Sentiment Analysis Project Documentation

Omar Hani Elsayed

July 20, 2024

---

1 Introduction

Sentiment analysis aims to determine the sentiment expressed in textual data. This project focuses on performing sentiment analysis on a dataset of tweets, categorizing them into positive, negative, or neutral sentiments. The objectives are to preprocess the data, handle class imbalance, and compare the performance of traditional machine learning models and deep learning models.

2 Data Description

The dataset used for this project is the TweetEval sentiment dataset from Hugging Face. It consists of labeled tweets with sentiments classified as positive, negative, or neutral. The dataset is divided into training, validation, and test sets as follows:

- Training set: 45615 instances
- Validation set: 2000 instances
- Test set: 12284 instances

	text	label
0	"QT @user In the original draft of the 7th boo...	2
1	"Ben Smith / Smith (concussion) remains out of...	1
2	Sorry bout the stream last night I crashed out...	1
3	Chase Headley's RBI double in the 8th inning o...	1
4	@user Alciato: Bee will invest 150 million in ...	2

Figure 1: Example of a subset from the dataset

3 Baseline Experiments

3.1 Goal

The goal of the baseline experiments is to establish initial performance benchmarks for sentiment analysis using traditional machine learning models. By doing so, we can compare the effectiveness of these models

against more advanced deep learning approaches. The baseline models include **Logistic Regression, SVM, Random Forest, Naive Bayes, and Gradient**

3.2 Methodology

The methodology for establishing baseline performance consists of the following steps:

3.2.1 Data Preprocessing

The first step in the baseline experiments is to preprocess the text data. This involves several sub-steps:

1. **Text Cleaning**: The raw tweet texts are cleaned by removing mentions, URLs, special characters, and stop words. The text is also converted to lowercase to ensure uniformity.
2. **Tokenization and Vectorization**: The cleaned text is tokenized, converting it into individual words or tokens, and then vectorized using TF-IDF (Term Frequency-Inverse Document Frequency) to transform the text data into numerical form. This step captures the importance of words in the context of the entire dataset.
3. **Address class imbalance using SMOTE and data augmentation techniques:**

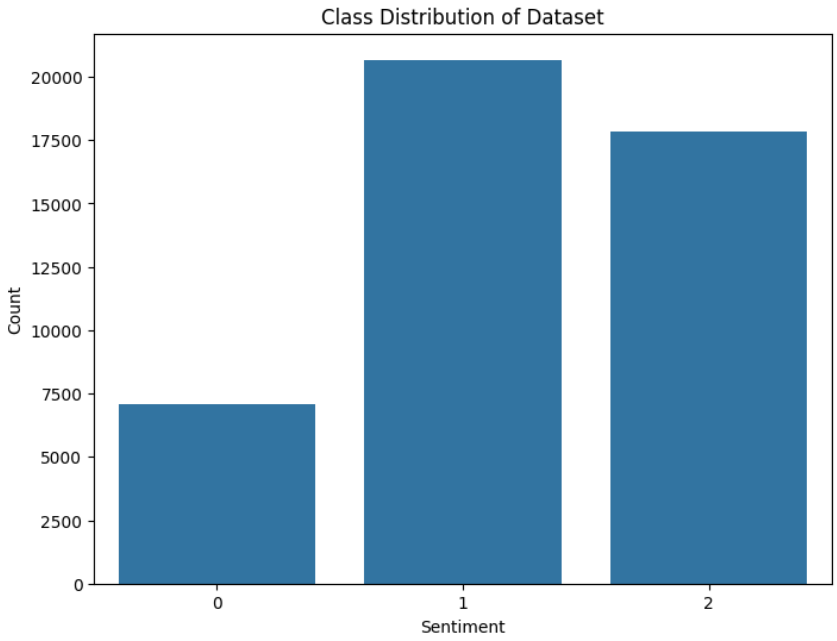


Figure 2: Class Distribution of Dataset

- SMOTE (Synthetic Minority Over-sampling Technique): Generate synthetic samples for the minority class to balance the class distribution. This helps the models to learn equally well from all classes and prevents bias towards the majority class.
- Data Augmentation: Apply techniques such as synonym replacement, random insertion, and random deletion to create additional training examples. This increases the diversity of the training data and helps the models generalize better.

3.3 Model Training

In the baseline experiments, several traditional machine learning models were trained to establish performance benchmarks for sentiment analysis. Each model was chosen for its unique strengths and suitability for text classification tasks.

