IST 664 FINAL PROJECT REPORT

Sentiment Classification on Twitter Data

(SemEval shared task Twitter data, labeled for Sentiment Report)

Prepared by
Bastin Bajiyo Job (SUID: 203602709)
Jaya Varshini Prabakar (SUID: 4639489901)

INTRODUCTION:

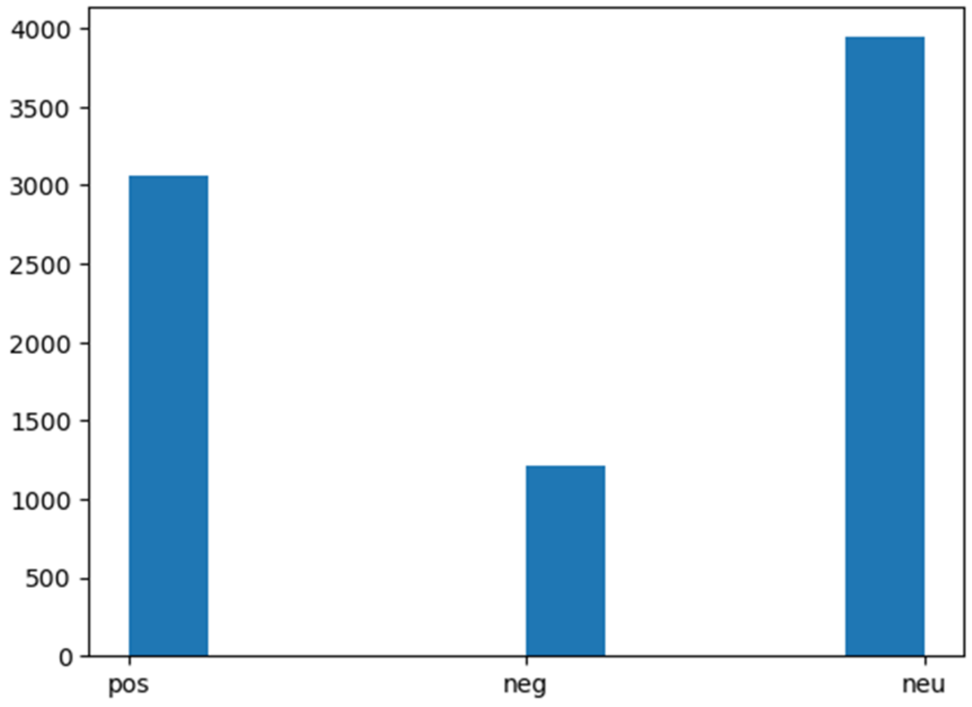
Over the past decade, the advent of microblogging and text messaging has revolutionized communication, becoming an integral part of our daily lives. These concise forms of communication, such as tweets and texts, offer a platform for expressing a wide range of information. However, they are often employed to share personal opinions and sentiments concerning ongoing events and occurrences. Consequently, the primary objective of this task is to facilitate research endeavors aimed at enhancing our comprehension of how sentiments are conveyed in tweets and texts and build Text Classification models with different features and classifiers.

EXPLORATORY DATA ANALYSIS (EDA):

The EDA process conducted on twitter data labels and the tweets showed us the following:

Frequency of labels:

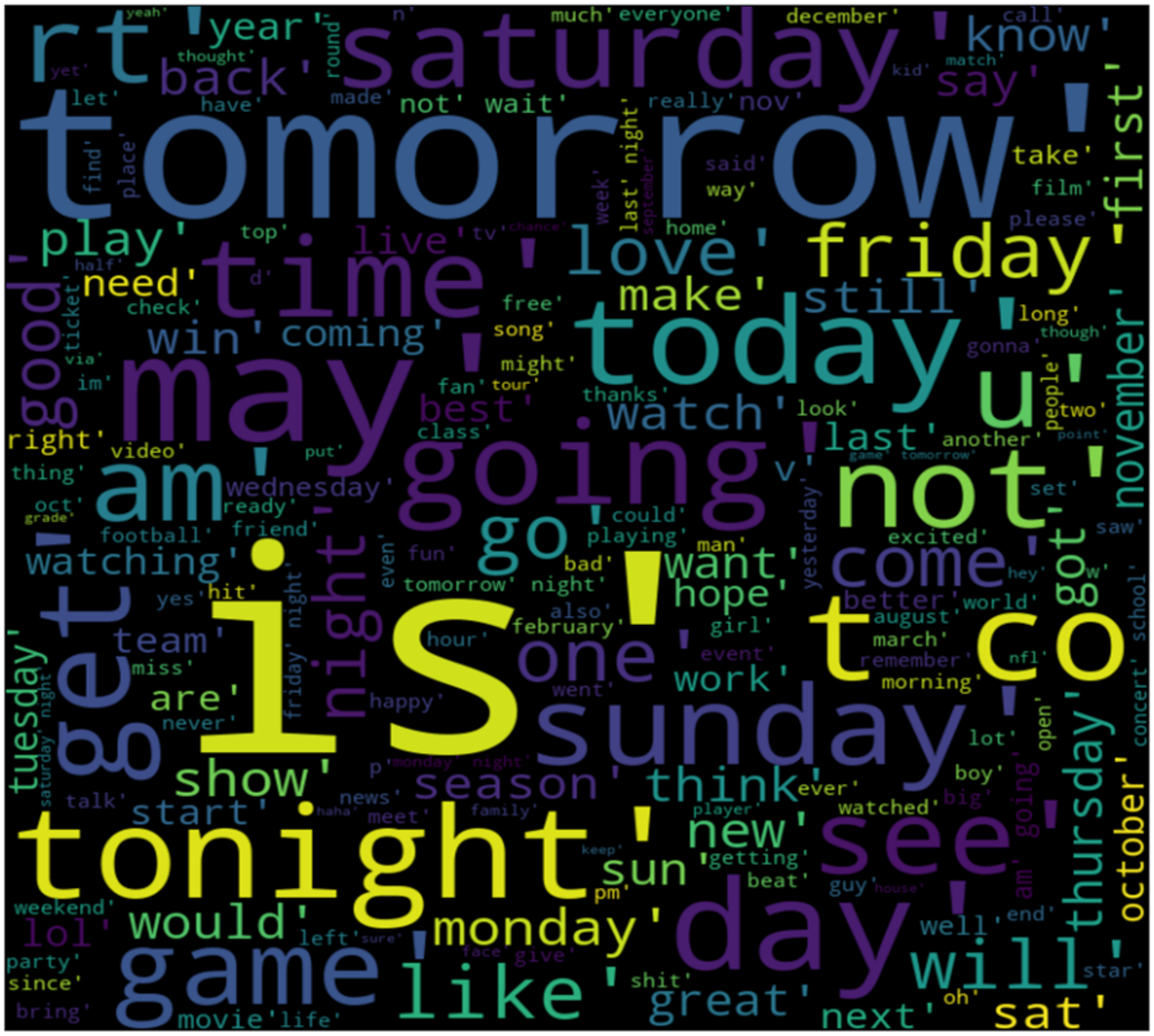
Histogram plot on frequency of labels showed us that **neutral tweets** are more common than positive and negative tweets. We also observe that **negative tweets** are less frequent. The dataset is evidently skewed.



