Sentiment Analysis on Twitter's Sentiment140

Umar Faruk Abdullahi[1]

Abstract

In this project, we build and evaluate supervised machine learning models for sentiment analysis using the TwitterSentiment140 dataset. We begin by training four different machine learning models using two different feature engineering techniques. We conclude by fine-tuning a pre-trained BERT model on the dataset and perform comparison of performance of the models. The project intricately details the data preprocessing steps, reasons for choice of approaches and provides a comprehensive analysis of the performance results associated with the selected machine learning models.

Link to Google Colab. Link to Fine-tuned Sentiment Analysis Model.

Keywords: Data Science, Machine Learning, Sentiment Analysis, NLP

1. Introduction

A vital signal to individuals, companies and governments alike remains the sentiments that others do hold. As a social species, humans survive by navigating the complex relationships of the world through understanding the thoughts, speeches or broadly, the sentiments of others. To this end, sentiment analysis remains an important problem in NLP in understanding the meaning of texts.

Perhaps the most benefit of sentiment analysis is derived by businesses, by analyzing customer feedback, sentiment analysis enables businesses to gain insights into consumer preferences, track brand reputation, and make data-driven decisions to improve products and services. It also aids in brand monitoring, reputation management, and crisis mitigation by identifying and addressing issues quickly. Additionally, sentiment analysis plays a crucial role in market research, helping businesses identify market trends, competitor strategies, and emerging opportunities. Beyond business applications, sentiment analysis is utilized in finance for risk management, in government for public opinion analysis, and in customer service for enhancing support interactions.

In this project, data is collected from the activities of users on the social media platform - Twitter. The project's objective was to understanding the processes involved in sentiment analysis, selection of appropriate text processing and modelling techniques including state-of-the-art models and finally, discussing the difference in performance between these approaches.

In the final phase of the project, we compared the performance of the trained models using different approaches.

2. Data

In this phase of the project, we identify the type of the data and distribution of the features with respect to the target variable. The dataset is a publicly available well know dataset for sentiment analysis by Twitter named the Sentiment140 dataset. The data collected was already cleaned to contain only two features:

- The text of the tweet
- Sentiment of the tweet - **Positive** or **Negative**

We treat this problem then, as a binary classification supervised learning problem where the model learns how to accurately the predict the outcome label - **Sentiment**.

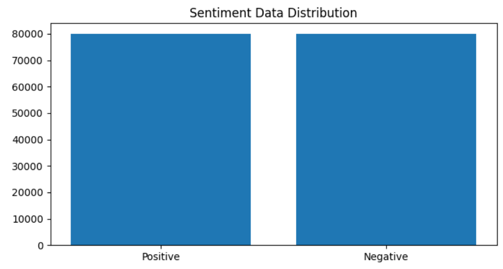

Figure 1: Sentiment distribution

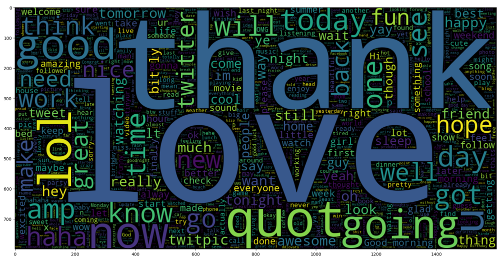

Figure 2: Positive sentiments

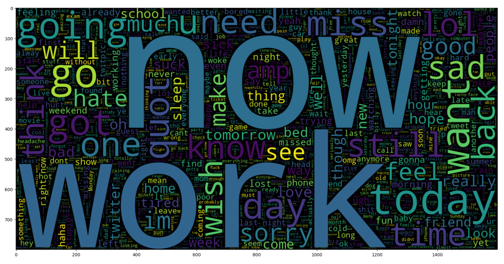

Figure 3: Negative sentiments

2.1. *Data Visualization*

Since the dataset consisted of only textual data, comprehensive data analysis is not feasible in this project. However, we display the distribution of the sentiments in Figure 1. From the figure, we could discern the distribution between the classes as even. In binary classification, achieving an even distribution of labels, where each class is represented equally in the dataset, is advantageous for several reasons. Firstly, it ensures balanced training, preventing the model from becoming biased toward the majority class and enabling it to effectively learn to distinguish between classes. Secondly, an even distribution helps prevent bias in model predictions, ensuring fair treatment of all classes. Additionally, models trained on balanced datasets tend to generalize better to unseen data and are more reliable in real-world scenarios where class distributions may vary. Balanced datasets also provide a fair basis for evaluating model performance across all classes, reducing the risk of misinterpretation.

We also use WordCloud to highlight the most commonly occurring words in the two sentiment classes.

