Introduction

Sentiment analysis is the process of using natural language processing (NLP) and machine learning techniques to automatically identify and extract subjective information from text data. The goal of sentiment analysis is to determine the attitude, opinion, or emotion expressed by a writer towards a particular topic or product. In this paper, I will explore the task of performing binary classification of Movie Reviews in the training dataset and to predict the Outcome (value of 1 or 0) of ratings in the test dataset.

Data Preprocessing

The data preprocessing part in my code is the section that includes the *'clean_tweet'* function and the subsequent code where *'df'* and *'df_test'* are read and cleaned using the *'clean_tweet'* function. The *'clean_tweet'* function performs various cleaning operations, such as removing HTML special entities, URLs, RTs, twitter handles, punctuation, special characters and numbers from the tweets, converting them to lowercase, and removing additional white spaces.

I decided to do this due to it being important in the case of sentiment analysis. This involves cleaning the text data and preparing it for feature extraction, which is a crucial step in training a machine learning model. By cleaning the text data, we can ensure that the model focuses only on the relevant information in the text, improving its accuracy and performance.

Feature Engineering

The feature engineering part in my code is where the data is tokenized and encoded using the pre-trained transformer model in *'AutoTokenizer'* and *'tokenizer'*, respectively. The encoded data is then converted into tensors and used to create *'LocalDataset'* instances, which are then used to create *'DataLoader'* instances for training and validation. The *'collate_fn'* function is used to pad the encoded data to a fixed length and create a batch of tensors suitable for model training. Finally, the *'Classifier'* class is defined, which uses the pre-trained transformer model to encode the input data and a linear layer to classify the encoded data.

I decided to do this due to that, overall, the feature engineering part is crucial for preparing the data for the machine learning model to learn from it and make accurate predictions.

How I performed validation of my predictions

My code includes a section where the model predictions are evaluated using validation data. After each epoch of training, my model's performance is evaluated using the validation set. The validation set is loaded using a *'DataLoader'* instance, and the model is evaluated on this set using the same metrics used for training. The metrics used for evaluation are the same as those used for training, which are the binary cross-entropy loss and accuracy. The code calculates the average loss and accuracy over the entire validation set and prints these values at the end of each epoch. This process allows the model's performance to be monitored during training and helps prevent overfitting.

Model exploration

Here I used the accuracy rate for the validation dataset as model measurement. The best model for this task will be the one that has the highest accuracy.

- Firstly, I selected CNN, CNNs have been shown to be effective in various natural language processing tasks, including sentiment analysis. So it is possible that a CNN model trained on the same dataset and with appropriate hyperparameters may perform better than others. The final accuracy of my CNN model is about 82%.
- Secondly, I selected ANN, which is a type of deep learning algorithm that can be used for a wide range of tasks including image classification, natural language processing, and time series prediction. ANN works by creating a network of interconnected nodes that can learn and adapt to patterns in the data. It's possible that ANN could perform better than others. The final accuracy of my CNN model is about 71%.
- Finally, I selected a pre-trained sentiment analysis model based on RoBERTa architecture. This model was trained on a large dataset of tweets to predict the sentiment of the tweets.This model has the highest accuracy rate (92%) for the validation dataset as model measurement and this is the model that I finally submitted.

Citation for the code I used:
Abdal, R., & Goyal, A. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv preprint arXiv:1907.11692.Hugging Face. (n.d.). RoBERTa. https://huggingface.co/roberta-base

Extra Analysis / Interesting Findings

We can see that the model achieves an accuracy of around 92% on the validation set, which is a good result. However, some interesting findings could be extracted from the confusion matrix, which can provide more insights into the model's performance.

By looking at the confusion matrix, we can see that the model misclassified more tweets with a negative sentiment as positive sentiment (*False Positive*) than the opposite (*False Negative*). This indicates that the model may be biased towards predicting positive sentiment, which could be further investigated and improved by adjusting the model's hyperparameters or training with a more balanced dataset.

Another interesting analysis could be to investigate the most common words or phrases used in the tweets and how they affect the sentiment prediction. This could be done by performing a frequency analysis or using techniques such as *'TF-IDF'* to weigh the importance of words in the dataset. By understanding the most influential words or phrases, we could further improve the model's accuracy and make more accurate predictions.

My code references the following websites:

Kaggle. (2021). NLP Getting Started - Disaster Tweets. Retrieved April 7, 2023, from https://www.kaggle.com/c/nlp-getting-started

Twitter US Airline Sentiment Dataset, Accessed on Kaggle, https://www.kaggle.com/crowdflower/twitter-airline-sentiment

Bentrevett. (2021). pytorch-sentiment-analysis [Computer software]. GitHub. https://github.com/bentrevett/pytorch-sentiment-analysis