Chat.py

The code snippet is a Python program that demonstrates how to build a simple chatbot using natural language processing (NLP) techniques and a neural network model. Here's what the code does:

1. The program imports necessary libraries and modules, including `random`, `json`, `textwrap`, `torch`, `sys`, and `base64`. Additionally, it imports two custom modules `model` and `nltk_utils`.
2. The program loads the intents file (intents.json) using the `json` module. The intents file is a JSON file that contains different intents (actions) and their associated tags and responses.
3. The program loads a pre-trained neural network model from a saved PyTorch model file ("data.pth") that contains the trained weights and biases of the model. The program also loads some data from the model file, including the input, hidden, and output sizes, all words, tags, and the model state.
4. The program creates an instance of the NeuralNet class defined in the model module and sets its device to run on either GPU or CPU, depending on availability.
5. The program enters an infinite loop to get user input and generate chatbot responses. Within the loop, the user's input is tokenized using the `tokenize` function from the `nltk_utils` module.
6. The program generates a bag of words representation of the tokenized sentence using the `bag_of_words` function from the `nltk_utils` module. The bag of words representation is a vector that contains binary values (0 or 1) indicating the presence or absence of each word in the sentence.
7. The program reshapes the bag of words vector to match the input shape expected by the neural network model (1 x len(all_words)).
8. The program converts the reshaped bag of words vector into a PyTorch tensor and moves it to the device specified in step 4.
9. The program passes the tensor through the neural network model to obtain the output vector.

10. The program applies the softmax function to the output vector to convert it into a probability distribution.
11. The program gets the index of the highest probability value in the output vector, which corresponds to the predicted tag.
12. The program checks if the probability of the predicted tag is higher than a threshold (0.75). If so, it selects a response associated with the predicted tag from the intents file and generates a chatbot response. If not, it generates a default "I do not understand..." response.
13. The program prints the chatbot response on the console.
14. The program continues to prompt the user for input until the user types "quit" to exit the program.