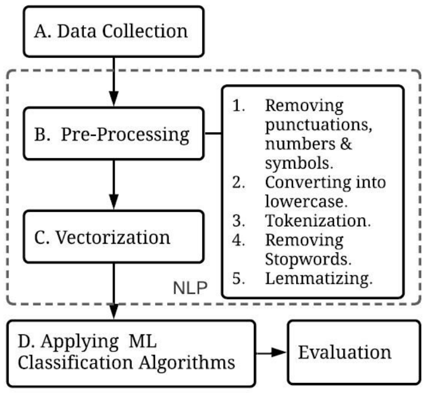

Figure 4: Sentiment Analysis Workflow

2.2. *Data Preprocessing*

In this stage, which can be argued as the most important step in any Natural Language Processing (NLP) project, we perform text processing to clean our text and use standard NLP techniques to prepare the text for further vectored processing to train our machine learning models. The steps performed in this stage are as follows:

1. Removal of unwanted characters: Since the source of the text is Tweets, one of the most common character occurances is then the **@** sign which signifies usernames. We remove the tag and subsequently remove unwanted spaces and single character words. This is because a single character words would contain no context and therefore not be useful for a machine learning model.
2. Tokenization: This refers to the process of breaking down a text or a sequence of characters into smaller units called tokens. These tokens can be words, phrases, symbols, or other meaningful elements, depending on the context and the specific task at hand. Tokenization is a fundamental step in NLP preprocessing because it enables computers to understand and process human language by treating each token as a discrete unit.
3. Text Normalization: The most common text normalization techniques are stemming and lemmatization. **Stemming** refers to the process of reducing words to their root or base form, known as the stem while **Lemmatization** which the process of reducing words to their base or dictionary form, known as the lemma. Unlike stemming, which simply chops off affixes from words, lemmatization takes into account the morphological analysis of words and aims to transform them into their canonical or dictionary form. We proceed with Lemmatization.

We then store the clean tweets in a separate column for further processing.

Figure 4 shows an example of a standard sentiment analysis workflow in NLP.

2.3. *Vectorization*

In this section, we perform processing with various techniques and then proceed to train machine learning models based on the engineered features.

1. TF-IDF: The TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer is a powerful tool in natural language processing (NLP) used for extracting features from text data. It operates by transforming a collection of raw text documents into a matrix of TF-IDF features. TF-IDF consists of two main components: Term Frequency (TF) and Inverse Document Frequency (IDF). TF measures how often a term appears in a document relative to the total number of terms in that document, while IDF measures the importance of a term across the entire corpus, assigning higher weights to terms that are rare across documents but common within a specific document. The TF-IDF score of a term in a document is the product of its TF and IDF scores, indicating the term's importance in the context of the entire corpus.
2. Word2Vec: This is a technique in natural language processing that helps computers understand the meaning of words by representing them as numerical vectors. Word2Vec learns these representations by analyzing large amounts of text data. It does this by training neural network models to predict words based on their surrounding context or to predict context words given a target word. As a result, similar words end up with similar vector representations in a continuous vector space. These word embeddings capture semantic relationships between words, enabling computers to better understand and process natural language.

3. Model Training and Evaluation

After carefully selecting the most important features and pre-processing the data for our machine learning task, we proceed to selecting and training four supervised machine learning algorithms and one deep-learning finetuned BERT model. For this project, we selected the following machine learning paradigms:

1. Regression: **Logistic Regression**
2. Support Vector Machines: **Linear SVC**
3. Ensemble Learning: **Random Forest**
4. Linear Models: **SGD Classifier**
5. Deep Learning Transformer Model: **BERT**

For the first four machine learning models, we trained two variants using the already highlighted vectorization techniques discussed above.

3.1. Machine Learning Models

1. Logistic Regression: With the TF-IDF vectorizer, the logistic regression model achieved an accuracy of 78% making it the most performant of the models. The model's accuracy slightly dropped to 76% accuracy using Word2Vec vectorizer. A summary of the results is shown in Table 1
2. Linear SVC: The model achieved an accuracy of 78% with Tf-IDF and noticed its performance slightly decrease by 2% using the Word2Vec.
3. Random Forest: Using the TF-IDF vectorizer, a random forest classifier was trained on the samples with a balanced class weight, maximum depth of 10 and 100 estimators. The model achieved 73% accuracy. With the Word2Vec vectorizer, the model performed the same.
4. SGD Classifier: A Stochastic Gradient Descent Classifier was also trained using log-likelihood error, an L2 regularization parameter and learning rate of 1e-4. The model performed somewhat similar to the already trained model exhibiting 75% accuracy on both Tf-IDF and Word2Vec.

Table 1: Comparison of Model Accuracy

Model	**Tf-IDF (%)**	**Word2Vec (%)**
Logistic Regression	78	76
Linear SVC	78	76
Random Forest	73	73
SGD Classifier	75	75

3.2. Deep Learning with BERT

The techniques we tried above are based on traditional machine learning approaches. To further enhance the performance of our model, we explore the usage of state-of-the-art pre-trained deep learning models that we will fine-tune with our data.

Fine-tuning is a process in machine learning where a pre-trained model is further trained on a specific task or dataset to adapt its knowledge and parameters to better suit that particular task. Instead of training a model from scratch, fine-tuning starts with a model that has already been trained on a large dataset for a general task, in our case - natural language processing. Then, the model's parameters are adjusted or fine-tuned by training it on a smaller, task-specific dataset like the TwitterSentiment140 in our project.

