Sentiment Analysis

Arda Derbent[1*] and Anton Yahorau[2†]

[†]These authors contributed equally to this work.

Abstract

Sentiment analysis plays a crucial role in analyzing and understanding public opinion. In this study, we evaluate the performance of two different sentiment analysis tools: BERTweet and Support Vector Machines (SVM) used with TF-IDF . We also investigate the impact of preprocessing techniques such as stemming and feature reduction on the overall performance of sentiment analysis. For Hypothesis 1, our results demonstrate that BERTweet outperforms SVM used with TF-IDF as a sentiment analysis BERTweet exhibited higher accuracy and precision in our sentiment classification tasks. The utilization of contextual embeddings in BERTweet allows it to capture the nuances and complexities of sentiment more effectively than the traditional SVM with TF-IDF.
In Hypothesis 2, we explore the influence of stemming during the preprocessing phase on sentiment analysis performance. Our findings indicate that employing stemming does not significantly improve sentiment analysis results. Despite the potential benefits stemming may offer in other natural language processing tasks, our experiments suggest that its impact on sentiment analysis is negligible. This implies for our dataset, stemming might not be necessary.
Lastly, Hypothesis 3 investigates the impact of reducing the maximum number of features in the TF-IDF representation on sentiment analysis performance. Our experiments reveal that when ngrams are introduced, reducing the maximum number of features does not significantly affect the model's performance. This suggests that some features in the TF-IDF representation may not be vital for sentiment analysis tasks, particularly when considering the contextual information captured by ngrams. By reducing the feature dimensionality, computational resources can be saved by sacrificing minimaal sentiment analysis accuracy.

Keywords: TF-IDF, Sentiment analysis, SVM, BERTweet

1 Introduction

In recent years, sentiment analysis has gained significant attention in the field of natural language processing (NLP) due to its relevance in understanding public opinion, customer feedback, and social media analysis. The ability to automatically classify sentiment from text is crucial for organizations to extract valuable insights and make informed decisions. To achieve accurate sentiment analysis, researchers and practitioners have explored various techniques and algorithms.

1.1 BERTweet

BERTweet, an adaptation of the BERT (Bidirectional Encoder Representations from Transformers) model specifically tailored for Twitter data, has emerged as a powerful language representation model for sentiment analysis. BERTweet utilizes transformer-based neural networks to capture contextual relationships and generate high-quality contextual word representations. By pretraining on a large corpus of Twitter data, BERTweet effectively captures the unique characteristics and linguistic nuances present in tweets. The resulting language representation can be fine-tuned for specific sentiment analysis tasks, allowing for improved sentiment classification accuracy.

1.2 TF-IDF

On the other hand, TF-IDF (Term Frequency-Inverse Document Frequency) is a traditional method widely used for representing text and quantifying the importance of words in a given document or corpus. TF-IDF assigns numerical values to words based on their frequency in the document and their rarity in the corpus. It has been successfully employed in various NLP tasks, including sentiment analysis. By capturing distinctive features of documents, TF-IDF allows for effective sentiment classification.

1.3 Support Vector Machines (SVM)

Support Vector Machines (SVM) is a popular supervised machine learning algorithm used in sentiment analysis. SVM learns a decision boundary based on a set of labeled training data to separate positive and negative sentiment instances. By mapping the input data into a high-dimensional feature space, SVM identifies the optimal hyperplane that maximally separates the sentiment instances. SVM's ability to handle high-dimensional feature spaces and capture non-linear relationships has made it a robust choice for sentiment analysis tasks.

1.4 Dataset Description

The dataset used in this study is derived from Crowdflower's Data for Everyone library, which originally collected Twitter data in February 2015. The dataset focuses on sentiment analysis of tweets related to major U.S. airlines. Contributors were tasked with classifying the sentiment of tweets towards each major U.S. airline, labeling them as positive, negative, or neutral.

The dataset contains a total of 14,640 instances(entries), each representing a tweet related to major U.S. airlines. It contains 9045 negative, 2945 neutral, 2188 positive sentiment labels. Scraped from February 2015.

1.5 Research Objectives

By analyzing this dataset, we aim to explore and compare the performance of different sentiment analysis techniques and preprocessing choices, such as the utilization of BERTweet and SVM with TF-IDF. Additionally, we will investigate the impact of preprocessing techniques such as stemming and feature reduction on sentiment analysis results.

