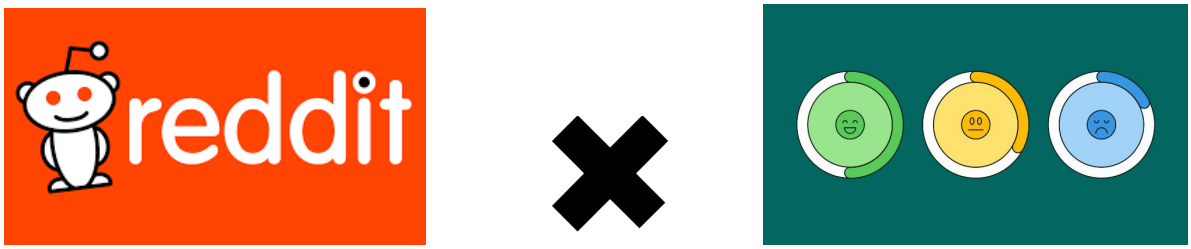

Sentiment Analysis Using Reddit for Brand Management

Introduction to Computational Sociology (SOC471)
Prof. Pradeep Swarnakar

Submitted by :

Gaurav Raj (200378)
Harsh Jain (200412)
Harsh Topno (200421)
Gavish Garg (200385)

Codes have been attached with the email to the instructor

1. Aim **3**
2. Choice of platform **4**
3. Setting up the working environment and installing dependencies **5**
4. API : PRAW (Python Reddit API Wrapper) **6**
5. Before Data Collection : Setting up a developer account **7**
6. Data Preprocessing **8**
a. Remove specific characters : 8
b. Remove stopwords : 8
c. Preprocessing and Tokenization : 9
7. Performing Sentiment Analysis **10**
a. DistilBERT Overview: 10
b. Hugging Face Transformers Integration: 10
c. Sentiment Analysis Process: 11
● Fitting the model 13
● Classifying sentiments 14
● Defining Key Metrics 14
● Visualizing results for a particular case 16
8. Customer Journey **18**
9. Challenges **18**
10. Scope of Improvements **18**

1. Aim

Big data has emerged as a transformative force in the realm of information technology, utilizing complex datasets that traditional data processing applications struggle to handle. With the exponential growth of digital information, industries and organizations are increasingly leveraging big data analytics to extract valuable insights, make data-driven decisions, and gain a competitive edge. From healthcare and finance to marketing and logistics, the applications of big data are diverse and far-reaching.

One particularly fascinating area of application is sentiment analysis, where massive amounts of textual data from platforms like Reddit and Twitter are analyzed to gauge public opinions and emotions. The real-time nature and sheer volume of user-generated content on these platforms make them rich sources for understanding sentiment trends. By using sentiment analysis through machine learning algorithms, businesses can make informed decisions about their perceived public image. As we continue to delve into the era of big data, the fusion of technology and data-driven insights holds immense potential for shaping our understanding of human sentiment and behavior.

We aim to utilize Reddit, a popular social platform that encourages discussions on a wide range of topics, including politics, science, finance and technology, to name a few, which are organized on “Subreddits”, which are platforms for specific topics within Reddit. To say that reddit covers almost everything known to society, in a peer-to-peer, mass collaboration would be an understatement. We hence use this platform to conduct sentiment analysis for Brand Management, which can help PR divisions of firms monitor their online footprints in a timely fashion, and take potential steps after identifying any lags.

2. Choice of platform

Reddit was chosen over twitter for this developing this product because of the following properties :

1. Contextual Depth: Reddit is known for longer and more in-depth discussions compared to the character-limited nature of Twitter. Reddit users often provide more context and detail in their posts, allowing sentiment analysis algorithms to better understand the nuances of opinions and emotions.

2. Subreddit Specificity: Reddit is organized into subreddits, which are communities focused on specific topics. This segmentation allows for more targeted sentiment analysis based on the particular interests or industries relevant to a user's needs. Twitter, while having hashtags, may not provide the same level of granularity.

3. Thread Structure: Reddit discussions are organized into threads, with replies and comments forming a structured conversation. This structure can be advantageous for sentiment analysis

algorithms to analyze the flow of sentiment over a conversation, identifying key sentiments and their evolution.

