**Conversational Chatbot using LSTM**

**Introduction**

This project aims to create a conversational chatbot using Sequence-to-Sequence Long Short-Term Memory (LSTM) models. Sequence-to-sequence learning involves training models to convert sequences from one domain to sequences in another domain. This approach is particularly useful for tasks like machine translation, speech recognition, and text generation.

**Code and Resources Used**

* **Language**: Python 3.8
* **Dataset**: Chatterbot Kaggle English Dataset
* **Packages Used**: numpy, tensorflow, pickle, keras
* **Model Used**: Seq2Seq LSTM model
* **API Built**: Keras Functional API

**Step 1: Data Extraction and Preprocessing**

The dataset is from chatterbot/english on Kaggle.com by kausr25, containing pairs of questions and answers based on various subjects like food, history, AI, etc.

1. **Parse each .yml file**:
   * Concatenate sentences if the answer has more than one.
   * Remove unwanted data types produced during parsing.
   * Append tags to all the answers.
   * Create a Tokenizer and load the entire vocabulary (questions + answers) into it.
2. **Prepare Data Arrays**:
   * **Encoder Input Data**: Tokenize the questions and pad them to the maximum length.
   * **Decoder Input Data**: Tokenize the answers and pad them to the maximum length.
   * **Decoder Output Data**: Tokenize the answers and remove the first element from all tokenized answers.

**Step 2: Defining the Encoder-Decoder Model**

The model consists of Embedding, LSTM, and Dense layers.

1. **Input Layers**: One for encoder\_input\_data and another for decoder\_input\_data.
2. **Embedding Layer**: Converts token vectors to fixed-size dense vectors (use mask\_zero=True).
3. **LSTM Layer**: Provides access to Long-Short Term cells.

**Workflow**:

* encoder\_input\_data goes into the Embedding layer.
* The output of the Embedding layer goes to the LSTM cell, producing two state vectors (h and c).
* These states are set in the LSTM cell of the decoder.
* decoder\_input\_data goes through the Embedding layer and then into the LSTM cell, which had the states, to produce sequences.

**Step 3: Long Short-Term Memory (LSTM)**

LSTM networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. They are used in complex domains like machine translation and speech recognition.

**Key Components**:

* **Memory Blocks (Cells)**: Responsible for remembering things.
* **Cell State**: The main conveyor belt running through the chain.
* **Gates**: Structures that regulate information flow, including input, forget, and output gates.

**Step 4: Training the Model**

The model is trained for 150 epochs using the RMSprop optimizer and the categorical\_crossentropy loss function. The model achieved a training accuracy of 96%.

**Step 5: Defining Inference Models**

1. **Encoder Inference Model**: Takes the question as input and outputs LSTM states (h and c).
2. **Decoder Inference Model**: Takes in LSTM states (output of the encoder model) and answer input sequences. It outputs the answers for the given question and updates the state values.

**Step 6: Talking with the Chatbot**

1. **Convert Questions to Tokens**: Use a method str\_to\_tokens to convert string questions to integer tokens with padding.
2. **Predict State Values**: Input a question and predict the state values using the encoder model.
3. **Generate Sequence**:
   * Set state values in the decoder's LSTM.
   * Generate a sequence with the initial element.
   * Input this sequence into the decoder model.
   * Replace the initial element with the predicted element and update the state values.
   * Repeat until the end tag or maximum answer length is reached.

**Results**

The chatbot was able to generate coherent and contextually appropriate responses to the input questions, demonstrating the effectiveness of the Seq2Seq LSTM model in conversational AI tasks.