**Addis Ababa University**

**AI Assignment**

**Simple AI cahtbot using pytorch**

**code Explanation**

**Abdurahmman Mohammed – ATE/8901/13  
Amanuel Ayalew – ATE/3871/13**

**Code repository** [**https://github.com/Amanuel-Ayal3w/ChatBot\_Assignment-**](https://github.com/Amanuel-Ayal3w/ChatBot_Assignment-)

**Model.py**

This code makes a special type of program called a "neural network." It uses a library called PyTorch to help with the math. The network is like a chain of connected parts, and each part does some math.

The NeuralNet part is like the brain of the network. It's made of smaller parts called layers. There are three layers in this network.

In the first part, the program gets ready by setting up these layers. It says, "I'm going to have three layers, and each layer will do a specific kind of math."

The forward part is where the real work happens. It's like the network thinking. It takes some input (like numbers o data) and does math on it.

First, the input goes through the first layer. Then, it does a special kind of math called "ReLU." After that, it goes through the second layer and does more ReLU math. Finally, it goes through the third layer.

At the end, the program gives out the result of all the math it did. This result is the output of the network.

**nltk.py**

This code sets up some functions for processing text data:

tokenize: Splits a sentence into individual words or tokens.

stem: Finds the root form of a word using stemming.

bag\_of\_words: Creates a bag of words representation for a sentence, where each element in the array represents whether a known word is present in the sentence or not.

These functions are commonly used in natural language processing tasks like text classification or sentiment analysis. They help in converting raw text data into a format that can be used as input for machine learning models**.**

**Train.py**

Data is loaded from a JSON file named 'intents.json', which presumably contains patterns for intent classification.

Tokenization, stemming, and preprocessing are performed on the data to prepare it for training.

Creating Training Data:

The preprocessed data is used to create training inputs (X\_train) and labels (y\_train) for the neural network.

Defining Dataset and DataLoader:

A custom ChatDataset class is defined to handle the training data.

A DataLoader is created to efficiently load the training data in batches during training.

Defining Model:

The neural network model (NeuralNet) is instantiated with input, hidden, and output sizes.

The model is moved to the appropriate device (CPU or GPU) for training.

Defining Loss Function and Optimizer:

Cross-entropy loss and Adam optimizer are chosen for training the model.

**Training the Model:**

The model is trained for a specified number of epochs.

Training data is loaded in batches using the DataLoader.

Forward pass, loss calculation, backward pass, and optimization steps are performed.

Training progress is printed at regular intervals.

**Saving the Model:**

After training, the model's state dictionary, along with other necessary information, is saved to a file named 'data.pth'.

Final loss and a message confirming the completion of training and saving are printed.

This script appears to be a complete pipeline for training a neural network model for intent classification using a dataset provided in a JSON format. It handles data loading, preprocessing, model training, and model saving.

**Chat.py**

This is a chatbot application that uses a pre-trained neural network model to respond to user queries based on predefined intents. Here's a breakdown of how it works:

1. Imports: The script imports necessary libraries and modules including random, json, torch, NeuralNet from a custom module called model, and utility functions bag\_of\_words and tokenize from another custom module called nltk\_utils.
2. Device Selection: It checks whether a GPU is available and sets the device accordingly.
3. Loading Data:
   * It loads intents data from a JSON file named 'intents.json', which contains predefined intents along with patterns and responses.
   * It loads pre-trained model data from a file named 'data.pth', which contains information about the neural network model.
4. Model Initialization:
   * It initializes the neural network model (NeuralNet) with the input size, hidden size, and output size obtained from the loaded data.
   * It loads the pre-trained model weights (model\_state) into the model.
5. Chat Interaction:
   * It prompts the user to start a conversation.
   * Inside a while loop, it continuously waits for user input.
   * It tokenizes the user input, creates a bag-of-words representation using the loaded vocabulary (all\_words), and converts it into a PyTorch tensor.
   * It passes the input tensor through the model to get the output.
   * It identifies the predicted tag by selecting the one with the highest probability.
   * If the predicted probability is above a threshold (0.75), it randomly selects a response associated with the predicted tag from the intents data and prints it.
   * If the predicted probability is below the threshold, it prints a default response indicating that it doesn't understand.
6. Termination: The conversation ends when the user types 'quit'.

This script demonstrates a basic chatbot functionality using a neural network model trained on predefined intents. The model classifies user queries into specific intents and generates appropriate responses based on those intents**.**