2 Methodology

2.1 Dataset Visualization

The visualization graphs were included to:

- To observe patterns and differences in tweet lengths categorized by sentiment, both before and after preprocessing, using histograms.
- To analyze the distribution of words in the dataset, categorized by sentiment, before and after preprocessing, through frequency distribution graphs.
- To identify the most frequently occurring words within each sentiment category, both before and after preprocessing, providing insights into common language patterns and key themes.
- To create word clouds using the most frequent words, before and after preprocessing, offering a visual representation of prominent sentiment-related terms.
- By comparing the graphs and word clouds before and after preprocessing, we can assess the impact of the techniques on sentiment analysis and gain deeper insights into sentiment classification.

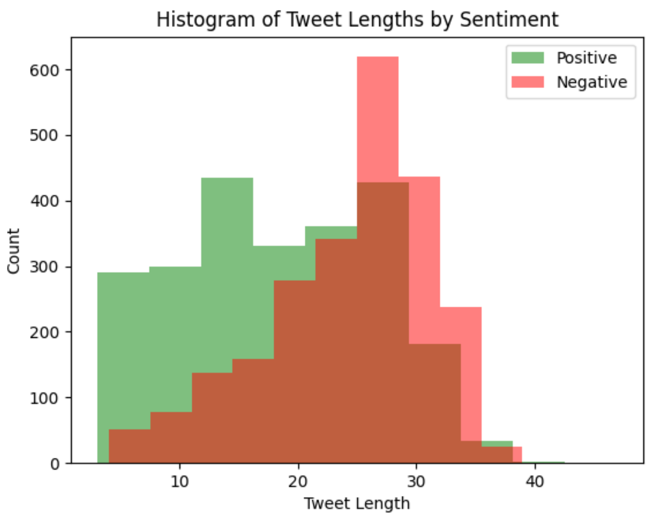

Fig. 1 Histogram of tweet lengths categorized by sentiment, before preprocessing

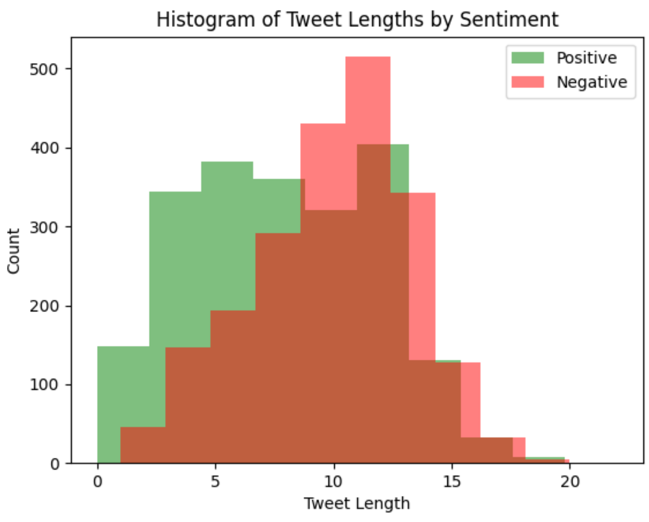

Fig. 2 Histogram of tweet lengths categorized by sentiment, after preprocessing

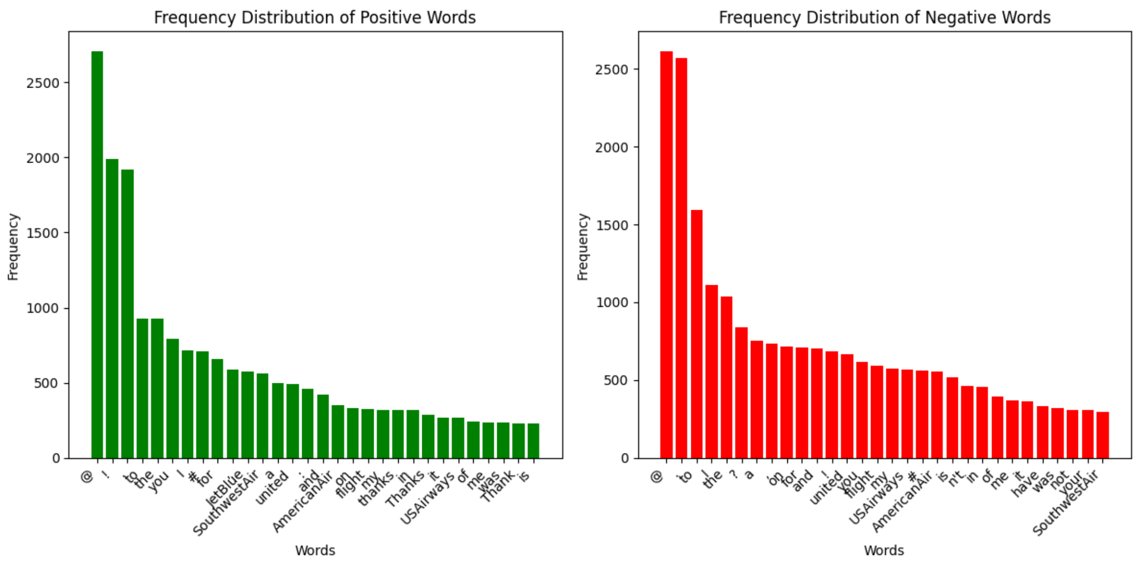

Fig. 3 Frequency distribution graph categorized by sentiment, before preprocessing

