Question Generation Using RNN

Sharmin Pathan
Artificial Intelligence Engineer | Wolters Kluwer
Masters in Computer Science | University of Georgia

Abstract

Question generation is to generate questions from a piece of text by extracting facts from it. A sentence from the text is read by the model and a corresponding question is generated based on it by the model. This is a reverse task of question answering activity. The primary focus is to have a proper sentence structure and understanding the overlapping words, question words (what, who, how, etc).

Keywords: RNN (Recurrent Neural Networks), LSTM (Long Short Term Memory), Attention Mechanism, Word Embedding, Encoder - Decoder

1 Overview

Earlier works on question generation used a rigid rule based engine to transform sentences into related questions. These systems highly rely on manually written transformation guidelines and rules and are domain specific.

Instead, this model uses the encoder – decoder model approach for RNNs with LSTM architecture. It doesn't use a rule based engine or an NLP pipeline. The encoder network is fed with facts and decoder understands the questions. Stanford Question Answering Dataset (SQuAD) was used for a preliminary study and training.

1.1 Problem Statement

To generate questions based on a piece of text (facts).

1.2 Approach

Develop a RNN using LSTM cells and Attention mechanism with encoder – decoder.

2 Preliminaries

The system was operational with the following libraries used for feature selection and building a classification model.

2.1 Technologies Used

Keras. Keras is Python's Deep Learning Library that facilitates working with high-level neural networks. It is an API written in Python and runs on top of TensorFlow, CNTK, or Theano.

Recurrent Neural Networks (RNN). RNNs are a kind of Artificial Neural Networks where its units form a directed cycle. This allows them to have an internal memory of their own to be able to process sequences of inputs.

Long-Short Term Memory (LSTM). It's an RNN architecture that allows data to flow forwards and backwards within the network. This architecture provides certain advantages over alternative RNNs.

Encoder – Decoder. These are sequence-to-sequence learning models. Here, the RNN takes a sequence of text as input and generates another sequence as the output. In this project, the sequence acting as the input is the extracted fact and the output is the corresponding question generated.

Global Attention Mechanism. Attention mechanisms have application in image analysis but lately have made their way into NLP. These are related to the visual attention mechanisms in humans, their ability to focus on a certain region of, for example, an image and perceiving the rest of the image in lower details.

Word Embeddings. The model uses a pre-trained word embedding vectors for initialization. Word embeddings are feature learning techniques in NLP where words are mapped to vectors of real numbers.

3 Dataset

The training dataset is a bunch of direct Question-Answer pairs, nearly 100,000 of them for efficient training. For example,

Extracted Fact: The primary objective of ankle fracture fixation is to recreate a tibiotalar articulation with physiological contact stresses.

Question: What is the primary objective of ankle fracture fixation?

The primary focus is to gather QA pairs with a complete sentence structure. The overlapping words are important for the model to study and generate the sentence structure whist making predictions on the test data.

Out of the 100k QA pairs, 20% is reserved as the validation set and the rest 80% for training.

4 Implementation
This section speaks about the preprocessing and model building strategies.

4.1 Preprocessing
The extracted facts are split into sentences and further tokenized using Python's NLTK. Some facts span up to two or more sentences, these can be concatenated. A vocabulary of about 45,000 tokens is created (including the tokens and placeholders). Tokens outside the vocabulary are replaces by 'UNK' (Unknown). Additionally, the 'ZERO' padding is done to make the sentences of equal length.

4.2 Model
The Sequential model from keras.models is used.

Layers of the Encoder Network include, Embedding, LSTM with hidden size of 1000, Repeat Vector.

Layers of the decoder network include, LSTM with hidden size of 1000, TimeDistributed, Softmax Activation

The loss is computed using categorical_crossentropy and the optimizer being rmsprop

4.3 Training
Optimization is done using Stochastic Gradient Descent (SGD) with an initial learning rate of 1.0. A mini batch size of 100 is selected. Training goes upto 5000 epochs. Checkpoints / Model snapshots are saved whilst training the model to resume with the saved weights next time.

5 Concluding Remarks
Referring to a few papers and other resources listed under references to help define the model, I tried a few models with different combination of layers and picked the one that gave the best performance. I tried tuning a few parameters of the layers, mostly changing the batch size, vocabulary size, hidden layers, etc.

6 References

[1] Sequence to Sequence Learning with Neural Networks.
Ilya Sutskever, Oriol Vinyals, Quoc V. Le (Google)

[2] A Neural Conversational Model.
Oriol Vinyals, Quoc V. Le (Google)

[3] Learning to Ask: Neural Question Generation for Reading Comprehension.
Xinya Du, Junru Shao, Claire Cardie (Cornell University)

[4] Neural Question Generation from Text: A Preliminary Study.
Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, Ming Zhou