Problem Statement:

Chatbot using Recurrent Neural Network.

Introduction:

we will create an AI chatbot using Natural Language Processing (NLP) in Python. Our goal is to help you build a smart chatbot. First, we'll explain NLP, which helps computers understand human language. Then, we'll show you how to use AI to make a chatbot to have real conversations with people. Finally, we'll talk about the tools you need to create a chatbot like ALEXA or Siri. Also, We Will tell in this article how to create ai chatbot projects with that we give highlights for how to craft Python ai Chatbot.

Dataset Used:

Local dataset for language preprocessing

Cornell_Movie-Dialogs_Corpus.txt

Movie interaction.txt

For better interactive chatbot we use Seq2Seq model for high accuracy throughput.

What is NLP?

Natural Language Processing or NLP is a prerequisite for our project. NLP allows computers and algorithms to understand human interactions via various languages. In order to process a large amount of natural language data, an AI will definitely need NLP or Natural Language Processing. Currently, we have a number of NLP research ongoing in order to improve the AI chatbots and help them understand the complicated nuances and undertones of human conversations.

Training Phase

1. Data Collection and Preprocessing
    - Collect Data: Gather a large set of conversations or dialogue data relevant to your chatbot's domain.
    - Clean Data: Remove any noise, irrelevant information, and perform normalization (lowercase conversion, punctuation removal, etc.).
    - Tokenization: Split text into tokens, words or subwords.
    - Create Vocabulary: Build a vocabulary of all unique tokens.
    - Padding: Ensure uniform input size by padding sequences to the same length.

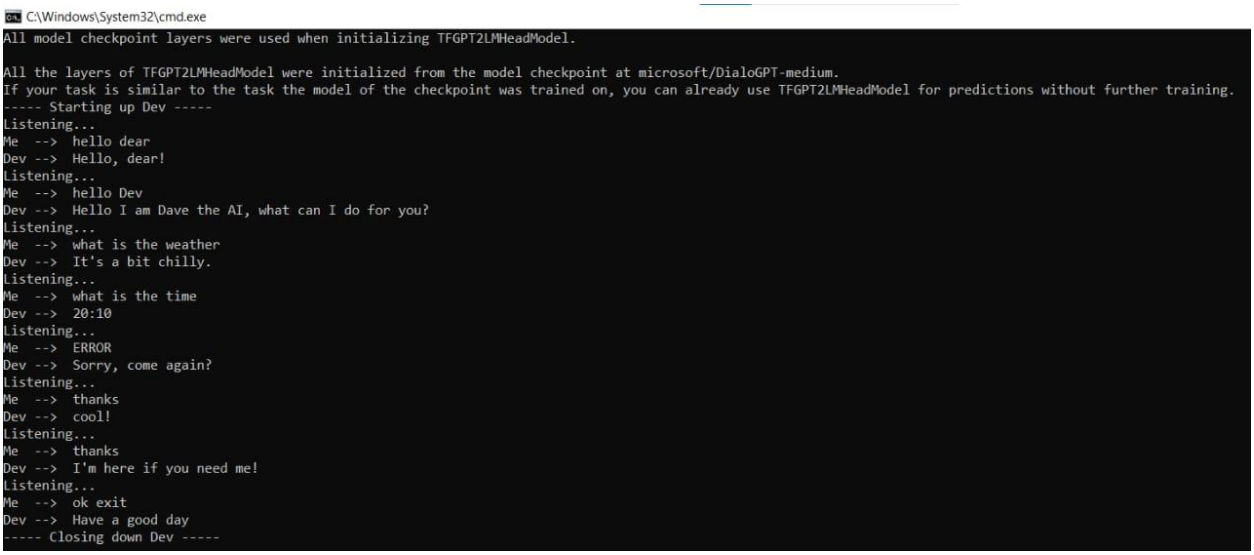
2. Model Selection and Training
    - Choose a Model: For instance, Sequence-to-Sequence (seq2seq) with attention mechanism.
    - Create Train-Test Split: Split data into training and testing datasets, typically 80-20 or 70-30.
    - Embedding Layer: Convert tokens into dense vectors using embedding layers or pre-trained embeddings like GloVe or Word2Vec.
    - Training the Model: Train your model using the training dataset. This involves feeding the data into the model, calculating loss, and optimizing the model's weights using backpropagation.
    - Save the Model: Once trained, save your model's weights and architecture.

Testing Phase

1. Model Evaluation
    - Load the Trained Model: Load the saved model weights and architecture.
    - Testing on Held-Out Data: Evaluate the model on the testing dataset.
    - Metrics Calculation: Use metrics like accuracy, BLEU score, and perplexity to evaluate the model's performance.
2. Performance Optimization
    - Fine-Tuning: Adjust hyperparameters, try different architectures, or incorporate additional data to improve performance.
    - Error Analysis: Analyze where the model performs poorly and understand failure modes.
    - Cross-Validation: Use cross-validation techniques to ensure robustness and avoid overfitting.
3. Deployment Preparation
    - Integration: Integrate the chatbot with your chosen platform (e.g., a website, app).
    - Real-World Testing: Conduct testing with real users to get feedback and identify any issues.

Output:

```
----- starting up Dev -----
listening...
me --> Hey Dev
AI --> Hello I am Dev the AI, what can I do for you?
listening...
me --> What is the time
AI --> 17:30
listening...
me --> thanks
AI --> cool!
listening...
```



Conclusion:

You can use this chatbot as a foundation for developing one that communicates like a human.