The following models were trained:

Model	**Description**
Logistic Regression	A linear model that predicts class membership probabilities using a logistic function. It performs well on text data due to its ability to handle large feature spaces.
Support Vector Machine (SVM)	Finds the hyperplane that best separates the classes in the feature space. Effective in high-dimensional spaces and robust to overfitting.
Random Forest	An ensemble method that builds multiple decision trees and outputs the mode of the classes. Robust and handles large datasets with higher dimensionality.
Naive Bayes	A probabilistic classifier based on Bayes' theorem, assuming independence between features. It is efficient and often performs well in text classification.
Gradient Boosting	An ensemble technique that builds models sequentially, with each new model correcting the errors of its predecessor. Known for its high accuracy and ability to handle various data types.
XGBoost	An optimized version of gradient boosting that includes advanced features like regularization, parallel processing, and handling of missing values. It is highly efficient and effective for classification tasks.

Table 1: Summary of Machine Learning Models Used for Sentiment Analysis

3.4 Results

The results of the baseline models using SMOTE and data augmentation techniques are presented below. The first figure shows the test accuracy of different models after applying SMOTE, while the second figure shows the test accuracy after using data augmentation.

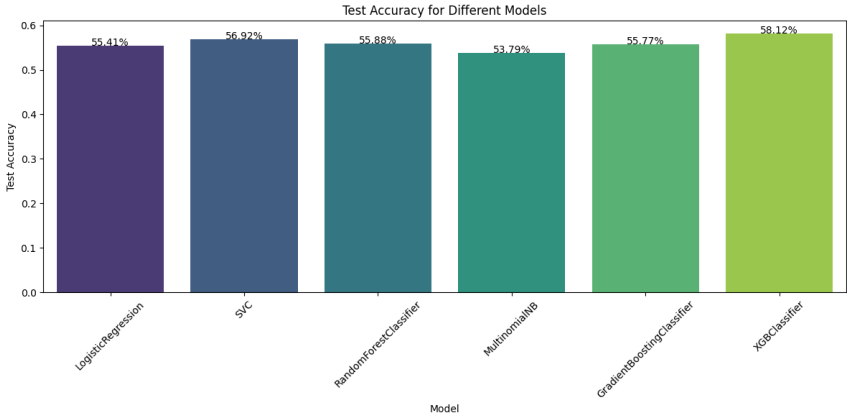


Figure 3: Test Accuracy of Models with SMOTE

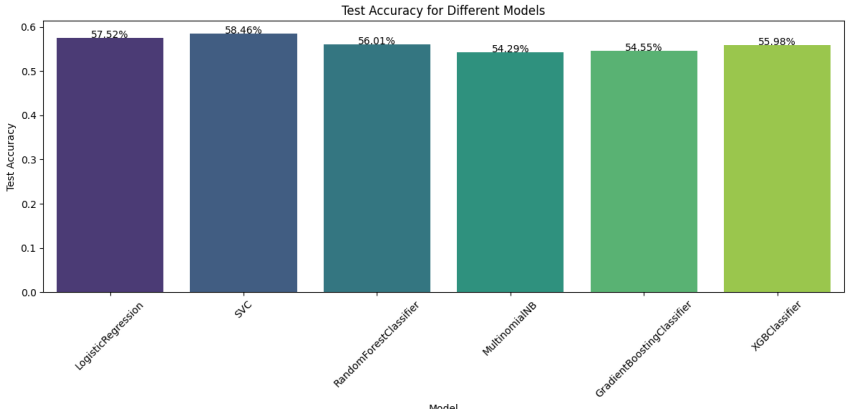


Figure 4: Test Accuracy of Models with Data Augmentation

The test accuracy of the models remained relatively low, even after applying SMOTE and data augmentation techniques. This could be due to several factors:

- **Class Imbalance**: Despite efforts to balance the classes, the intrinsic complexity and nuances of tweet sentiments may still pose a challenge.
- **Model Limitations**: Traditional machine learning models might not be capturing the contextual information effectively, which is crucial for sentiment analysis.

3.5 Conclusion

Traditional machine learning models provide a good starting point but may not capture the complexities of textual data effectively.

