CS4395 Assignment 2

https://github.com/JayTeaK/NLP\_assignment2

Jefferson Taeksung Kim

jtk200000

**1 Introduction and Data (5pt)**

This project was to test feed forward neural network (FFNN) and recurrent neural network (RNN) to perform sentiment analysis on yelp reviews and predict their rating based on the text. The project goal is to complete the forward passes for these neural network models, and examine the results.

The yelp reviews come in the format of JSON files that contain 100 unique reviews that include the text of the review and the rating of the review.

**2 Implementations (45pt)**

**2.1 FFNN (20pt)**

An FFNN uses a bag of words that ignores word order and turns it into a vector to train. All the forward pass does is take the vector and applies ReLU to activate the vector, then applies the output layer to produce a score. Then adds softmax for a probability distribution. The rest of the code seems to translate the JSON file and turn the review into a vector that the neural network can train on.

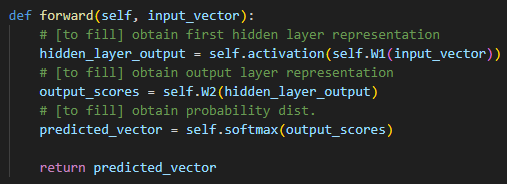


Figure 2.1: The FFNN forward pass implementation

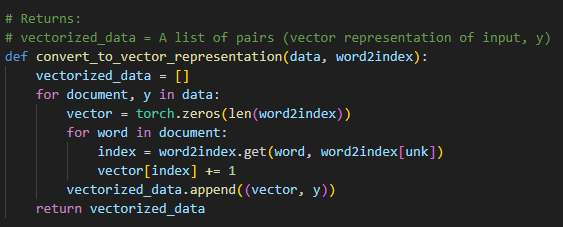
The following bit of code transforms the review into a “bag of words” for the program to vectorize where each word represents an index in the vector. ****

Figure 2.2: FFNN method to convert review into vector

Unlike RNN the FFNN will not rely on context, so there is no information to share from previous iterations.

**2.2 RNN (25pt)**

The RNN is obtaining vectors by using pre-trained embeddings and taking sequence of word indices to represent the data with better context and order. The forward pass Takes advantage of this and uses previous steps in calculating the predictions.

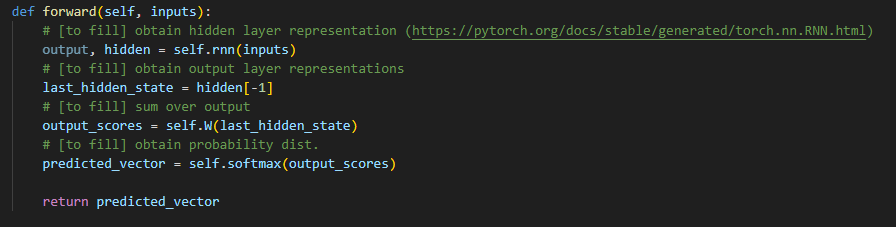


Figure 2.3: RNN forward pass implementation

The hidden[-1] utilizes the previous step to influence predictions of the next step to give the neural network the ability to see context. Everything else is the standard softmax and other forward pass practices. The only thing to note is the pre-trained embeddings that are used to help with learning context better since that is RNN’s strength.

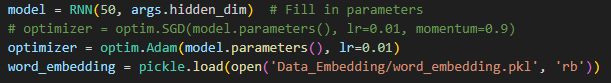


Figure 2.4: RNN word embedding from pickle

**3 Experiments and Results (45pt)**

To examine the performance of these models, we will measure how close the predictions are to the actual rating. This will be measured as accuracy.

Then, we will determine how accurate each model is based on epoch size and hidden dimensionality.

