Twitter Sentiment Analysis - Final Deliverable

Problem Statement: Social media platforms contain a vast amount of public sentiment data, crucial for understanding prevailing attitudes. However, the sheer volume makes manual analysis impractical. Using machine learning to categorize sentiment into positive or negative categories can help uncover deeper insights into sentiment dynamics.

Dataset: The dataset from Stanford CS Datasets comprises tweets from April to June 2009, with 1.6 million rows and 6 columns: Polarity, Serial Number, DateTime, No_Query, Username, and Tweet content. This dataset will be used to explore unsupervised machine learning techniques for uncovering inherent sentiment groupings in social media data.

Exploratory Data Analysis (EDA): In our initial EDA phase, we used basic Python functions like head(), isnull(), and describe() to understand the dataset, including its size, missing values, and duplicates. We also created visualizations, including two graphs showing the behavior of the top 10 Twitter users and a plot displaying the distribution of positive vs. negative tweets. Interestingly, the distribution appears even, suggesting possible biases in the data collection method where neutral tweets might be misclassified as positive or negative.

Preprocessing:
Text Cleaning: We converted all text to lowercase for uniformity. We removed non-alphanumeric characters, including punctuation and URLs.
Stop Words Removal: Using NLTK's TweetTokenizer, we split tweets into words, preserving case and removing user mentions and repeated characters. Common English stop words were then removed, improving the quality of our tokenized data.
Stemming and Lemmatization: Stemming and lemmatization were performed to standardize word forms, reduce dimensionality, and enhance the performance of our analysis algorithms.
Including Emojis: Emojis, important for sentiment analysis, were retained, focusing on commonly used 'old school' emojis to capture nuanced sentiments in tweets.
Handling negations: Handling negations is a crucial step as negation words change the meaning of the sentences. These words help to form an accurate interpretation of the sentences.

Phase 1:

Analysis Plan:

In Phase 1, our baseline model for sentiment analysis will be Bag of Words (BoW), which will help us determine the frequency of specific words in the dataset to understand their importance. However, BoW doesn't consider word order or structure, so we'll also use TF-IDF to capture the importance of words in the context of the entire corpus. For a more nuanced understanding of text semantics, we'll employ Word2Vec embeddings to explore contextual meanings and associations of words, which is crucial for capturing sentiment nuances in tweets. Finally, we'll use cosine similarity to quantify the similarity between tweet embeddings derived from Word2Vec, allowing us to identify clusters of tweets with similar sentiment or topics, complementing our BoW and TF-IDF analyses.

Preliminary Results:
In Figure 4 we can see our preliminary ‘WordCloud’ using BoW, it doesn’t seem the most useful for sentiment analysis as we cannot gather any insights related to sentiments of the tweets. We do see words such as ‘love’, ‘like’, ‘good’ - but again, since context was not considered in BoW it is hard to draw any conclusions about the sentiment. Figure 5 is the “WordCloud" for TF-IDF, it is not that different from Figure 4 and as a result, not very helpful.

Next Steps:
We will enhance sentiment analysis by incorporating n-grams and building a classification model since we have the labeled data. N-grams will capture contextual information, improving language structure representation. The labeled dataset will be split into training and testing sets for model training. We'll select a suitable classification algorithm and evaluate the model's performance using metrics like accuracy, precision, recall, and F1-score. Additionally, we'll perform cosine similarity analysis on Word2Vec vectors to refine sentiment analysis of tweets.

Phase 2:
Adjusted Preprocessing: We ran word2vec using a pretrained model, a model that was trained on twitter data. Between the different models we selected the one with 50 dimensions to reduce the computational effort of the model employed later.

Supervised Machine Learning Methods + Analysis:
We took the supervised ML method to classify and explore the results. We used the Random Forest classifier, because amongst all the supervised models it tends to perform the best without needing to be hyperparameter tuned. We trained it using various models

- **Word2Vec**: The results were not that great, we received an F1 score of 0.59. While this is not the result we were expecting, we established that the original classifications of each tweet may not have been completely accurate since a lot of neutral tweets were also given a polarity of either 0 or 1. We tried with a vector size of 50 and 100, but we were able to successfully run the model with only 50 vectors.
- **Pre-trained model**: The results were a tad worse in comparison as the F1 score was 0.58. This came as a surprise since the pretrained model was specifically trained on twitter data, however, one thing to note is that the pre-trained model was trained on recent twitter data and our dataset is from 2009. We hypothesize that a lot of the language used on twitter, including the usage of emojis has evolved a lot since 2009.
- **N-grams**: The results were the best amongst the 3, with a F1 score of 0.73 for ngrams range of n (1,2). We suspect this is because n-grams capture more contextual information than individual words, which can be particularly beneficial for sentiment analysis. We tried to perform n-grams with a range of n (1,3) and (2,3). However, they were computationally way too slow, so we decided to not pursue them anymore. This result is particularly good since this model, unlike the others, was trained with only 1.75% of the data (10% of the data required more resources).

