**Introduction**

Sentiment analysis is a technique to detect the opinion (e.g., positive, negative) of a given text, typically using machine learning algorithms. This report develops models for sentiment analysis using Naïve Bayes (NB), Convolutional Neural Network (CNN), and Long Short-Term Memory (LSTM) models.

**Dataset**

The datasets used are RottenTomatoesMovieReviews (RMTR) and Sentiment140 (S140). Training and testing datasets are separated from each other originally. The sentiment scores are 0 for negative, 1 for somewhat negative, 2 for neutral, 3 for somewhat positive, and 4 for positive. Overall, the RMTR training set contains 148256 reviews, with 75555 labelled 2, 31289 labelled 3, 25872 labelled 1, 8774 labelled 4 and 6766 labelled 0. In S140 training set, 1600000 tweets are evenly divided into 0 and 4 while S140 testing set contains 0, 4, and 2. It is apparent that the imbalance between classes and lack of certain labels on some datasets contribute to bias.

**Data Preparation**

In terms of formats, datasets are first read into data frames and then converted to list of strings for NB. Then, processed texts are then stored as csv files for convenience. Data for neural networks are converted to Field object as required by Pytorch. Vocabularies for text and label are built according to that.

In terms of quantities, original data are used for NB models as there is no need tuning parameters. For the neural networks, however, RMTR training data is further divided into training set and validation set with some reviews labelled 0 removed for data balance. The rationale for splitting dataset is to prevent model from seeing test set during validation.

In terms of contents, abbreviation normalised tokens with no less than 2 characters are stored in a list if they are not punctuations or stop words. Such tokens are then joint by white spaces for further use.

**Text Representation**

Text representation used in this report are term frequency-inversed document frequency (TF-IDF) for NB models and word embeddings for neural networks. TF-IDF is the product of term frequency and inverse document frequency, the latter of which is calculated by log (n/df). df (document frequency) is the frequency a term appears in a document. Term frequency is thus penalised so that frequent words (e.g., stop words) are assigned low weights. This helps with extract meaningful information of each tweet/review. The reason for applying TF-IDF to NB models is that naïve bayes as a bag-of-word method does not need word embeddings, and that TF-IDF is better than one-hot coding in terms of extracting meaningful words.

Word embedding creates vectors of words according to the vocabulary, with consideration on the relation of the given word to the other words. Therefore, GloVe is used for CNN while LSTM trains a brand-new embedding layer with random initial weights using the training data. The rationale behind it is that neural networks require word embeddings and to compare the performance between models with different embedding layers.

**Machine Learning Models**

Overall, the models developed are multinomial NB trained on RMTR, S140, and combination of the two, respectively, 2D CNN trained on RMTR, and bidirectional LSTM trained on RMTR. The number of epochs is set to 20 due to limited timeframe.

The 2D CNN consists of an embedding layer with weights from pre-trained GloVe embedding, four convolution layers with relu activation function and kernel size at 1 and 2 respectively, 2 max-pooling layers corresponding to the convolution layers, a fully connected layer, a dropout layer, and a fully connected layer followed by softmax as activation function. While it is generally considered that conv2d is for image classification, it is also used for text classification in some papers (e.g., (Baishya et al, 2021)). Softmax is applied to the final output as it is a multiclass classification problem, where outputs are scaled between 0 and 1 with the sum of 1, thus suitable for multiclass classification. Moreover, Adam optimiser is used with learning rate at 0.00001 to prevent gradient vanishing which sometmes happens to relu. Dropout rate is set to 0.5 for preventing overfitting.

The bidirectional LSTM model contains a self-trained embedding layer, two LSTM layers with hidden nodes of 16, and a dense layer followed by softmax activation function. A bidirectional LSTM model works by training 2 LSTM layers—one in normal sequence and the other in reversed sequence, so that future as well as past information is stored. Similar as the previous model, softmax and cross-entropy are used for multiclass classification, learning rate is set at 0.001 for gradient vanishing prevention, and dropout rate is set to 0.2 to prevent overfitting.

**Experimental Results**

Metrics applied for evaluation are accuracy, F1-score, precision, recall, and heatmaps. The NB model trained on RMTR received 59.68% accuracy, 36.33% f1-score, 57.78% precision, and 34.37% recall when tested on RMTR, but received only 27.71% accuracy, 9.37% f1-score, 25.96% precision, and 19.82% recall when tested on S140. The performance for NB trained on S140 and tested on RMTR and S140 are 7.34% and 55.62% for accuracy, 5.27% and 43.27% for f1-score, 2.88% and 37.40% for precision, and 30.40% and 51.47% for recall, respectively. That for NB trained on combined data and tested on RMTR and S140, respectively, are 14.25% and 56.02% for accuracy, 10.58% and 43.57% for f1-score, 43.83% and 37.64% for precision, and 35.63% and 51.84% for recall. Heatmaps are summarised in table 1.