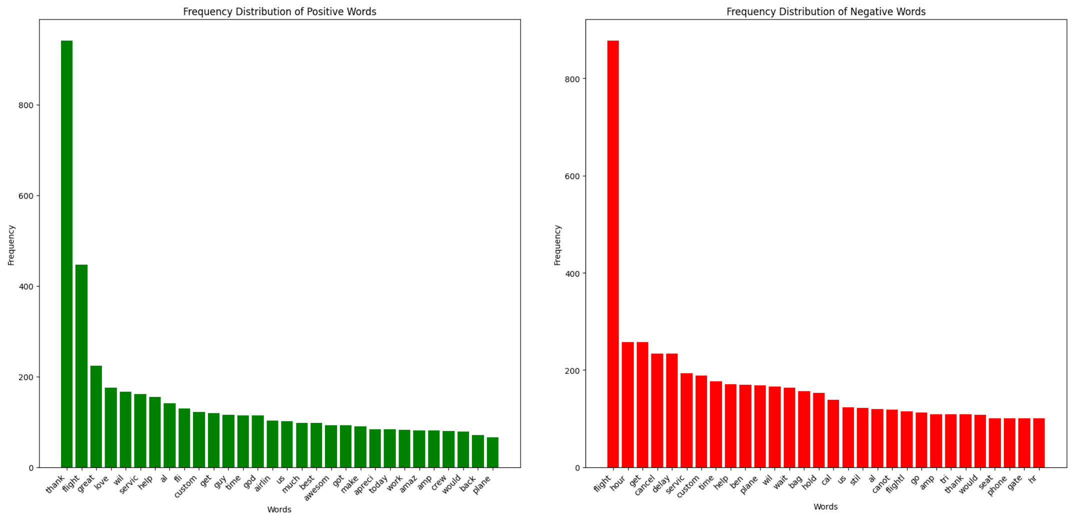

Fig. 4 Frequency distribution graph categorized by sentiment, after preprocessing