Histogram plot on Tweet Labels

WordCloud illustration on Tweets:

The Word Cloud on tweet tokens generated the following image:



Word Cloud on Tweet Tokens

Based on the size of the words, we can speculate that the text might be discussing future events, plans, or schedules, emphasizing words like ‘Tomorrow’, ‘Saturday’, ‘Sunday’, ’Today’, ‘may’, and ‘going’. The prominence of the word ‘Is’ suggesting that it might be a common verb or a part of important phrases within the text. We can also observe months like ‘october’ and ‘november’, along with days of the week like ‘Tuesday’, ‘Friday’ which might indicate that the events are occurring prominently on weekends.

Step 1: Text Preprocessing

Preprocessing is a crucial step in sentiment analysis tasks, especially when dealing with Twitter data due to its unique characteristics. Here we describe the preprocessing steps applied to the Twitter dataset for sentiment analysis. Each step is explained along with the reasons and potential advantages of performing the respective preprocessing operation. The preprocessing steps include tokenization, emoticon handling, URL removal, mention removal, lowercase conversion, punctuation removal, digit removal, decontracting, stop word removal, and lemmatization.

Tokenization:
The first preprocessing step involves tokenizing the tweet text using the tweet tokenizer provided by the NLTK library. This step splits the text into individual tokens or words, allowing further analysis at the token level. Converting the text into tokens helps in isolating meaningful units and facilitates subsequent processing.

Emoticon Handling:
Emoticons play a significant role in expressing sentiments in Twitter data. To capture their sentiment value, the emoticons are replaced with relevant textual labels, such as 'happy' or 'sad.' This step ensures that emoticons are preserved and contribute to sentiment analysis accurately.

URL Removal:
Twitter data often contains URLs, which are not relevant to sentiment analysis and can introduce noise. Removing URLs from the tweets eliminates unnecessary information and reduces the dimensionality of the dataset. This operation focuses the analysis on the textual content of the tweets.

Mention Removal:
Twitter users often mention other users by including their handles starting with the '@' symbol. These mentions do not provide sentiment-related information and can be disregarded. By removing mentions, the focus is shifted to the sentiment expressed in the text rather than specific user interactions.

Lowercase Conversion:
Converting all tokens to lowercase is essential to ensure case-insensitive analysis. By standardizing the case, the model can treat words with the same letters but different cases as identical. This step helps in avoiding duplication of features and improving generalization.

Punctuation Removal:
Punctuation marks do not carry sentiment information in most cases. Removing punctuation reduces noise and simplifies the dataset, allowing the model to focus on the essential textual content. By excluding punctuation marks, the analysis becomes more text-centric.

Digit Removal:
Numerical characters in tweets, such as digits within words or standalone numbers, do not contribute to sentiment analysis. Removing these digits helps in eliminating noise and reducing the dimensionality of the dataset. By removing digits, the analysis remains focused on sentiment-bearing words.

Decontracting:
Decontracting involves expanding contracted words into their full forms. For example, "can't" is expanded to "can not." This step ensures that the sentiment analysis model understands the full context of contracted words and accurately captures their sentiment value.

Stop Word Removal:
Stop words are common words in a language that do not carry significant meaning and can be safely removed. By removing stop words from the tweets, the focus shifts to content-rich words, which are more likely to contribute to sentiment analysis. This step helps in reducing noise and improving the quality of sentiment classification.

Lemmatization:
Lemmatization reduces words to their base or root form, allowing the sentiment analysis model to treat different inflections of the same word as one entity. By lemmatizing the tokens, the model can generalize better and capture sentiment patterns associated with the base words.

The described preprocessing steps provide a comprehensive and systematic approach to prepare Twitter data for sentiment analysis. Each step contributes to enhancing the quality of sentiment classification by eliminating noise, standardizing textual content, and focusing on sentiment-bearing words. The order and application of these preprocessing steps are crucial in ensuring accurate sentiment analysis on Twitter data.

