IT1244 Group 13: Tweet Sentiment Analysis

Chionh Wan Sim, Hiew Shani, Kushioka Nodoka, Yeo Fu Kai Marcus

Introduction

The task at hand revolves around the development of an AI/ML model with the capability to accurately categorize the sentiment (positive or negative) conveyed within a given tweet. This challenge holds paramount significance as it equips us with the means to comprehend the prevailing public sentiment on diverse subjects, offering substantial benefits for businesses, policymakers, and researchers alike. For businesses, it serves as a tool to gauge customer satisfaction and receive invaluable feedback on their products or services. In the sphere of policymaking, it provides a dynamic gauge of public sentiment towards policies and societal concerns.

We referenced a few previous works to determine appropriate methods to approach this problem. We first referenced Sayar Ul Hassan et al.[2] which provided a clear comparison of machine learning-based algorithms for text classification, which helped us narrow down which methods to utilise. However, little detail on preprocessing was provided. Thus, we referenced Ismail et al.[1], which provided detailed pre-processing steps. However, a limitation was that their models were trained on minimal data points. Lastly, to corroborate, we referenced a final report by Wang et al. [3], which described multiple approaches to tweet sentiment analysis, including non-machine learning methods. They highlighted that the prediction accuracy of models is highly dependent on the classifiers that were taught by the target domain, however, they did not go into detail on how to counter this. We have thus adjusting our pre-processing steps to accommodate the above considerations.

Overall, we used Natural Language Processing (NLP) techniques for feature extraction and thereafter utilised supervised machine learning algorithms that would be suitable for our binary classification task.

Dataset

The dataset we are utilising for this project comprises 100,000 tweets, each accompanied by a sentiment label. These sentiment labels categorise each tweet as either 1, indicating a positive sentiment, or 0, representing a negative sentiment. Notably, within the dataset, we have a balanced distribution of 50,000 instances for both positive and negative labels as shown in Figure 1 below.

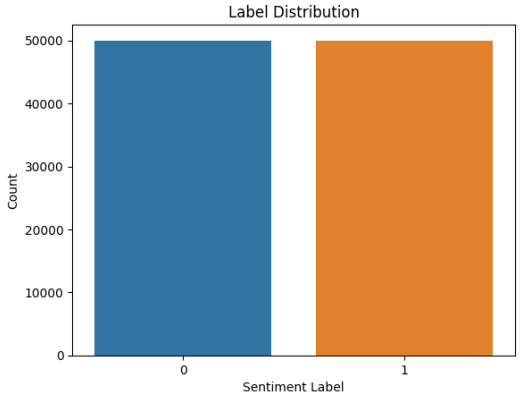

Figure 1: Label distribution of Twitter dataset

Issues with the Dataset

Like any real-world data, this dataset has issues. Some of the common problems with Twitter data include:

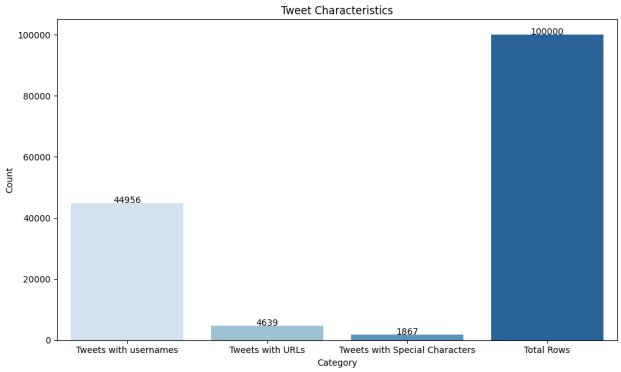

Figure 2: Counts of different characteristics of tweets

1. Noise - Tweets often contain a lot of noise, such as irrelevant words, hashtags, user mentions, and URLs.

```
"@Kenichan I dived many times for the ball. Managed to save 50% The rest go out of bounds"
```

Example 1: Twitter user mentions "@"

```
"@switchfoot http://twitpic.com/2y1zl - Awww, that's a bummer. You shoulda got David Carr of Third Day to do it. ;D"
```

Example 2: URLs in tweets

2. Short Texts - Tweets are limited to a certain number of characters, which can make it challenging to understand the context.

