(Wysa ML Assessment) | Hritik Akolkar | hritikakolkar@gmail.com

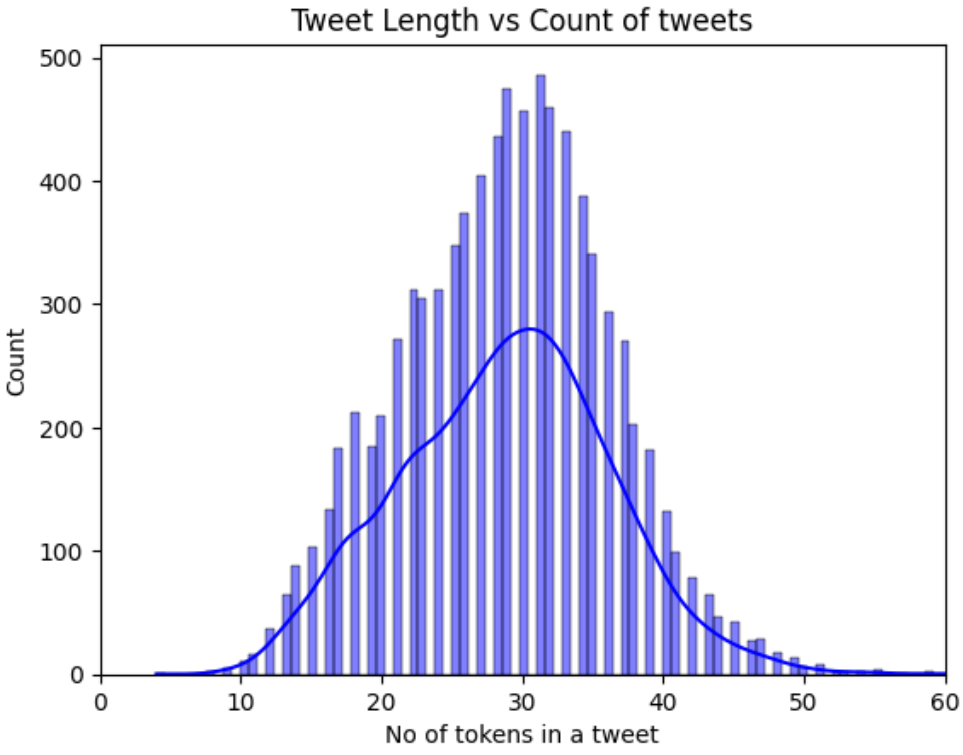
Exploratory Data Analysis

Dataset description	Users assessed tweets related to various brands and products, providing evaluations on whether the sentiment conveyed was positive, negative, or neutral. Additionally, if the tweet conveyed any sentiment, contributors identified the specific brand or product targeted by that emotion.
Columns	tweet_text emotion_in_tweet_is_directed_at is_there_an_emotion_directed_at_a_brand_or_product

Converted the names of columns for ease:
Tweet_text :- **tweet**
Emotion_in_tweet_is_directed_at:- **entity** (this can be a product, brand or service)
Is_there_an_emotion_directed_at_a_brand_or_product:- **emotion**