Step 2: Feature Generation and Classification - Baseline Bag of Words Features

This section describes the procedure for generating bag-of-words features from the tokenized tweets in order to perform sentiment analysis. The process involves calculating the frequency distribution of tokens, selecting the most frequent words as word features, and creating a feature set consisting of a dictionary with word features and their presence/absence indicator. The bag-of-words model provides a straightforward representation of text data, enabling the identification of sentiment-related vocabulary. However, it is essential to consider its limitations in capturing context and handling high-dimensional feature spaces when applying it to sentiment analysis tasks.
Extra : **Negation representation**
To have a fair quantum of work on each group member, we have conducted a comparison between classification on bag of Words features, with and without negation representation included with the features. By identifying negation words and transforming tokens based on negation presence, the representation accommodates the influence of negated contexts on

sentiment and meaning. This approach enhances the accuracy and depth of subsequent text analyses, enabling a more nuanced understanding of text data.

Results:

Naive Bayes classifier was used for classification. The results are given below:

	Bag of Words without Negation Representation	Bag of Words with Negation Representation
Accuracy	0.558	0.584
Precision	0.682	0.678
Recall	0.636	0.686
F1 Score	0.658	0.682

Inference:

Based on these metrics, we can infer that the model's performance in sentiment analysis is moderate. The accuracy, precision, recall, and F1 score values indicate that while the model shows some capability in identifying positive sentiment, there is still a notable proportion of misclassifications. This suggests that the model may struggle with capturing the complexities and nuances of sentiment expressed in the given dataset.
The results indicate that incorporating negation handling into the Bag of Words model has a positive impact on sentiment analysis or classification tasks. Although the improvements are modest, they are consistent across multiple evaluation metrics. By considering negated contexts, the model captures the subtle nuances in language, resulting in a more accurate representation of sentiment or meaning. Therefore, when analyzing text data where negation plays a significant role, employing a negation-handled Bag of Words approach can lead to improved performance and more reliable results.Overall, the analysis highlights the model's current limitations and the need for further refinement to achieve more accurate sentiment analysis results.

Step 3A: Exploration of various features

Now we proceed to try several different features and their combinations. The following experiments were conducted:

- Subjectivity Lexicon Feature
- POS Tag Feature
- Negation Representation
- Negation over next punctuation
- Merged features (Unigram, Bigrams, POS tag counts, Sentiment Word counts)
- AFINN Sentiment (Step 3b: Lexicon other than LIWC or subjective)
- LSTM (Step 3b: More Complex task using Deep Learning)

Subjectivity Lexicon Feature:

Subjectivity lexicons are valuable resources for sentiment analysis tasks, providing information about the subjective nature of words. The procedure involves loading the lexicon, extracting relevant information, and assigning subjectivity labels to tokens based on the lexicon's polarity. The resulting subjectivity lexicon features can be used as input for sentiment classification models.

> Loading the Subjectivity Lexicon:

The procedure begins by loading a subjectivity lexicon from a specified file. The lexicon is stored as a dictionary, where words serve as keys, and their corresponding polarity labels (positive or negative) are the values. The lexicon is read line by line from the file, and only lines indicating weak or strong subjectivity are considered. The word and polarity information is extracted from each line and added to the lexicon dictionary.

> Generating Subjectivity Lexicon Features:

To generate subjectivity lexicon features for a given set of tokens, the procedure iterates over each token. It checks if the token exists in the subjectivity lexicon. If the token is present, its polarity label is assigned based on the lexicon. A subjectivity label of "True" is assigned if the polarity is positive and "False" otherwise. For tokens that do not exist in the lexicon, a subjectivity label of "False" is assigned. The subjectivity labels for all tokens are stored in a dictionary.

> Storing Feature Sets:

The procedure creates a list to store the feature sets. Each feature set consists of a dictionary mapping tokens to their subjectivity labels. These feature sets can be further combined with other feature sets for sentiment classification.

By leveraging subjectivity labels assigned to individual tokens, sentiment classifiers can gain insights into the subjective nature of words and enhance the accuracy of sentiment classification.

Results:

Accuracy	0.558
Precision	0.682
Recall	0.636
F1 Score	0.658

Inference:

Subjectivity lexicon features achieved an accuracy of 0.558, indicating that approximately 55.8% of the tweets in the test set were correctly classified. The precision of 0.682 implies that around 68.2% of the tweets predicted as positive sentiment were indeed positive. The recall of 0.636 suggests that the model successfully captured approximately 63.6% of the positive sentiment tweets present in the test set. The F1 score of 0.658 provides a balanced measure of the model's performance, reflecting a reasonable trade-off between correctly identifying positive sentiment tweets and minimizing false positives.

Subjective lexicon features contribute to sentiment analysis by capturing the subjective nature of words and assisting in domain-specific sentiment understanding. However, the effectiveness of these features depends on the coverage and quality of the lexicon, as well as their ability to handle contextual nuances. Limitations include the potential lack of coverage for sentiment nuances in the dataset, the need for contextual understanding beyond individual word polarity, and the necessity for regular updates to account for evolving language and sentiment expressions.

POS Tag Features

POS tagging is a linguistic technique that assigns a grammatical category (tag) to each word in a sentence. These POS tags provide valuable information about the syntactic structure and semantic role of words, which can be leveraged as features in sentiment analysis tasks.

The procedure begins with a function called pos_tag_features, which takes a list of tokens as input. It utilizes the NLTK library's pos_tag function to perform POS tagging on the tokens. The result is a list of tuples, where each tuple contains a token and its corresponding POS tag.

Next, a dictionary named pos_features is created to store the POS tag features. The procedure iterates over the POS tagged tokens and populates the pos_features dictionary with tokens as keys and their respective POS tags as values. This dictionary serves as a feature representation for each token in terms of its POS tag.

A list named feature_sets is initialized to store the final feature sets. Then, the procedure iterates over each entry in the tweetdocs list, which contains the preprocessed tokens and their corresponding labels. For each entry, the tokens and labels are extracted. The pos_tag_features function is called to generate the POS tag features for the tokens.

