Sentiment Analysis using Natural Language Processing

Group 45, Alagammai Lavanya Muthiah, Tay Yong, Ethan Wong Zyen Byng

Introduction

In the digital age, advanced tools are needed for understanding sentiments in textual data. Sentiment analysis, powered by NLP, automates text classification into positive or negative sentiment. Through NLP and ML, we efficiently process text to regulate online content. This project demonstrates the practical use of ML algorithms and NLP techniques, addressing limitations of traditional sentiment analysis methods like rule-based strategies.

Exploring past works, Erwin & Alvi (2022) also examined the limitations inherent in analyzing short texts related to public policy, which often lead to reduced semantic meaning. They proposed smart feature extraction using Word2Vec, enhancing accuracy by filling in missing details and rectifying mismatches. Their approach employed three vectorization methods—Bag of Words (BOW), Word2Vec, and TF-IDF—alongside logistic regression, all of which we will follow closely in our analysis. Rizal et al. (2022) supported the use of Word2Vec, achieving an accuracy of 72% with the Naive Bayes classifier on movie reviews. As its fast and easy implementation proved their method's effectiveness in sentiment analysis, we will adopt it. Their research also followed a preprocessing protocol involving label encoding, case folding, cleaning, tokenization, stopword removal, and stemming (lemmatization), laying a groundwork for data handling that our study will adhere to. Additionally, we will follow Kavitha et al. 's (2022) approach of Random Forest Classifier (RFC) with TF-IDF on classifying tweets, which reported an accuracy of 95%. Erwin's (2022) focus on topic-specific datasets potentially narrows the model's applicability, so we decided to use a more general dataset of tweets to increase usability of our model on other various datasets. Bahrawi (2019) also explored RFC, achieving decent results and suggested the exploration of additional machine learning algorithms paired with various vectorization to identify optimal combinations further sentiment analysis accuracy and to identify optimal combinations for context-specific data versus general mixed data, which is what our paper aims to do. This overcomes the limitation of classification models being only more appropriate for specific types of data on

social media or movie reviews separately and hopes to identify a model with more general usability on data with unsorted themes/domains.

Dataset

The dataset 'dataset.csv' contains 10,000 sentences, evenly split between positive and negative sentiments labeled as 1 or 0, respectively. Our subsequent data visualization revealed frequent occurrences of words like 'movie' and 'film,' indicating that the dataset primarily comprises movie reviews. Hence, we decided to augment our dataset with a general dataset of tweets for broader coverage.

Creating a wider dataset

We used a dataset of 1.6 million tweets uploaded by user 'ΜΑΡΙΟΣ ΜΙΧΑΗΛΙΔΗΣ ΚΑΖΑΝΟΥΑ' from kaggle. To process the data, we first filtered out extra columns such as datetime and user and left only the tweets. To minimise computation power needed, we scaled it down, similar in size to our original dataset, with tweets that were strictly positive or negative. To do so, we first randomly sampled 20000 tweets from the dataset, 10000 of each value (4 or 0). Reviews of this dataset indicated that some data were false positives and false negatives, or neutral. Hence, to minimize the probabilities of us extracting this falsely labelled data, we imported vaderSentiment, a module with a function SentimentIntensityAnalyzer that could give an accurate numerical value of sentiment to an input text string. It was recommended that a value of 0.05 or greater be considered positive, and -0.05 or lesser be negative.

We removed specific features of tweets - usernames, as they could contain positive/negative sentiment, and emojis as they could disrupt the tokenisation/training process. The new dataset of cleaned tweets correctly labelled 1 or 0 (positive or negative respectively) is named 'filteredtweets.csv'.

Data Preprocessing

We first label encoded our data to ensure that all our data is in numerical form as integers for ML algorithms to process. Then to clean our textual data, we identified and removed or filled missing values, removed duplicate rows, changed all texts to lowercase, and removed punctuation and contractions. Afterwards, we employed tokenization and lemmatization methods.

