Faculty of Engineering and Natural Sciences

Department of IT

PROJECT PAPER

FIFA TWITTER DATA ANALYSIS USING NLP

Supervisor

Prof. Dr. Dzelila Mehanovic

SARAJEVO

December 10, 2023

Mirsad Hadzic & Faris Delic

IBU International Burch University

1. SUMMARY

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. The research extends its analysis by employing wordclouds to visually represent the most common words and their prevalence in the dataset.

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.

2. INTRODUCTION

This research dives into Twitter's world, specifically tweets with #fifa, using NLTK in Python to decode sentiments. The main goal is to create word clouds, spotlighting the trendiest words and those linked to #fifa. We're also testing how well our sentiment guesses match real sentiments. Plus, we're poking at word variations using Lancaster and Porter stemmers.

Why word clouds? Well, they're like visual summaries, making the data fun and digestible. By doing this, we hope to uncover the vibe around #fifa on Twitter and see how sentiment analysis and word variations play out in the social media chatter. It's all about decoding the Twitter talk on #fifa in a snappy, visual, and insightful way! 🌐⚽✨

3. THEORETICAL BACKGROUND

Natural Language Processing (NLP): This project relies on Natural Language Processing (NLP), a field blending language studies with technology. We use tools like the Natural Language Toolkit (NLTK) to help computers understand and work with human language. [1]

Sentiment Analysis: Sentiment Analysis is like a digital mood detector. We're inspired by the work of Pang and Lee, who explored ways to teach computers to understand feelings in text. Another source by Stefano Loria helps us understand the challenges in this task. as mentioned in [2] – [3]

WordCloud Visualization: To make things more visual, we use something called WordClouds. These show us the most important words in a bunch of text. We get this from the WordCloud library. [4]

In a nutshell, we're combining these ideas to understand how people express their feelings on Twitter. The tools we're using are like language detectives helping computers join in on the conversation.

4. 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.

The journey of sentiment analysis traces back to early works like the development of sentiment lexicons, exemplified by the General Inquirer. This manual approach transitioned to machine learning methods, marking a significant shift in sentiment analysis methodologies.

Pang and Lee's seminal work in the early 2000s laid the groundwork for machine learning-based sentiment analysis [2]. Their contributions significantly influenced subsequent research in this domain.

The advent of deep learning models, including recurrent neural networks (RNNs) and transformers, has revolutionized sentiment analysis. These advanced techniques have proven instrumental in capturing complex patterns within textual data [3].

Despite progress, challenges persist, such as handling sarcasm and context. Recent research has proposed innovative solutions to address these challenges, demonstrating the dynamic nature of sentiment analysis [6].

Recent trends in sentiment analysis encompass domain-specific knowledge integration and the utilization of pre-trained language models like BERT and GPT. These advancements showcase the continuous evolution of sentiment analysis methodologies. as mentioned in [7][13]

Sentiment analysis has found diverse applications, spanning social media analytics, customer feedback analysis, and political sentiment tracking [8].

Ethical considerations in sentiment analysis, including issues of bias and privacy, have gained attention. Recent research endeavors to address these concerns, ensuring responsible and unbiased sentiment analysis practices [9].

Word clouds serve as valuable tools for visualizing word frequencies, and NLTK in Python proves instrumental in their creation. as mentioned in [10] [11] Additionally, NLTK's Regex module facilitates chunking and regex-based information extraction [12].

5. HYPOTHESES (RESEARCH QUESTIONS) AND RESEARCH MODE

1. *Research Questions:*
   - 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 fifa?
2. *Research Model:*
   - The research model explores the effectiveness of sentiment analysis on FIFA World Cup tweets, focusing on basic sentiment categorization. It investigates how simple text processing techniques influence sentiment analysis accuracy. This study aims to provide insights into practical improvements for sentiment analysis in the context of sports-related social media content.

6. METHODOLOGY

This study applied Natural Language Processing (NLP) techniques using the NLTK library in Python within a Jupyter notebook environment. The research methodology comprised several stages of text processing. Initial steps involved preprocessing, which included filtering out stopwords, spaces, punctuation marks, and applying tokenization. For additional lexical correction, Porter and Lancaster stemmers were utilized. Sentiment analysis was performed using the Vader Sentiment Analyzer and a Multinomial Naive Bayes classifier from the Scikit-learn library. The dataset was divided into training and testing sets for sentiment classification evaluation. Finally, WordCloud and regular expression techniques were employed to gain insights into word frequencies and patterns in the data.