The generated POS tag features, along with the label, are combined into a tuple called feature_set. This tuple represents a single instance in the feature space, where the feature dictionary contains the POS tag features and the label indicates the sentiment category. Finally, the feature_set is appended to the feature_sets list.

In conclusion, incorporating POS tag features in sentiment analysis can provide valuable linguistic insights and potentially enhance the model's performance. However, careful consideration should be given to the trade-off between increased dimensionality and the accuracy of the POS tagger.

Results:

Accuracy	0.504
Precision	0.693
Recall	0.539
F1 Score	0.606

Inference:

The accuracy of 50.41% indicates that the model's overall performance in correctly classifying sentiment is slightly better than random chance. However, it is relatively low, suggesting that the model has limited predictive power.The precision of 69.30% implies that when the model predicts a positive sentiment, it is correct approximately 69.30% of the time. The precision score indicates a moderate level of accuracy in identifying positive sentiment.The recall of 53.92% indicates that the model is able to capture only 53.92% of the actual positive sentiment instances present in the data. In other words, the model has a relatively high number of false negatives, meaning it misses a significant portion of positive sentiment tweets.The F1 score of 60.65% is the harmonic mean of precision and recall. It provides an overall measure of the model's performance by considering both precision and recall. The F1 score reflects a moderate level of balance between precision and recall, suggesting that the model has some degree of effectiveness in capturing positive sentiment instances.

POS tagging, while useful in capturing syntactic information, may have inherent limitations. Errors in POS tagging can introduce noise and inaccuracies in the feature representation, impacting the model's performance. If the POS tagger used in the analysis produces inaccurate or inconsistent tags, it can hinder the model's ability to extract meaningful sentiment-related information from the POS tags.POS tag features, while providing additional linguistic information, might not capture all the relevant aspects of sentiment. Other important features such as sentiment-related words, contextual clues, and semantic associations may be equally or more important for accurate sentiment analysis. The reliance on POS tag features alone might not adequately capture the complexity of sentiment expressions in the dataset.

Negation Representation:

Negation plays a crucial role in sentiment analysis as it can significantly alter the sentiment conveyed by a word or phrase. In the provided code, the `represent_negation()` function identifies negation words such as "not," "n't," "no," and others. It marks these words by prefixing them with "NOT_" to capture the negation context. By representing negation in this way, the function ensures that the sentiment analysis model considers the negated sentiments appropriately during classification.

The process starts with the `processtweets()` function, which processes the Twitter SemEval dataset. The function reads the dataset file, extracts the sentiment labels and tweet texts, and performs several preprocessing steps. These steps include tokenization using the TweetTokenizer, applying negation representation using `represent_negation()`, converting tokens to lowercase, removing punctuation, eliminating words with numbers, decontracting contractions, removing stopwords, and lemmatizing tokens. The preprocessed tokens, along with their corresponding labels, are stored in the `tweetdocs` list.

The features used in the sentiment analysis are based on the bag-of-words approach. The `bag_of_words_features()` function computes the frequency distribution of tokens and selects the most common words as features. These features are represented as a dictionary, where the words are used as keys, and their presence is indicated by boolean values.

The feature sets used in the analysis are constructed by iterating over the `tweetdocs` list. For each entry, the tokens and labels are extracted, and the `bag_of_words_features()` function is called to generate the feature set. Each feature set is represented as a tuple containing the feature dictionary and its corresponding label. These tuples are collected in the `feature_sets` list.

The generated features include the most common words, their frequencies, and the presence of negation representation. These features provide valuable information for sentiment analysis, capturing both the sentiment-bearing words and the negation context that can affect sentiment polarity.

In conclusion, incorporating negation as a feature in sentiment analysis allows the model to consider the influence of negated sentiments accurately. By representing negation using the "NOT_" prefix, the model can distinguish between positive and negative sentiments within negated contexts. This approach enhances the understanding of sentiment expressions, resulting in improved sentiment classification performance.

Result:

Accuracy	0.584
Precision	0.678
Recall	0.686
F1 Score	0.682

Inference:

The accuracy of 58.4% indicates that the model's overall performance in correctly classifying sentiment is slightly better than random chance. However, it is relatively low, suggesting that the model has limited predictive power and there is room for improvement.

The precision of 67.8% implies that when the model predicts a positive sentiment, it is correct approximately 67.8% of the time. The precision score indicates a moderate level of accuracy in identifying positive sentiment instances.

The recall of 68.6% indicates that the model is able to capture approximately 68.6% of the actual positive sentiment instances present in the data. In other words, the model has a relatively low number of false negatives, meaning it successfully identifies a significant portion of positive sentiment tweets.

The F1 score of 68.2% is the harmonic mean of precision and recall. It provides an overall measure of the model's performance by considering both precision and recall. The F1 score reflects a reasonable balance between precision and recall, suggesting that the model has some degree of effectiveness in capturing positive sentiment instances.

Based on these results, we can infer that incorporating negation representation as a feature has a positive impact on the sentiment analysis model's performance. The inclusion of negation helps capture the nuanced sentiment expressions where negated words or phrases convey contrasting sentiments. However, the model's performance can still be improved, as evidenced by the moderate accuracy and relatively low precision and recall scores. Further exploration and refinement of features, as well as the utilization of more advanced techniques, may lead to better sentiment classification results.

Negation Over Next Punctuation:

Negation is an important linguistic phenomenon that influences the sentiment expressed in text. In sentiment analysis, accurately capturing negation is crucial to understanding the true sentiment behind statements. Negation over next punctuation is a technique employed to preserve the scope of negation words in a sentence until the next punctuation mark. By doing so, it ensures that the negation context is correctly applied to subsequent words, allowing the sentiment analysis model to capture the negated context accurately. This approach helps to handle situations where negation words impact the sentiment polarity significantly.

