CSIT 696: Research Methods in Computing

Final Project Report

Project Tile: Sentiment Analysis of YouTube comments

Name: Siva Chandra Kakarlapudi

---

1. Description

1.1 Basic Information

The motivation behind embarking on this project was a personal curiosity fueled by a deep passion for anime. As a dedicated fan of Onepiece, I was keen to explore and understand how fellow anime enthusiasts and the broader online community were responding to this particular show. YouTube, being a platform with a significant presence of anime-related content, offered a rich source of opinions and discussions through its comments section.

Anime has a unique ability to evoke strong emotions and connections among its viewers. Onepiece holds a special place in my heart, and I was eager to delve into the sentiments and discussions surrounding it.

Choosing sentiment analysis as the primary method allowed me to quantitatively assess the emotional responses and prevailing opinions within the YouTube comments. This analytical approach provided a structured way to uncover patterns, identify trends, and gain a comprehensive understanding of the community's reception of the anime.

1.2 Project Objectives

Sentiment Distribution:

Explore how sentiments (positive, negative, neutral) are distributed among YouTube comments related to Onepiece.

Word Cloud for Video Creation:

Identify key words and phrases in comments to inform the creation of engaging and relevant new videos. Analyze frequently mentioned topics or themes in positive comments and consider incorporating common fan phrases into video content.

Model Comparison for Sentiment Analysis:

Compare different sentiment analysis models to assess their effectiveness in classifying sentiments within YouTube comments.

Accuracy Assessment with Different Models:

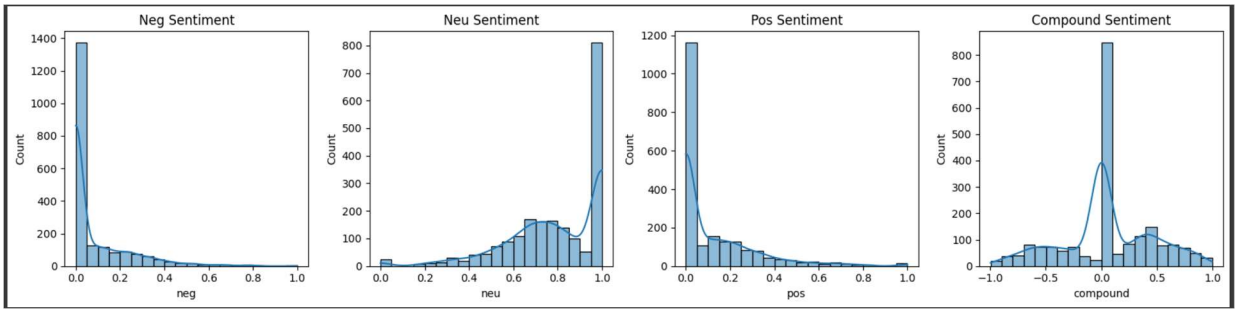
Assess the accuracy of various machine learning models, including Logistic Regression, Naive Bayes, and deep learning models like LSTM.

1.3 Description of the Data Set

The selected dataset for analysis was obtained by utilizing the YouTube API v3 to extract comments from the Hyperkage YouTube channel. This channel is dedicated to content related to Onepiece, and the comments retrieved represent the diverse range of sentiments, opinions, and discussions expressed by viewers and fans of the anime on the platform.

The dataset comprises a collection of textual comments associated with various videos on the Hyperkage channel, providing a snapshot of the audience's reactions and engagement with the anime content. By leveraging the YouTube API, I was able to gather a substantial volume of comments, forming the basis for sentiment analysis and other exploratory analyses to uncover patterns, sentiments, and insights within the fan community.

Distribution of comments in the Dataset:



2. Design of the Project

2.1 Technique Methodology

The project focused on analyzing sentiments in YouTube comments related to Onepiece. The following methodology was employed:

Data Collection:

Utilized the YouTube API v3 to extract comments from the Hyperkage YouTube channel. Captured a diverse range of sentiments expressed by viewers and fans of the anime.

Sentiment Analysis:

Applied VADER sentiment analysis to assign sentiment scores (positive, negative, neutral) to comments. Preprocessed text data by filtering non-alphabetic and non-numeric characters.

