Carter Auer and Willow Keenan-Harte

P.S. I attached a zip containing the final version of our code just in case, but here is the link to the github for the bonus points:
https://github.com/JollyOmnivore/NLP-NN-Final-Project

NN Final assessment and Writeup

Problem statement/project overview:

For this project, we wanted to create a model that can solve the problem of predicting the next word(s) of a sentence based on a portion of a sentence. For example when a model receives “I like to” the model would predict a user-selected number of words to continue off of that with. For example, it would respond with “I like to go outside.” This can be used for things like search engines or keyboards on mobile like the keyboard on Apple iPhones that uses your old message history to train a model to predict what you are going to text.

For this project, we used Pytorch in order to build an LSTM mode as well as testing with a standard RNN in conjunction with testing different datasets and word embedding methods such as Bert, Glove, Fasttext, and W2V. We experimented with all of this with the first goal of lowering loss and the second and ultimate goal of getting meaningful responses from the model. As well as storing all responses the model gives along with hyperparameter data.

Data description - where is it from? What does the data look like? What kind of preprocessing (if any) needed to be done?

We tested with a few datasets 2 coming from Kaggle and one from Amy’s original week nine assignment. Since these were sentence datasets we had to first break down each sentence and from that, we would break those sentences down to each word to train our word embedding model. As for the subject of our data, we used a few different sources. We messed around with a dataset with every line in Southpark which was not a good idea since it wasn’t very politically correct. Next, we tried a data set that consisted of blog posts from all over the internet. Unfortunately, that dataset was very large and would bring my RTX3070 to a dead stop or just overflow the GPU V-ram. Even when we reduced the size of the blog post dataset our final choice still outperformed it in terms of responses and loss. This final data set was the one provided by Amy which is built from Wikipedia articles as well as questions and answers hosted on Wikipedia with a total of 30,000 sentences in total.

Test results: what datapoints/tests are you using to prove your concept works?

To test the quality of our model we did two things: we measured loss (specifically cross-entropy loss) and we created our own set of benchmarks that we would run through on each version and manually observe the results by hand. In our GitHub and the folder submitted, there is a txt file named Responses, and there are all of our previous tests and responses from the model.

Limitations: Where does your model break? What are the limits for input data?
One very key place that our model breaks is when a user enters a word for phrase that was not included in the training dataset. Originally this would get passed into our predict function and would error out the program, however, I was able to use the word vocab we create during training to make sure every word entered is something the model has seen before. Unfortunately, this is pretty limiting since a random word in a sentence that you may try and enter causes your entire sentence to get rejected.

Future directions
Going forward there are a few things we can do to truly make this model great. That is our dataset size and variety as well as increased computer power. This is because even going from 30,000 words to 50,000 the model would train 3-4 times slower. With the amount of data, we would need to have consistently meaningful responses from the LSTM being in the range of 200,000 - 500,00 sentences; we would need multiple of the Western deep learning servers running at once to even get a few rounds of hyper parameter tuning done. Another direction I was interested in taking this project was to take over that data set and rebuild it using a transformer since for next word prediction transformers typically outperform LSTM models.

Final PS I attached a video of me demonstrating the program running. However, I don’t believe it's good enough for CS and WCU marketing departments but I had to record it for Amy’s submission so if you think it's good enough, great! Otherwise no problem.