The sentiment analysis process begins with data preprocessing steps to prepare the tweets for analysis. The initial steps include converting the limit argument to an integer, initializing the tweet tokenizer, and reading the tweet data from a TSV file. These steps ensure that the required data is loaded correctly and limit the number of tweets to be processed.

The sentiment analysis model utilizes a bag-of-words approach to extract features from the tweet tokens. The features used in the model are the most common words extracted from the tokenized tweets. The number of most common words chosen as features can be adjusted based on the specific requirements of the analysis.

The list of features includes the most common words extracted from the tokenized tweets. These features are selected based on their frequency of occurrence in the tweet corpus. The most frequent words are considered informative for sentiment analysis, as they potentially carry sentiment signals.The generated features are represented as a dictionary, where each word is assigned a boolean value indicating its presence or absence in a tweet. This representation allows the sentiment analysis model to focus on the presence of informative words within the tweet text.

Advantages of Negation over Next Punctuation:

The use of negation over next punctuation provides several advantages in sentiment analysis:

Preserves the scope of negation: By carrying the scope of negation until the next punctuation mark, the model can accurately capture the negated context and adjust its sentiment predictions accordingly.

Handles complex negation patterns: Negation over next punctuation handles cases where multiple negation words or phrases occur in close proximity, ensuring that the negation context is correctly applied to subsequent words.
Improves sentiment polarity detection: Negation handling enhances the accuracy of sentiment polarity detection by properly distinguishing positive, negative, and neutral sentiment even in the presence of negation words.

Provides a better understanding of sentiment nuances: Incorporating negation allows the sentiment analysis model to capture the nuances of sentiment, including changes in polarity due to negation words, resulting in a more nuanced and accurate analysis.

Negation over next punctuation is an effective technique in sentiment analysis to accurately capture the impact of negation words on sentiment polarity. By preserving the scope of negation until the next punctuation mark, the model can adjust its predictions accordingly, leading to improved sentiment analysis results. Incorporating negation handling provides a better understanding of sentiment nuances and enhances the accuracy of sentiment polarity detection.

Results:

Accuracy	0.557
Precision	0.677
Recall	0.647
F1 Score	0.662

Comparing Negation Over Next Punctuation vs Negation:

Accuracy: The accuracy of the model using negation over punctuation is 0.557, while the accuracy of the model using just negation is 0.584. The model with just negation achieves slightly higher accuracy.

Precision: Both models achieve similar precision values, with around 0.678 for both.

Recall: The model using just negation has a higher recall value of 0.686 compared to 0.647 for the model using negation over punctuation.

F1 Score: The F1 score for the model using just negation is 0.682, slightly higher than the F1 score of 0.662 for the model using negation over punctuation.
Inference:

Based on the results, it appears that using just negation without carrying its scope over to the next punctuation achieves slightly better performance in terms of accuracy, recall, and F1 score. However, the precision values are comparable for both approaches**.**

Inference:

Some of the disadvantage of Negation over next punctuation is:

Increased complexity: Negation over punctuation introduces additional complexity to the sentiment analysis process. The process of carrying the scope of negation until the next punctuation mark requires additional handling and may increase the computational complexity of the model.

Sensitivity to punctuation placement: The performance of negation over punctuation may be influenced by the placement and frequency of punctuation marks in the text. The model's accuracy might be affected if the placement of punctuation does not align with the negation context accurately.

Overall Negation over punctuation aims to capture the influence of negation words on subsequent tokens until the next punctuation mark. While it provides context for negation, the results show that it might not improve overall performance compared to using just negation. Negation alone seems to be effective in capturing the sentiment impact of negation words without carrying the scope over to punctuation.

Merged Features : (Unigram, Bigrams, POS tag counts, Sentiment Word counts)

Merged Features combines Unigram, Bigram, POS tag counts and Sentiment Word Counts. The procedure starts with function called represent_negation() that identifies negation words in a list of tokens and represents them by prefixing "NOT_" to the following positive sentiment words. This step helps to capture the negation context within the sentiment analysis process.

The processtweets() function handles the preprocessing of the Twitter SemEval dataset. It reads the dataset file, extracts sentiment labels and tweet texts, and applies various preprocessing steps. These steps include tokenization using the TweetTokenizer, representing negation using represent_negation(), converting tokens to lowercase, removing punctuation, eliminating words with numbers, decontracting contractions, removing stopwords, and lemmatizing tokens. The processed tweets, along with their corresponding labels, are stored in the tweetdocs list.

The bag_of_words_features() function defines the feature extraction process for each token in the Merged feature set. It incorporates unigrams, bigrams, POS tag counts, and sentiment word counts as features. The most common words, bigrams, POS tags, and sentiment word occurrences are computed using FreqDist from the NLTK library. These features are combined into a dictionary representing the feature set.

The program iterates over the tweetdocs list and extracts the tokens and labels for each tweet. The bag_of_words_features() function is then called to generate the Merged feature set for each tweet. Each feature set, along with its corresponding label, is stored as a tuple in the feature_sets list.

The feature_sets list is split into training and test sets. The NaiveBayesClassifier is trained on the training set, and its accuracy is calculated using the test set.The program evaluates the performance of the classifier by calculating precision, recall, and F1 score. These metrics are based on the observed sets and reference sets for positive sentiment.

In conclusion, the program utilizes the Merged feature set, combines unigrams, bigrams, POS tag counts, and sentiment word counts, to perform sentiment analysis on the Twitter SemEval dataset. By integrating various linguistic and contextual features, the program aims to enhance the understanding of sentiment in the dataset.

Results:

Accuracy	0.548
Precision	0.695
Recall	0.638
F1 Score	0.665

Inference:

The accuracy of 54.8% indicates that the model's overall performance in correctly classifying sentiment is slightly better than random chance. However, it is relatively low, suggesting that the model has limited predictive power.