To achieve this, we will be using a model called BERT. BERT, short for Bidirectional Encoder Representations from Transformers, is a pre-trained language model created by Google in 2018. It's really good at understanding the meaning of words in a sentence because it looks at both the words before and after each word. BERT is trained on a lot of text data using a special technique, and then it can be used for different tasks like understanding what a sentence means, finding important information in a paragraph, or answering questions. We will be employing BERT as the base model for our tokenization and modelling process through using popular machine learning libraries from HuggingFace.

The steps are listed below:

1. Load Pre-trained BERT Model
2. Prepare Data: This involves formatting the dataset in a way that BERT can understand. We tokenized the text data and converted it into input features that BERT expects.
3. Fine-tune BERT Model: We fine-tune the pre-trained BERT model on our converted Twitter dataset using the 'Trainer' module from the Hugging Face library.
4. Evaluate Model Performance: Lastly, we evaluate the performance of the fine-tuned BERT model using a separate validation dataset to assess its accuracy and effectiveness on our task.

For this task, we selected a variant of the BERT model - DistilBert that has been shown to be more efficient while demonstrating similar performance to the base BERT model. The model performed better than all the previously trained machine learning models. It achieved an accuracy of 82% which also demonstrating superior precision and recall scores. This further elucidates the ability of deep learning models to learn intricate features better than traditional machine learning models.

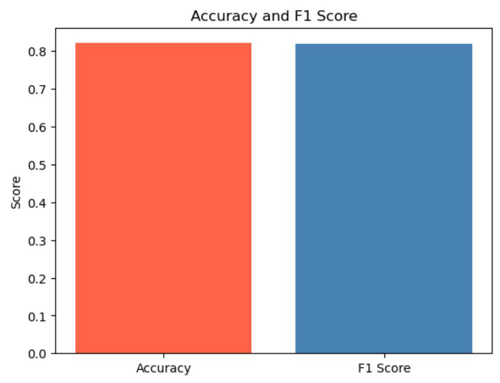

Figure 5: Fine-tuned BERT

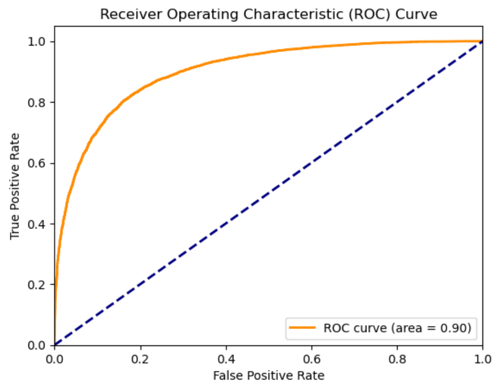

Figure 6: Fine-tuned BERT ROC-AUC

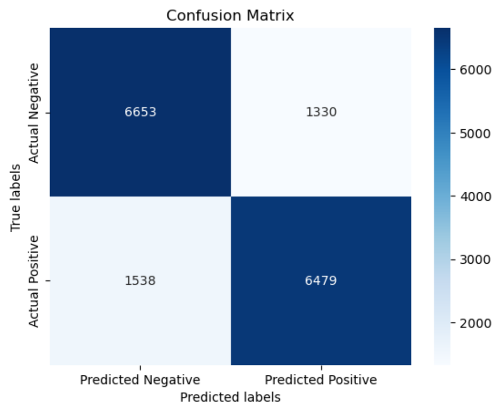

Figure 7: Fine-tuned BERT Confusion Matrix

4. Discussion

The findings from our study clearly demonstrate that the deep learning model outperformed all previously trained machine learning models, achieving an impressive 82% accuracy rate. This success highlights the significant advantages of deep learning in natural language processing (NLP), especially in tasks like sentiment analysis. Unlike traditional machine learning methods, deep learning excels at recognizing complex patterns and connections within text data. It can understand context, grasp long-distance relationships, and pick up on subtle meanings in language, resulting in more accurate and nuanced text representations. Additionally, deep learning models benefit from pre-trained language models like BERT, which have been trained on vast amounts of text data. By leveraging this prior knowledge, deep learning models can enhance their performance on tasks like sentiment analysis, as demonstrated in our project. In summary, the effectiveness of deep learning in NLP lies in its ability to understand and interpret text data with precision and efficiency, making it a powerful tool for various applications.

5. Conclusion

In this project, we began by performing the basic text preprocessing steps in natural language processing. We then pre-processsed the data to a format compatible with machine learning models for text classification and then trained four machine learning algorithms using two different approaches and finally, compare the performance of the models. We then fine-tuned a transformer based deep learning model - BERT on our dataset which demostrated superior performance to all the previously trained deep learning models. This further underscores the ability of deep learning to perform better than machine learning models. In our case, the models training on large corpus of data gives it a greater ability to specialize on our sentiment analysis task after fine-tuning.