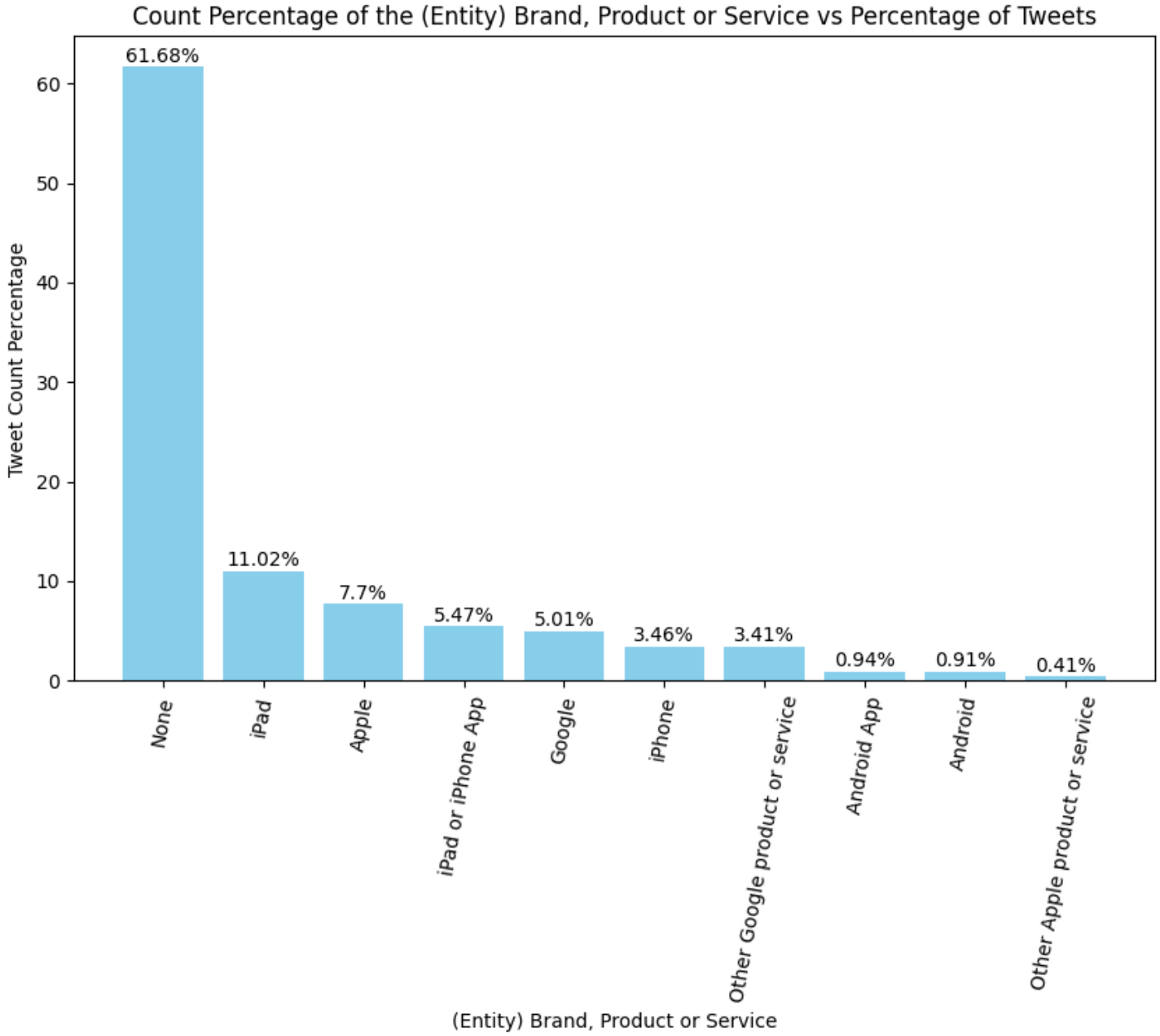
Tweet Section

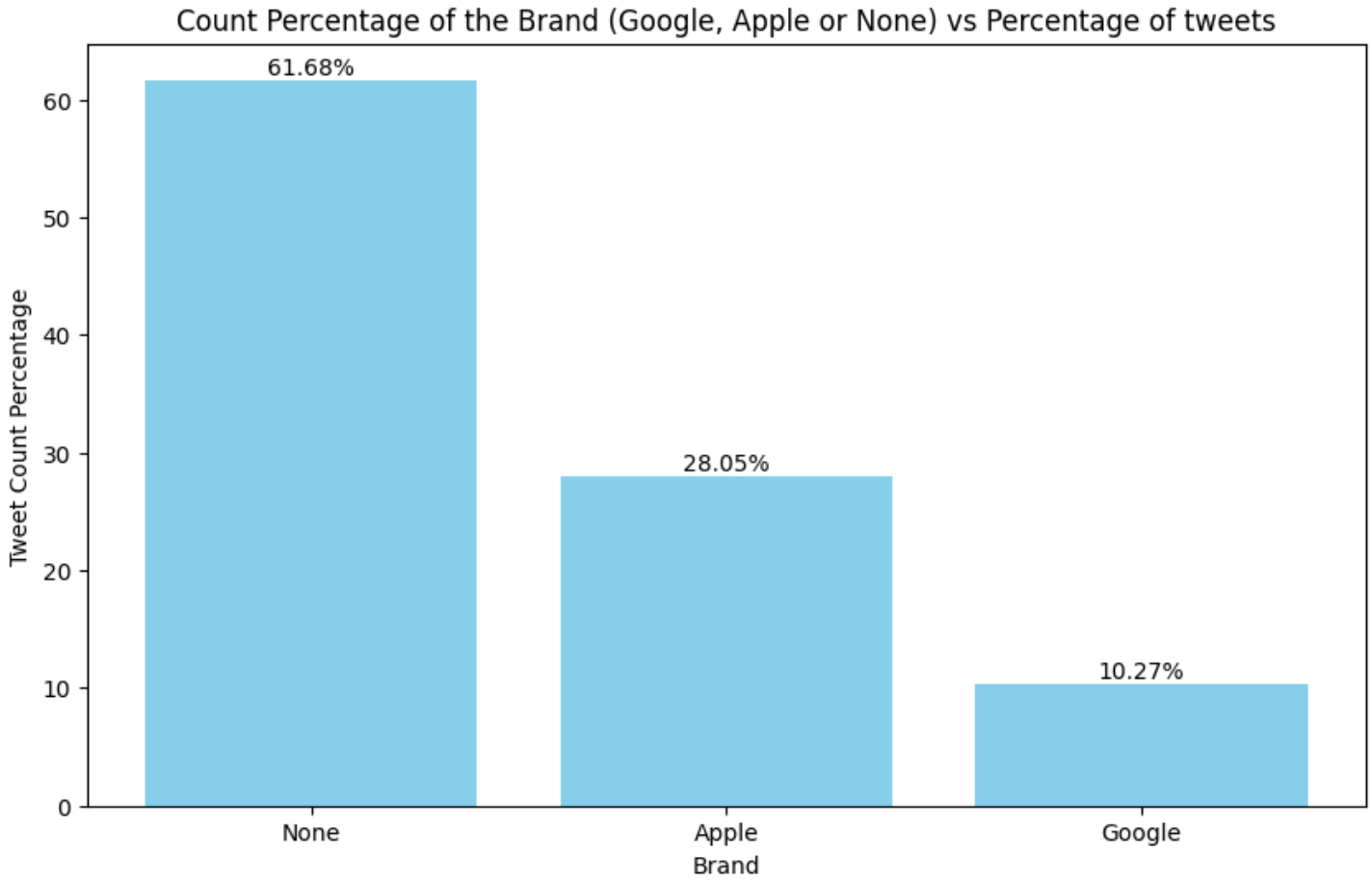
- **Preprocessing**
  a. Removed non-ascii characters like "ऄ⅁ऄऄऄऄ_ऄऄυऄ"
  b. Removed all rows with tweet containing NaN Value
  c. Removed extra spaces
- **Insights**
  a. Looking at the belo plot we can use our max_length to be around 70 while training the model