4 Advanced Experiments

4.1 Experiment 1: Deep Learning Models

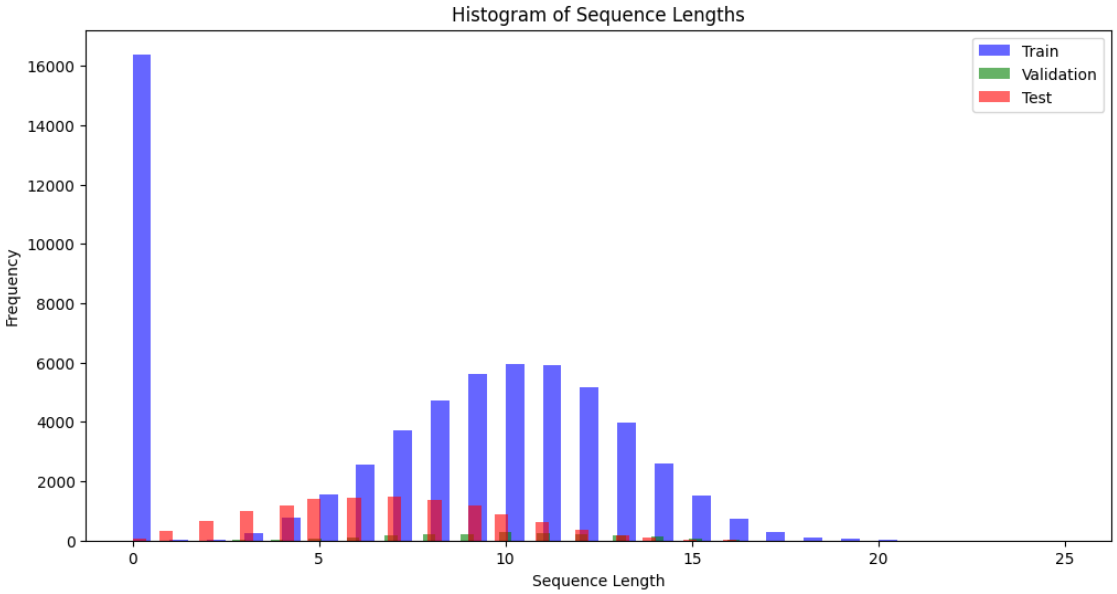
4.1.1 Goal

Improve performance using various deep learning models with different strategies for handling text data and leveraging pre-trained embeddings and transformer models.

4.1.2 Methodology

1. **Data Preparation:**

   - **Data Cleaning:** The text data is cleaned by removing mentions, URLs, special characters, and stop words.
   - **Data Augmentation:** Augment the data using NLP augmentation techniques from the 'nlpaug' library. The methods used include:
     - Synonym Replacement
     - Random Insertion
     - Random Deletion
   - **Handling Class Imbalance:** Address class imbalance by augmenting the minority classes until the dataset is balanced.
   - **Tokenization and Vectorization:**
     - Tokenize the text and convert it into sequences.
     - Apply padding and truncation based on the maximum sequence length, which is determined as the 98th percentile of sequence lengths across the entire dataset.
     - Below is the histogram of sequence lengths with the 98th percentile indicated:



   - **Data Conversion:** Convert the tokenized data into TensorFlow datasets with shuffling and prefetching for efficient training. Shuffling ensures that the data is presented in a different order each epoch, which helps the model generalize better. Prefetching allows the data to be loaded in the background while the model is training, improving training speed and efficiency.

2. **Model Training:** Various deep learning models were trained and validated, including:

Model	Description
BiLSTM Model	The model includes an embedding layer, followed by bidirectional LSTM layers, and fully connected layers with dropout and batch normalization.
Word CNN Model	The model includes an embedding layer, convolutional layer, global max pooling, and fully connected layers with dropout and batch normalization.
Transformer Model	The model includes self-attention mechanisms, token and position embedding layer, transformer blocks, global average pooling, and fully connected layers.
Enhanced Transformer Model	An advanced transformer model with multiple transformer blocks and fully connected layers.
Universal Sentence Encoder Model	This model is pretrained and obtained from TensorFlow Hub. It utilizes the Universal Sentence Encoder followed by fully connected layers.
Hybrid CNN-LSTM Model	The model includes an embedding layer, convolutional layer, bidirectional LSTM layers, and fully connected layers.
Model with Pre-trained Embeddings	The model uses pre-trained embeddings, convolutional layers, bidirectional LSTM layers, and fully connected layers with dropout.

