Sentiment Analysis Using Deep Learning

Task Overview

The goal of this task is to create a sentiment analysis model that is capable of taking user inputs and predicting their sentiment. To create a model that can generalise well and be effective in many situations, the IMDB and Twitter sentiment datasets will be used. This will allow the models to learn from a larger variety of contexts to attempt to improve the model. This work will test 3 different models for sentiment analysis including standard logistic regression, linear SVC and a bidirectional LSTM model.

Initial Data Exploration

The datasets were loaded into Pandas DataFrames and resulted in the following findings:

- IMDB dataset consisted of 40000 rows and 2 columns, with a text column and label column.
- Twitter dataset consisted of 74682 rows and 4 columns, which included a text, context, tweet_id and label columns
- The IMDB labels consisted of 2 values, 0 and 1 indicating negative and positive respectivelly.
- The Twitter labels consisted of 4 different values being, positive, negative, neutral and irrelevant.

To create the full dataset, the context and tweet_id columns were dropped from the twitter dataset and the labels were changed to be 0 and 1 for negative and positive. The entries relating to neutral and irrelevant were dropped as this task was creating a binary classifier. This resulted in the final dataset containing 114682 rows and 2 columns. There was 42561 entries labelled negative and 40813 labelled positive which is an almost even class split.
The 2 classes can be visualised with wordclouds to show what sort of words are mainly present in these classes, as seen in Figure 1. These were generated using the nltk set of stopwords with some additions such as movie and film which are in a lot of texts from the IMDB dataset. TF_IDF was also applied so it shows the words with the largest impact on the text.

(a) Word cloud of positive sentiment texts

(b) Word cloud of negative sentiment texts

Figure 1: Word clouds illustrating positive and negative sentiment distributions

Model Selection

Before creating a model, the text various preprocessing methods. Firstly, it was stripped of all html tags, which were mostly found from twitter entries, and removed all non-lexical entries such as punctuation and numbers. Then only words that are not in the common stopwords were excluded to focus on the words that provide more information for the models to learn from. The next step was to apply a lemmatizer to ensure that misspelt words are treated as their correctly spelt counterpart to ensure the model learns more efficiently. This was then tokenized and the tokens were stored in a column named tokens that is ready to be used by the models.
For both the logistic regression model and the linear SVC model, the tokens column was TF_IDF vectorized with max features of 20000 to allow the models to learn. These models ended up with test accuracies of 86.4% for logistic regression and 87.6% for linear SVC while being very fast to learn.
The deep learning methods consisted of a deep neural network that can be seen in Figure 2

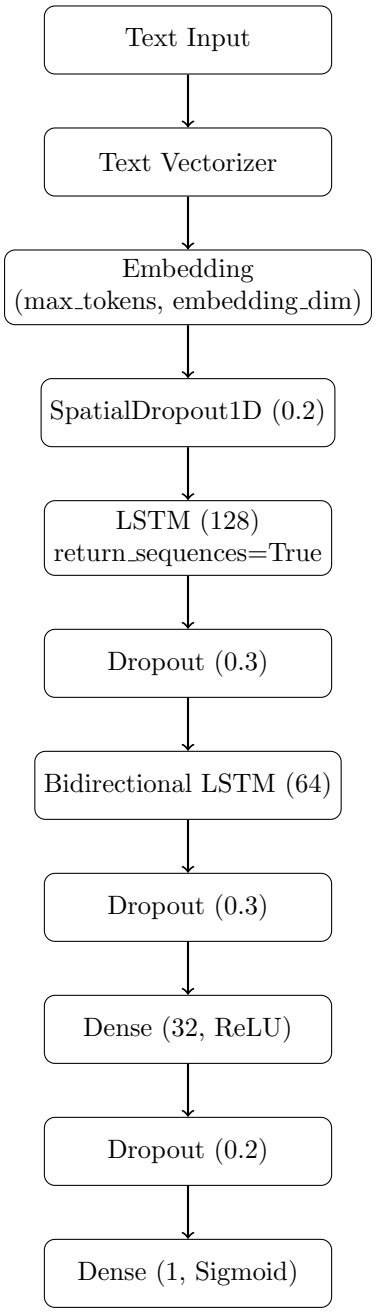

Figure 2: Recurrent neural network architecture for sentiment classification

This RNN architecture begins with a text vectorization layer that converts raw text into numerical representations, followed by an embedding layer that captures semantic relationships between words. The model then employs an LSTM layer and a bidirectional LSTM layer to learn sequential dependencies in both forward and backward directions, ensuring that contextual information is fully captured. Dropout layers are incorporated as a regularization technique to mitigate overfitting, while the final dense layers perform nonlinear transformations and generate the final sentiment prediction.
This model was trained for 10 epochs which took a total of 1 hour 41 minutes 55 seconds, with the training accuracy reaching 96.4% and validation accuracy reaching 88.3%, with signs of overfitting appearing after the 5th epoch. This model achieved 88.3% accuracy on the final test data which is the highest overall. Alongside this, a predict sentiment function was added which can label the sentiment of any given text alongside a confidence percentage.

Key Findings, Insights and Next Steps

Overall the deep learning model performed slightly better than the non-deep learning models for this task however took much longer to train ($\tilde{1}0$ seconds vs $\tilde{2}$ hours). However for this task, the final accuracy is the most important as the goal is to find a sentiment analysis model that can most accurately predict sentiments. The process of using the model to predict is fast so the initial increase in training time is a worthy trade-off.
The next steps are to deploy the model and observe it to see any problems that can occur while continuously improving through further hyper-parameter optimization and model changes.
While the logistic regression and linear SVC models did not perform as well, they could be further optimised by hyperparameter tuning and by using more data. All evaluated models are suitable for this task, with the optimal choice depending on specific application requirements such as computational efficiency and performance constraints.