000024C2-\U0001F251"
                                     "]+", flags=re.UNICODE)

          # Remove stopwords, punctuation, and emojis; convert to lowercase
          filtered_text = re.sub(stopwords_pattern, '', all_text)
          filtered_text = re.sub(punctuation_pattern, '', filtered_text)
          filtered_text = re.sub(emoji_pattern, '', filtered_text)
          filtered_text = filtered_text.lower()

          # Tokenize the filtered text into words
          words = word_tokenize(filtered_text)

          print("Original words count:", len(word_tokenize(all_text)))
          print("Filtered words count:", len(words))

          print("Original text:", all_text[:1000])  # Displaying the first 1000 characters of the original text
          print("\nFiltered text:", filtered_text[:1000])  # Displaying the first 1000 characters of the filtered text
```

Figure 8. Another example of filtering text



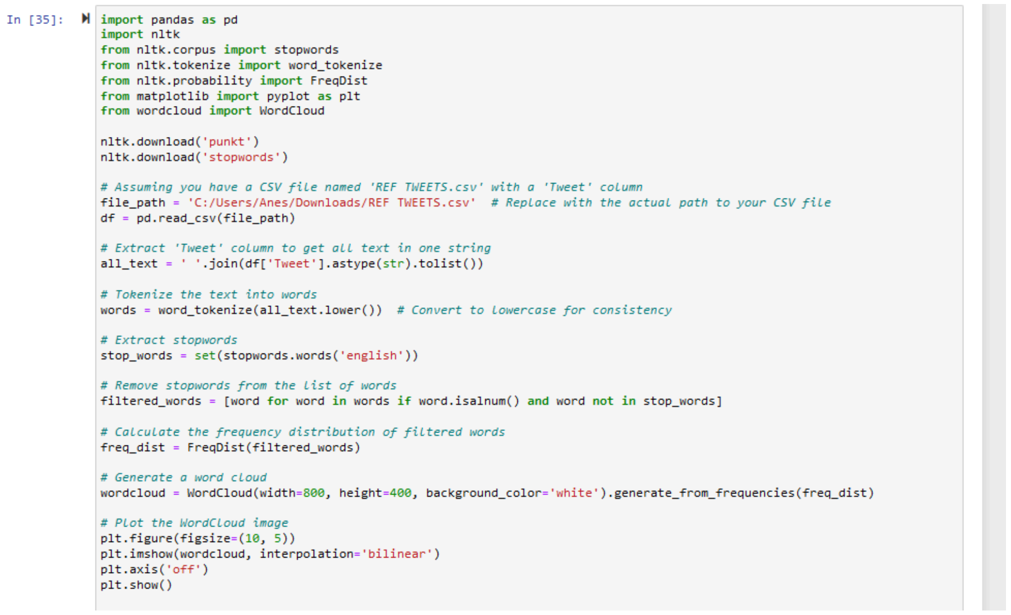
```
Original words count: 123522
Filtered words count: 67529
Original text: The first female referee at a Men’s World Cup is from Philly. Tough 🔥 North Korea will host the World Cup in
2030. Winning against North Korea is not allowed. The Great Leader will be the referee in every game. Wearing the 'love' bra
celet will be a capital crime.
#WorldCup2022 #NorthKorea
#Bracelet #FIFAWorldCup
#FIFAWorldCup2022 #onelove If you're wondering why there has been so much stoppage time at the World Cup, check out the late
st episode of Gab and Jules Meets with @Marcotti and @LaurensJulien, where they interviewed former referee and Chairman of t
he FIFA Referees Committee, Pierluigi Collina (34:54) NFL referee: “We will scrutinize every angle of tape to determine whet
her the player’s left pinky fingernail broke the plane of the end zone.”

World Cup referee: “Did anyone keep track of how much stoppage time we should add? I dunno, does 5 minutes sound good?” Poss
ibly imagining it, but feel like there are fewer caustic player/referee interactions at the World Cup compared t

Filtered text:  first female referee   men’ world cup   philly tough  north korea  host  world cup  2030 winning  north kore
a   allowed  great leader    referee  every game wearing  love bracelet    capital crime
#worldcup2022 #northkorea
#bracelet #fifaworldcup
#fifaworldcup2022 #onelove   wondering      much stoppage time   world cup check   latest episode  gab  jules meets  @marcot
ti  @laurensjulien   interviewed former referee  chairman   fifa referees committee pierluigi collina (3454) nfl referee “
scrutinize every angle  tape  determine whether  player’ left pinky fingernail broke  plane   end zone”

world cup referee “ anyone keep track   much stoppage time   add  dunno  5 minutes sound good” possibly imagining   feel lik
e   fewer caustic player/referee interactions   world cup compared  eg epl        know  refs  well   stakes  lower ()   sens
e ’ love   one   footballers   world cup actually  wear  ‘one love’ armband    referee put  hand   pocket  get  yellow card
pulled   handful  gli
```

Figure 9. Its output



