Ask to Mentor Chatbot

This project is a chatbot application named Ask to mentor, which helps users find mentors or tutors for their doubts and questions. It uses a neural network to process user inputs and provide appropriate responses.

Code Flow

1. app.py

- This is the main application file that sets up the Flask web server.
- It initializes the model, all_words, and tags by calling load_model from model_utils.py.
- It defines routes:

1. /: Renders the home page (index.html).
2. /get_response: Handles POST requests to get chat responses. Calls get_chat_response from stories.py.
3. save_user_data: Saves user data to user_data.json.

2. neural_network.py

- Defines the neural network structure using PyTorch.
- Contains train_model function to train the model on the intents data from intents.json.
- Saves the trained model to data.pth.

3. nltk_utils.py

1. Utility functions for text processing:

- tokenize: Tokenizes a sentence into words.
- stem: Reduces words to their root form.
- bag_of_words: Converts a tokenized sentence into a bag-of-words representation.

4. model_utils.py

- Contains load_model function to load the trained model from data.pth.

5. stories.py

- Contains get_chat_response function to handle user messages and provide appropriate responses.
- Uses predefined responses for specific conversation tags and utilizes the neural network model for others.

6. intents.json

- Contains predefined intents with patterns and responses to guide the conversation flow.

3. Flow of Code Execution

1. Initialization

- app.py starts the Flask application.
- It attempts to load the model using load_model from model_utils.py.
    - If the model is not found, it prints an error and sets model, all_words, and tags to None or empty lists.

2. Serving the Home Page

1. The '/' route renders the index.html page.

3. Handling User Messages

- The '/get_response' route handles POST requests with user messages.
- It calls 'get_chat_response' from 'stories.py' with the loaded model, all_words, and tags.
    - If the model is not loaded, it returns an error response.
- 'get_chat_response' processes the user message and determines the appropriate response based on the current tag and message content.

4. Flow of Messages

1. Start Conversation

- Initial message triggers the start_conversation tag.
- Responses offer a welcome message and options to look for a mentor or explore.

2. Offer Options

- Based on user choice, it transitions to either offer_options:
    - If "mentor" is mentioned, it provides mentor-related responses and options for subjects.
    - If "exploring" is mentioned, it provides exploration-related responses and options.

3. Select Subject

- User selects a subject, transitioning to select_subject tag.
- Offers further options for assistance format (written or video calls).

4. Select Assistance Format

- Based on user preference, it provides options for notes format or mentor preferences.

5. Get Mentor Preferences

- User chooses mentor preferences, leading to either male, female, or both options.

6. Get User Details

- Sequentially collects user mobile number, name, and email through respective tags (get_mobile_number, get_name, get_email).

7. Fallback to Neural Network

- If a tag isn't predefined, the neural network processes the user message and predicts the appropriate tag, returning a relevant response from intents.json.