Chatbot with LSTM - Project Manual

Project Overview

This project is about creating a simple chatbot using Python and deep learning techniques. The chatbot is trained on a dataset of dialogues, where each line contains an input sentence and its corresponding response. The goal is to build a model that can generate meaningful replies to user queries.

The model is based on a sequence-to-sequence architecture using Recurrent Neural Networks (RNNs) with Long Short-Term Memory (LSTM) layers. This approach helps the chatbot remember context and produce better responses compared to simple rule-based systems.

Objectives

1. To build a chatbot from scratch without using pre-trained models.
2. To learn about tokenization, embeddings, and sequence modeling.
3. To implement an Encoder-Decoder LSTM architecture for text generation.
4. To train the chatbot on a dataset of conversations and evaluate its performance.

Dataset

- File: dialogs.txt
- Format: Tab-separated values with input and response pairs.

Example from the dataset:

hi, how are you doing? i'm fine. how about yourself?
i'm fine. how about yourself? i'm pretty good. thanks for asking.

Preprocessing

- Remove extra spaces.
- Add special start (`\t`) and end (`\n`) tokens to the target sentences.
- Tokenize the text into numbers that can be used by the model.
- Pad sequences so that all inputs have the same length.

Methodology

1. Preprocessing

- Convert sentences into sequences of integers using a tokenizer.
- Prepare encoder inputs (user queries) and decoder targets (responses).

2. Model Architecture

- Encoder: Embedding layer followed by an LSTM that creates hidden states.
- Decoder: Embedding layer and LSTM that uses the encoder’s states to generate output.
- Dense layer with softmax activation to predict the next word in the sequence.

3. Training

- Loss function: Sparse categorical crossentropy.
- Optimizer: RMSProp.
- Teacher forcing is used during training to improve learning.
- Training is done for 20 to 40 epochs depending on dataset size.

4. Inference (Chatting)

- Encoder processes the user input and produces state vectors.
- Decoder generates words one by one until the end token is produced.

Implementation Steps

1. Load the dataset (dialogs.txt).
2. Preprocess the data (tokenization, padding, adding tokens).
3. Build the encoder-decoder LSTM model.
4. Train the model on the dataset.
5. Save the model and tokenizers for later use.
6. Build separate encoder and decoder models for inference.
7. Write a reply function to generate responses.
8. Test the chatbot with interactive input.

Example Results

Example conversation after training:

You: hi, how are you?
Bot: i'm fine. how about yourself?

You: what school do you go to?
Bot: i go to pcc.

How to Run

1. Open the notebook in Google Colab (recommended for GPU).
2. Upload the `dialogs.txt` dataset.
3. Run the preprocessing and training steps.
4. Use the interactive loop to chat with the model.

Dependencies

- Python 3.x
- TensorFlow / Keras
- Pandas
- Numpy

To install the libraries:

```
pip install tensorflow pandas numpy
```

Future Improvements

- Add attention mechanism for improved context handling.
- Train on a larger dataset for better results.
- Create a web interface using Flask or Streamlit.
- Experiment with Transformer-based models for advanced performance.

Author

Amandeep

Data Analytics Enthusiast | Learning NLP and Deep Learning