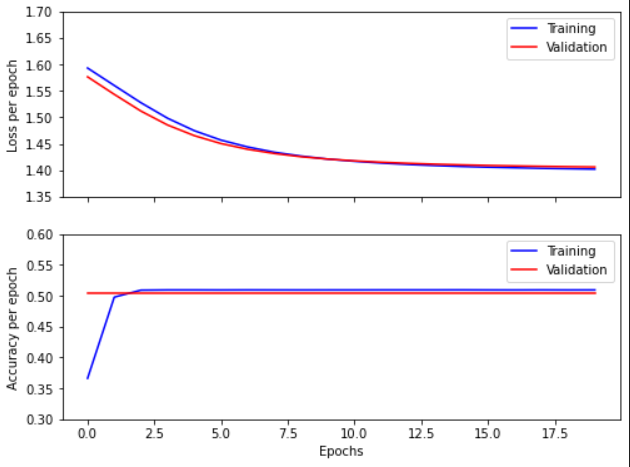
|  |  |  |
| --- | --- | --- |
|  | Tested on RMTR | Tested on S140 |
| NB trained on RMTR |  |  |
| NB trained on S140 |  |  |
| NB trained on combined |  |  |

*Table 1. Heatmaps for NB Models*

For CNN, best model is saved at epoch 19 (the 20th epoch as well as the final epoch) with train loss at 1.402, train accuracy at 50.92%, validation loss at 1.406, and validation accuracy at 50.36%. The metrics at testing set on RMTR and S140, respectively, are 51.61% and 27.91% for accuracy, 13.62% and 14.55% for f1-score, 10.32% and 9.30% for precision, and 20% and 33.33% for recall. Heatmaps are summarised in table 2 and loss is plotted in figure 1. It should be noted that although the loss seems flat after a certain point, it actually keeps decreasing throughout the training period.

|  |  |  |
| --- | --- | --- |
|  | Tested on RMTR | Tested on S140 |
| CNN trained on RMTR |  |  |

*Table 2. Heatmaps for CNN*

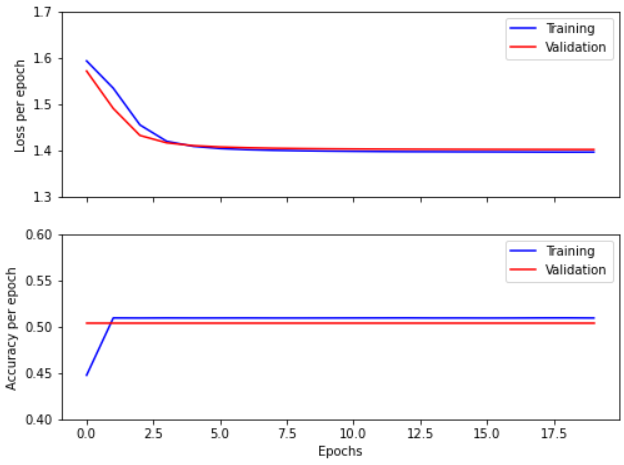


*Figure 1. Accuracy and Loss for CNN*

For LSTM, best model is also saved at epoch 19 (the 20th epoch as well as the final epoch) with train loss at 1.396, train accuracy at 50.92%, validation loss at 1.402, and validation accuracy at 50.36%. When tested on RMTR, the model receives 51.61% in accuracy, 13.62% in f1-score, 10.32% in precision, and 20% in recall. When tested on S140, the model receives 27.91% in accuracy, 14.55% in f1-score, 9.30% in precision, and 33.33% in recall. Heatmaps are summarised in table 3 and loss are plotted in figure 2. It should be noted that although the loss seems flat after a certain point, it actually keeps decreasing throughout the training period.

|  |  |  |
| --- | --- | --- |
|  | Tested on RMTR | Tested on S140 |
| LSTM trained on RMTR |  |  |

*Table 3. Heatmaps for LSTM*



*Figure 2. Accuracy and Loss for LSTM*

**Discussion**

Overall, models that are trained and tested on the same dataset generally receive higher accuracy than those on different ones. However, the relatively large distance between accuracy and f1-score indicates overfitting problems. Also, the results reflect bias from datasets, where most texts are predicted 2 by NB trained on RMTR but are predicted either 0 or 4 by NB trained on S140 as shown in table 1. Therefore, it is advisable to merge classes in terms of further work.

For CNN and LSTM, it is interesting that they both end up keeping predicting 0. The steadily downward lines for validation losses indicate that both models are not fully trained. Therefore, it is unlikely that overfitting is the reason. The reason is probably due to the imbalance training data, which add bias that cannot be recognised by cross entropy. One potential attempt is to replace cross entropy with focal loss to deal with imbalance data by assigning weights according to errors. Alternatively, oversampling or undersampling can be implemented to artificially create balance training data by randomly duplicate or dropping samples. Another likely reason is that the number of epochs is too small for the models to be fully trained. The seemingly flat accuracy graphs do not indicate convergence as losses are decreasing. Rather, they only indicate an unreached threshold, by which the predictions could be very different. Therefore, it is advised to try more epochs, and implement focal loss, undersampling, or oversampling before applying these two models.

**Conclusion**

In conclusion, NB, 2D CNN, and bidirectional LSTM are built for sentiment analysis. Results show that NB models suffer from overfitting while neural networks may not have been fully trained. Moreover, imbalance training data is another potential reason for poor test results. Hence, more epochs, focal loss, and over/under-sampling strategies are recommended for improvement.

References

Baishya, D., Deka, J.J., Dey, G. *et al.* SAFER: Sentiment Analysis-Based FakE Review Detection in E-Commerce Using Deep Learning. *SN COMPUT. SCI.* **2,**479 (2021). https://doi.org/10.1007/s42979-021-00918-9