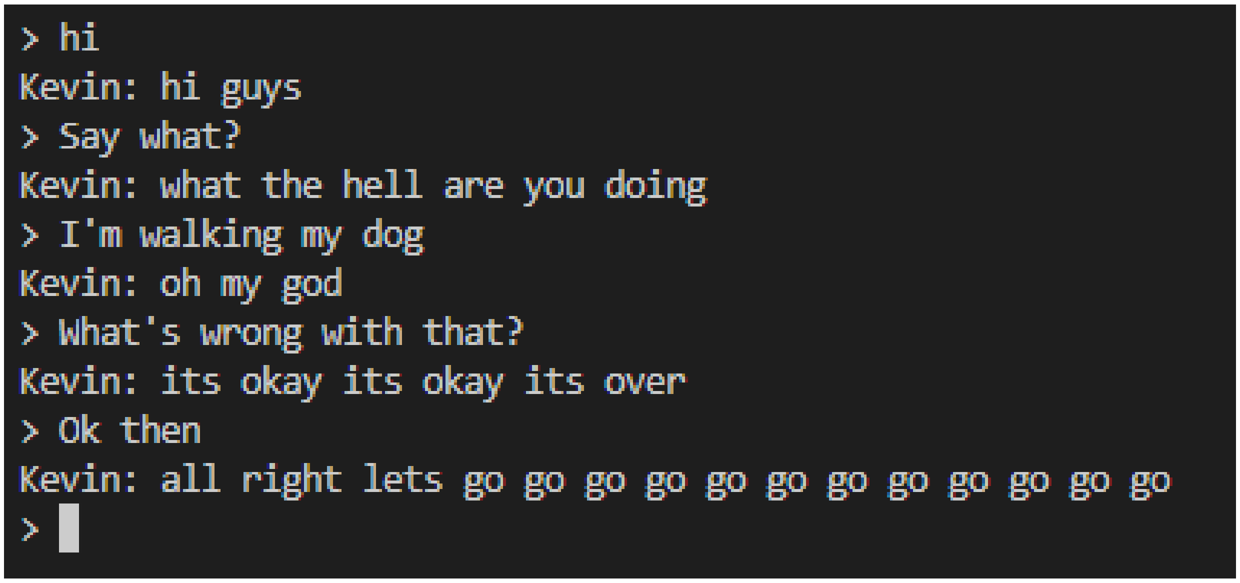
Chatbot Report

CS 4395 - Noah Gonzales

The chatbot presented is a RNN built using pytorch's neural network library. A dataset of South Park lines was used to train the chatbot. The dataset included all lines from South Park and the season and episode they are from. First the quotes were extracted from the dataset, then they were processed. Processing includes removing punctuation, standardizing some phrases ("oh" with more than one "h" or "ah" with more than one "h", removing accented characters or characters from a different language that wasn't similar to English, removing special unicode characters (a latin space or soft hyphen), then removing white space.The text was then saved as both conversations and the plain text. Conversations are just neighboring quotes strung together as pairs with a tab as the delimiter. The text is then imported to build the model. By going through the text, a vocabulary is built and the tokens are vectorized. The model is then defined, along with any needed helper functions. Helper functions include functions to change batch data to training data and converting batch data to input and output that can be trained on. The model is then defined as an encoder and decoder, where the encoder takes a variable length input sequence to a fixed-length vector and the decoder takes an input word and context and returns a guess for the next word. The model is then trained and saved as checkpoints. The checkpoints can be loaded to run the model.

The text used to build the vocabulary can be found here and the conversations used to build the model can be found here. The text in both files has already been processed. The following is a sample dialog with the chatbot:



```
> hi
Kevin: hi guys
> Say what?
Kevin: what the hell are you doing
> I'm walking my dog
Kevin: oh my god
> What's wrong with that?
Kevin: its okay its okay its over
> Ok then
Kevin: all right lets go go go go go go go go go go go go
> 
```

There were many different options for making a chatbot but none seemed as open to customization as I would have liked. My first attempt was using keras, but that was difficult to implement with my dataset. My second attempt was just a reattempt of that but in Google colab and a little more fleshed out, enough to warrant calling it a second attempt. My third attempt was using the Chat GPT API. I did not have enough tokens to be able to call the API, so I abandoned the effort. I also looked into using ChatterBot, AIML, and chatscript. I decided to use PyTorch because it seemed to be able to train to adopt a certain way of speaking better than the other options. Options such as AIML or ChatterBot had ways to program certain responses to specific prompts but did not have the ability to vary its responses from something that has already been seen. PyTorch can generate responses. The downside of PyTorch, which I encountered after running the model, is that the responses, although generated, are still very closely related to what the training data is and there is not variance in the response with the same input because the response is generated by selecting the word with the highest likelihood given an input, and the trained likelihoods don’t change. Creating variance in the output would be a future endeavor. PyTorch also has no ability to interpret words that are outside of its trained vocabulary. Solutions to this would be to expand the dataset to include a broader range of words and conversations, but this would reduce the amount of South Park personality the bot would possess. Another solution would be to change words out using synonyms or hypernyms until they are in the vocabulary.