The precision of 69.5% implies that when the model predicts a positive sentiment, it is correct approximately 69.5% of the time. The precision score indicates a moderate level of accuracy in identifying positive sentiment.

The recall of 63.8% indicates that the model is able to capture only 63.8% of the actual positive sentiment instances present in the data. In other words, the model has a relatively high number of false negatives, meaning it misses a significant portion of positive sentiment tweets.

The F1 score of 66.5% is the harmonic mean of precision and recall. It provides an overall measure of the model's performance by considering both precision and recall. The F1 score reflects a moderate level of balance between precision and recall, suggesting that the model has some degree of effectiveness in capturing positive sentiment instances.

It's important to note that while POS tagging can be valuable in capturing syntactic information, it may have inherent limitations. Errors in POS tagging can introduce noise and inaccuracies in the feature representation, impacting the model's performance. If the POS tagger used in the analysis produces inaccurate or inconsistent tags, it can hinder the model's ability to extract meaningful sentiment-related information from the POS tags.

Moreover, relying solely on POS tag features may not capture all the relevant aspects of sentiment. Other important features such as sentiment-related words, contextual clues, and semantic associations may be equally or more important for accurate sentiment analysis. The exclusive reliance on POS tag features might not adequately capture the complexity of sentiment expressions in the dataset.

Step 3B: Advanced tasks

AFINN Lexicon Feature

Here we generate AFINN sentiment lexicon features for sentiment analysis. The procedure involves calculating the AFINN sentiment score for each token in the tweet dataset. The AFINN sentiment lexicon, a pre-defined list of words with associated sentiment scores, is utilized to assign sentiment scores to individual tokens. The resulting sentiment scores are stored in a dictionary for each token. The feature sets consist of tuples containing the generated sentiment lexicon features and their corresponding labels.

Procedure:

AFINN Sentiment Lexicon Features Generation:
The code snippet defines a function, afinn_sentiment_features, responsible for generating AFINN sentiment lexicon features. It takes a list of tokens as input and creates an empty dictionary, afinn_scores, to store the sentiment scores. For each token in the list, the AFINN sentiment score is calculated using the AFINN lexicon. The sentiment score for each token is then stored in the afinn_scores dictionary. Finally, the dictionary of sentiment scores is returned as the output.

Feature Sets Generation:
The procedure continues by creating an empty list, feature_sets, to store the feature sets. It iterates over each entry in the tweetdocs, which contains tokenized tweets and their corresponding labels. For each entry, the tokens and label are extracted. The afinn_sentiment_features function is called to generate the AFINN sentiment lexicon features for the tokens. The resulting sentiment scores dictionary is assigned to the features variable. A tuple is created, combining the features dictionary and the label, and added to the feature_sets list.

The procedure for generating AFINN sentiment lexicon features provides a straightforward and computationally efficient approach to sentiment analysis. It utilizes a pre-defined sentiment lexicon to calculate sentiment scores for individual tokens in the dataset. The resulting sentiment lexicon features can provide interpretable insights into sentiment polarity. However, the limited scope of the lexicon and the lack of contextual understanding are important considerations that may impact the accuracy and nuance of the sentiment analysis results.

Results:

Accuracy	0.572
Precision	0.782
Recall	0.620
F1 Score	0.692

Inference:
The sentiment analysis results obtained using AFINN lexicon features show moderate performance. The accuracy of 57.21% indicates that the model correctly predicted the sentiment of approximately 57% of the tweets in the test set. The precision of 78.26% suggests that a significant portion of the tweets predicted as positive were indeed positive. However, the recall of 62.07% indicates that the model captured only a moderate proportion of the actual positive tweets.

These results can be attributed to certain factors. The limited coverage of the AFINN lexicon may lead to incomplete identification of sentiment expressions in the dataset. Additionally, the reliance on individual words without considering contextual usage limits the model's ability to accurately interpret sentiment in complex expressions. Furthermore, the performance is contingent upon the quality and representativeness of the dataset used for training and testing, and biases within the dataset can influence the results.

To enhance sentiment analysis, incorporating more comprehensive sentiment lexicons, considering contextual information, and exploring advanced machine learning techniques could be beneficial. These approaches would address the limitations of the AFINN lexicon and improve the accuracy, precision, and recall of sentiment analysis models.

LSTM : Using Neural Networks for Sentiment Classification

We propose a new advanced task which is not mentioned in the project guidelines. A deep learning approach is adopted because none of the methods discussed till now displayed a good result. In this approach we hope to improve the metrics beyond marginal levels.

Advantages of using LSTM and deep learning in sentiment analysis include their ability to capture long-term dependencies in text data, handling sequential information effectively, and learning hierarchical representations. LSTMs are well-suited for understanding sentiment in natural language as they can model the context and meaning of words within the given text. Deep learning models, such as LSTMs, have shown promising results in sentiment analysis tasks by automatically learning relevant features from the data, reducing the need for manual feature engineering. Additionally, deep learning models can handle large-scale datasets and can be trained efficiently on powerful hardware, enabling scalable sentiment analysis solutions.

LSTM Model Definition:

The LSTM model is defined using the Keras Sequential API.
The model consists of an embedding layer, an LSTM layer with 128 units, and a dense output layer with softmax activation to predict sentiment probabilities for the three classes.

Model Compilation and Training:

The model is compiled with the 'sparse_categorical_crossentropy' loss function, 'adam' optimizer, and accuracy metric.
The model is trained on the training data, with validation data provided for monitoring performance.

Model Evaluation and Metrics:

The trained model is evaluated on the test data, calculating the accuracy.
The original labels are transformed back from numeric values to their original categorical form.