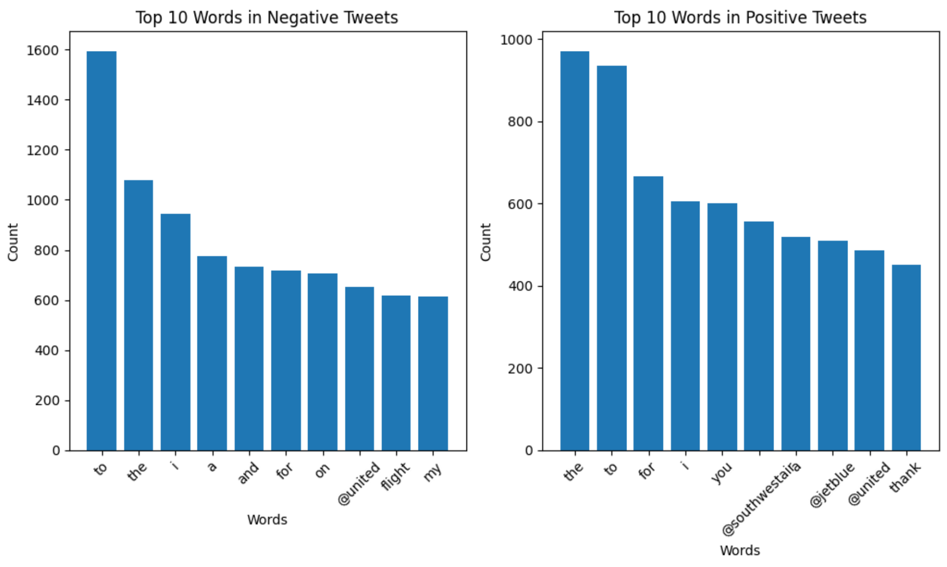

Fig. 5 Top occuring words categorized by sentiment, before preprocessing

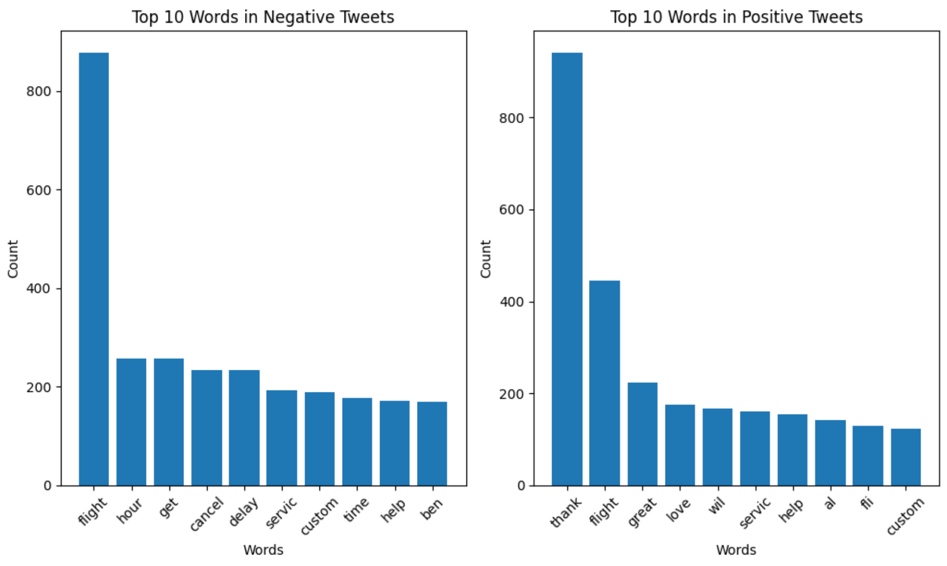

Fig. 6 Top occuring words categorized by sentiment, after preprocessing

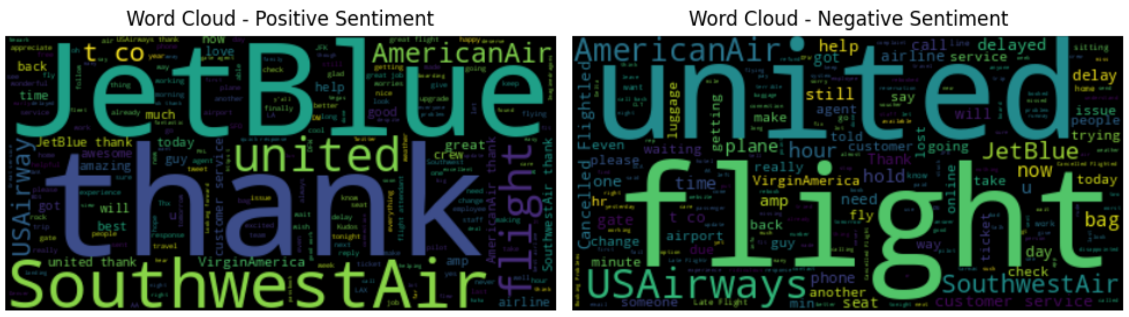

Fig. 7 Wordclouds using most frequent words, before preprocessing

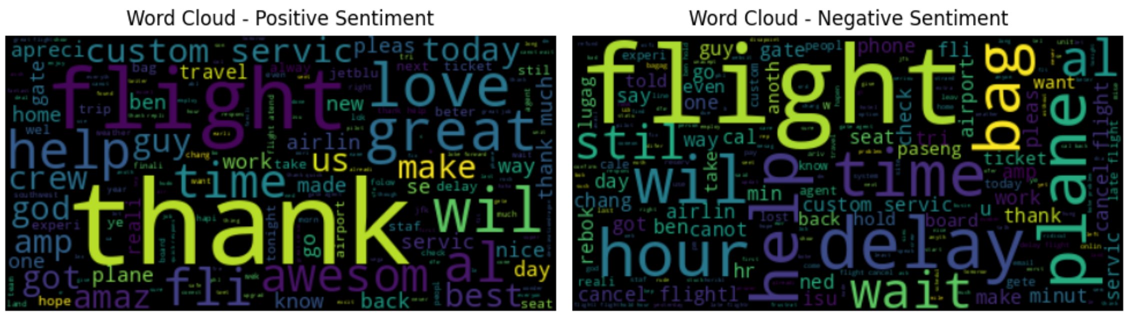

Fig. 8 Wordclouds using most frequent words, after preprocessing

2.2 SVM Preprocessing components

tweet_to_words: This nested function takes a tweet as input and applies several preprocessing steps to it, including:

- Removing URLs from the tweet.
- Removing user mentions (if specified by the *mentions* parameter).
- Converting the tweet to lowercase.
- Expanding contractions.
- Removing non-alphabetic characters.
- Removing duplicate letters.
- Tokenizing the tweet into individual words.
- Removing stopwords (common English words that do not carry much meaning).
- Performing stemming on the words (if specified by the *stemming* parameter) using either the SnowballStemmer or PorterStemmer from the NLTK library.
- Finally, the stemmed words are joined back into a single string and returned.

extract_mentions: This nested function takes a tweet as input and extracts user mentions (e.g., usernames) from the tweet using a regular expression pattern. It returns a list of mentions.

extract_hashtags: This nested function takes a tweet as input and extracts hashtags (e.g., words or phrases preceded by the "#" symbol) from the tweet using a regular expression pattern. It returns a list of hashtags.