4. Pseudonymity and Longer User History: Reddit users often have longer posting histories and may use pseudonyms, allowing for a more extended analysis of a user's sentiments over time. Twitter, with its character limit and real-time nature, might provide a more fleeting snapshot of sentiment.

5. Community Consistency: Each subreddit on Reddit tends to have a more consistent user base, which can be beneficial for sentiment analysis. This consistency allows for a more stable understanding of the sentiment within a specific community compared to the broader and more diverse user base on Twitter.

6. Less Noise from Automated Accounts: While both platforms are susceptible to automated accounts (bots), Twitter may have a higher prevalence. Reddit's community moderation and structure may mitigate some of the noise generated by automated accounts, contributing to a potentially more authentic dataset for sentiment analysis.

3. Setting up the working environment and installing dependencies

```
import praw
from transformers import pipeline, DistilBertTokenizerFast,
TFDistilBertModel
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
import re
import string
import matplotlib.pyplot as plt
import datetime
```

1. `import praw`: Imports the PRAW (Python Reddit API Wrapper) library, which facilitates interaction with the Reddit API.

2. `from transformers import pipeline, DistilBertTokenizerFast, TFDistilBertModel`: Imports the Hugging Face Transformers library, including pre-trained models for natural language processing tasks like sentiment analysis, and the DistilBERT tokenizer and model.

3. `from nltk.corpus import stopwords`: Imports the Natural Language Toolkit (NLTK) library's stopwords module, which provides a list of common English stopwords for text processing.

4. `from nltk.tokenize import word_tokenize`: Imports the word_tokenize function from the NLTK library, which tokenizes text into individual words.

5. `from nltk.stem import PorterStemmer`: Imports the PorterStemmer from NLTK, which is used for stemming words (reducing words to their root form) in text processing.

6. `from nltk.stem import WordNetLemmatizer`: Imports the WordNetLemmatizer from NLTK, which is used for lemmatizing words (reducing words to their base or dictionary form) in text processing.

7. `import re`: Imports the regular expression (regex) module, allowing the use of regular expressions for string manipulation and pattern matching.

8. `import string`: Imports the string module, which provides a collection of string constants and functions for string manipulation.

9. `import matplotlib.pyplot as plt`: Imports the Matplotlib library for data visualization, specifically the pyplot module for creating plots and charts.

10. `import datetime`: Imports the datetime module, providing classes for working with dates and times.

These import statements set the stage for using various functionalities and tools throughout the script, enabling interactions with Reddit, applying natural language processing, and visualizing results.

4. API : PRAW (Python Reddit API Wrapper)

PRAW, as a Python library, streamlines the process of extracting data from Reddit, offering a user-friendly interface for developers and researchers alike. With its well-documented functions and easy integration into Python scripts, PRAW empowers users to programmatically access

posts, comments, and other relevant data from Reddit, facilitating in-depth sentiment analysis.Utilizing the PRAW (Python Reddit API Wrapper) for sentiment analysis on Reddit provides a robust and efficient means of accessing the vast reservoir of user-generated content on the platform.

In the context of sentiment analysis, PRAW enables the extraction of textual data that serves as the foundation for understanding user opinions and emotions. By accessing individual posts or entire threads within specific subreddits, analysts can gather a wealth of contextual information. The PRAW API's versatility allows for the customization of queries, enabling users to focus on particular topics, time frames, or communities of interest. This flexibility is particularly valuable when tailoring sentiment analysis to niche subjects or when exploring sentiment trends over specific periods.

Moreover, PRAW provides features to navigate through comments, replies, and user profiles, facilitating a comprehensive understanding of the sentiment landscape. The ability to retrieve historical data and track sentiment changes over time enhances the analytical depth, offering insights into evolving opinions within Reddit communities. PRAW's adherence to Reddit's API guidelines ensures ethical and responsible data extraction, fostering a positive relationship between developers and the platform.

In conclusion, leveraging the PRAW API for sentiment analysis on Reddit equips analysts with a powerful toolset to navigate the vast sea of user-generated content. Its simplicity, customization options, and compatibility with Python make it an ideal choice for researchers seeking to extract meaningful insights from Reddit's diverse and dynamic discussions. The subsequent sections of this report will delve into the methodology, challenges, and outcomes of employing PRAW for sentiment analysis on Reddit, shedding light on the valuable contributions this API makes to the field of data-driven analysis.