We also tried to run the random forest using TF-IDF but our computational resources were not sufficient to run our model. We employed PCA to reduce dimension to 100 but only about 15% of the total explained variance was kept which drove our decision to drop this model

Sentiment Analysis:
We wrote out sentences full of positive/negative sentiment lexicon and the word2vec and the pretrained models were used to find the similarities between the tweets. At first we gave a sample sentence to understand if our model showed capabilities in distinguishing positive sentiment from negative and then we passed our twitter dataset to it. By doing so, we can understand the contextual meanings of words and their associations, especially in the context of tweets, which often contain informal language and abbreviations. Adding a few words to our sentences of ideal positive and negative words helped improve our model capability. We got an F1 score of 0.529 for the pretrained model and 0.508 for our word2vec model. This score was calculated in comparison to the given label which we know may be misclassified.

Conclusion:
In conclusion, our exploration of sentiment analysis techniques encompassed adjusted preprocessing, supervised machine learning, and sentiment analysis methodologies. While each approach yielded valuable insights, the effectiveness varied across different models. Adjusted preprocessing using a Word2Vec model trained on Twitter data facilitated computational efficiency without compromising semantic understanding. Supervised machine learning, particularly the Random Forest classifier, showcased robust performance, with N-grams proving to be the most successful approach, achieving a decent F1 score of 0.73. However, in sentiment analysis, both Word2Vec and pre-trained models demonstrated only moderate performance, suggesting a need for further refinement and adaptation to capture the nuances of sentiment within Twitter data. Continual improvement and adaptation of methodologies are crucial in navigating the complexities of sentiment analysis in the evolving landscape of social media language. Finally, we observed that concentrating on our ideal positive and negative sentences can significantly enhance tweet classification. By identifying the key positive and negative aspects and eliminating the borderline cases, we can substantially improve the outcomes of the analysis.

Recommendations/ Future Work:
We can make use of deep learning models like LSTM, BERT etc., in capturing contextual information in text data. We can incorporate this model in real life analysis and process the tweets as they are being posted. We can also try to identify the degree of sentiment of the tweets, instead of just interpreting whether they are positive or negative. In considering future improvements to our sentiment analysis approach, we aim to explore practical enhancements that align with the challenges highlighted in our problem statement. While sophisticated models like LSTM and BERT are powerful, we'll focus on more manageable steps. Implementing real-time sentiment analysis pipelines for timely insights into public sentiment is a feasible goal, addressing the issue of manual analysis impracticality. We also plan to refine our sentiment categorization by incorporating additional positive and negative words, acknowledging that simple adjustments can contribute to model improvement. Exploring domain-specific sentiment analysis tailored to our industry or domain is a practical extension, allowing for more contextually relevant insights. Lastly, continuous learning and adaptation of our sentiment analysis models, along with considering ethical implications and bias mitigation, reflect our commitment to responsible and effective sentiment analysis. Through these pragmatic steps, we aim to enhance the accuracy and relevance of our sentiment analysis efforts in the context of social media data.

Appendix:

Figure 1:

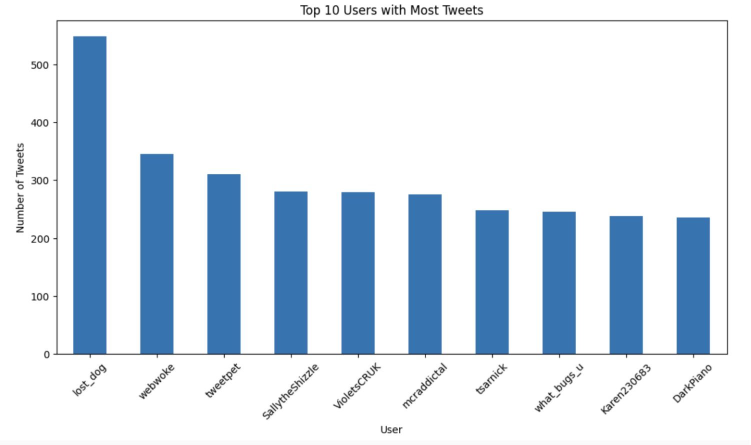


Figure 2:

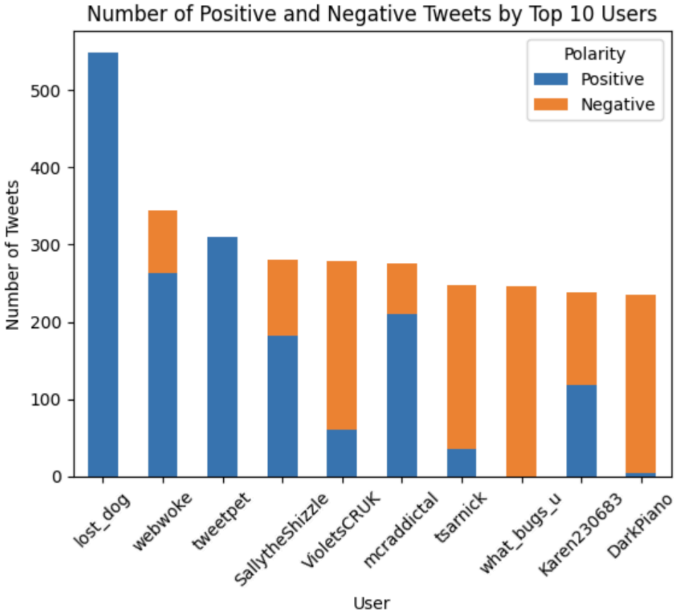
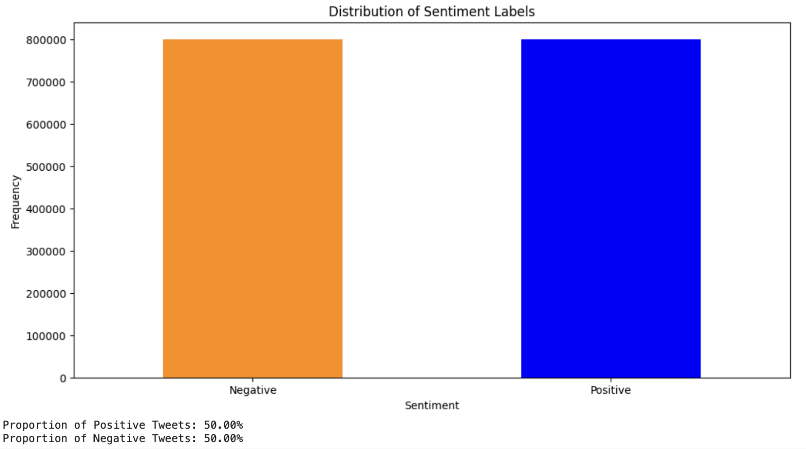


Figure 3:



```
Proportion of Positive Tweets: 50.00%
Proportion of Negative Tweets: 50.00%
```

Figure 4:

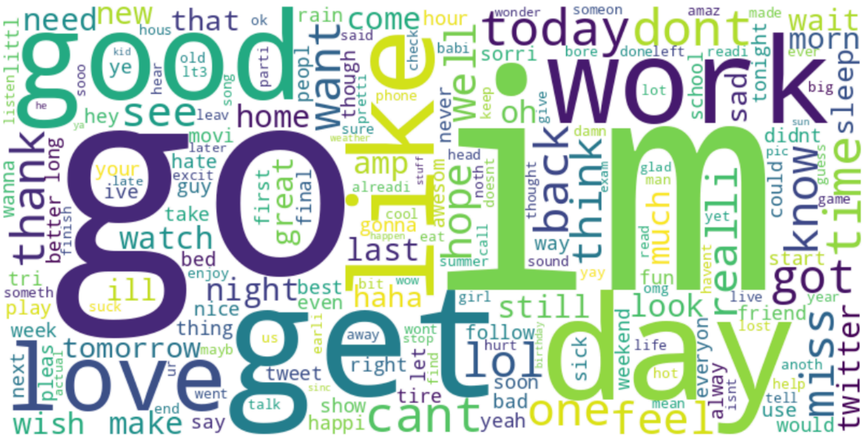
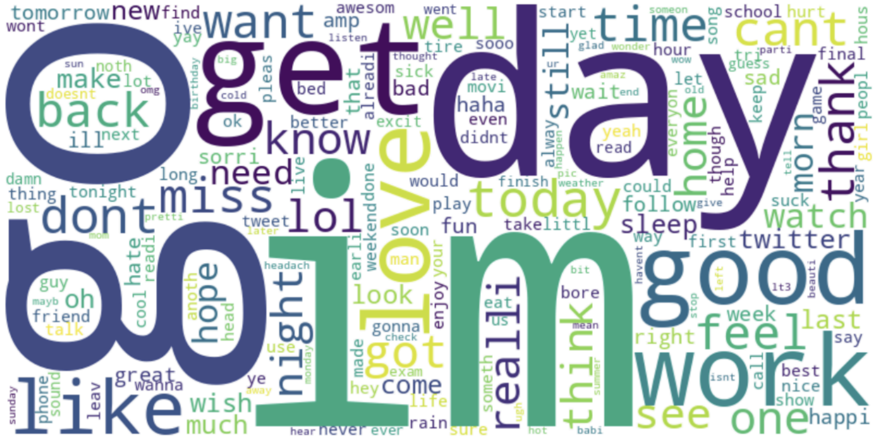