Data modification:

- The function adds a new column named 'mentions' to the input dataset, which contains the extracted user mentions for each tweet.
- It also adds a new column named 'hashtags' to the dataset, which contains the extracted hashtags for each tweet.
- The function applies the *tweet_to_words* function on the 'text' column of the dataset and stores the preprocessed text in a new column specified by the *new_column_name* parameter.
- Duplicate entries based on the 'text' column are removed from the dataset and new dataframe is returned

Data Balancing: If the data will be balanced:

- The negative sentiment data are undersampled to match the desired number of instances which is queal to positive sentiment data.
- The undersampled negative, and positive are concatenated to create the processed data.
- The neutral sentiment is removed from the processed data.

2.3 TF-IDF

TF-IDF parameters

- **ngram range**: Specifies the range of n-grams to be extracted. In this case, the ngram range is set to the desired range. Our default value was (1,3)

- **max features**: Determines the maximum number of features to be included in the TF-IDF matrix. Here, the maximum features value is set to the desired maximum value.
- **min df**: Specifies the threshold value for document frequency. Terms with a document frequency lower than the min df value will be ignored.
- **max df**: Specifies the threshold value for document frequency. Terms with a document frequency higher than the max df value will be ignored.

SVM from scikit-learn

- SVM can handle binary classification tasks as well as multiclass problems through the use of one-vs-one or one-vs-rest strategies. In our cased we used it as binary classification method.
- During training, SVM determines support vectors, which are the data points closest to the decision boundary. These support vectors play a crucial role in defining the decision boundary and are used for classification or regression predictions.
- The hyperparameters of SVM, such as the regularization parameter C and the kernel-specific parameters, can be tuned to optimize the model's performance using techniques like cross-validation or grid search. We used Bayesian optimization to find these parameters
- After training, SVM can make predictions on new, unseen data by classifying it based on its location with respect to the decision boundary and the learned support vectors.

2.3.1 BERTweet preprocessing

- **Normalization**: The input text is normalized to ensure consistency in representations. This step may involve converting Unicode characters to their standard form and handling special characters or symbols.
- **Tokenization**: The text is split into individual tokens. The BERTweet tokenizer uses a variation of the WordPiece tokenization algorithm, which divides the text into subword units called "wordpieces." Wordpieces are usually shorter than complete words and can capture more fine-grained linguistic information.
- **Lowercasing**: The tokens are converted to lowercase. This step helps in reducing the vocabulary size and ensures that words with different capitalization are treated as the same.
- **URL, @-mention, and hashtag handling**: The BERTweet tokenizer has specific rules for handling URLs, @-mentions, and hashtags. It often treats them as single tokens or splits them into meaningful segments, depending on the specific implementation.
- **Emoticon handling**: Emoticons or emojis present in the text may be replaced with special tokens or handled in a way that preserves their meaning or sentiment.
- **Special character and whitespace handling**: Special characters and whitespace are handled appropriately. This may involve removing or preserving specific characters, depending on the tokenizer's rules and requirements.
- **Token truncation and padding**: The BERTweet tokenizer may truncate or pad the tokens to a fixed length to fit the model's input requirements. Truncation

involves cutting off tokens from the beginning or end of the text if it exceeds the maximum token limit. Padding involves adding special tokens to ensure all input sequences have the same length.

2.3.2 BERTweet parameters

- **num train epochs** (default: `3`): The total number of training epochs.
- **per device train batch size** (default: `16`): The batch size per device during training.
- **per device eval batch size** (default: `64`): The batch size for evaluation.
- **warmup steps** (default: `500`): The number of warmup steps for the learning rate scheduler.
- **weight decay** (default: `0.01`): The strength of weight decay regularization.

2.3.3 BERTweet

- BERTweet utilizes the architecture of BERT (Bidirectional Encoder Representations from Transformers) specifically trained on Twitter data.
- The model consists of multiple layers of self-attention mechanisms, known as transformer layers, which capture the contextual information of the input text.
- BERTweet employs a masked language modeling objective during pretraining. It randomly masks some tokens in the input and learns to predict those masked tokens based on the surrounding context.
- During both pretraining and fine-tuning, BERTweet takes advantage of the tokenization process to split the input text into subword units called "wordpieces." These wordpieces capture both complete words and subword information.
- BERTweet learns contextual representations of the input text by considering the relationships between the wordpieces, enabling it to capture the meaning and syntactic structure of the text.