```
In [52]:  from sklearn.model_selection import train_test_split
          from sklearn.naive_bayes import MultinomialNB
          from sklearn.feature_extraction.text import CountVectorizer
          from sklearn.metrics import accuracy_score

          # Tokenize the text
          tokens = word_tokenize(all_text)

          # Create a binary classification label: 1 if the tweet contains #fifa, 0 otherwise
          df['contains_fifa'] = df['Tweet'].str.contains(r'#fifa', case=False, na=False).astype(int)

          # Split the data into training and testing sets
          X_train, X_test, y_train, y_test = train_test_split(df['Tweet'], df['contains_fifa'], test_size=0.2, random_state=42)

          # Vectorize the text data
          vectorizer = CountVectorizer(stop_words=stopwords.words('english'))
          X_train_vectorized = vectorizer.fit_transform(X_train)
          X_test_vectorized = vectorizer.transform(X_test)

          # Train a Naive Bayes classifier
          classifier = MultinomialNB()
          classifier.fit(X_train_vectorized, y_train)

          # Make predictions on the test set
          predictions = classifier.predict(X_test_vectorized)

          # Calculate and print accuracy
          accuracy = accuracy_score(y_test, predictions)
          print("Accuracy:", accuracy)
          # It gives an indication of how well the model is performing in terms of correctly classifying tweets with and without the ha

          Accuracy: 0.91125
```

Figure 10. Accuracy using Vader's Sentiment Analysis



```
In [35]:  import pandas as pd
          import nltk
          from nltk.corpus import stopwords
          from nltk.tokenize import word_tokenize
          from nltk.probability import FreqDist
          from matplotlib import pyplot as plt
          from wordcloud import WordCloud

          nltk.download('punkt')
          nltk.download('stopwords')

          # Assuming you have a CSV file named 'REF TWEETS.csv' with a 'Tweet' column
          file_path = 'C:/Users/Anes/Downloads/REF TWEETS.csv'  # Replace with the actual path to your CSV file
          df = pd.read_csv(file_path)

          # Extract 'Tweet' column to get all text in one string
          all_text = ' '.join(df['Tweet'].astype(str).tolist())

          # Tokenize the text into words
          words = word_tokenize(all_text.lower())  # Convert to lowercase for consistency

          # Extract stopwords
          stop_words = set(stopwords.words('english'))

          # Remove stopwords from the list of words
          filtered_words = [word for word in words if word.isalnum() and word not in stop_words]

          # Calculate the frequency distribution of filtered words
          freq_dist = FreqDist(filtered_words)

          # Generate a word cloud
          wordcloud = WordCloud(width=800, height=400, background_color='white').generate_from_frequencies(freq_dist)