The following is one instance of using the PRAW API for the purposes of our Sentiment Analysis. Extensive documentation for the PRAW library can be found here.

```
reddit_obj = praw.Reddit(
    client_id=client_id,
    client_secret=client_secret,
    user_agent=user_agent,
    cache_timeout=0  # Disable caching
    )
```

5. Before Data Collection : Setting up a developer account

Setting up a developer account is a crucial step for accessing PRAW. Establishing a developer account ensures that users can make authenticated requests, adhering to the API provider's guidelines and restrictions. For PRAW, this typically involves creating a Reddit developer account and registering an application, which grants access to the required API keys. This process not only serves as a means of authentication but also allows developers to monitor their API usage and, in some cases, access additional features or higher rate limits. By obtaining API keys through a developer account, users contribute to responsible and ethical data extraction, respecting the terms of service of the platform. Moreover, developer accounts often come with documentation and support resources, providing users with valuable insights into best practices, troubleshooting, and any updates to the API.

6. Data Preprocessing

Preprocessing raw data extracted from the Reddit API involves several steps, all of which are enumerated below along with their codes :

a. Remove specific characters :

```
# Function to clean text
def clean_text(text):
    text = re.sub(r"[@#]", "", text)
    text = re.sub(f"[{string.punctuation}]", "", text)
    text = text.lower()
    text = re.sub(r"\d+", "", text)
    return text
```

Input	Raw text of a Reddit post.
Output	Cleaned text with specific characters removed.
Function	Removes Twitter-specific characters (such as '@' and '#') and other punctuation, converts text to lowercase, and removes digits from the text.

b. Remove stopwords :

```
def remove_stopwords(text):
    stopwords_list = set(stopwords.words("english"))
    words = text.split()
    filtered_words = [word for word in words if word not in stopwords_list]
    filtered_text = " ".join(filtered_words)
    return filtered_text
```

Input	Cleaned text
Output	Text with common English stop words removed.
Function	Uses NLTK's stopwords list to remove common words (e.g., 'and', 'the', 'is') from the text to focus on more meaningful content

c. Preprocessing and Tokenization :

```
def preprocess_posts(posts):
    preprocessed_data = []
    for post in posts:
        cleaned_text = clean_text(str(post))
        filtered_text = remove_stopwords(cleaned_text)
        encoded_chunks = []
        for j in range(0, len(filtered_text), 512):
            chunk = tokenizer.encode(
                filtered_text[j : j + 512], truncation=True, max_length=512
            )
            encoded_chunks.extend(chunk)
        preprocessed_data.append(encoded_chunks)
    return preprocessed_data
```

Input	List of Reddit posts
Output	List of preprocessed and tokenized data
Function	Iterates through each Reddit post, cleans and removes stopwords from the text,

	and then tokenizes the text using DistilBERT tokenizer. It also handles encoding the text in chunks to fit within the model's maximum sequence length, which is 512

7. Performing Sentiment Analysis :

a. DistilBERT Overview:

- BERT Architecture: DistilBERT (Distill + BERT) is a compact version of the BERT (Bidirectional Encoder Representations from Transformers) model. BERT, introduced by Google, is a transformer-based architecture that processes words bidirectionally in a contextualized manner, capturing intricate dependencies in language.
- Distillation Process: DistilBERT is distilled from the original BERT model using a knowledge distillation process. This process involves training a smaller model (DistilBERT) to mimic the behavior of a larger model (BERT) while using a fraction of the parameters. This results in a more lightweight yet effective model for various NLP tasks.

b. Hugging Face Transformers Integration:

- Pipeline for Sentiment Analysis: the line pipeline("sentiment-analysis") sets up a sentiment analysis pipeline using Hugging Face Transformers. This pipeline simplifies the process of using pre-trained models for sentiment analysis without the need for extensive coding.
- Tokenization: DistilBertTokenizerFast.from_pretrained("distilbert-base-uncased") loads the DistilBERT tokenizer, which breaks down input text into tokens. Tokenization is a critical step in preparing text for analysis, converting it into a format suitable for input into the model.
- Model Loading: TFDistilBertModel.from_pretrained("distilbert-base-uncased") loads the DistilBERT model itself. This pre-trained model has learned representations of words and their context from vast amounts of text data.
- Sentiment Prediction: The pipeline applies the loaded DistilBERT model to the tokenized input text, predicting sentiment labels and associated confidence scores. The model has been trained on sentiment-labeled datasets, allowing it to generalize and make predictions on new text inputs.

