**CPS525 Final Project Paper**

**Reddit Sentimental analysis using Text Mining**

**Team members:** Yasaswi Sai Niharika Avula(avula1y), Charan Sai Guntupalli(guntu1c), Lakshmi Sudha Aluru(aluru1l)

Department of Computer Science, Masters Studies

**avula1y@cmich.edu, guntu1c@cmich.edu**, **aluru1l@cmich.edu**

**Abstract** Reddit Sentimental Analysis is a program used to analyze users' sentiments related to different issues, as Reddit is a social news aggregation and discussion website that provides a ton of data to explore. By analyzing the sentiment of words from specific posts, we can determine whether a piece of writing is positive, negative or neutral.

**Introduction**

Reddit is categorized into smaller communities called subreddits. A subreddit is simply a board devoted to specific topics like memes, food, travel, entertainment etc. Users are the building blocks of Reddit, they join these communities and submit their thoughts and experience. Other users then show their sentiments via comments on those thoughts. Here we use Sentimental Analysis to understand the subreddit polarity on different topics.

**Dataset Description**

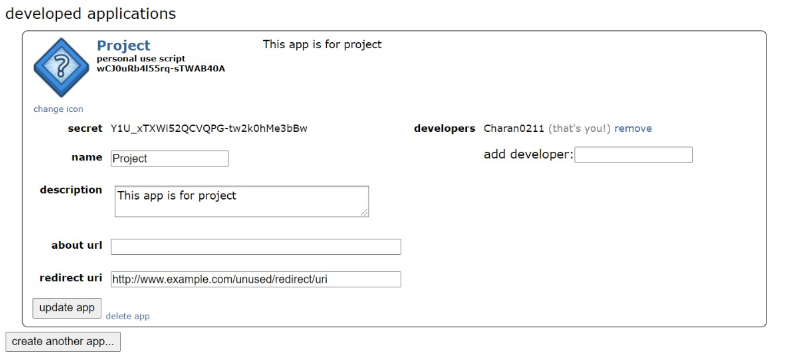
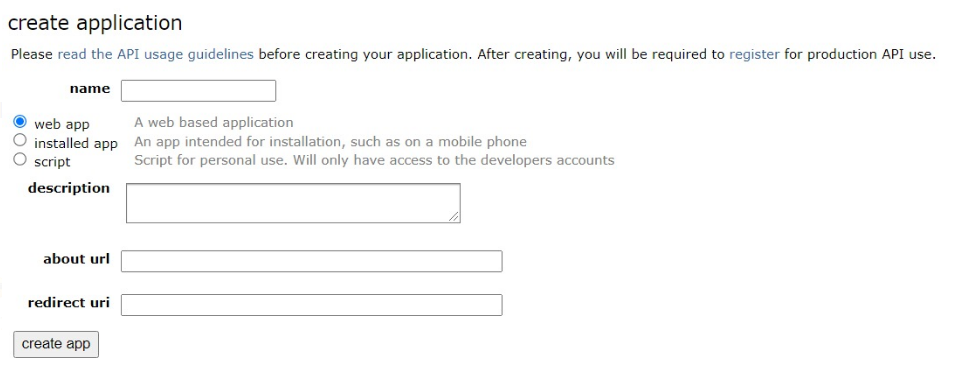
We are applying the sentimental analysis to the retrieved subreddit comments. These comments had been collected from a selected Reddit post using Data Scraping. It is also known as web scraping or data extraction where we automatically extract the data from the websites.

**Tool used for accessing Reddit API**

For this We use Praw(Python Reddit API Wrapper), a Python library, to interact with Reddit API.

PRAW is an easy to use and fun module to start collecting data from Reddit. And also follows all of Reddit's API rules, In which it requires the credentials like Client ID and client Secret to access the Reddit API.

Create a Reddit App,



As Reddit’s API uses OAuth2, a Reddit connection must be established with a client ID and client secret.