Entity Section (`emotion_in_tweet_is_directed_at`)

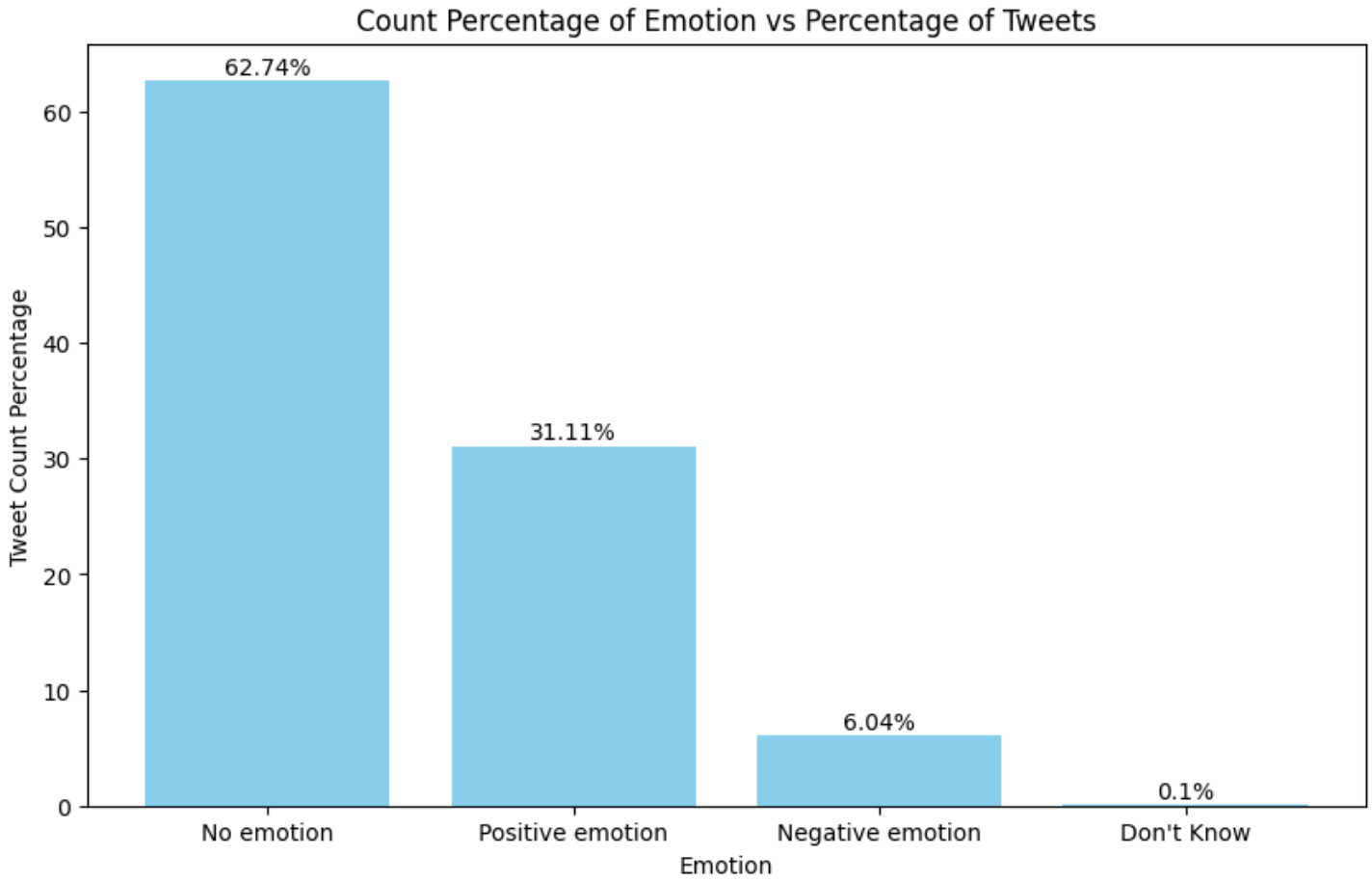
- **Preprocessing**
  - Contains a lot of None value to fill the NaN values we can identify the keywords like Ipad, Iphone, Apple, Google, Android in the corresponding tweet and fill the value I didn't do it because I am not going to use entity anyway
  - Grouped the entities into two brands Apple and Google, please the no of tweets vs brands plots below