c. Sentiment Analysis Process:

- Input Preprocessing: Before sending text for sentiment analysis, the script preprocesses the Reddit posts by cleaning the text, removing stopwords, and tokenizing it with the DistilBERT tokenizer.
- Model Inference: The sentiment analysis model, powered by DistilBERT, then processes the tokenized text and produces sentiment predictions. These predictions include labels (positive, negative, neutral) and confidence scores.
- Post-Processing: The script further classifies sentiment based on the confidence scores and specific thresholds, resulting in the final sentiment classifications.

DistilBERT, integrated with Hugging Face Transformers, serves as a powerful tool for sentiment analysis. DistilBERT's knowledge distillation process ensures efficiency and speed in processing, making it suitable for analyzing Reddit's posts. The Hugging Face Transformers library simplifies the usage of such advanced models, allowing developers to leverage state-of-the-art NLP techniques with ease. Most of the processes are illustrated in the figures that follow.

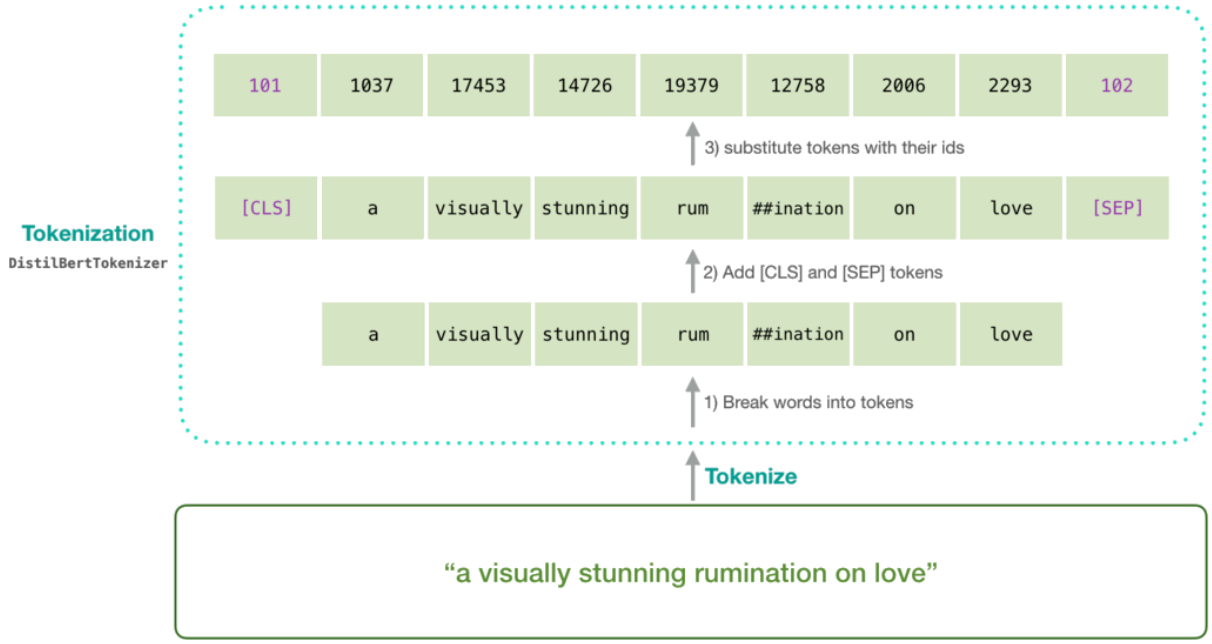

Fig. DistilBERT Tokenizer

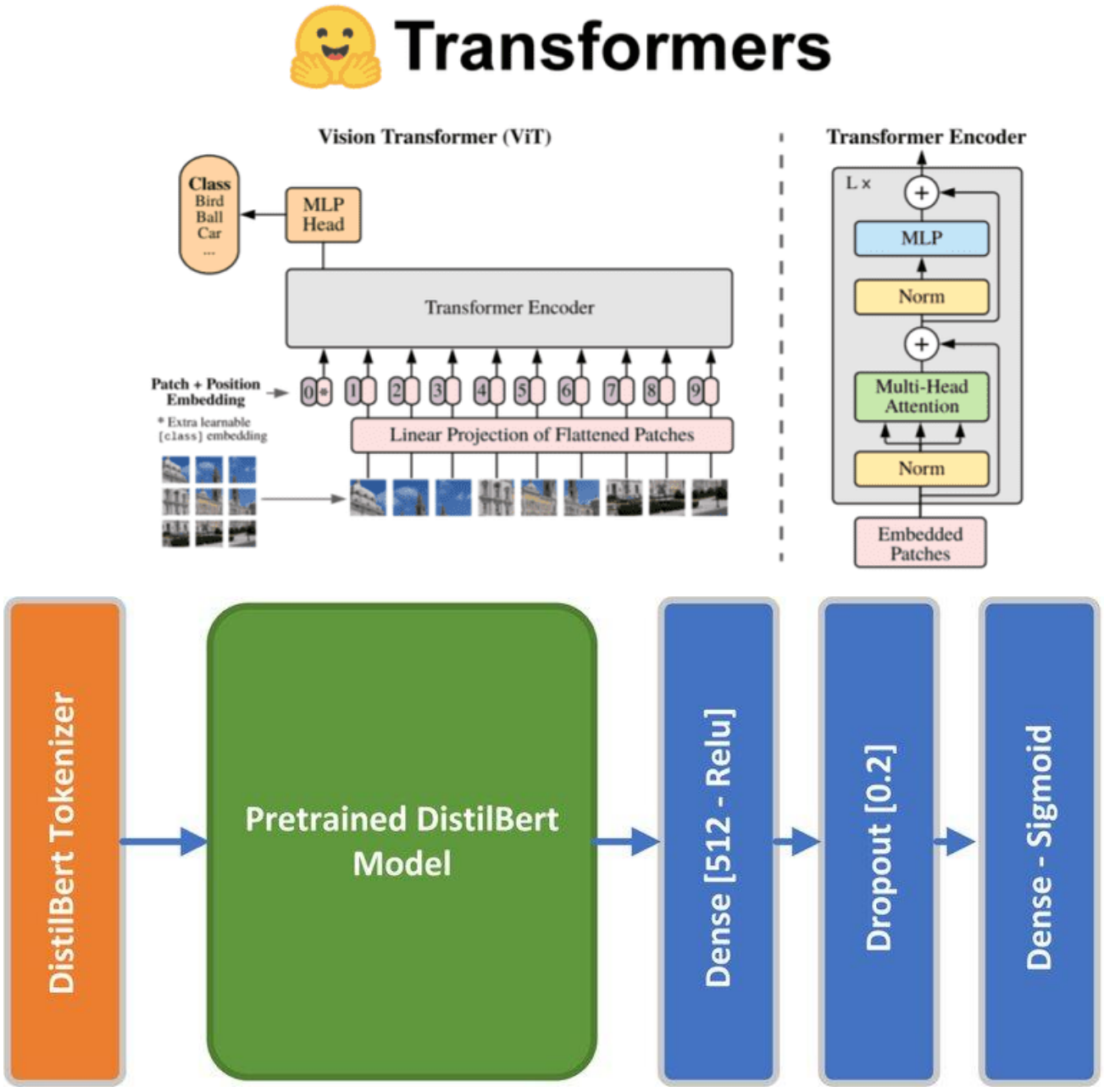

Fig. DistilBERT Architecture

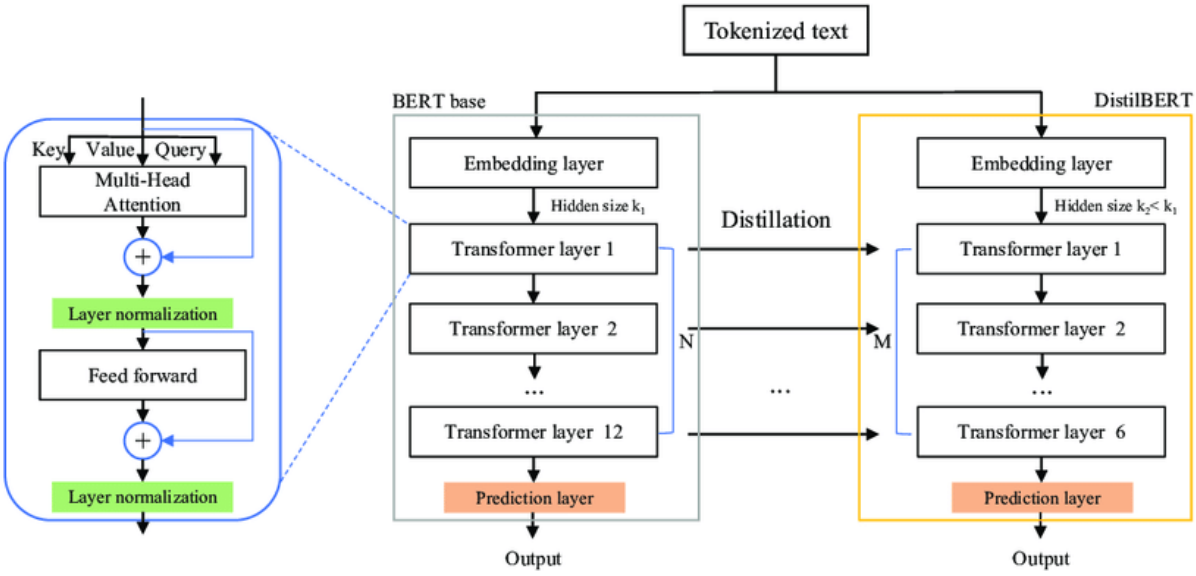

Fig. DistilBERT Architecture

- Fitting the model

```
def perform_sentiment_analysis(texts):
    sentiment_analysis_model = pipeline("sentiment-analysis")