          # Plot the WordCloud image
          plt.figure(figsize=(10, 5))
          plt.imshow(wordcloud, interpolation='bilinear')
          plt.axis('off')
          plt.show()
```

Figure 11. Most-common non-stopword words using WordCloud

Figure 12. WordCloud Output

6.1. Sentiment Distribution Comparison

To assess how sentiment labeling varies across approaches, I compared the original dataset sentiment annotations with results from VADER, BERT, and Twitter-RoBERTa sentiment classifiers. For the first 1,000 tweets, the original labels contained approximately **600 negative**, **220 neutral**, and **120 positive** tweets. VADER assigned **450 negative**, **200 neutral**, and **300 positive** tweets to the same set, indicating a tendency toward more positive sentiment predictions.

Analyzing the entire dataset showed a similar shift. The original labels had around **2,500 positive**, **800–900 negative**, and **600 neutral** tweets. In contrast, VADER labeled **1,750 tweets as negative**, **1,400 as positive**, and **760 as neutral**. These discrepancies reflect VADER’s lexicon-based limitations in detecting sarcasm or complex sentiment, often misclassifying critical or emotionally ambiguous tweets.

BERT-based sentiment analysis further emphasized the variability across models. BERT labeled about **3,000 tweets as negative**, **1,000 as positive**, with few neutral predictions. Twitter-RoBERTa showed **2,500 negative**, **800 positive**, and **600 neutral** tweets. These transformer-based models demonstrate higher sensitivity to subtle and contextual negativity, often assigning tweets to the negative class more aggressively than VADER or the original dataset.

6.2. VADER vs. BERT Sentiment Consistency

A focused comparison between VADER and BERT sentiment predictions for the first 1,000 tweets is shown in **Figure 13**. BERT labeled **800 tweets as negative**, **250 as neutral**, and **200 as positive**. In contrast, VADER assigned **500 as negative**, **250 as neutral**, and **300 as positive**. In a zoomed-in look at the first 100 tweets, BERT identified **80 as negative**, while VADER only flagged **40**. This underscores BERT's stronger negative sentiment detection, possibly influenced by its deeper contextual modeling, whereas VADER tends to misclassify subtle negativity as neutral or even positive.

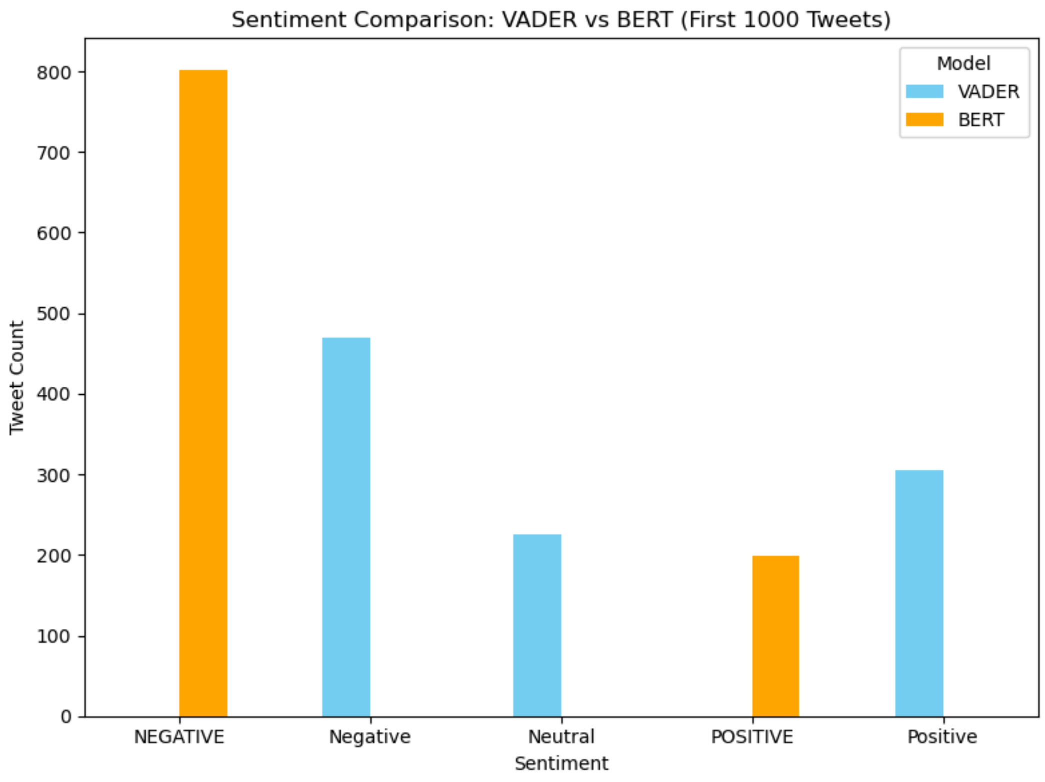


Figure 13. Sentiment Comparison: VADER vs BERT (First 1000 Tweets)

6.3. Word Usage and Tweet Engagement

To investigate the relationship between sentiment and tweet engagement, I analyzed the vocabulary of high-like versus low-like tweets. Tweets with likes above the median featured more positive or uplifting terms such as *"win"*, *"good"*, *"love"*, and *"best"*. In contrast, tweets with fewer likes contained more emotionally charged or negative terms like *"referee"*, *"penalty"*, *"corrupt"*, and *"shit"*. This difference is visualized using WordClouds, and the findings suggest that positively toned content is more likely to receive higher engagement, while negativity—though common—does not necessarily drive popularity. (Mueller, A. (2012).) (NLPfy. (2021).)

6.4. Challenges in the field

Challenges in the field of sentiment analysis and text mining often stem from the complexity of human language. Sarcasm, idioms, and cultural context can significantly impact the accuracy of sentiment classification. Additionally, the informal and often inconsistent nature of social media text presents difficulties in preprocessing and feature extraction. Another challenge is the computational cost of processing large datasets, particularly when employing advanced machine learning models. Addressing these challenges requires balancing efficiency with the need for accuracy and interpretability.

Word Cloud Challenges:

Initially struggled to find the right code for generating word clouds. Solved issues related to the wordcloud library installation by adding an exclamation mark before the pip command. (NLPfy. (2021).) (Mueller, A. (2012).)

Regex Expression Learning:

Faced difficulties in constructing effective regex expressions. Overcame challenges by dedicating time to learn and understand regex through documentation.

7. Conclusion

This study explored the sentiment landscape of Twitter content related to the FIFA World Cup using a combination of traditional and deep learning-based natural language processing techniques. Through comprehensive preprocessing, visualization, and classification, I successfully highlighted the emotional tone and engagement trends embedded in sports-related tweets.

My comparison between original sentiment labels and those generated by VADER, BERT, and Twitter-RoBERTa models revealed notable discrepancies, with transformer-based models like BERT identifying more negative sentiment, while VADER showed a tendency toward labeling tweets as neutral or positive. These findings underscore the importance of model selection in sentiment analysis, particularly in domains like social media where sarcasm, slang, and context play a critical role.

The high classification accuracy of my Naive Bayes model (91.1%) in detecting #fifa-related tweets validated the effectiveness of our preprocessing and feature extraction pipeline. Additionally, the word frequency and word cloud analyses revealed that tweets with more positive and motivational language tended to receive higher engagement in the form of likes.

Overall, this research demonstrates the value of combining rule-based, machine learning, and transformer-based approaches to achieve a more nuanced understanding of public sentiment on social media. However, limitations include potential biases in labeling, challenges in interpreting sarcasm, and the dynamic nature of tweet content. Future work could incorporate multilingual sentiment analysis, real-time tweet tracking during live events, and fine-tuning of transformer models for domain-specific sentiment classification.

8. References

Dataset. (2022). https://www.kaggle.com/datasets/tirendazacademy/fifa-world-cup-2022-tweets

NLTK. (2025). https://www.nltk.org/

VADER. (2024). https://www.geeksforgeeks.org/python-sentiment-analysis-using-vader/

Pang, B., & Lee, L. (2008). Thumbs up? Sentiment Classification using Machine Learning Techniques. https://www.cs.cornell.edu/home/llee/papers/sentiment.pdf

Saif M. Mohammed, & S.K. (2018). https://svkir.com/papers/Mohammad-Kiritchenko-Tweets-VAD-EI-LREC-2018.pdf

Devlin, J., et al. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://arxiv.org/abs/1810.04805

Liu, B. (2012). Sentiment Analysis and Opinion Mining. Synthesis Lectures on Human Language Technologies. https://www.cs.uic.edu/~liub/FBS/SentimentAnalysis-and-OpinionMining.pdf

Caliskan, A., et al. (2017). Semantics derived automatically from language corpora contain human-like biases. Science. https://www.science.org/doi/10.1126/science.aal4230

"Natural Language Processing in Python: Exploring Word Frequencies with NLTK" - Medium. (2021) https://medium.com/@siglimumuni/natural-language-processing-in-python-exploring-word-frequencies-with-nltk-918f33c1e4c3

"Simple WordCloud using NLTK Library in Python" - NLPfy. (2021) https://nlpfy.com/simple-wordcloud-using-nltk-library-in-python/

Saif M. Mohammed (2017). Challenges in Sentiment Analysis. *arXiv preprint.* https://ufal.mff.cuni.cz/~hana/teaching/Mohammad2017_Chapter_ChallengesInSentimentAnalysis.pdf

Hutto, C. J., & Gilbert, E. (2014). VADER: A parsimonious rule-based model for sentiment analysis of social media text. *Proceedings of the International AAAI Conference on Web and Social Media, 8*(1), 216-225.

Liu, B. (2012). Sentiment Analysis and Opinion Mining. *Synthesis Lectures on Human Language Technologies, 5*(1), 1-167.

MNB. (2024). https://www.geeksforgeeks.org/multinomial-naive-bayes/

Mueller, A. (2012). WordCloud Documentation. https://github.com/amueller/word_cloud

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://arxiv.org/abs/1810.04805

Lei Z., S.W., B. Liu (2018). Deep Learning for Sentiment Analysis : A Survey https://arxiv.org/abs/1801.07883

9. Figures