How to Connect to Reddit API, As you can see the code bit below we used the created application’s Client ID and Client secret.

ex:

reddit = praw.Reddit(client\_id="wCJ0uRb4l55rq-sTWAB40A",

client\_secret="Y1U\_xTXWi52QCVQPG-tw2k0hMe3bBw",

username="Charan0211",

password="cherry@123",

user\_agent="Project")

libraries used for sentiment analysis:

#Installing All the requirements

! pip install praw #installs Python Reddit API Wrapper

! pip install emoji #installs emoji

import praw #importing praw

import pandas as pd #importing pandas dataframe as pd

from praw.models import MoreComments

from nltk.corpus import stopwords

from nltk.tokenize import RegexpTokenizer

from nltk.stem import WordNetLemmatizer

from nltk.stem import PorterStemmer

from nltk import FreqDist

import emoji #imports emoji

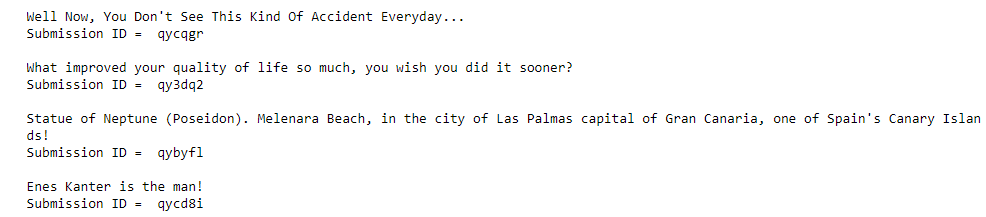
import re #imports re

import en\_core\_web\_sm #imports en\_core\_web\_sm

import spacy #imports spacy

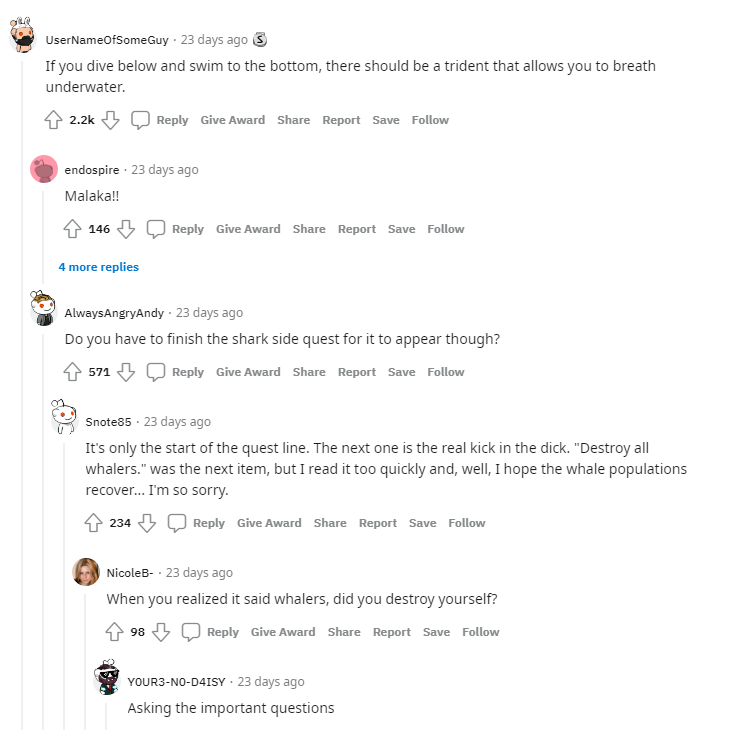
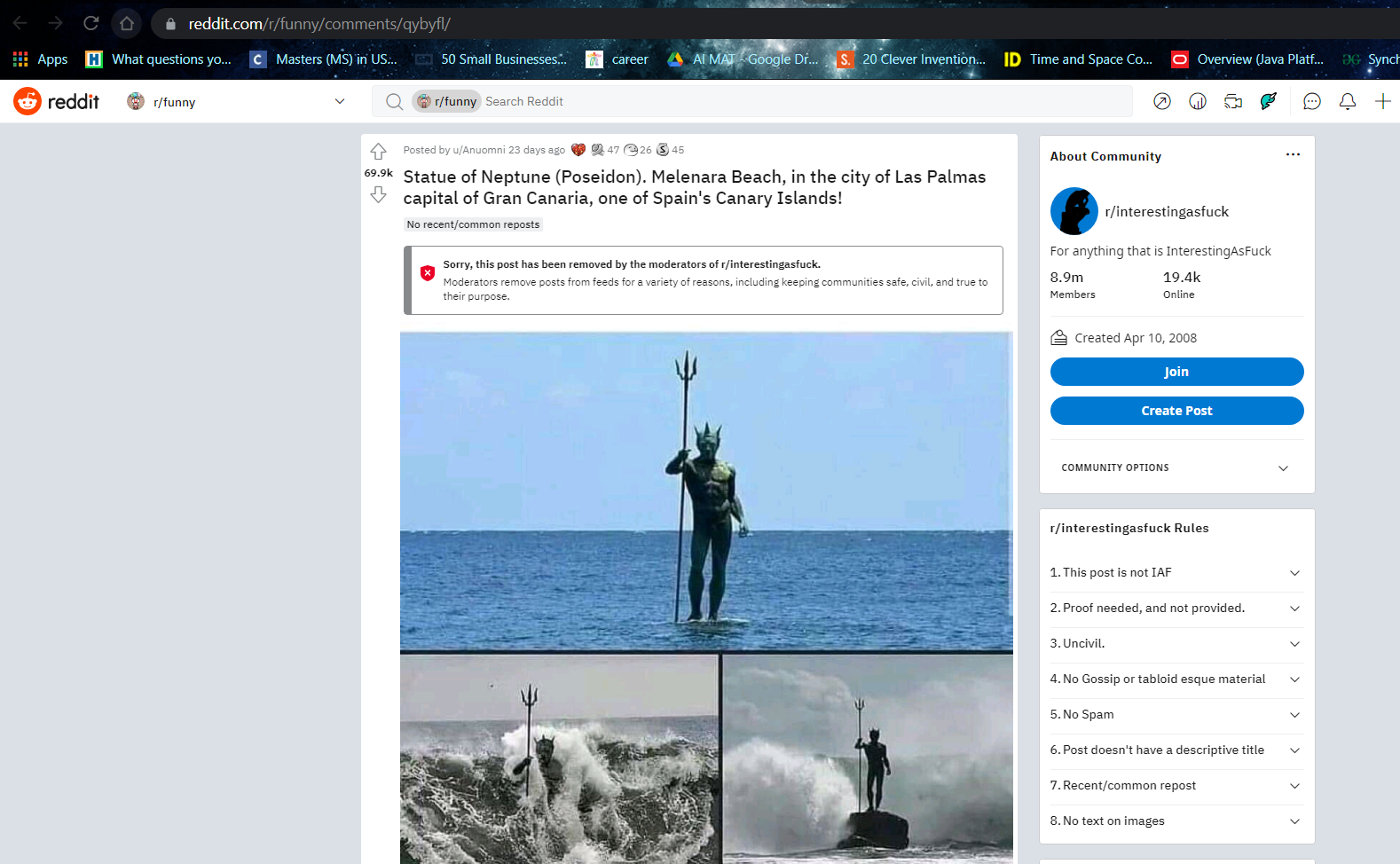
**Obtaining comments from Reddit post**

Here, we retrieve the top trending posts from the /popular category. Where we use the below code to obtain those top posts in /popular category and select one specific post to conduct the sentimental analysis.

Code:

subreddit = reddit.subreddit('popular')

Here we’ve selected a post with submission id: qybyf1



the comments retrieved from the post are stored into a list, instead of a single string - to navigate easily

**Code:**

Comments\_All = []

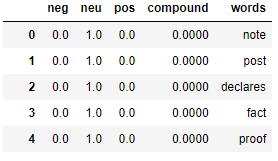
Post1.comments.replace\_more(limit=None)