Tokenisation and Lemmatization

Tokenization is the process of breaking down a text into smaller units called tokens, allowing our algorithm to analyse it more effectively. In our analysis, we used NLTK's word_tokenizer function. Next, we removed stopwords

from our data NLTK's stopwords, a list of words that NLTK considers to be of little value for NLP tasks as they occur frequently but typically do not carry significant meaning. However, the list had some words that could impact sentence sentiment such as 'not' or 'over', so we identified all such words and kept them in our data. This scaled down our data significantly, leaving mostly important data. Lemmatization was done using used NLTK's WordNetLemmatizer to reduce words to their base or root form. This reduces the dimensionality of our vocabulary list. Overall, these processes improve the performance of our model by allowing the models to focus on only key words and reduce computational resources.

Data Visualization

The generate_wcloud function creates a word cloud visualization by utilizing the WordCloud class, representing the frequency of words in provided sentences. We used N-grams, which are contiguous sequences of words, to analyze the text dataset. With the plot_top_ngrams function, we visualized the frequency of these n-grams, focusing on the top 10 most frequent trigrams. These plots, displayed using matplotlib, provide a comprehensive overview of common words (in the case of word clouds) and specific phrases (in the case of n-grams), facilitating quick comprehension of prominent themes or patterns within the text data. In our analysis, we observed these patterns to be mainly related to movies and their reviews, as illustrated in Figures 1 and 2 below.

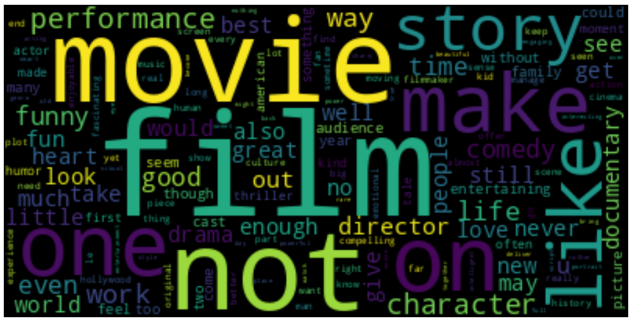


Figure 1: Word Cloud of Positive labelled data

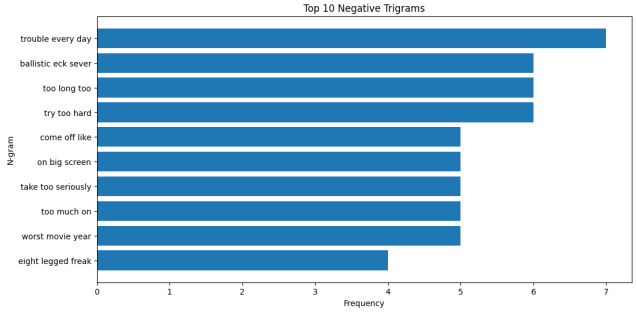


Figure 2: Top 10 Positive Trigrams

Methodology

To observe the effect of adding the new dataset of tweets, we carried out our methods on two datasets, the original 'dataset.csv' and a new dataset created from appending 'filteredtweets.csv' into it. We used the 'train_test_split' method to implement an 80/20 split of each dataset separately into training and testing data. We used 3 different vectorizations on our data, then fed them into different ML algorithms for classification. The model will then learn the relationships between the vectorized data and the class labels, helping it to make predictions for new data. We also defined a function 'train_and_eval' that took in our model and the training and testing data, and trained the model with the training data, which made predictions of all the data and returned train and test accuracy values respectively.

Bag-of-Words Vectorizer

We utilized the Bag-of-Words (BoW) Vectorizer to convert raw text data into numerical features. By representing each document by tallying the individual occurrences of each word and discarding structural information such as grammar, it makes data easier for the ML algorithms to process. This feature extraction process enables machine learning models to learn patterns and relationships between words and make predictions based on these patterns, enabling tasks like sentiment analysis.