    sentiment_predictions = sentiment_analysis_model(texts)
    return sentiment_predictions
```

Input	List of tokenized texts
Output	Predictions of sentiment scores for each text.
Function	Uses the Hugging Face Transformers library to load a pre-trained sentiment analysis model (DistilBERT-based) and applies it to the list of tokenized texts, providing sentiment predictions and scores from 0 to 1, 1 indicating highest confidence in the predicted label

- Classifying sentiments

```
def classify_sentiments(predictions):
    sentiment_classifications = []
    for prediction in predictions:
        label = prediction["label"]
        score = prediction["score"]
        if label == "POSITIVE" and score >= 0.5:
            sentiment = "positive"
        elif label == "NEGATIVE" and score >= 0.5:
            sentiment = "negative"
        else:
            sentiment = "neutral"
        sentiment_classifications.append(sentiment)
    return sentiment_classifications
```

Input	List of sentiment predictions
Output	List of sentiment classifications (positive, negative, neutral)
Function	Classifies sentiments based on the sentiment scores obtained from the sentiment analysis model. If the score is greater than or equal to 0.5, it is classified as positive or negative; otherwise, it is considered neutral

- Defining Key Metrics

At this stage, we have all our scores and labels with posts for us. We now move on to define some key metrics that will be used to understand the sentiments from a particular subreddit for a particular brand.