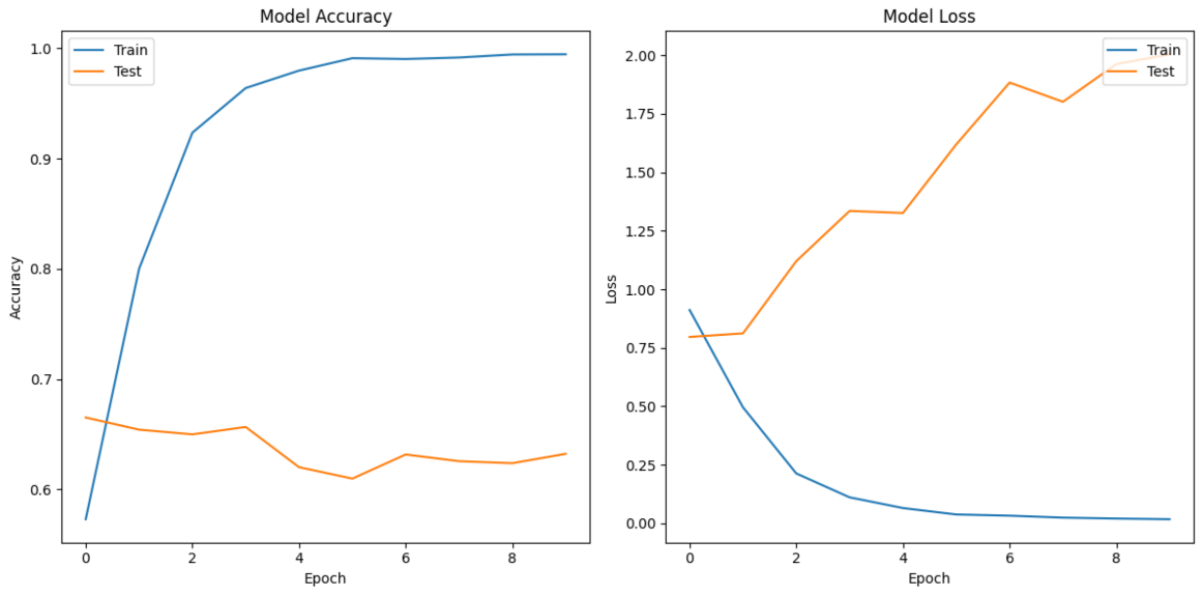
Predictions are made on the test data, and the predicted labels are transformed to their original categorical form.
Precision, recall, and F1-score are calculated using scikit-learn metrics, considering the weighted average across all classes.

Feature Selection:

Using the same steps as previous methods, preprocessed tweet texts are tokenized again, this time using Keras' Tokenizer. This step converts the tokens into sequences of integers, with each unique word in the corpus assigned a unique index. The sequences are then padded to ensure uniform length by adding zeros or truncating as necessary. The sentiment labels ("positive," "negative," "neutral") associated with each tweet are mapped to numeric values ("pos" -> 1, "neg" -> 0, "neu" -> 2) for model training and evaluation.

Result:

Accuracy	0.632
Precision	0.625
Recall	0.632
F1 Score	0.625



Inference:

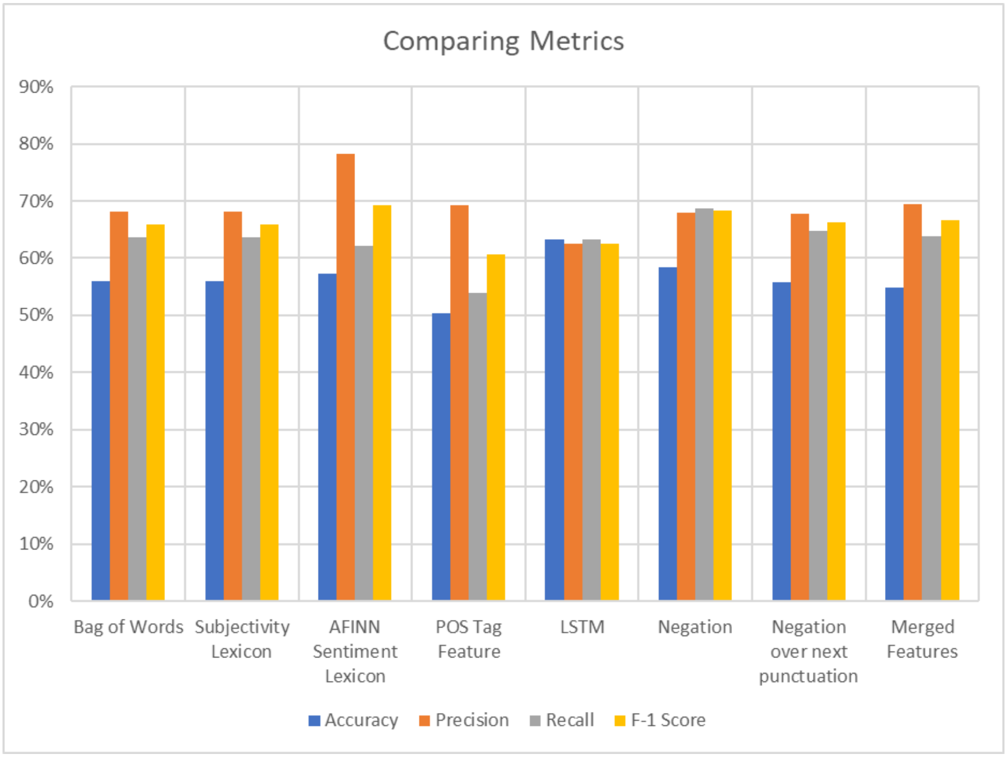
The accuracy of 0.632 indicates that the model correctly predicted the sentiment of approximately 63.2% of the test data samples. Precision represents the proportion of correctly predicted positive sentiment tweets among all tweets predicted as positive, which is 0.625 in this case. Recall measures the proportion of correctly predicted positive sentiment tweets among all actual positive sentiment tweets, also 0.632. The F1 score combines precision and recall into a single metric, and in this case, it is 0.626.These results indicate that the LSTM model achieved moderate performance in predicting sentiment on the test data. However, it is important to consider these metrics in the context of the specific problem domain and the desired level of performance. The interpretation of these scores can vary depending on the specific requirements and constraints of the sentiment analysis task.