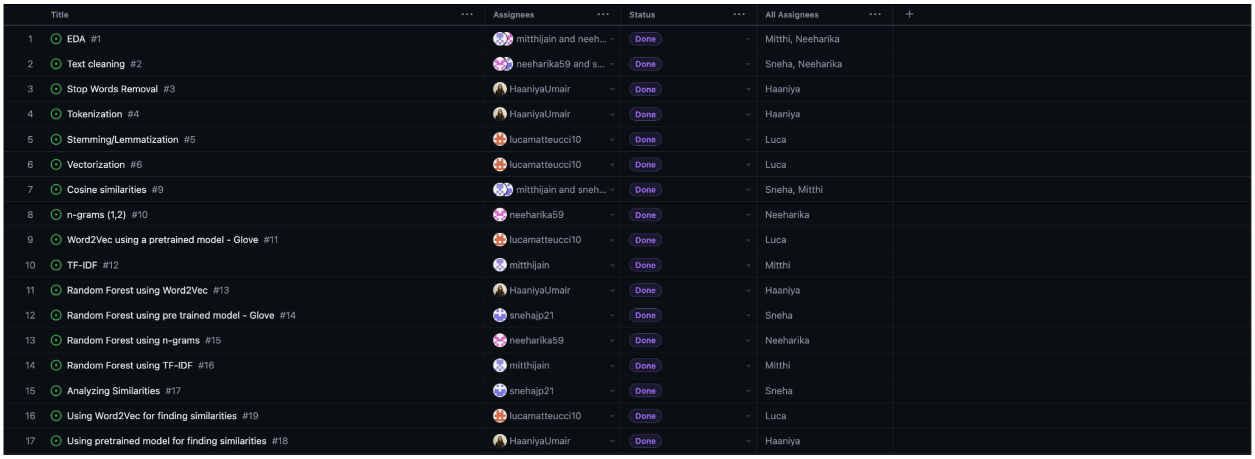
Figure 5:



Github Project: https://github.com/users/lucamatteucci10/projects/1

Repository: https://github.com/lucamatteucci10/BA820-team-10.git

Coding Contribution:



	Title	Assignees	Status	All Assignees
1	EDA #1	mitthijain and neeh...	Done	Mitthi, Neeharika
2	Text cleaning #2	neeharika59 and s...	Done	Sneha, Neeharika
3	Stop Words Removal #3	HaaniyaUmair	Done	Haaniya
4	Tokenization #4	HaaniyaUmair	Done	Haaniya
5	Stemming/Lemmatization #5	lucamatteucci10	Done	Luca
6	Vectorization #6	lucamatteucci10	Done	Luca
7	Cosine similarities #9	mitthijain and sneh...	Done	Sneha, Mitthi
8	n-grams (1,2) #10	neeharika59	Done	Neeharika
9	Word2Vec using a pretrained model - Glove #11	lucamatteucci10	Done	Luca
10	TF-IDF #12	mitthijain	Done	Mitthi
11	Random Forest using Word2Vec #13	HaaniyaUmair	Done	Haaniya
12	Random Forest using pre trained model - Glove #14	snehajp21	Done	Sneha
13	Random Forest using n-grams #15	neeharika59	Done	Neeharika
14	Random Forest using TF-IDF #16	mitthijain	Done	Mitthi
15	Analyzing Similarities #17	snehajp21	Done	Sneha
16	Using Word2Vec for finding similarities #19	lucamatteucci10	Done	Luca
17	Using pretrained model for finding similarities #18	HaaniyaUmair	Done	Haaniya