```
def calculate_sentiment_metrics(sentiment_classifications,
sentiment_scores):
    total_posts = len(sentiment_classifications)
    positive_posts = sentiment_classifications.count("positive")
    negative_posts = sentiment_classifications.count("negative")
    positive_rate = positive_posts / total_posts
    negative_rate = negative_posts / total_posts
    net_positive_rate = positive_posts / negative_posts if negative_posts >
0 else 0
    net_score = sum(sentiment_scores) / total_posts
    day_sentiment = "Positive" if net_score >= 0 else "Negative"
```

```
    day_score = sum(sentiment_scores) / positive_posts if day_sentiment ==
"Positive" else abs(sum(sentiment_scores)) / negative_posts
    sentiment_metrics = {
        "total_posts": total_posts,
        "positive_posts": positive_posts,
        "negative_posts": negative_posts,
        "positive_rate": positive_rate,
        "negative_rate": negative_rate,
        "net_positive_rate": net_positive_rate,
        "net_score": net_score,
        "day_sentiment": day_sentiment,
        "day_score": day_score
    }
    with open(f"{brand_keyword}_sentiments.csv", 'w', newline='') as
csvfile:
        writer = csv.DictWriter(csvfile,
fieldnames=sentiment_metrics.keys())
        writer.writeheader()
        writer.writerow(sentiment_metrics)
    return sentiment_metrics
```