Count Vectorization:

- Utilized CountVectorizer to transform text data into a numerical representation.
- Obtained training and testing sets for machine learning models.

Machine Learning Models:

- Employed NLTK machine learning models for sentiment classification.
- Trained models using sentiments derived from VADER analysis.

Neural Network Model (LSTM):

- Tokenized and encoded text data for input into the neural network.
- Utilized a deep learning approach with an LSTM model for sentiment classification.

Word Cloud Generation:

Created word clouds to visually represent frequently mentioned words in comments.

2.2 Implementation of the Project

Code to pull data from Youtube using Youtube API v3:

We can choose the channel we want the data from by using channel_id which can be found in the About channel section in the channel info.

I choose the channel ID for the Hyperkage channel

```
import googleapiclient.discovery

api_service_name = "youtube"
api_version = "v3"
DEVELOPER_KEY = "AIzaSyDRm7QjpHwFZyZhsICaog2JUaB89LLOLMw"

youtube = googleapiclient.discovery.build(
    api_service_name, api_version, developerKey=DEVELOPER_KEY)

channel_id = "UCxx4MuVpQxG9BtNfG3o8hGg"  # Replace with the actual

request = youtube.search().list(
    part="snippet",
    channelId=channel_id,
    maxResults=100
)
response = request.execute()

for item in response['items']:
    print(item['snippet']['title'])
```

Cleaning the Dataset:

```
import re
def clean_comment(comment):
    # Remove non-alphanumeric characters and symbols, except spaces
    comment = re.sub(r'[^a-zA-Z0-9\s]', '', comment)

    # Convert to lowercase (optional)
    comment = comment.lower()

    return comment
comments = df['Content']
Cleaned_Comments = [clean_comment(comment) for comment in comments]
cleaned_df = pd.DataFrame(Cleaned_Comments, columns = ['Clean_Content'])
cleaned_df.head()
```

Cleaned Comments:

	Clean_Content
0	black beard pirates will win
1	what is blackbeard mumbling about in the backg...
2	obviously the straw hats win but it39s shady t...
3	kuzan ice age
4	bb 73

```
comment_text = df['Content']
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Combine all comments into a single string
combined_text = " ".join(comment for comment in comment_text)

# Create a WordCloud object
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(combined_text)

# Display the word cloud using Matplotlib
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

The above code is to generate word cloud in order to get it we join all the comments to get combined_text

The resulting word cloud is:

Sentiment analyser to classify the comments:

```
from nltk.sentiment import SentimentIntensityAnalyzer

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

Encoding the sentiments to numeric:

```
sentiment_mapping = {'Positive': 2, 'Neutral': 1, 'Negative': 0}
data_copy['Sentiment'] = data_copy['Sentiment'].map(sentiment_mapping)
```

Sentiment Count in each category:

```
processed_data['Sentiment'].value_counts()
```

```
1    848
2    701
0    513
Name: Sentiment, dtype: int64
```

Sampled the dataset to train machine learning model. Inorder to remove bias

```
from sklearn.utils import resample

df_neutral = processed_data[(processed_data['Sentiment']==1)]
df_negative = processed_data[(processed_data['Sentiment']==0)]
df_positive = processed_data[(processed_data['Sentiment']==2)]


# Upsample the neutral and negative sentiments to 250 samples each
df_negative_upsampled = resample(df_negative,
                                 replace=True,
                                 n_samples=599,
                                 random_state=42)

df_neutral_upsampled = resample(df_neutral,
                                replace=False,
                                n_samples=599,
                                random_state=42)
df_positive_unsampled = resample(df_positive,
                                replace=False,
                                n_samples=599,
                                random_state=42)


# Concatenate the upsampled and downsampled dataframes
final_data = pd.concat([df_negative_upsampled, df_neutral_upsampled, df_positive_unsampled])
```

As the number of negative samples is less than the required number of samples, I have used sampling with replacement: replace=True
Each time we run the entire code the API pulls different dataset and the counts also changes. So we need to make changes to the sampling everytime.