Table 2: Deep Learning Models Used

4.1.3 Results

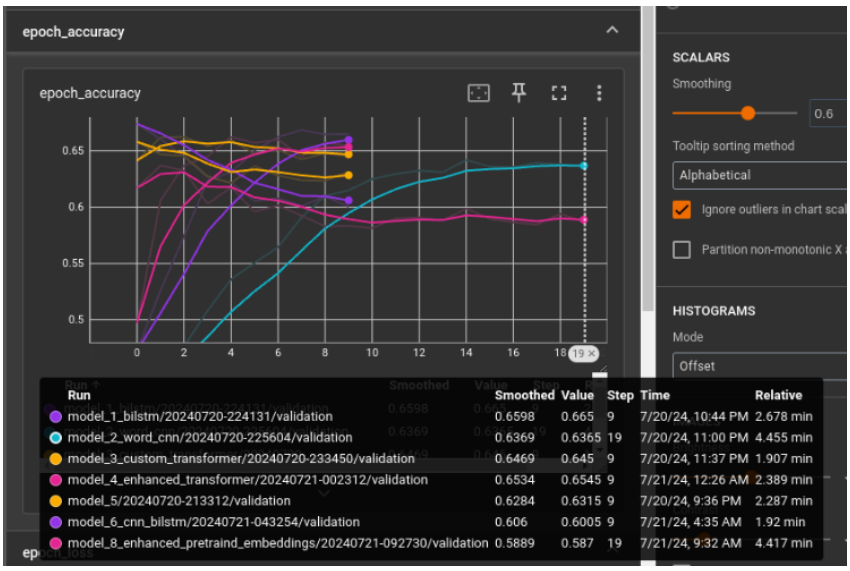


Figure 5: Test Accuracy of Models with Data Augmentation

4.1.4 Conclusion

- Pre-trained embeddings and deep learning models can capture more complex patterns in text data, resulting in improved performance.
- Although the accuracy of the models is not very high despite using some pre-trained models, this highlights the challenges in creating effective models from scratch.
- However, using transformer-based models, such as DistilBERT, demonstrated the potential for achieving higher performance due to their ability to capture contextual information effectively.
- The use of transformers is a promising approach and represents a significant advancement in the field of sentiment analysis.

4.2 Experiment 2: Transformer-Based Model (DistilBERT)

4.2.1 Goal

Leverage transformer-based models for better contextual understanding in sentiment analysis.

4.2.2 Methodology

1. **Data Preparation:**
    - **Data Cleaning:** The text data is cleaned by removing mentions, URLs, special characters, and stop words.
    - **Data Augmentation:** Augment the data using NLP augmentation techniques from the 'nlpaug' library.
    - **Handling Class Imbalance:** Address class imbalance by augmenting the minority classes until the dataset is balanced.
    - **Tokenization and Vectorization:**
        - Tokenize the text using the DistilBERT tokenizer.
        - Apply padding and truncation based on the maximum sequence length
        - **Data Conversion:** Convert the tokenized data into TensorFlow datasets
    - **Model Training:** The DistilBERT model is fine-tuned for sequence classification. The model architecture and hyperparameters are as follows:
        - **Model Architecture:** Fine-tune the DistilBERT model for sequence classification with a dense layer for classification.
        - **Optimizer:** AdamW optimizer with a learning rate of 5e-5.
        - **Loss Function:** Sparse Categorical Crossentropy.
        - **Training Parameters:** Trained for 3 epochs with a batch size of 16.

4.2.3 Conclusion

- Transformer-based models like DistilBERT significantly outperform traditional and other deep learning models for sentiment analysis due to their ability to capture contextual information effectively.

- However, training such models requires significant computational resources. In our case, the lack of adequate GPU resources prevented us from completing the training process, thereby limiting our ability to fully assess the model's performance.

5 Overall Conclusion

In this project, we explored two primary approaches for sentiment analysis: traditional machine learning models and advanced deep learning models. Each approach was implemented and evaluated based on various metrics, and the results provided valuable insights into the strengths and limitations of each method.

- **Traditional Models:** Easier to implement and interpret but limited in handling complex patterns and context.
- **Deep Learning Models:** Showed superior performance, especially with transformer-based models, but required more computational resources and extensive experimentation.
- **Pre-trained Models:** Leveraging pre-trained embeddings and transformer models proved to be highly effective in capturing the nuances of sentiment in text data.
- **Data Augmentation:** Essential for addressing class imbalance and improving model robustness, particularly in deep learning approaches.

5.1 Tools and Libraries Used

- Python
- Jupyter Notebook
- scikit-learn
- imbalanced-learn
- TensorFlow
- Keras
- PyTorch
- Hugging Face Transformers
- NLPaug

5.2 External Resources and Pre-trained Models Used

- GloVe pre-trained embeddings
- DistilBERT model from Hugging Face

6 Reflection Questions

1. What was the biggest challenge you faced in implementing sentiment analysis?

The biggest challenge was addressing the class imbalance in the dataset. Building effective deep learning models from scratch that could match the performance of benchmark models was challenging and required extensive experimentation.

2. What insights did you gain about NLP and sentiment analysis through this project?

Through this project, I gained a deeper understanding of how different models handle textual data and the importance of preprocessing steps. I learned that while traditional models provide a good baseline, advanced models like transformers significantly improve performance by capturing contextual information more effectively. Moreover, working with pre-trained transformer models in the future will likely yield better results, as they leverage vast amounts of data and fine-tuning to achieve high performance, making them more reliable for complex NLP tasks.