Input	List of sentiment classifications, list of sentiment scores
Output	Calculates various sentiment metrics and exports them to a CSV file
Function	Calculates various metrics to classify overall sentiment for the Brand

Metric	Formula	Significance
Total Posts	# of posts analyzed	Total number of posts analyzed
Positive Posts	# of positive posts	Posts labeled as positive by the model
Negative Posts	# of negative posts	Posts labeled as negative by the model
Positive Rate	# of posts labeled positive/total # of posts	Shows the proportion of positive posts out of total posts
Negative Rate	# of posts labeled negative/total # of posts	Shows the proportion of negative posts out of total posts

Net Score	Sentiment scores added up for all posts (positive for positive posts and negative for negative posts)	Shows the overall sentiment score of the population for the brand and subreddit analyzed
Day Sentiment	Positive if net score > 0 Negative if net score < 0	Shows the label of the sentiment for the entire day
Day Score	Average of net score	Average of the scores scaled from 0 to 1

These metrics when analyzed together give a picture of the sentiments that have been expressed on top posts for the brand on a particular subreddit. More than one of these metrics can be utilized to understand the trend of sentiments for a particular brand. For example, an increase in the positive rate signifies a greater positive feeling towards the brand and vice versa. An increase in day score measures the growing strength of these sentiments towards this particular brand. It must be noted that the results of more than one subreddit can be aggregated to come to a conclusion about the overall sentiment and trend of these sentiments regarding the brand.

- Visualizing results for a particular case

```
def generate_sentiment_chart(subreddit_name, brand_keyword, date,
sentiment_scores, sentiment_classifications):
    # Convert date to string
    date_str = date.strftime("%Y-%m-%d")

    # Plotting
    colors = []
    for sentiment in sentiment_classifications:
        if sentiment == "positive":
            color = "green"
        elif sentiment == "negative":
            color = "red"
        else:
            color = "blue"

        colors.append(color)

    plt.figure(figsize=(10, 5))
    plt.scatter(range(len(sentiment_scores)), sentiment_scores, c=colors,
marker="o")
```

```
    plt.title(f"Sentiment Analysis for {brand_keyword} in {subreddit_name}
on {date_str}")
    plt.xlabel("Post number")
    plt.ylabel("Sentiment Score")
    plt.grid(True)
    plt.show()
```

Input	Subreddit name, brand keyword, target date, sentiment scores, sentiment classifications
Output	Displays a scatter plot of sentiment scores with color-coded points
Function	Uses Matplotlib to generate a scatter plot where each point represents a post, color-coded according to its sentiment classification. The chart provides a visual representation of sentiment distribution within the specified subreddit

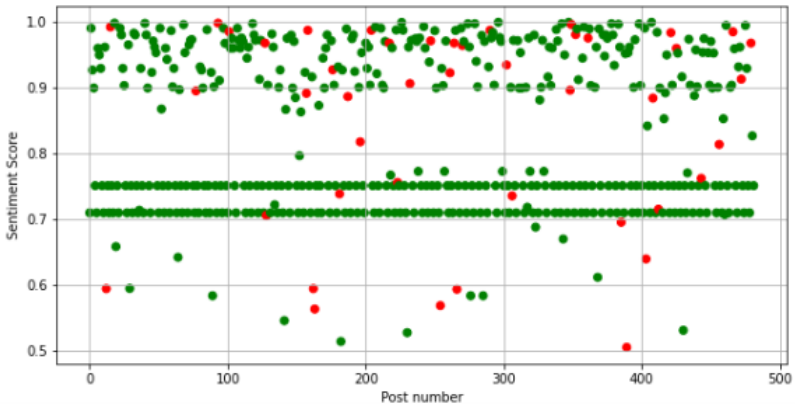

Fig. Sentiment Scores for Zomato (Subreddit : India)

8. Customer Journey

9. Challenges

- Rate limit of Reddit's APIs makes it difficult to analyze posts after a certain number
- The free PRAW version does not allow for filtering directly on the basis of dates and trends must be understood indirectly

10. Scope of Improvements

- Automatizing the aggregation process from various subreddits
- Developing better visualizations on the basis of new constructed metrics
- Developing workarounds for date filters
- Other NLP models and their performance can also be checked and results aggregated to give the best scores.