Converting the words in to tokens to train the classification models:

```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=150)
X = cv.fit_transform(corpus).toarray()
y = final_data.iloc[:, -1].values
```

2.3 Evaluation of the Project

Testing the sentiment analyser:

```
sia.polarity_scores('Luffy is awesome')
```

```
{'neg': 0.0, 'neu': 0.328, 'pos': 0.672, 'compound': 0.6249}
```

We can clearly see that the sentiment of the sentence is positive and the model correctly predicted it.

Accuracy:

Accuracies obtained by different Machine learning models in classifying the sentiments

1. GaussianNB

```
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

```
▾ GaussianNB
GaussianNB()
```

```
from sklearn.metrics import confusion_matrix, accuracy_score
y_pred = classifier.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[ 56,  60,  18],
       [  1, 106,   6],
       [ 13,  73,  27]])
```

```
nb_score = accuracy_score(y_test, y_pred)
print('accuracy',nb_score)
```

```
accuracy 0.525
```

2.MultinomialNB:

```
from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()
model.fit(X_train, y_train)
```

```
▾ MultinomialNB
MultinomialNB()
```

```
y_pred = model.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
```

```
n_score = accuracy_score(y_test, y_pred)
print('accuracy',n_score)
```

```
accuracy 0.5694444444444444
```

3.Logistic Regression:

```
from sklearn.linear_model import LogisticRegression
model1 = LogisticRegression(multi_class='ovr')  # or mu
model1.fit(X_train, y_train)
```

```
▾          LogisticRegression
LogisticRegression(multi_class='ovr')
```

```
y_pred1 = model1.predict(X_test)
cm = confusion_matrix(y_test, y_pred1)
cm
```

```
array([[75, 42, 17],
       [ 7, 94, 12],
       [20, 39, 54]])
```

```
n1_score = accuracy_score(y_test, y_pred1)
print('accuracy',n1_score)
```

```
accuracy 0.6194444444444445
```

4.SVC:

```
from sklearn.svm import SVC
model2 = SVC(kernel='linear', decision_function_shape='ovr')
model2.fit(X_train, y_train)
```

```
SVC
SVC(kernel='linear')
```

```
y_pred2 = model2.predict(X_test)
cm = confusion_matrix(y_test, y_pred2)
cm
```

```
array([[88, 31, 15],
       [14, 92,  7],
       [25, 35, 53]])
```

```
n2_score = accuracy_score(y_test, y_pred2)
print('accuracy',n2_score)
```

```
accuracy 0.6472222222222223
```

5.Random Forest Classifier:

```
from sklearn.ensemble import RandomForestClassifier
model3 = RandomForestClassifier()
model3.fit(X_train, y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
y_pred3 = model3.predict(X_test)
cm = confusion_matrix(y_test, y_pred3)
cm
```

```
array([[98, 26, 10],
       [17, 77, 19],
       [14, 33, 66]])
```

```
n3_score = accuracy_score(y_test, y_pred3)
print('accuracy',n3_score)
```

```
accuracy 0.6694444444444444
```

6.LSTM:

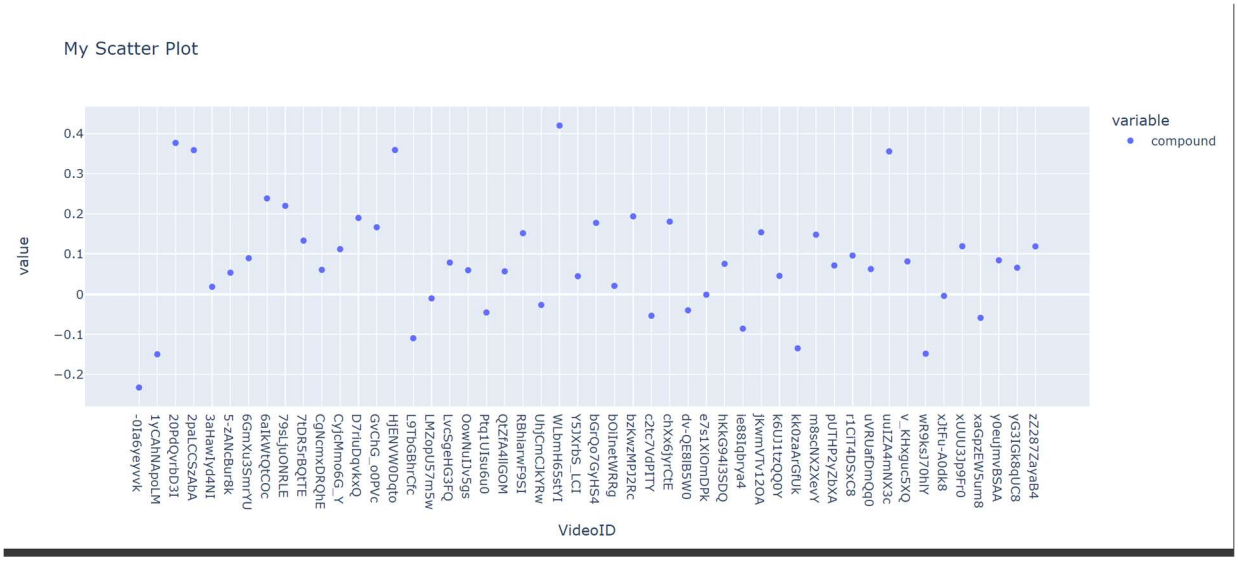
```
# Example using TensorFlow and Keras for an LSTM-based model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense