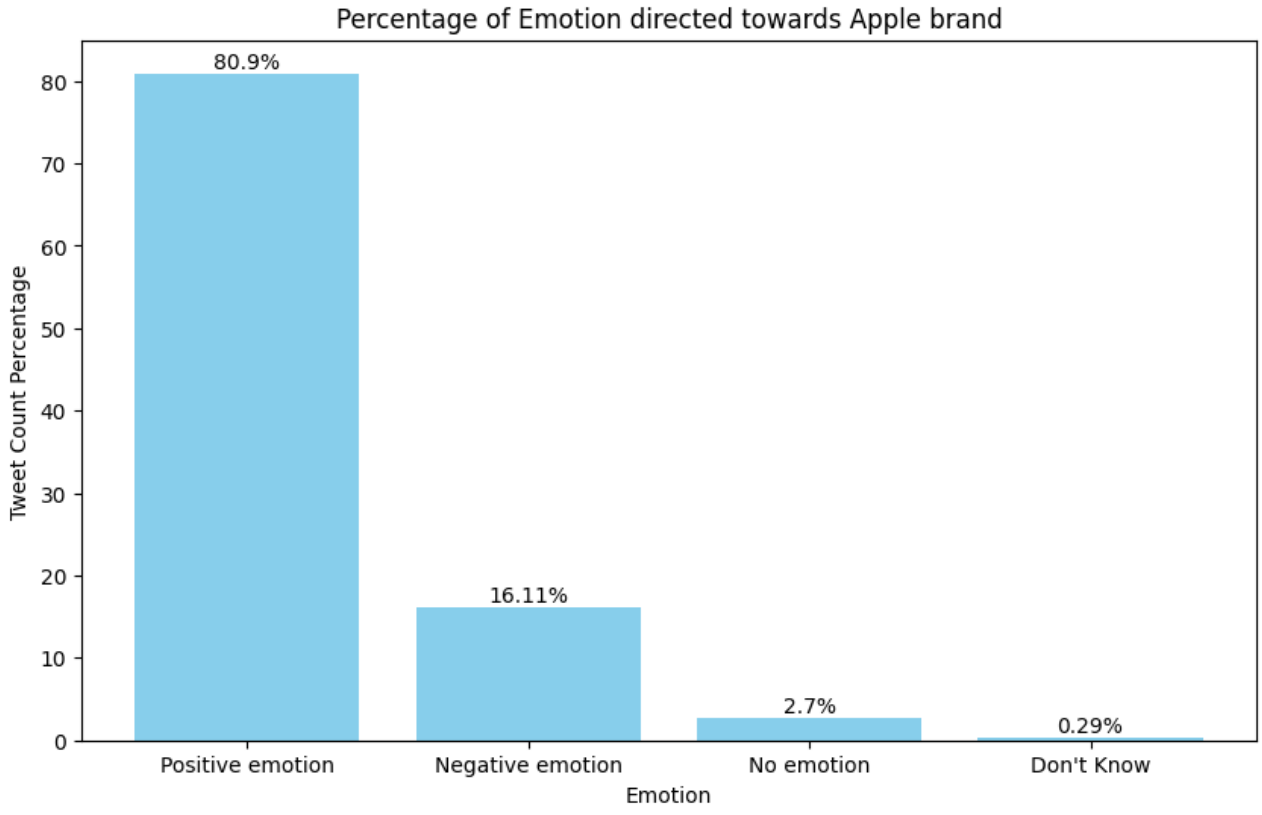


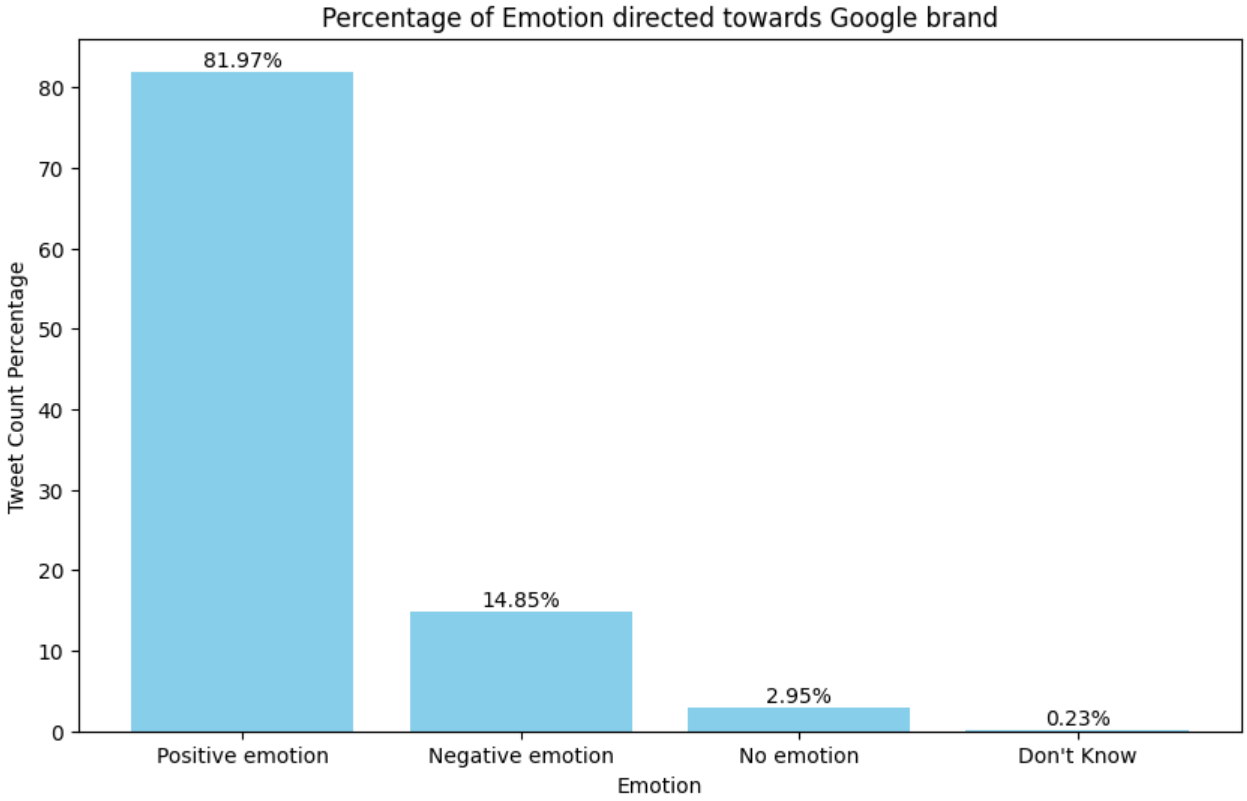
Emotion Section (`is_there_an_emotion_directed_at_a_brand_or_product`)