3 Experiments and Results

3.1 Experiment 1

Hypothesis 1: BERTweet is significantly a better sentiment analysis tool compared to SVM used with TF-IDF

In order to test this hypothesis, we ran our algorithms in their default configurations with the best currently obtained results from hyperparameter optimizations. We observed that BERTweet outperformed SVM with TF-IDF even when it was overfitting data:

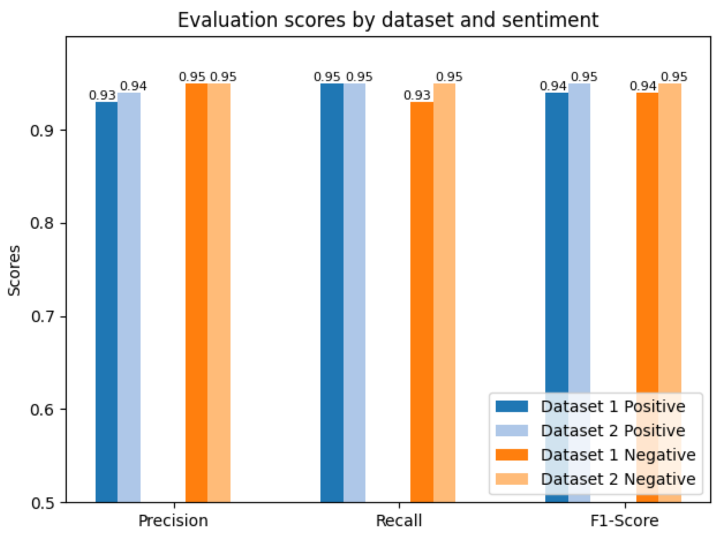

Fig. 9 BERTweet Dataset 1- Epoch size 4, Dataset 2 - Epoch size 3 default configurations

Training Argument Values

The values of the training arguments used in BERTweet are as follows:

- **num_train_epochs**: `4 and 3`
- **per_device_train_batch_size**: `16`
- **per_device_eval_batch_size**: `64`
- **warmup_steps**: `500`
- **weight_decay**: `0.01`

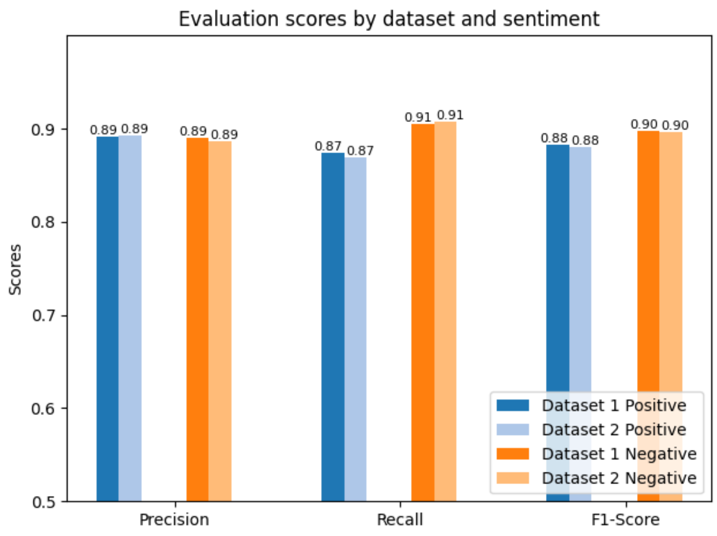

Fig. 10 SVM max features 53404 and max features 4000

SVM Dataset 1 parameters

- **C**: 48.47923420462904
- **kernel**: 'sigmoid'
- **gamma**: 0.9864697981315979
- **max_features**: 47047
- **vocabulary**: 47047

SVM Dataset 2 parameters

- **C**: 0.66410841803048891
- **kernel**: 'sigmoid'
- **gamma**: 0.4622714817696367
- **max_features**: 4000
- **vocabulary**: 4000

Our Analysis:

- Preprocessing and vectorization were identified as the most important factors in improving the SVM results.
- Limiting the number of features did not lead to a significant reduction in evaluation scores.
- From the dataset visualization, it was clear that the skewed distribution of negative sentiment tweets had an impact on the SVM results.
- BERTweet's performance was affected during epoch size=3, resulting in a decrease of 0.01 in the evaluation score of precision.
- Overfitting the data with epoch size=4 led to a reduction in the evaluation scores of positive data in BERTweet.

3.2 Experiment 2

Hypothesis 2: Using stemming during preprocessing does not significantly improve the performance of sentiment analysis.

In order to test this hypothesis, we ran our preprocessing algorithms with and without PorterStemmer stemming functionality and obtained the best results from hyper-parameter optimizations.