model4 = Sequential()
model4.add(Embedding(input_dim=vocab_size, output_dim=50, input_length=150))
model4.add(LSTM(100))
model4.add(Dense(3, activation='softmax'))
```

```
# Compile the model
model4.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Train the model
model4.fit(X_train, y_train, epochs=30, batch_size=40, validation_split=0.2)
```

```python
# Evaluate the model on the test set
test_loss, test_accuracy = model4.evaluate(X_test, y_test)

# Print the test loss and accuracy
print(f'Test Loss: {test_loss:.4f}')
print(f'Test Accuracy: {test_accuracy * 100:.2f}%')
```

```
12/12 [==============================] - 0s 37ms/step - loss: 1.0297 - accuracy: 0.4316
Test Loss: 1.0297
Test Accuracy: 43.16%
```

3. Conclusion



Ranking of the videos based on their sentiment scores:

	VideoID	Video_title
0	LvcSgeHG3FQ	All 12 Members of Red Hair Pirates, Explained!
1	xUUU3Jp9Fr0	All 10 of Zoro Swords in One Piece, Explained
2	D7riuDqvkxQ	The Great Power (Intensity) that Daemon Felt i...
3	6GmXu3SmrYU	Explaining detail about Humphrey Bogard (Garp'...
4	RBhiarwF9SI	All 4 Basic and Advanced Techniques from Armam...
5	hKkG94i3SDQ	4 Secrets that Robin Hides from Luffy
6	chXx6jyrCtE	The Real Connection Between Monkey D Dragon an...
7	1yCAhNApoLM	Finally! All 12 Devil Fruit Users in Blackbear...
8	QtZfA4llGOM	5 Artificial Devil Fruits Created by Vegapunk
9	xaGpzEW5um8	All 15 Members of Revolutionary Army and Their...
10	-0Ia6yeyvvk	The Real Reason Why Makima Killed Power, EXPLA...
11	kk0zaArGfUk	Benn Beckman who cut Eustass Kid's Left arm
12	Y5JXrbS_LCI	STRAW HAT PIRATES IN THE FUTURE #onepiece #shorts
13	ie88Iqbrya4	STRAW HAT PIRATES VS BLACKBEARD PIRATES #onepi...
14	LMZopU57m5w	How Strong is Garp's Galaxy Impact?

WordCloud:

We can observe from the above word cloud that Luffy, Shanks, and Roger are the characters that are most discussed by the viewers. Luffy is the main character of the anime. Roger was the person who previously achieved the goal Luffy is trying to reach which is to become a pirate king. Shanks is also Luffy's mentor and one of the most hyped-up characters in the series. So, it makes sense to make more videos related to these characters.

Models Comparison:

By comparing the accuracies achieved by various modes in classifying sentiments we can see that in our case which has limited amount of comments data obtained through API call. The Random Forest classifier got the highest accuracy and the Neural network based model LSTM has the lowest accuracy due to the limitation of the size of data.