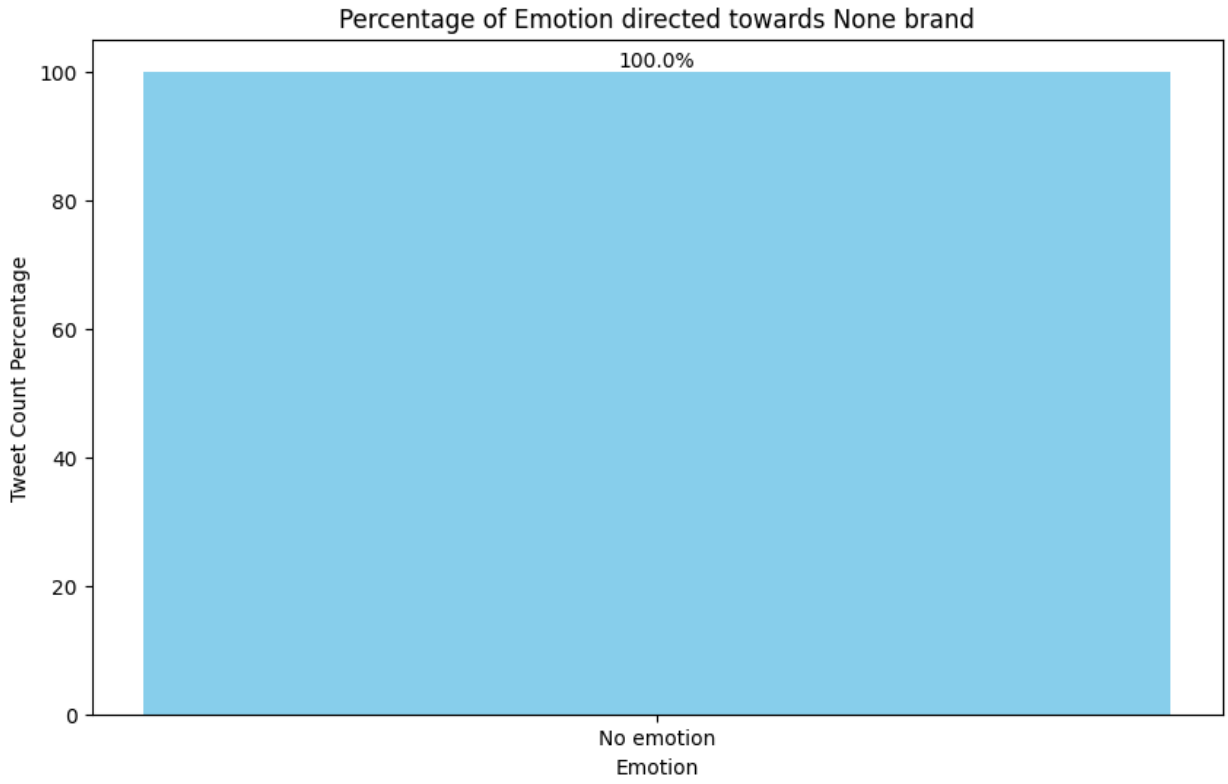
- **Preprocessing**
  - For training Combined the I can't tell with No emotion because it's only 0.1% of the data so we need to either add it or delete it.



Combine Analysis on all fields (tweet, entity, emotion):-



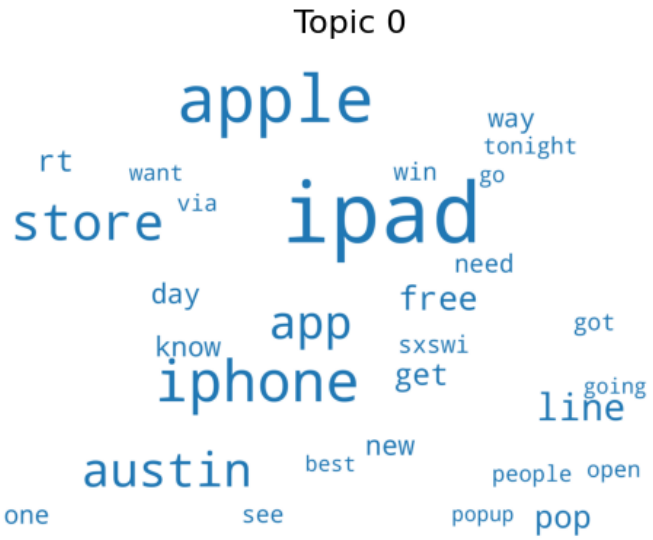




- **Insights**
  - There is not much difference between the emotion of tweets toward brand either it be Google or Apple
  - Almost 94.35% of tweets with no emotion falls under brand for which we don’t have data

Topic Modelling on Tweets to get distribution of words towards a particular topic

- When no of topic = 2, this shows how the words in the tweets are associated and grouped with each other. Hashtag and mentions like #sxsw, @mention and {link} were removed
- We can see below mostly all Apple and Google entities are grouped together.



Data Augmentation and Improvement Suggestions

- ## Data Augmentation
    - Fill NaN values of entity column by looking keywords like ipad, iphone, apple, android, google in the respective tweets. As I am not using entity for predictions so I didn't work on that.
    - Used only three labels (Positive, Negative and No Emotion) instead of four labels in the emotion section (Removed I can't tell label).
- ## Data Improvements
    - While scraping the twitter or using twitter api, we can also take the information of whether tweet was posted using android, or iphone, etc. As this will also help to analysis the sentiment based on what users are using now.
    - In case we have date information of the tweet posted, then we can use that to collect news related to the entity (brand or product) using any news api. And then finetune or use RAG to make our LLM model context aware of the situation, based on that context we can check the sentiment using any prompt (zero shot if works fine) or finetune the LLM model for sentiment classification.