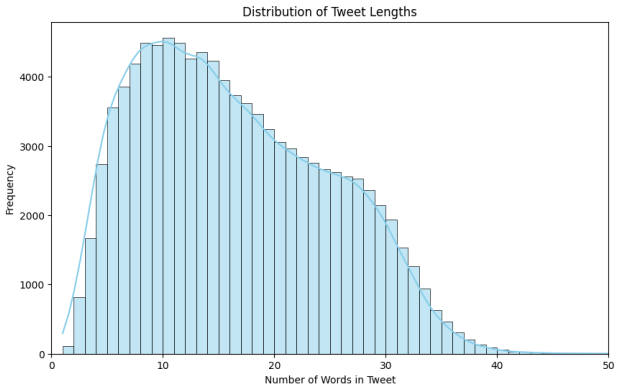

Figure 3: Distribution of tweet lengths

3. Slang and Abbreviations - Twitter data often contains slang, abbreviations, and misspellings which traditional NLP techniques might struggle with.

Data Analysis and Preprocessing

In our pursuit of accurate sentiment analysis, the initial dataset presented several challenges that required meticulous preprocessing to ensure the data's quality and readiness for modelling. Below, we elaborate on the various preprocessing steps we undertook:

Step 1. Removed Usernames: To focus on tweet content, we removed Twitter handles.

Step 2. Eliminated Links: We removed URLs and hyperlinks for a cleaner dataset as they often lack sentiment value.

Step 3. Stripped Punctuations, Numbers, and Special Characters: We removed these elements for data quality enhancement.

Step 4. Tokenization: We broke down the text into tokens for easier analysis.

Step 5. Stemming and Lemmatization: We standardized text by simplifying word variations (stemming) and reducing words to their base form (lemmatization).

These steps ensured consistent and reliable sentiment analysis results by harmonizing words with similar meanings.

Following data preprocessing, we have additionally created word clouds to examine the most frequently occurring words within each sentiment category, as depicted in Figures 4 and 5. This serves as a validation step to confirm the accuracy of our preprocessing efforts.

Figure 4: Most common “Negative” words

Figure 5: Most common “Positive” words

Methods

Before training our models, we first split the data into training (80%) and testing (20%) sets using sklearn’s train_test_split. This ensures we have unseen data to test the model’s performance. The random_state parameter set to 101 ensures the reproducibility of results.

Next, we convert text data into a numerical format that our model can understand. We use TfidfVectorizer from sklearn, which transforms text into a matrix of Term Frequency-Inverse Document Frequency (TF-IDF) features. TF-IDF reflects how important a word is to a document in a collection or corpus.

The ngram_range parameter set to (1, 3) allows us to consider unigrams, bigrams, and trigrams in our sentiment analysis. This broader n-gram range captures more contextual information from the text. We have chosen to include trigrams, which go beyond what some referenced reports have considered. This decision is made to enhance our model's ability to address the challenge of domain dependency. This approach is particularly valuable since our analysis encompasses a wide range of tweets and is not context-specific.

The max_features parameter set to 10000 considers only the top 10,000 most frequent terms, reducing dimensionality.

The stop_words parameter set to 'english' removes common English words that usually don't carry much information.

Finally, we call fit_transform on the training data to learn the vocabulary and idf, and transform it into TF-IDF features. The test data is then transformed into a matrix of TF-IDF features using the learned vocabulary and idf with the transform method. This ensures consistency in the way features are extracted from the training and test data.

Next, we have also shortlisted the following machine learning models to be trained and evaluated:

Logistic Regression

Logistic Regression, a statistical model for binary classification, estimates the probability of a data point belonging to a specific class through a logistic function. Its simplicity and computational efficiency make it an ideal baseline model for tasks like Twitter sentiment analysis, especially when working with large datasets. Additionally, the coefficients in a trained Logistic Regression model offer insights into feature importance. Key hyperparameters include the regularization strength (C) and maximum iteration limit (max_iter).