TF-IDF

TF-IDF stands for Term Frequency – Inverse Document Frequency. It is a numerical statistic used typically for information retrieval and sentiment analysis. TF measures how frequently a term appears in a document, calculating the relative importance of a term in a specific document. IDF measures the rarity of a term across a collection of documents. Utilizing TF-IDF in sentiment analysis endeavors is advantageous as it bestows greater significance upon terms prevalent within a particular document while being infrequently encountered across the corpus at large. This characteristic effectively addresses the challenges posed by imbalanced data distributions, ensuring a more nuanced and equitable analysis. The formula to calculate TF-IDF is as follows.

$$TF_{(i,j)} = \frac{Term\ i\ frequent\ in\ document\ j}{Total\ words\ in\ document\ j} \qquad (1)$$

$$IDF(i) = \log\left(\frac{Total\ documents}{documents\ with\ term\ i}\right) \qquad (2)$$

Word2Vec

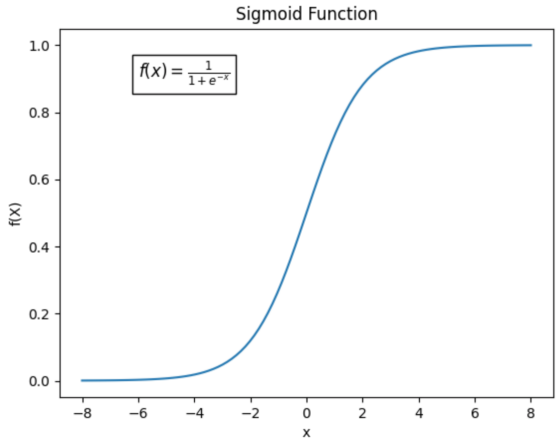
In this research, Word2Vec is used as feature extraction from data processed previously. It is a neural network implementation that learns from distributed representations of

words and does not require labels. We used the Skip-Gram model which works by making predictions from a word in the current word or around the predicted word in a sentence. In our code, we used the Skip-gram model of Word2Vec together with our machine learning algorithms, which makes predictions from a word itself or around the predicted word in a sentence, reducing the dimensionality of the word space while preserving vital semantic relationships. Additionally, considering the surrounding words is important in sentiment analysis, as the sentiment of a word can vary depending on its surrounding context. This model was chosen because this model can count the maximum of average log probability in one word as show by this equation:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} logp(w_{t+j}|w_t)$$

Logistic Regression (covered in IT1244)

In logistic regression, a decision boundary is created based on a logistic function. The model is trained by optimising the weights and biases to minimize the loss function, typically done using algorithms such as the Gradient Descent Algorithm. We used a "max_iter" value of 500. Our logistic regression model is trained and evaluated multiple times using different values of "C". This seeks to find the optimal regularization strength, balancing model complexity and generalization performance. Our code implements the sigmoid function as seen below:



Logistic regression is preferred over linear regression as it is better suited for binary classification tasks.

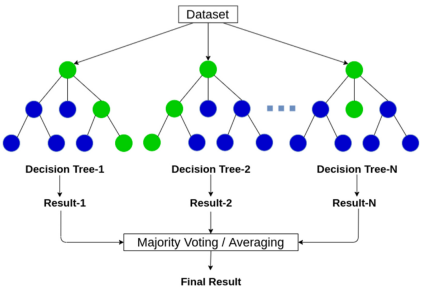
Naive Bayes Classifier

Naive Bayes classification is a probabilistic ML algorithm commonly used for classification tasks. This is based on computing the prior probability of each class based on the frequency of class labels in the training data. For sentiment analysis, the model counts the frequency of each word in documents belonging to the respective class. In our code, "alphas" represents a list of alpha values used for Laplace smoothing in the Multinomial Naive Bayes classifier. Laplace smoothing is commonly used in Naive Bayes classification to handle unseen features in the data. This process involves adding a small value (alpha) to the observed counts of features during probability estimation, smoothing the probability estimates and preventing zero probabilities. Reducing alpha leads to less smoothing, and vice versa. In our code, we iterate over different alpha values to assess the classifier's performance across varying smoothing conditions. The formula for Naive Bayes Classifier is given below:

$$P(c_j|w_i) = \frac{P(c_j)P(w_i|c_j)}{P(w_i)}$$

Random Forest Classifier

The Random Forest Classifier (RFC) is a supervised machine learning method that combines multiple decision trees to make predictions. By creating a "forest" of trees instead of relying on a single tree, RFC reduces the risk of overfitting, where the model learns excessively from the training data and struggles with new data. Introducing randomness enhances the model's reliability and accuracy. The algorithm works as such: It starts by taking many bootstrap samples from the original dataset, which are subsets of the original data. For each bootstrap sample, an unpruned decision tree is grown. However, at each node of the tree, only a random subset of predictors is sampled. The best split is then chosen from among these randomly selected predictors. This introduces variability into the decision-making process, reducing the risk of overfitting and enhancing the generalizability of the model. After growing all the trees, predictions for new data are made by aggregating the predictions of all the trees. We chose RFC as it considers complex non-linear relationships which often occur in sentiment analysis. RFC also provides a measure of feature importance, indicating which words contribute the most to sentiment prediction. Below is a visualization of RFC.



Results and Discussion

Our study aimed to enhance the accuracy of sentiment analysis models applied to sentence-level assessments. We defined three separate functions for hyperparameter tuning for each of our models, which iterated through different hyperparameters, and used techniques such as GridSearchCV to find the best one for each classifier, optimising performance for our model.

For Logistic Regression, the optimal setting was found to be {'C': 1}, indicating a balance between model complexity and regularization. The Naive Bayes model performed best with {'alpha': 5}, suggesting a slight preference for smoothing to address the problem of zero probabilities. RFC achieved optimal results with {'n_estimators': 300, 'criterion': 'entropy'} both with and without Word2Vec, emphasising the model's reliance on a large ensemble of decision trees to improve accuracy.

Exploration done using three machine learning models were evaluated: Naive Bayes, Logistic Regression, and Random Forest Classifier. The performances of these models were primarily assessed using accuracy metrics derived using the 'sklearn.metrics' package, utilising its 'accuracy_score' and 'classification_report' functions. The 'accuracy_score' function measures the proportion of correct predictions out of all predictions made, providing a straightforward metric of model performance. In contrast, the 'classification_report' function offers a comprehensive overview of a model's precision, recall, and F1-score for each class, facilitating a deeper understanding of model strengths and weaknesses in classification tasks.

Impact of Dataset Enrichment (New Dataset)

Classifier	Accuracy
Naive Bayes with TF-IDF (dataset.csv)	0.769
Naive Bayes with TF-IDF (new dataset)	0.818
Logistic Regression	0.807
Random Forest Classifier	0.790

From the above table, adding the new dataset of tweets improved the accuracy of all the models significantly. This enhanced accuracy suggests that the additional data provided more varied examples for model training, contributing to better generalization and performance.

We also considered the precision of each model, as for certain applications of sentiment analysis, the cost of having a false positive (FP) is higher than a false negative (FN). For example, flagging positive speech as hate speech on a social media platform is not as detrimental as the converse. Therefore, we should seek to minimize the number of FP, and hence maximize the precision. We used a confusion matrix, a representation of true and false positives and negatives, using which we found that the Naive Bayes Classifier had the best precision of 0.83, given by the ratio of true positives to all positives.

Classification Report:

	precision	recall	f1-score	support
0	0.83	0.77	0.80	1905
1	0.81	0.86	0.83	2163
accuracy			0.82	4068
macro avg	0.82	0.81	0.82	4068
weighted avg	0.82	0.82	0.82	4068

Figure 3: Classification Report of Naive Bayes Classifier

Our results are quite promising when compared to other works that used Naive Bayes Classifier, such as Hariguna et al. (2019) who achieved 0.771 accuracy. Some works such as Baid et al. (2017) achieved similar results of 0.815 accuracy, while others like Ramya & Kumati (2018) achieved high precision of 0.900.

Results of Word2Vec Feature Expansion

The fifth and last experiment used Word2Vec feature extraction with RFC, yielding an accuracy of 0.584, which was significantly lower than expected. From a paper by Kasri et. al. (2022), Word2Vec uses vector spaces to store words based on sentiment, however, it may not accurately reflect relative sentiment polarity of different words based on several factors such as corpus size. This could explain our results when attempting to use Word2Vec, affecting our model's performance.