Model Architecture

Encoder based architecture roBERTa (short for "robustly optimized BERT approach")) is used. The pretrained model I choose is Twitter-roBERTa-base for Sentiment Analysis as this model is trained on ~124M tweets from January 2018 to December 2021, and finetuned for sentiment analysis with the TweetEval benchmark on labels (Labels: 0 -> Negative; 1 -> Neutral; 2 -> Positive). Please click on [cardiffnlp/twitter-roberta-base-sentiment-latest] for more info related to pretrained model

We have two cases,

CASE 1 :- Only one label feature i.e emotion

In this case I added a Linear layer for classification of emotion

CASE 2 :- Two label feature i.e. entity, emotion

In this case we can add two Linear layer for classification of emotion and entity, then combine the loss of both for training.

Or

We can add only one layer with num_classes =num_emotion_classes + num_entity_classes

Model Training and Experiment Tracking (wandb)

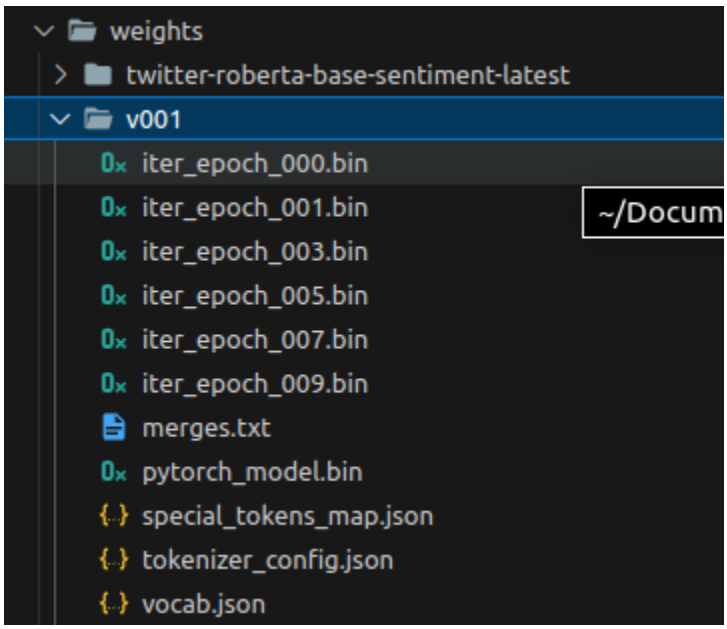
While training model it's a very good practice to track all you experiments I tracked only two - three experiment the major changes was in either using last_hidden_state or pooler_output from the pretained_model in our model architecture and the calculation of Evaluation Metrics using either weighted or macro while evaluating the models.

Proper versioning of models is done like v001, v002 … while experimenting on various things.

The model is overfitting and the things I thought to work on to improving model performance are as follows:

- Bayesian hyperparameter tuning using wandb sweeps, especially working with learning_rate and batch_size.
- Adding some additional Linear Layers for good classification results with working on dropouts more to reduce overfitting

Some examples of storing and managing model training workflows



The iter_epoch models are just for our reference to select the models. pytorch_model.bin is the model saved with best f1 score

Experiment Tracking

Ran multiple experiment with looking to use last hidden state and pooler output didn't work much on hyperparameter tuning as my focus was on model architecture and the calculation of evaluation metrics with average = "macro" or "weighted".

Name (4 visual	Notes	Runtime	average	batch_size	learning_	loss	num_epochs
v004	last_hidden_state with macro	27m 36s	macro	8	0.00002	0.09562	10
v003	last_hidden_state	27m 14s	weighted	8	0.00002	0.1451	10
v002	pooler_output	26m 16s	macro	8	0.00002	0.1765	10
v001	pooler_output	25m 33s	weighted	8	0.00002	0.1028	10

To check the the visualization of training models and comparison between the experiments I created some reports in wandb do check it out