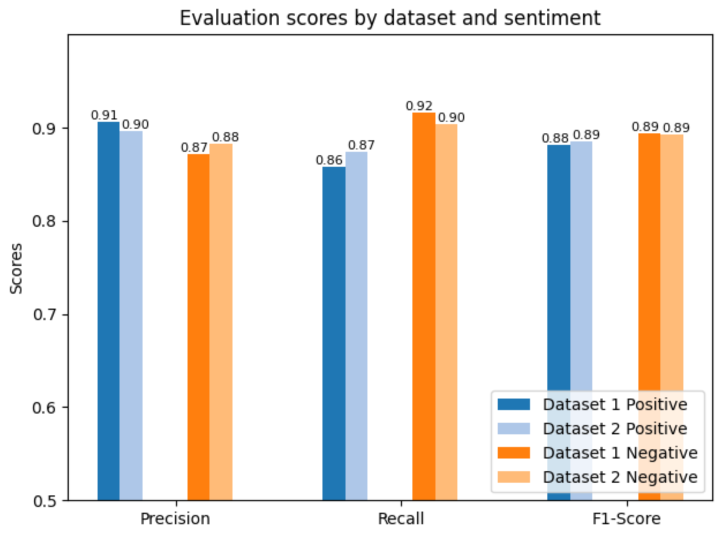

Fig. 11 SVM No stemming vs Porter Stemmer

- Stemming was not very impactful according to our results. It changed overall evaluation scores in small amounts, the only improvement that can be clearly seen is the 0.01 improvement in f1 score of positive sentiment text. So for our dataset and limited parameters, this hypothesis was proved true.
- Stemming, in the end, allows us to correct words in our dataset that are in different forms. In tweets, since we are dealing with relatively small sentences, the effect of PorterStemmer was not drastic. However, since it balanced the sentiment evaluation scores a bit in terms of positive sentiment, we used it in all experiments.
- The reason why stemming is used instead of lemmatization is that we don't care about preserving the words since they are not going to a text generator for displaying to the user.

3.3 Experiment 3

Hypothesis 3: Reducing the maximum number of features in the TF-IDF representation does not significantly impact the performance of the model, as some features are not vital especially when ngrams are introduced.

In order to test this hypothesis, We ran our algorithm with multiple max features values to observe the relation between our evaluation scores and the max features. We always used hyper-parameter optimization to get the best model for the current max features.