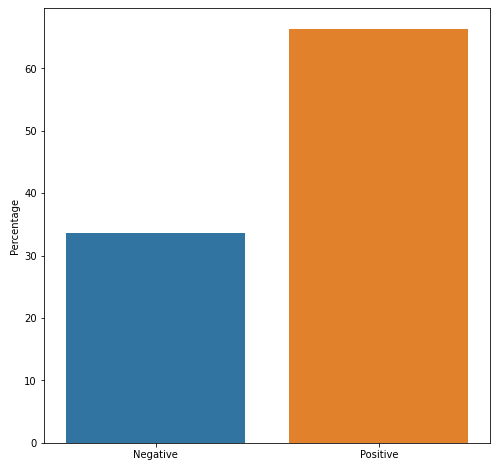
for comments in Post1.comments.list():

Comments\_All.append(comments.body)

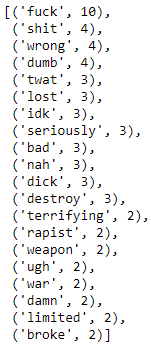
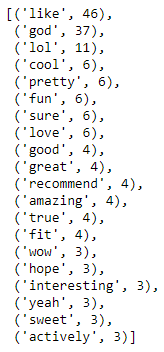
**Data Cleaning**

**1)** Join all the strings separated by a comma  
**Code:**   
string\_uncleaned = ' , '.join(List1)   
**2)**Remove all the emojis  
**Code:**   
string\_emojiless = emoji.get\_emoji\_regexp().sub(u'',string\_uncleaned)   
**3)**Tokenize and clean the strings- breaks the raw text into words, sentences called tokens  
**Code:**   
tokenizer = RegexpTokenizer('\w+|\$[\d\.]+|http\S+')  
tokenized\_string = tokenizer.tokenize(string\_emojiless)  
**4)**Convertings all the strings in to lowercase  
**Code:**  
lower\_string\_tokenized = [word.lower() for word in tokenized\_string]  
**5)**Removing all the stopwords  
**Code:**nlp = en\_core\_web\_sm.load()  
all\_stopwords = nlp.Defaults.stop\_words  
text = lower\_string\_tokenized  
tokens\_without\_sw = [word for word in text if not word in all\_stopwords]  
**6)**Normalizing the words using lemmatization & Stemming  
**Code:**  
lemmatizer = WordNetLemmatizer()  
lemmatized\_tokens = ([lemmatizer.lemmatize(w) for w in tokens\_without\_sw])  
stemmer = PorterStemmer()  
stem\_tokens = ([stemmer.stem(s) for s in tokens\_without\_sw])

After doing the data cleaning, we get   
Original length of words = 30575  
Number of words after removing emojis = 30565  
Number of words after removing tokenizing and cleaning = 5532  
Number of words after removing tokenizing,cleaning and removing stop words = 2759  
Number of words after removing tokenizing,cleaning,removing stop words and lemmatized = 2759  
Number of words for final output = 2759

