NLP 243 Assignment 1
Ryan Okimoto

The Task

The goal of this assignment is to use a machine learning model to help classify user input into categories on what the intention is behind the question. We then label these data points with any number of labels from the 19 categories available which is what makes this unique to other classification techniques where you would place the phrase into a single category. The data given to us contains pairs of natural language and the corresponding correct tags on what the content of the dialogue contains.

Preprocessing

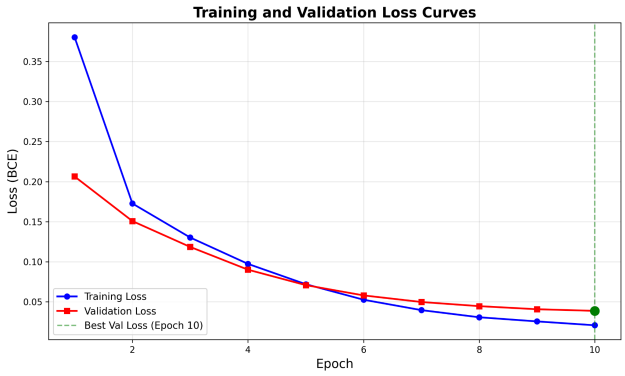
I used the given data preprocessing set although I acknowledge that it doesn't account for the class imbalances present in the dataset where there are more examples for one class than another. The preprocessing starts with lemmatization where it uses NLTK's WordNetLemmatizer to reduce words to their most basic form while still maintaining its semantic meaning as much as possible. Then the data is converted into a "bag-of-words" representation using sklearn CountVectorizer to filter out words that occur in 95% or more of the documents. Then finally the data goes under multi-hot encoding where the classes are represented by 0s and 1s where 0 means that the tag is not associated and 1 meaning that it is associated with the sentence.

Model Architecture

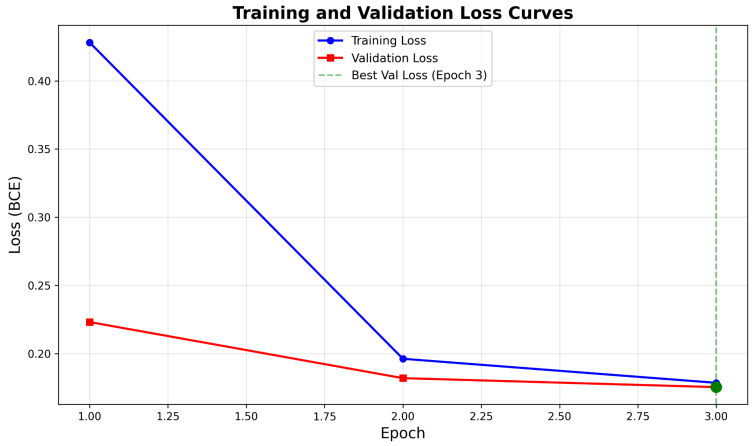
The model is a Multi Layer Perceptron neural network partially based on the one done in class. Where the processed data first passes through a linear transformation layer 128-dimensional hidden layer. Then I applied a layer normalization function, a ReLU activation function, and introduced a dropout of 30% to help prevent overfitting. Then it would be finally passed through the output layer containing 19 neurons for each class. Next I apply a sigmoid function to get the probabilities of each label and use a threshold of 0.5 to decide if it will be included in the final output. The model is trained using Binary Cross Entropy loss with the Adam optimizer, with a learning rate of 0.001 and utilizes early stopping after 3 epochs. This is run for 20 epochs because I have found that it tends to stop gaining accuracy around 16-19 epochs

Experimentation

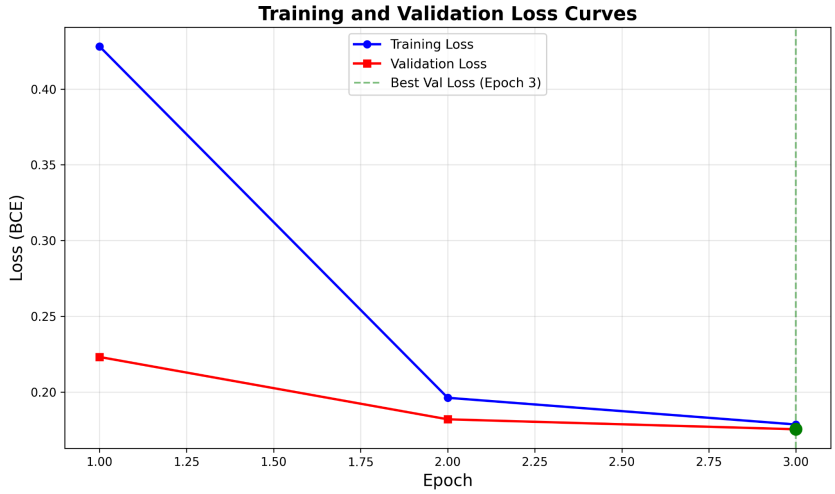
The following curve displays my best model with the preceding hyperparameters



However I also tried adjusting the learning rate to 0.01 and switching to the SGD optimizer with a momentum of 0.9 which performed much worse than the Adam optimizer.



I then experimented with changing the learning rate to 0.01 and found that although it performs slightly worse it performs similarly to the best model. I also found this to be true with or without the use of layer normalization.



Conclusion

What I have found after experimenting with many different hyperparameters, optimizers, normalization, dropout, learning rate, and epochs is that the most important part is in the model's core architecture. In the beginning I experimented simple linear layers with a ReLU activation function and found very little success with an F1 around 0.4. And switching optimizers of activation functions could jump my F1 values from 0.3-0.5 all the way to 0.85-0.9. Another key feature was implementing features like early stopping to prevent overfitting once the model has plateaued in the training process. Overall I have found that using a very simple model with standard hyperparameters will succeed as long as you choose the correct core activation functions, optimizers, and appropriate data preprocessing.