K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is an instance-based learning algorithm that classifies data points, such as tweets, based on the sentiment of their 'nearest neighbors' in the feature space. It stores all cases and classifies new ones using a similarity measure, like distance functions, assigning a data point to the most common class among its k nearest neighbors. As a non-parametric method, KNN makes no explicit assumptions about the data distribution, offering flexibility in modelling complex relationships and adaptability to different data characteristics. This makes it a versatile model for this project.

Random Forest

Random Forest, an ensemble learning method, constructs multiple decision trees and outputs the mode class. It's chosen for its robustness against overfitting, its ability to handle large, high-dimensional datasets, and its feature importance measure, which indicates which words or n-grams drive sentiment scores. Overfitting is controlled by adjusting the size of the feature subset at each split point, the number of trees, and the tree depth. The key parameters are the number of trees (n_estimators) and the number of features considered for splitting at each leaf node (max_features), with commonly used values of 100 and 'auto' respectively.

Hyperparameter tuning

RandomizedSearchCV and GridSearchCV are hyperparameter tuning techniques with differing approaches. While GridSearchCV exhaustively explores all parameter combinations, it can be computationally expensive, especially for extensive search spaces or slow-training models. In contrast, RandomizedSearchCV efficiently samples a fixed number of candidates from a parameter space using a specified distribution, making it a faster option. Therefore, in this project, RandomizedSearchCV is chosen for hyperparameter tuning due to its efficiency in exploring a vast hyperparameter space and finding an optimal model within an acceptable balance of runtime and performance.

The following flowchart in Figure 6 is a summary of our methodology used.

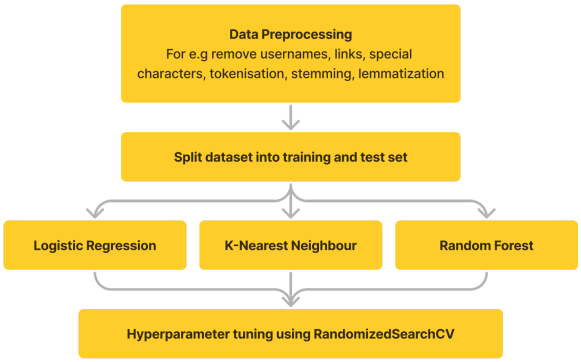

Figure 6: Overview of the methodology used

Results and Discussions

To evaluate the performance of our models, we focused on the use of accuracy, which is the proportion of correct predictions (both positive and negative) out of all predictions made, given by the formula below.

$$Accuracy = \frac{TrueNegatives + TruePositive}{TruePositive + FalsePositive + TrueNegative + FalseNegative}$$

In our tweet sentiment analysis, we aim for balanced classification, prioritizing accurate identification of both positive and negative sentiments. Our focus is on high accuracy to minimize false negatives and positives, reflecting model effectiveness. Despite the value of precision, recall, and F1 score, accuracy remains our key evaluation metric, especially given that we have an equal number of data points for each class.

```
Model: Logistic Regression
Best Hyperparameters: {'C': 1}
Accuracy on Validation Set: 0.75

Evaluation on Test Set:
Accuracy on Test Set: 0.75

Classification Report on Test Set:
              precision    recall  f1-score   support

    Negative       0.77      0.73      0.75     10037
    Positive       0.74      0.78      0.76      9963

    accuracy                           0.75     20000
   macro avg       0.75      0.75      0.75     20000
weighted avg       0.75      0.75      0.75     20000
```

Figure 7: Classification Report for Logistic Regression

```
Model: Random Forest
Best Hyperparameters: {'n_estimators': 175, 'max_depth': None}
Accuracy on Validation Set: 0.74
F1 Score on Validation Set: 0.74

Evaluation on Test Set:
Accuracy on Test Set: 0.74
F1 Score on Test Set: 0.74

Classification Report:
              precision    recall  f1-score   support

           0       0.75      0.73      0.74     10037
           1       0.74      0.75      0.74      9963

    accuracy                           0.74     20000
   macro avg       0.74      0.74      0.74     20000
weighted avg       0.74      0.74      0.74     20000
```