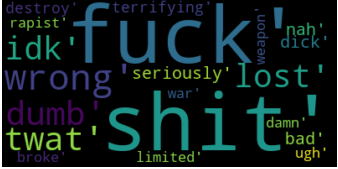
**Applying Sentimental Analyzer:**

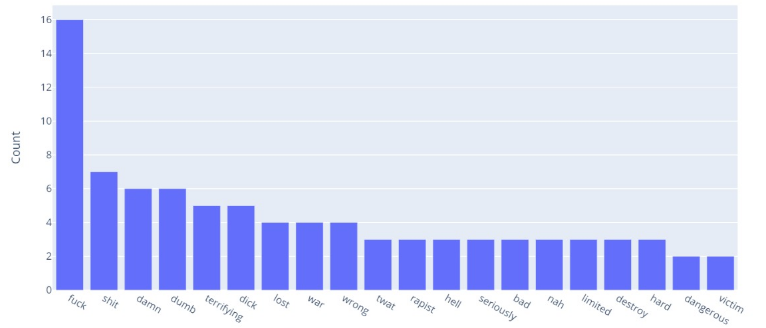
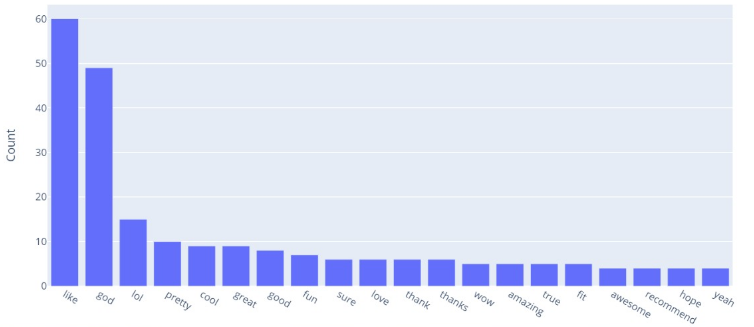
Here we use VADER (Valence Aware Dictionary and sentiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media.To find the polarity of words and analyze a body of text for understanding the opinion expressed by users. Either positive or negative.(we can perform this by nltk.sentiment.vader importing from SentimentIntensityAnalyzer)  
**Code:**   
from nltk.sentiment.vader import SentimentIntensityAnalyzer as SIA  
nltk.download('vader\_lexicon')  
sia = SIA()  
results = []  
for sentences in cleaned\_output:  
 pol\_score = sia.polarity\_scores(sentences)  
 pol\_score['words'] = sentences  
 results.append(pol\_score)  
pd.set\_option('display.max\_columns', None, 'max\_colwidth', None)  
df = pd. DataFrame.from\_records(results)  
Add label to each word,  
**Code:**  
df.loc[df['compound'] > 0.10, 'label' ] = 1  
df.loc[df['compound'] < -0.10, 'label'] = -1  
  
**Representation of Results:  
1)**Plotting the graph for negative and positive words only

**Code:**fig, ax = plt.subplots (figsize=(8, 8))  
counts = df\_positive\_negative.label.value\_counts(normalize=True) \* 100  
sns.barplot(x=counts. index, y=counts, ax=ax)  
ax.set\_xticklabels (['Negative' ,'Positive'])  
ax.set\_ylabel ("Percentage")

**2)** Frequency distribution of the 20 most common Positive and negative words:

**Code:**  
positive\_frequency = FreqDist(positive\_words)  
pos\_freq = positive\_frequency.most\_common(20)  
negative\_frequency = FreqDist(negative\_words)  
neg\_freq = negative\_frequency.most\_common(20)

**3)Visualization using WordClouds**  
Technique to show which words are most frequent among the given text. Basically it performs high level analysis and visualization of text data then gives a glance into the most important keywords in the data .  
**Code:**from wordcloud import WordCloud  
import matplotlib.pyplot as plt  
wordcloud\_positive = WordCloud (background\_color= 'white').generate(Pos\_words\_string)  
wordcloud\_negative = WordCloud().generate (Neg\_words\_string)  
plt.imshow(wordcloud\_positive, interpolation='bilinear')# Display the generated image for Positive words  
plt.axis ("off")  
plt.show()  
plt.imshow(wordcloud\_negative, interpolation='bilinear')# Display the generated image for Negative words  
plt.axis( "off")  
plt.show()

**4)Bar Chart display**! pip install plotly  
import plotly.express as px  
pos\_freq\_df = pd.DataFrame(pos\_freq)  
pos\_freq\_df = pos\_freq\_df.rename(columns = {0: 'Bar Graph Of Frequent Words', 1: 'Count'}, inplace = False)  
fig = px.bar(pos\_freq\_df, x= "Bar Graph Of Frequent Words" , y= "Count", title = "Commonly Used Positive Words By Count")  
fig.show()  
neg\_freq\_df = pd.DataFrame(neg\_freq)  
neg\_freq\_df = neg\_freq\_df.rename(columns = {0: 'Bar Graph Of Frequent Words', 1: 'Count'}, inplace = False)  
fig = px.bar(neg\_freq\_df, x= "Bar Graph Of Frequent Words" , y= "Count", title = "Commonly Used Negative Words By Count")  
fig.show()  


First bar chart shows the 20 most common positive words by count and the second bar chart shows the 20 most common negative words by count.

**Conclusion :**

By applying the Vader sentimental analyzer to the data retrieved post, we got more positive words than negative one’s. Which we conclude that the post likely to impact the users in more positive way.