First two tables will be the results of varying dimensionality for 5 epochs.

| FFNN Dimensionality | Best Training Accuracy | Best Validation Accuracy |
| --- | --- | --- |
| 50 | 0.651 | 0.5975 |
| 100 | 0.655125 | 0.6325 |
| 200 | 0.65775 | 0.61625 |

| RNN Dimensionality | Best Training Accuracy | Best Validation Accuracy |
| --- | --- | --- |
| 64 | 0.426625 | 0.405 |
| 128 | 0.39325 | 0.41125 |
| 256 | 0.388875 | 0.415 |

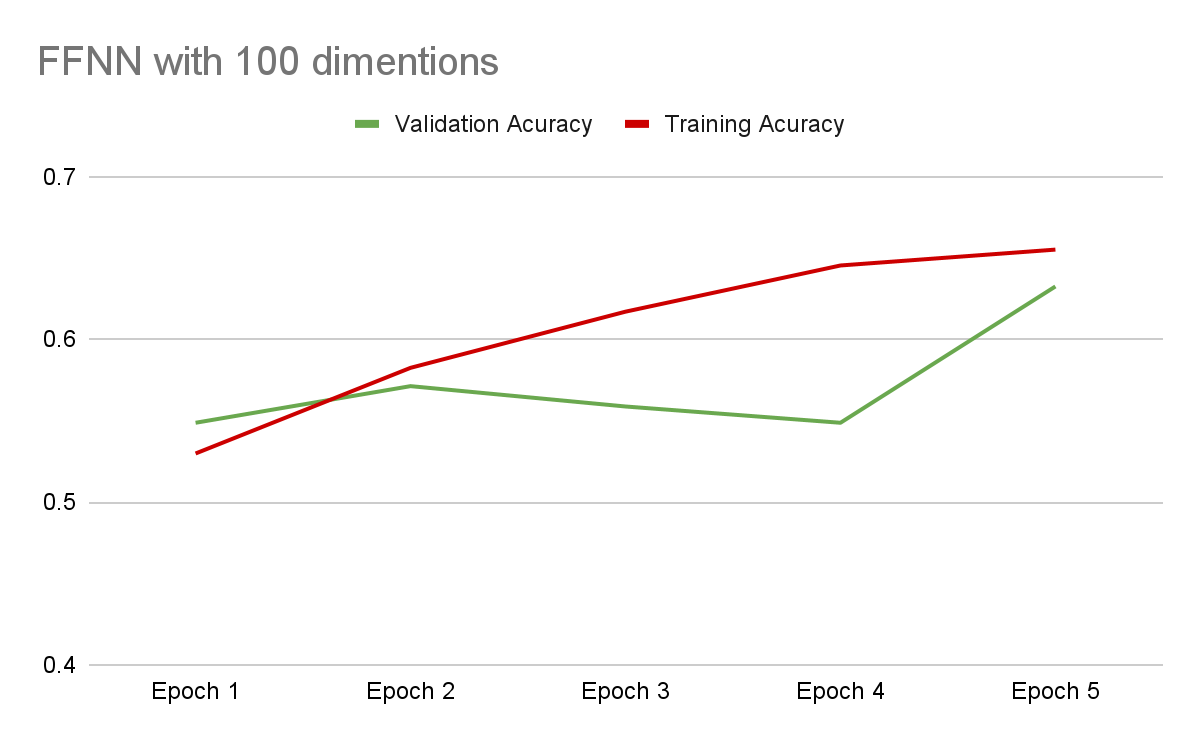
A note for RNN is the stop feature that pauses for overfitting. Some of the accuracies have stopped due to overfitting.

**4 Analysis (bonus: 1pt)**

The FFNN model outperformed the RNN across all tested hidden dimensions. The best validation accuracy for FFNN was 63.25% with 100 dimensionality. This might be because the size balances complexity and generalization well. While increasing the FFNN's hidden size beyond 100 slightly improved training accuracy, it led to a drop in validation accuracy, hinting at potential overfitting.

The RNN model performed significantly worse overall. Validation accuracy peaked at 41.5% with 256 hidden units, and training accuracy decreased as hidden size increased. This could suggest that the RNN is underfitting or struggling to optimize, possibly due to inadequate training time from the manual stop.

Overall, the FFNN model with a 100-dimensional hidden layer provides the most effective and stable performance for the task, leading me to believe that context isn’t too important when reading the sentiment of reviews.

I believe we could improve the RNN if the model used a LSTM or GRU since I believe it underperforms due to the lack of long-term dependency. FFNN is simpler, but the method is strong enough to work on simple data sets. The dataset given is just simple enough for the FFNN to outperform RNN despite logically assuming the opposite.

**5 Conclusion and Others (5pt)**

Time spent on the project was 72 hours, and the project difficulty was challenging enough to learn, but not overwhelming. I’m really happy with what I learned through this project.