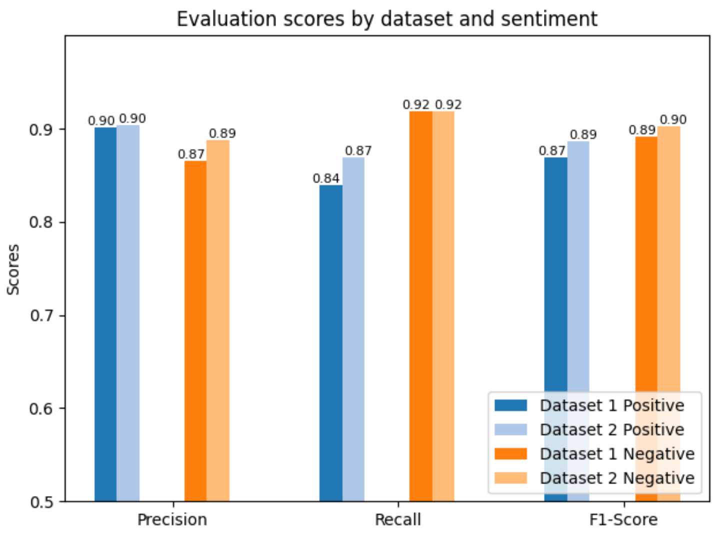

Fig. 12 SVM 500 max features vs 1500 max features

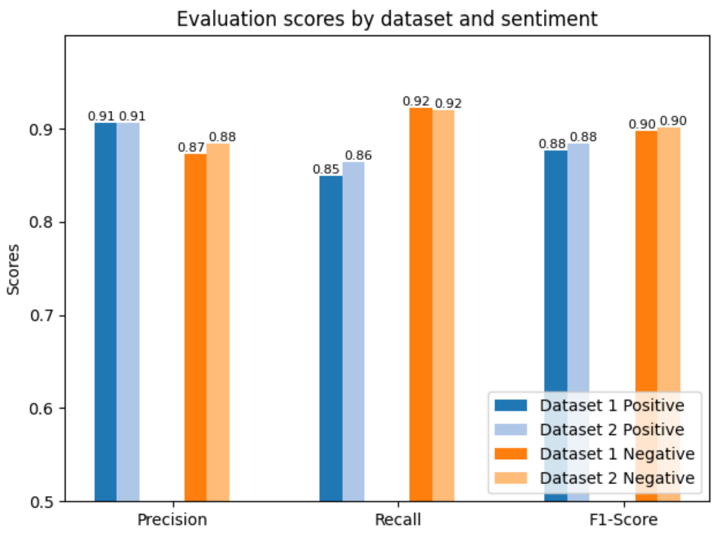

Fig. 13 SVM 1000 max features vs 2000 max features

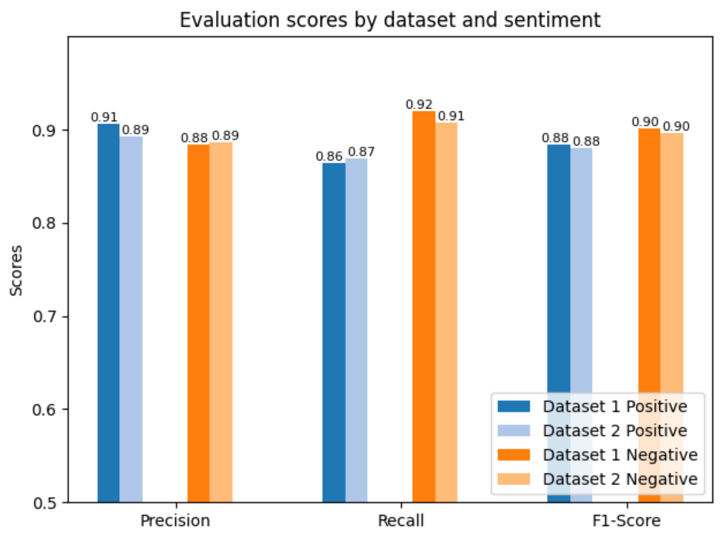

Fig. 14 SVM 2000 max features vs 4000 max features

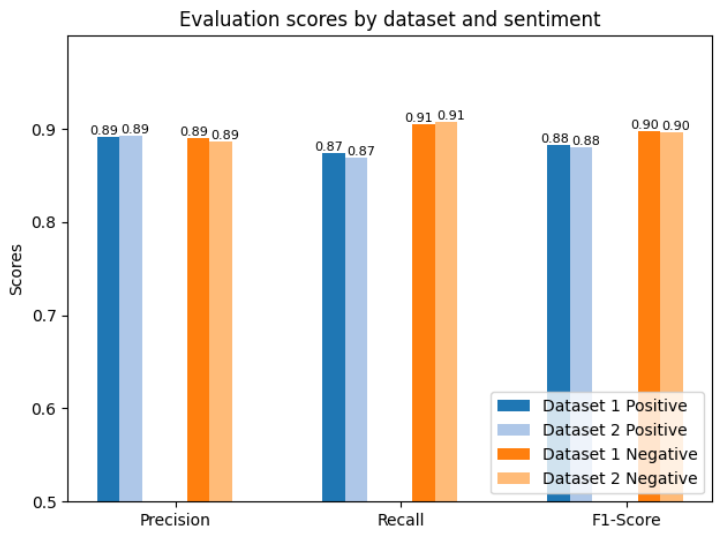

Fig. 15 SVM 53404 max features vs 4000 max features

- From Figure 15, it was confirmed that reducing the maximum number of features from 53,404 to 4,000 did not result in significant changes in evaluation scores.
- From Figure 14, it was observed that even when the maximum number of features was further reduced from 4,000 to 2,000, the changes in evaluation scores did not exceed 0.02.

- Therefore, it was concluded that TF-IDF n-grams introduce many unnecessary features that are not needed for sentiment classification.
- It is recommended to reduce the maximum number of features to save memory and improve the understanding of the dataset.

4 Conclusion

- BERTweet outperformed SVM in terms of evaluation scores, indicating its superiority in sentiment classification tasks.
- Preprocessing and vectorization were identified as the most important factors in improving the SVM results.
- Limiting the number of features did not lead to a significant reduction in evaluation scores.
- The skewed distribution of negative sentiment tweets had an impact on the SVM results, as observed from the dataset visualization.
- In the case of BERTweet, performance was affected during epoch size=3, resulting in a decrease of 0.01 in the precision evaluation score.
- Overfitting the data with epoch size=4 led to a reduction in the evaluation scores of positive data in BERTweet.
- Stemming was not highly impactful in terms of overall evaluation scores, but it showed a clear minmal improvement of 0.01 in the f1 score of positive sentiment text.
- The reduction of the maximum number of features from 53,404 to 4,000 and further to 2,000 did not result in significant changes in evaluation scores. This suggests that TF-IDF n-grams introduce many unnecessary features for sentiment classification.
- It is recommended to reduce the maximum number of features to save memory and improve the understanding of the dataset.