Figure 8: Classification Report for Random Forest

```
Model: K-Nearest Neighbors
Best Hyperparameters: {'n_neighbors': 2}
Accuracy on Validation Set: 0.61

Evaluation on Test Set:
Accuracy on Test Set: 0.61

Classification Report on Test Set:
              precision    recall  f1-score   support

    Negative       0.57      0.85      0.68     10037
    Positive       0.71      0.36      0.48      9963

    accuracy                           0.61     20000
   macro avg       0.64      0.61      0.58     20000
weighted avg       0.64      0.61      0.58     20000
```

Figure 9: Classification Report for K-Nearest Neighbours

Logistic Regression (0.75 accuracy): This model performed the best among the three. An accuracy of 0.75 means it correctly predicted the sentiment of 75% of tweets in the test set. This aligns with the project's goal of accurately determining sentiment. The result signifies that Logistic Regression was effective in distinguishing between positive and negative sentiments within the Twitter dataset.

Random Forest (0.74 accuracy): This model performed slightly worse than the Logistic Regression model but still achieved a respectable accuracy. The small difference might be due to the inherent randomness in the Random Forest algorithm or the specific characteristics of the dataset.

K-Nearest Neighbors (KNN) (0.61 accuracy): This model had the lowest accuracy. KNN models can be sensitive to the high dimensionality of text data and the choice of the 'k' parameter. Its lower performance might suggest that the tweet sentiment classification problem is not well-suited to the KNN algorithm, or that further tuning of the model's parameters is required.

Future Improvements

Given more time, exploring ensemble methods, such as model stacking, could enhance prediction robustness by combining multiple models. Deep learning models, like Recurrent Neural Networks (RNNs), could be investigated for their potential in sentiment analysis. If the dataset is imbalanced, techniques like oversampling, undersampling, or synthetic data generation could improve model performance. Lastly, incorporating sentiment lexicons, annotated lists of words with their sentiment polarity, could provide additional information to the models. These strategies present promising avenues for future research and model improvement.

References

[1] H. Ismail, S. Harous, and Boumediene Belkhouche, "A Comparative Analysis of Machine Learning Classifiers for Twitter Sentiment Analysis," *ResearchGate*, Apr. 03, 2016. https://www.researchgate.net/publication/300169929_A_Comparative_Analysis_of_Machine_Learning_Classifiers_for_Twitter_Sentiment_Analysis (accessed Nov. 05, 2023).

[2] Sayar Ul Hassan, J. Ahamed, and K. Ahmad, "Analytics of machine learning-based algorithms for text classification," *Sustainable Operations and Computers*, vol. 3, pp. 238–248, Jan. 2022, doi: https://doi.org/10.1016/j.susoc.2022.03.001.

[3] Y. Wang, J. Guo, C. Yuan, and B. Li, "Sentiment Analysis of Twitter Data," *Applied sciences*, vol. 12, no. 22, pp. 11775–11775, Nov. 2022, doi: https://doi.org/10.3390/app122211775.

[4] J. Brownlee, "Hyperparameter Optimization With Random Search and Grid Search - MachineLearningMastery.com," *MachineLearningMastery.com*, Sep. 13, 2020. https://machinelearningmastery.com/hyperparameter-optimization-with-random-search-and-grid-search/ (accessed Nov. 05, 2023).

[5] "What Is an N-Gram?," *Mathworks.com*, 2023. https://www.mathworks.com/discovery/ngram.html#:~:text=An

%20n%2Dgram%20is%20a,text%20classification%2C%20and%20text%20generation (accessed Nov. 05, 2023).

[6] A. Mishra, “Metrics to Evaluate your Machine Learning Algorithm,” *Medium*, Feb. 24, 2018. https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234 (accessed Nov. 05, 2023).