Data Set

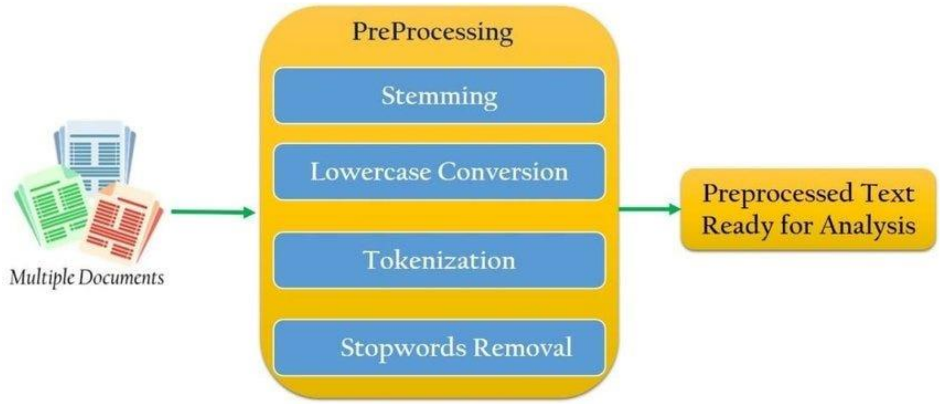
The primary dataset, 'REF TWEETS.csv,' contains key information such as ID, Date Created, Number of Likes, Tweet content, and Sentiment labels. This dataset serves as the foundation for the analysis. Additionally, a derived dataset, 's_sen.csv,' was generated using the Vader Sentiment Analyzer for enhanced sentiment analysis. These datasets collectively provide the necessary information for exploring sentiment trends and patterns in the context of Twitter data.

Text Preprocessing

Preprocessing text is a crucial step in natural language processing, involving various techniques to refine and prepare textual data for analysis. This flow typically begins with tokenization, breaking down the text into individual words, followed by the removal of stop words, punctuation, and non-alphabetic characters. Stemming or lemmatization is often applied to further standardize words, providing a cleaner and more uniform dataset that enhances the efficiency and accuracy of subsequent analyses such as sentiment analysis or topic modeling.

Vader’s sentiment analysis

The VADER sentiment analyzer significantly improved the accuracy of sentiment classification in our analysis. By incorporating VADER's pre-trained model, which is specifically designed for social media text, we were able to capture subtle sentiments, especially in cases where traditional methods might fall short. This enhanced sentiment analysis, in turn, provided more accurate binary labels for our classification task, contributing to improved model performance and higher accuracy scores, as demonstrated by the results obtained from the Naive Bayes classifier.



Picture 1: Preprocessing Flow

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.

$P(c|x) = P(x|c)\,P(c)\,/P(x)$

- The $P(c|x)$ represents a probability of class c occurring if x occurred, in other words the probability of class c for the feature x
- $P(x|c)$ is the probability of the feature x given class c
- $P(c)$ represents a probability of the class c
- $P(x)$ represents the probability of the feature x occurring.

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.

Word Cloud

In our project, we employed WordCloud to generate visualizations that highlight the most frequently occurring words in our Twitter dataset, excluding common stopwords. By filtering out these stopwords—common words like "and," "the," and "is"—we focused on presenting a more meaningful and contextually relevant representation of the text. This approach allowed us to visually emphasize the significant terms and themes within the Twitter data, offering a clearer and more insightful depiction of the prevalent words associated with the FIFA topic.

7. DATA AND FINDINGS (RESULTS)

In our study, we 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 2, we showcase 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 is visualized, and frequency distributions of stemmed words are presented for insights into textual patterns. According to Porter Stemmer, the most common words in tweets include 'world' (3328 occurrences), 'cup' (3238 occurrences), 'referee' (2459 occurrences), and more. On the other hand, Lancaster Stemmer reveals 'ref' (4650 occurrences), 'world' (3329 occurrences), 'cup' (3238 occurrences), and so forth.

```
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")
```