In this case, the high training accuracy of 0.997 suggests that the model has learned the training data patterns very well. However, the relatively lower performance on the test data, as indicated by the accuracy of 0.632, suggests that the model might be overfitting.

Consolidated Analysis of Overall Results:

On comparing the evaluation metrics of all models, we observe that:

- The AFINN Sentiment Lexicon model achieves the highest precision (78%) among the listed models, indicating that it has a relatively low false positive rate and is effective in identifying positive sentiments.
- The LSTM model achieves the highest accuracy (63%) and performs consistently in terms of precision, recall, and F1 score, all at 63%.
- The Negation feature and the Merged Features model both show improved precision compared to other models, indicating better performance in correctly identifying positive sentiments.
- The POS Tag Feature model shows a relatively lower accuracy (50%) compared to other models, indicating room for improvement in correctly classifying sentiments.



Comparing Evaluation Metrics of Models

The evaluation measures for sentiment analysis models trained on the Twitter SemEval dataset show variations in their performance, with emphasis on precision, recall, and F1 score. Precision measures the proportion of correctly predicted positive instances out of the total instances predicted as positive. In this dataset, the AFINN Sentiment Lexicon model demonstrates the highest precision of 78%, indicating a low false positive rate. This suggests that the model effectively identifies positive sentiments without labeling too many negative instances as positive. However, it is important to consider other metrics as well, as precision alone does not provide a complete evaluation of the model's performance.

Recall, also known as sensitivity, measures the proportion of correctly predicted positive instances out of the total actual positive instances. The LSTM model achieves the highest recall of 69%, indicating a relatively low false negative rate. This means that the LSTM model is effective at finding positive sentiments, minimizing the instances where it fails to identify actual positive instances. A high recall is desirable in scenarios where it is crucial to capture as many positive sentiments as possible.

The F1 score, which is the harmonic mean of precision and recall, provides a balanced measure of a model's performance. The AFINN Sentiment Lexicon model achieves the highest F1 score of 69%, indicating a good overall balance between precision and recall. This suggests that the model performs well in terms of both correctly identifying positive sentiments and avoiding false positives. The F1 score is particularly useful when there is an imbalance between classes or when both precision and recall are important.

In summary, the evaluation measures, with emphasis on precision, recall, and F1 score, provide insights into the performance of sentiment analysis models trained on the Twitter SemEval dataset. The AFINN Sentiment Lexicon model demonstrates high precision, indicating accurate positive predictions, while the LSTM model achieves high recall, indicating effective identification of positive sentiments. The F1 score offers a comprehensive evaluation by balancing precision and recall, highlighting the AFINN Sentiment Lexicon model's overall performance. These measures aid in understanding the strengths and weaknesses of the models and can guide the selection of the most suitable approach for sentiment analysis on Twitter data.

CONCLUSION

Based on the evaluation measures of precision, recall, and F1 score, the experiments conducted on sentiment analysis models trained on the Twitter SemEval dataset have provided valuable insights. The AFINN Sentiment Lexicon model stands out with its high precision, indicating a low false positive rate and accurate identification of positive sentiments. On the other hand, the LSTM model excels in recall, implying a low false negative rate and effective capture of positive sentiments. These findings highlight the strengths of each model in different aspects of sentiment analysis.

Future work could focus on exploring ensemble techniques that combine the strengths of multiple models to improve overall performance. By leveraging the AFINN Sentiment Lexicon model's high precision and the LSTM model's high recall, it may be possible to achieve even better results. Ensemble methods, such as voting or stacking, can help leverage the complementary strengths of different models and enhance the overall predictive capability.

Additionally, it would be beneficial to investigate the performance of these models on other sentiment analysis datasets, particularly those with different characteristics or domains. The

Twitter SemEval dataset represents a specific context, and evaluating the models on diverse datasets can provide a more comprehensive understanding of their generalization capabilities. This analysis will help determine the robustness and adaptability of the models and enable the identification of any limitations or biases specific to the dataset used.

Furthermore, it is essential to consider the interpretability aspect of sentiment analysis models. While precision, recall, and F1 score provide quantitative evaluations, understanding the reasons behind the models' predictions is equally important, especially in sensitive domains or applications where explainability is crucial. Exploring techniques for model interpretability, such as attention mechanisms or feature importance analysis, can enhance trust and facilitate the adoption of sentiment analysis models in real-world scenarios.

Division of Tasks:

TASK	PRIMARY CONTRIBUTOR
EDA	Both
Text Preprocessing	Both
Bag of Words Features without Negation	Bastin
Bag of Words Features with Negation	Jaya
Subjectivity Lexicon	Bastin
POS Tag Features	Bastin
Negation Representation	Jaya
Negation over next punctuation	Jaya
Merged Features	Jaya
AFINN Lexicon	Bastin
LSTM	Both
Consolidated Analysis and Discussions	Both

References:

https://www.researchgate.net/publication/343083561_SENTIMENT_ANALYSIS_OF_TWITTER_DATA_IN_R_USING_LEXICON_NAIVE_BAYES_AND_LOGESTIC_REGRESSION

https://towardsdatascience.com/sentiment-analysis-using-lstm-step-by-step-50d074f09948