simple-torch-chatbot

This repository contains an implementation of a Seq2Seq chatbot using PyTorch and the Microsoft WikiQA corpus. The chatbot uses an encoder-decoder architecture with LSTM cells to generate responses to user input.

Installation

Clone the repository:

```
git clone https://github.com/mohammadforouhesh/simple-torch-chatbot.git
```

Install the required packages:

```
pip install -r requirements.txt
```

Usage

Preprocessing the Data Download the Microsoft WikiQA corpus from the official website and place the files in the data directory.

preprocess.py

Tokenizes the input and output sequences. Removes stop words from the input sequences. Filters out input sequences that are too short or too long. Builds the vocabulary file, which maps words to unique integer IDs. The output of this script is a set of preprocessed data files (in the data/processed directory) and the vocabulary file (in the data directory).

Training the Model

train.py

This script trains the Seq2Seq model on the preprocessed WikiQA data using the hyperparameters specified in config.py. During training, the model periodically saves checkpoint files (in the models directory) and logs the training progress to TensorBoard.

Once training is complete, the final model parameters will be saved to a file named best_model.pt.

Evaluating the Model

evaluate.py

This script loads the trained Seq2Seq model and allows you to test it on sample input. You can enter a question, and the model will generate a response based on the input.

Note that you can modify the input sequences and expected output sequences in the evaluate.py script to test the model on different examples.

Configuration

You can configure the hyperparameters of the model and the training process by modifying the values in the config.py file. The following hyperparameters can be modified:

- batch_size: The number of examples in each batch.
- num_epochs: The maximum number of epochs to train for.
- learning_rate: The learning rate for the optimizer.
- hidden_size: The size of the hidden state in the LSTM cells.
- num_layers: The number of layers in the LSTM encoder and decoder.
- dropout: The dropout probability for the LSTM cells.
- clip: The gradient clipping threshold.

You can also modify the file paths and other settings in `params.py` as needed.

References PyTorch documentation: https://pytorch.org/docs/stable/index.html TorchText documentation: https://torchtext.readthedocs.io/en/latest/index.html Microsoft WikiQA corpus: https://www.microsoft.com/en-us/download/details.aspx?id