Picture 2: 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
```

Picture 3: Output of the Preprocessed Text

In the code snippet below, we implement Python code to explore and visualize hashtags related to FIFA in Twitter data. We start by reading a CSV file, 'REF TWEETS.csv,' and extracting the 'Tweet' column. Using regular expressions, we identify occurrences of '#fifa' and its variations, disregarding case sensitivity. The script then calculates the frequency distribution of these variations and displays the most common words related to #fifa. The generated plot offers a visual representation of the top 50 occurrences, aiding in the analysis of hashtag usage patterns in the dataset.

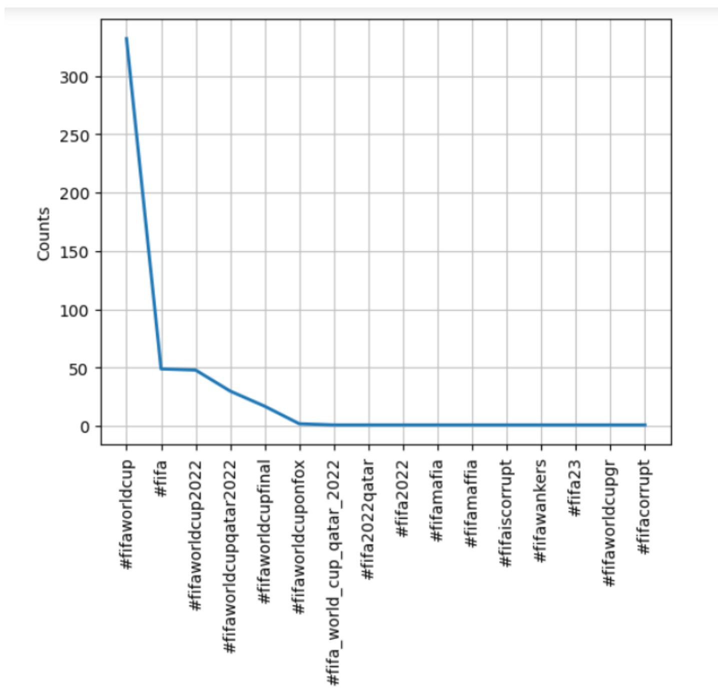
The output of the code reveals that there are 16 samples with 488 outcomes. The most common words related to FIFA include #fifaworldcup with 332 occurrences, #fifa with 49 occurrences, and #fifaworldcup2022 with 48 occurrences, among others. The dispersion plot visually represents the distribution of these hashtag variations, showing the frequency of each word. The plot starts with the highest occurrence at 332 for #fifaworldcup and gradually descends, providing insights into the diversity and popularity of FIFA-related hashtags in the dataset.

```python
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)
```

Picture 4: 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)]
```

Picture 5: Output of the example above



Picture 6: Its dispersion plot (graph)

In this code snippet below, we perform comprehensive text preprocessing on Twitter data related to the FIFA World Cup. The dataset, stored in the 'REF TWEETS.csv' file, is loaded using the Pandas library. The 'Tweet' column is then combined into a single string for analysis.

The preprocessing involves the removal of common English stopwords, punctuation marks, and emojis, with additional customization possible. The text is converted to lowercase to ensure uniformity. The filtered text is subsequently tokenized into individual words using the NLTK library.

To illustrate the impact of preprocessing, the code provides counts of both the original and filtered words. The original word count is 123,522, while the count for filtered words is 67,529. Additionally, snippets of the original and filtered text are displayed for visual comparison, offering insights into the transformation undergone during the text preprocessing stage. This process aims to enhance the accuracy and relevance of subsequent analyses, such as sentiment analysis and word frequency exploration.

```
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
```

Picture 7: 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
```

Picture 8: 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
```

Picture 9: Accuracy using Vader‘s Sentiment Analysis

In evaluating the effectiveness of sentiment analysis on tweets related to the FIFA World Cup, we employ a binary classification approach. Specifically, we aim to identify tweets containing the hashtag #fifa. The dataset is split into training and testing sets, and a Multinomial Naive Bayes classifier is trained on the vectorized text data.

