Hate Classification using Machine Learning

Joseph Adams

Overview This model is a ternary classification model that classifies text as either 'hate', 'maybe hate', or 'not hate'. I decided to make it as a personal project to learn more about machine learning (as well as just for fun). None of the data used in the training of this model is owned by me.

Dataset The model is trained on a dataset composed of selected entries from 5 other datasets. Each entry is labeled as either 2 (hate), 1 (maybe hate), or 0 (not hate). The data used to create this dataset was taken from the following sources:

- Automated Hate Speech Detection and the Problem of Offensive Language [1]
- Hate Speech Dataset from a White Supremacy Forum [2]
- Constructing interval variables via faceted Rasch measurement and multitask deep learning: a hate speech application [3]
- HateXplain: A Benchmark Dataset for Explainable Hate Speech Detection [4]
- Large-Scale Hate Speech Detection with Cross-Domain Transfer [5]

The dataset comprises 137,343 entries. A stratified version of k-Folds cross-validation (where the class distribution remained consistent for each fold) was used to split the dataset into $k = 10$ folds for training and testing. The stratified version was used to address the class imbalance, where there are 84,016 'not hate', 37,827 'maybe hate', and 15,500 'hate' entries. Each entry in the dataset was normalised with `normalise.py`, which made the text lowercase, removed most special characters (punctuation, emojis, unnecessary whitespace, etc.), and replaced various text segments to keep the data consistent. The text data was also written to individual files for each entry, for each dataset using `build.py` if it does not already exist, in order to provide a hard copy for the tokenisation process. The assembled data object created in `model.py` is stored as *data.pkl* for faster access. The Hugging Face tokenizer library's *Byte-Pair-Encoding* tokenizer was used. It generated an overall vocabulary of 30,000.

Design Process When I first set about this task, I used a deep feed-forward network, consisting of an embedding layer, two fully connected layers, and an output layer. I then returned to this project and decided to replace this with a transformer architecture, due to the success of transformers in other natural language processing tasks. However, my transformer model took an arduous amount of time to train and tweak, offering minimal improvements in validation loss. I finally settled on a Bi-directional Long Short-Term Memory (Bi-LSTM) model with an added attention layer. This model was much faster to train, and offered a significant improvement in validation loss and accuracy. Each model was fairly easy to implement using PyTorch. I could have fine-tuned BERT for this project, but I wanted to learn more about the underlying architectures of neural networks, and get more hands-on experience with the PyTorch library.

Neural Network Architecture The model consists of the following layers:

1. An embedding layer with a dimension of 256.
2. Two Bi-LSTM layers with a dimension of 128. The padded sequences are packed for performance. Dropout is applied here with a rate of 0.3.
3. An attention layer with a dimension of 32. *tanh* is applied as the activation function.
4. A dropout layer with a rate of 0.3.
5. A fully connected classification layer with a dimension of 3, which outputs the logits for each class.

Performance The model's performance was evaluated using cross entropy loss. Over each fold, the model had an average validation accuracy of 0.795, as well as an average validation loss of 0.561. It is worth noting that validation loss decreased and then increased as it began to overfit, and so the final saved model achieved an accuracy of **0.809** and a validation loss of **0.492**. The average accuracies and losses across training and validation for each fold are shown in *Fig* 1.

The number of epochs the model was trained across varied with each fold. This was due to early stopping when the validation loss did not improve past $\delta = 1e\text{-}4$ of the previous best validation loss. The AdamW optimizer was used to train the model, with a learning rate of $5e\text{-}4$, and a weight decay of $1e\text{-}5$. A scheduler was used that reduced the learning rate by a factor of 0.5 when it started to plateau. The model had a false positive rate for 'hate' of 0.0155, and a false positive rate for 'maybe hate' of 0.0278. The model had a false negative rate for 'hate' of 0.0237, and a false negative rate for 'maybe hate' of 0.0326.

Evaluation & Next Steps The model's classification abilities are satisfactory, although there is room for improvement. The model is overfitting, and the validation loss is quite high. In future I plan to better tune the hyperparameters and add more data, especially data that diverges more from typical online speech. It is worth noting that the model could also be improved by fine-tuning BERT.

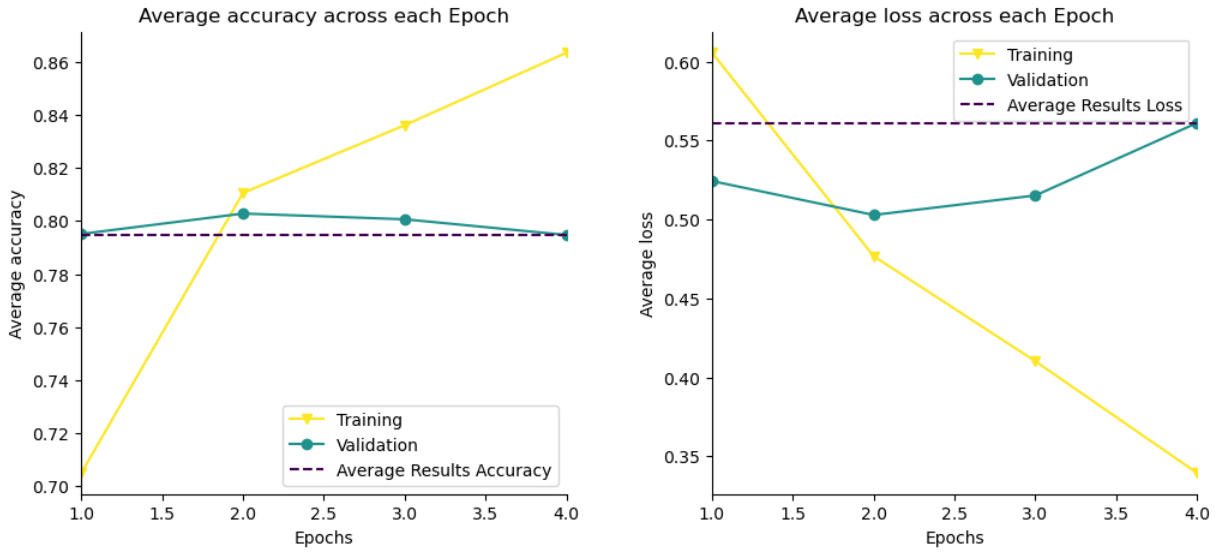

Figure 1: The average accuracy and loss across each epoch.

References

[1] Thomas Davidson et al. "Automated Hate Speech Detection and the Problem of Offensive Language". In: *Proceedings of the 11th International AAAI Conference on Web and Social Media*. ICWSM '17. Montreal, Canada, 2017, pp. 512–515.

[2] Ona de Gibert et al. "Hate Speech Dataset from a White Supremacy Forum". In: *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 11–20. DOI: 10.18653/v1/W18-5102. URL: https://www.aclweb.org/anthology/W18-5102.

[3] Chris J Kennedy et al. "Constructing interval variables via faceted Rasch measurement and multitask deep learning: a hate speech application". In: *arXiv preprint arXiv:2009.10277* (2020).

[4] Binny Mathew et al. "HateXplain: A Benchmark Dataset for Explainable Hate Speech Detection". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 17. 2021, pp. 14867–14875.

[5] Cagri Toraman, Furkan Şahinuç, and Eyup Halit Yilmaz. "Large-Scale Hate Speech Detection with Cross-Domain Transfer". In: *Proceedings of the Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, June 2022, pp. 2215–2225. URL: https://aclanthology.org/2022.lrec-1.238.