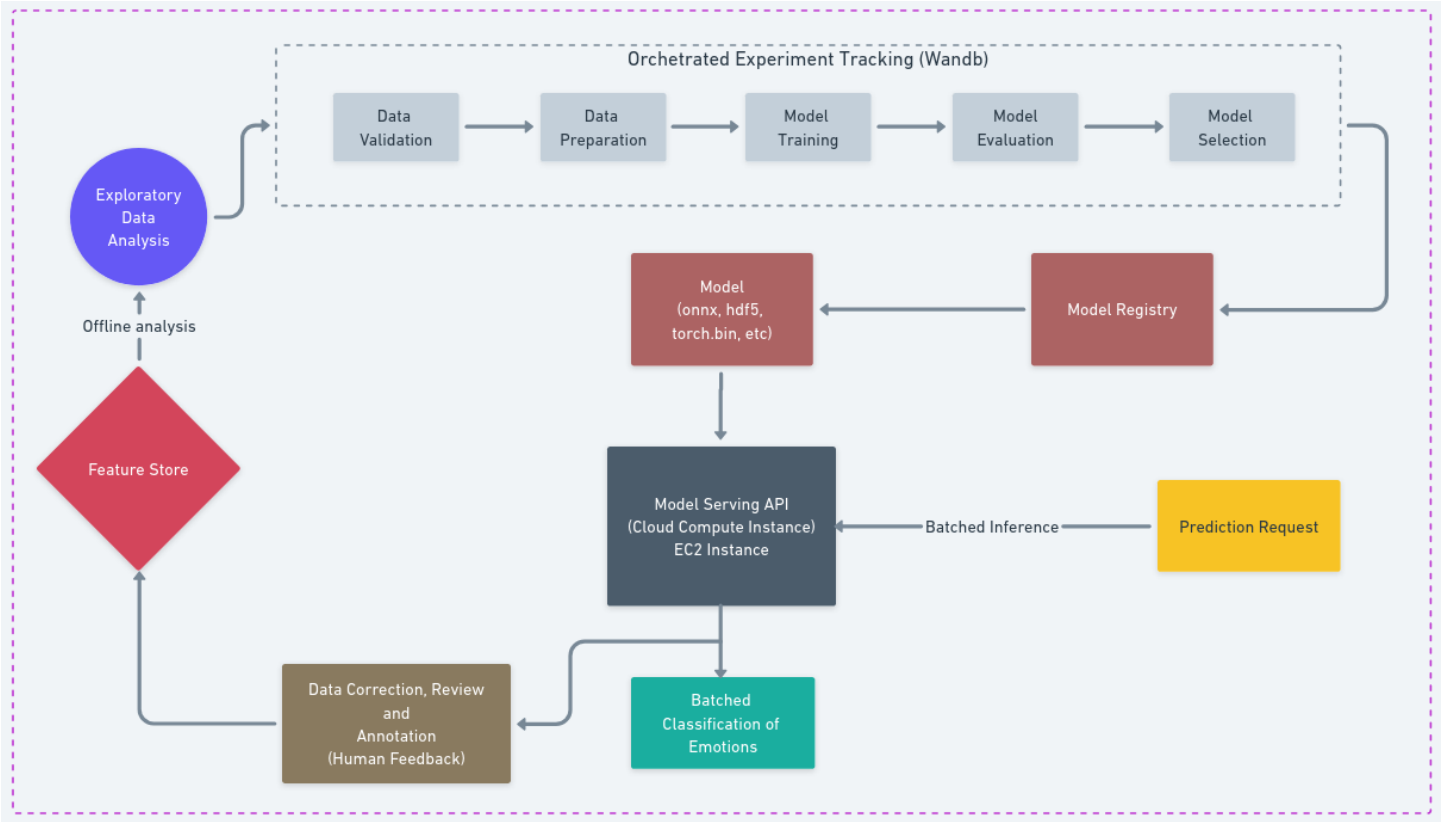
- Multiple Experiment Comparison Report
  https://api.wandb.ai/links/hritikakolkar/avm8upcx
- Fintuned Model Training Report
  https://api.wandb.ai/links/hritikakolkar/70rvxa5h

Evaluation Metrics

As this is the classification problem especially class imbalance so for that reason precision, recall and f1_score are one of the best apart from that we can also use confusion matrix and Precision-Recall Curve.

Deployment

- Requirements
  - EC2 instance with 16 GB RAM and 8 core CPU
  - S3 for model registry
  - Sagemaker Access



Monitoring

- Model Evaluation Metrics
  - Precision
  - Recall
  - F1 Score
  - Accuracy (error rate)
  - Confusion Matrix
  - Precision-Recall Curve

- Model Drift:
    - Metric: Data Drift
    - Description: Monitors changes in the distribution of input data over time.

- Availability:
    - Metric: Uptime
    - Description: Measures the percentage of time the model is available and responsive.

- Model System Performance Metrics
    - Latency
        - Metric: Inference Time
        - Description: Measures the time it takes for the model to process and respond to a single prediction request.
    - Throughput
        - Metric: Requests Per Second (RPS)
        - Description: Measures the number of prediction requests the model can handle per second.

- System Resource Utilization
    - Metric: CPU/GPU/Memory Usage
    - Description: Monitors the utilization of system resources during model inference.

- Logging to check any cause of failure or debugging system failure.