CHKTAP011 - NLP Assignment 1

Implementation

A small amount of data processing was required. This involved removing samples classified as neutral and removing unneeded symbols like words and URLs. For tokenization, I implemented both Byte Pair Encoding (BPE) and word-based tokenization. The tokenized sentences were further converted to binary vector embeddings. This improved performance by simplifying arithmetic and reducing the amount of memory required to represent the features.
There are three main steps in my Naive Bayes classifier

1. Counting - I use a `frequency_dict` function to calculate the frequency of each unique token in each class. These help to show the spread of tokens for each sentiment.
2. Training - Token frequencies are used to calculate probabilities. For ease of working, probabilities are converted to likelihoods. Furthermore, ‘plus-one smoothing’ was used to avoid any token having a frequency of zero. These ‘zero probabilities’ led to errors in likelihood calculations. The logarithmic function is also applied to normalize values and prevent underflow errors.
3. Predicting - The total likelihood of all tokens in a sentence is calculated. A positive likelihood corresponds to a positive sentiment and negative likelihood means negative sentiment.

Strengths and Weaknesses

On top of the already mentioned smoothing and applying log to values, the general code structure was designed for efficient memory use. Dictionaries were used extensively to speed up memory access. This can be seen in the relatively fast training times. Using the Hausa dataset with nearly 10,000 training samples, training both the byte pair encoder and text classifier takes approximately 8 minutes. Vectorization also allowed for compact representation of the training data. However, the vectors are less readable than text and not all available vector calculations were fully utilized. Another weakness is its sequential nature. Parallelization could be implemented to further speed up training. Time constraints prevented further exploration of this and different vocabulary sizes for BPE. Memory cleanup was also lacking and could be better integrated.

Results

Comparisons are made based on the results for the Hausa language dev dataset in `hau_output.txt`. In all cases, the BPE encoder was trained for 1000 merges. Both tokenization methods achieved an accuracy and average F1 of 85%. They also had identical precision values for the two classes. For this reason, I conclude that both tokenization methods are equally effective for this dataset.
Besides comparing the different tokenization methods, I also experimented with the Multinomial Naive Bayes model found in the Scikit-learn package. This only offered a 2% increase in accuracy and average F1 over my model. The two classifiers are therefore quite comparable. However the Scikit model is more efficient. Training and predicting can all be done in under a minute.