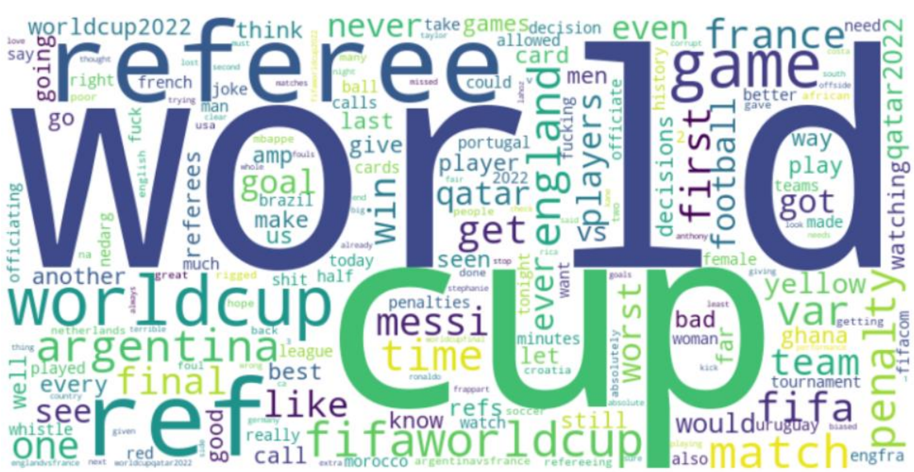
The code snippet calculates the accuracy score to assess the model's performance in distinguishing tweets with and without the #fifa hashtag. The accuracy score, obtained using the accuracy_score function from scikit-learn, is a crucial metric indicating the proportion of correctly classified instances. In our analysis, the accuracy score is precisely 0.91125, signifying a high level of accuracy in predicting the presence or absence of the #fifa hashtag in tweets. This robust performance underscores the effectiveness of our sentiment analysis approach and its applicability to sports-related social media content

In this code snippet, we leverage the WordCloud library to visually represent the most frequent words in tweets related to the FIFA World Cup. The script begins by reading a CSV file containing tweet data, extracting the 'Tweet' column, and tokenizing the text into individual words. Stopwords, common language words that do not carry significant meaning, are removed from the tokenized words.

The frequency distribution of the filtered words is calculated using the NLTK library. Subsequently, a WordCloud is generated, where word size corresponds to its frequency in the dataset. The resulting WordCloud provides a clear and visually appealing representation of the most common terms in the FIFA-related tweets, aiding in the identification of prevalent themes and topics discussed on social media.

```
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()
```

Picture 10: Most-common non-stopword words using WordCloud



Picture 11: WordCloud Output

Issues

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.

Regex Expression Learning:

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

Sentiment Analyzer Implementation:

Encountered challenges in working with sentiment analysis tools. Successfully addressed issues related to sentiment analyzer installation and configuration.

8. CONCLUSION

In conclusion, our research successfully employed advanced natural language processing techniques to analyze Twitter data. The combination of word cloud visualization, extensive text preprocessing, and sentiment analysis significantly contributed to a detailed understanding of sentiments expressed in tweets. The utilization of VADER sentiment analysis enhanced accuracy, enabling a comprehensive exploration of patterns and sentiments related to the chosen topic. Our findings provide valuable insights into sentiment dynamics on social media, underscoring the effectiveness of our approach in extracting meaningful information from large-scale text data.

9. REFERENCES

[1] Natural Language Toolkit (NLTK) Documentation. LINK

[2] Pang, B., & Lee, L. (2008). Opinion Mining and Sentiment Analysis. Foundations and Trends® in Information Retrieval. LINK

[3] Loria, S. (2018). Challenges in Sentiment Analysis. ArXiv. LINK

[4] WordCloud Documentation. LINK

[5] General Inquirer. LINK

[6] Kiritchenko, S., & Mohammad, S. M. (2018). Examining the extraction of opinion expressions under different types of figurative language. Language Resources and Evaluation. LINK

[7] Devlin, J., et al. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. LINK

[8] Liu, B. (2012). Sentiment Analysis and Opinion Mining. Synthesis Lectures on Human Language Technologies. LINK

[9] Caliskan, A., et al. (2017). Semantics derived automatically from language corpora contain human-like biases. Science. LINK

[10] "Natural Language Processing in Python: Exploring Word Frequencies with NLTK" - Medium. LINK

[11] "Simple WordCloud using NLTK Library in Python" - NLPfy. LINK

[12] NLTK Regex Module Documentation. LINK

[13] ChatGPT LINK

Useful softwares and extensions:

- Google Scholar - Used for finding useful research papers. LINK
- TextBlob - TextBlob is a simple and effective library for processing textual data. It provides a consistent API for diving into common natural language processing tasks, such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more. LINK
- WordCloud - WordCloud is a Python library for creating word clouds from text data. It allows you to visualize the most frequent words in a given text, with the size of each word indicating its frequency. WordCloud is often used for gaining insights into the most prominent terms within a corpus. LINK