Conclusion

The incorporation of an enriched dataset notably improved accuracy across all models, emphasizing the importance of data selection. Naive Bayes stands out for its effectiveness in sentiment analysis, with both accuracy and precision as metrics. However, the attempt to enhance Random Forest Classifier performance with Word2Vec features decreased accuracy. Future research could overcome this limitation through integrating diverse models such as Support Vector Machines, in conjunction with deeper linguistic analysis to achieve a more accurate and nuanced understanding of semantic relationships of sentiment in text.

References:

Sentiment140 dataset with 1.6 million tweets. (n.d.). Retrieved April 07, 2024 from https://www.kaggle.com/datasets/kazanova/sentiment140/data

Bahrawi, Bahrawi. (2019). SENTIMENT ANALYSIS USING RANDOM FOREST ALGORITHM ONLINE SOCIAL MEDIA BASED. JOURNAL OF INFORMATION TECHNOLOGY AND ITS UTILIZATION, VOLUME 2, ISSUE 2, 29-33

Muthuvelu, Kavitha & Naib, Bharat & Mallikarjuna, Basetty & Ravindran, Kavitha & Rajkumar, Srinivasan. (2022). Sentiment Analysis using NLP and Machine Learning Techniques on Social Media Data. 112-115. DOI: 10.1109/ICACITE53722.2022.9823708

S. Rizal, Adiwijaya and M. D. Purbolaksono, "Sentiment Analysis on Movie Review from Rotten Tomatoes Using Word2Vec and Naive Bayes," 2022 1st International Conference on Software Engineering and Information Technology (ICoSEIT), Bandung, Indonesia, 2022, pp. 180-185, DOI: 10.1109/ICoSEIT55604.2022.10030009.

SETIAWAN, ERWIN BUDI. "Feature Expansion Word2Vec for Sentiment Analysis of Public Policy in Twitter." Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi) (2022) Vol. 6 No. 1 (2022) 78 – 84 DOI: https://doi.org/10.29207/resti.v6i1.3525

Qader, Wisam & M. Ameen, Musa & Ahmed, Bilal. (2019). An Overview of Bag of Words;Importance, Implementation, Applications, and Challenges. 200-204. 10.1109/IEC47844.2019.8950616.

Antim, Antim. (2023). Application of ML in Text Analysis: Sentiment Analysis of Twitter Data Using Logistic Regression.

Annex:

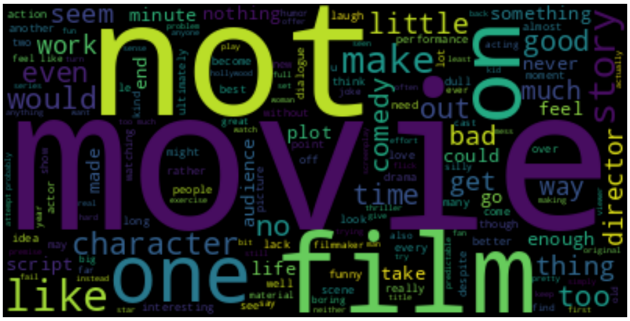


Figure 4: Word Cloud of Negative labelled data

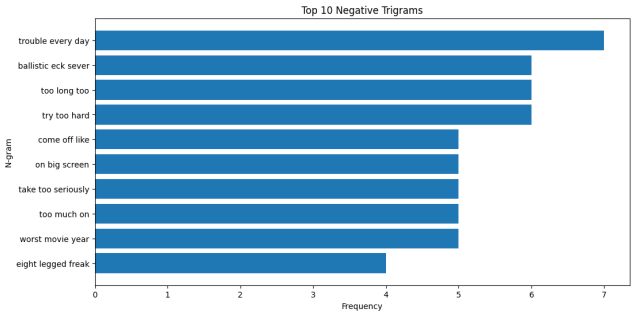


Figure 5: Top 10 Negative Trigrams