Question Generation

Prajneya Kumar - 2019114011
Shivansh S - 2019114003

Advanced NLP Project Final Report

Introduction

In the age of technology, it is natural to ingrain it with all possible fields, including education. With the ever growing amount of educational content it is becoming more and more difficult to generate multiple questions manually. Hence, our project works on question generation based on educational content which will automatically generate questions based on text data as input.

Literature Review

Majority of the research work carried out so far has been focused on three models:

1. Rule Based Generation Model
2. LSTM + Attention + Linguistic Features Model
3. The QG-Net Model

Apart from the rule-based model, we explore each one of the two NN-based models, as explained below:

Learning to Ask: Neural Question Generation for Reading Comprehension

This paper focuses on generating questions for sentences from text passages in reading comprehension by introducing a LSTM+Attention+Linguistic Features model. Seq2Seq Learning is used to train the model and is shown to significantly outperform earlier rule-based models. [**du2017learning**]

QG-Net: A Data-Driven Question Generation Model for Educational Content

This paper introduces a RNN-based model, particularly built for the generation of quiz questions from educational content. The paper compares the results of the earlier model (LSTM+Attention+Linguistic Features) with QG-Net and proves that it outperforms the same. [**inproceedings**]

Our Methodology

As mentioned in our proposal, we aimed to implement the following models:

1. Baseline: LSTM + Attention + Linguistic Features Model
2. Baseline+: The QG-Net Model (trained on GLoVe)
3. Baseline++: The QG-Net Model (trained on BERT)

First part of our problem deals with basic Question Generation, and how well they can be generalised. We will explore how well we can generalise LSTM based encoder-decoder model trained on SQuAD dataset onto question generation based on Educational Data.

Second part of our project will be focused on improving upon our baseline, both based on better word representations and better generalisation. We would use contextual word

embeddings like BERT instead of basic ones like GloVE for the former, and for the latter we would try implementing QGNet, a SoTA generalizable model for question generation.

Methods

Before we look at the methods we employed, let us define certain paradigms under which we can see how our model(s) perform:

1. Just like machine translation, where given a sentence in language A, we translate the sentence to language B, we can employ a similar structure to solve: **given a sentence A, generate a question for the same**. However in the question generation case, the output dimension is much higher since a lot of questions can be constructed from one sentence.

2. Given a sentence and an answer, generate a question whose answer is the given input. The output dimension of this is much lower as compared with the previous paradigm.

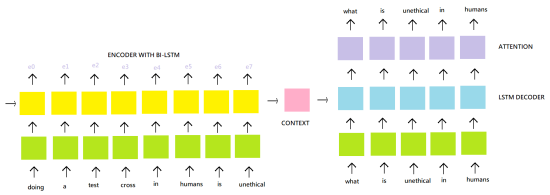
3. Given a paragraph context, generate any question whose answer can be found in the context.

4. Given a paragraph context, and an answer, generate a question that has the answer as the given input.

We now proceed to look at our models to see how we tackle these paradigms.

Proposed Question Generation System - Baseline: LSTM+Attention

We aim to solve paradigm-1 as mentioned in the above section with this model to start with. We build a Bi-LSTM encoder and Decoder Model with Attention as shown in the diagram to proceed with this task.



We try to implement the code for the paper *Learning to Ask: Neural Question Generation for Reading Comprehension*, therefore the model is partially inspired by the way in which a human would solve the task.

To ask a natural question, people usually pay attention to certain parts of the input sentence, as well as associating context information from the paragraph.

Encoder

The encoder is a 2-layer Bi-LSTM, where for a multi-layer RNN, the input sentence, X, after being embedded goes into the first (bottom) layer of the RNN and hidden states, $H = \{h_1, h_2, ..., h_T\}$, output by this layer are used as inputs to the RNN in the layer above. Thus, representing each layer with a superscript, the hidden states and cell states in the first layer are given by:

$$(h_t^1, c_t^1) = EncoderLSTM^1(e(x_t), (h_{t-1}^1, c_{t-1}^1))$$

$$(h_t^1, c_t^1) = EncoderLSTM^1(e(x_t), (h_{t+1}^1, c_{t+1}^1))$$

and similarly for the second layer:

$$(h_t^2, c_t^2) = EncoderLSTM^2(h_t^1, (h_{t-1}^2, c_{t-1}^2))$$

$$(h_t^2, c_t^2) = EncoderLSTM^2(h_t^1, (h_{t+1}^2, c_{t+1}^2))$$

The final hidden state and final cell state is our context vector.

Decoder

The Decoder class does a single step of decoding, that is, it outputs single token per time-step. The first layer will receive a hidden and cell state from the previous time-step, (s_{t-1}^1, c_{t-1}^1), and feeds it through the LSTM

with the current embedded token, y_t, to produce a new hidden and cell state, (s_t^1, c_t^1). The subsequent layers will use the hidden state from the layer below, s_{t-1}^l, and the previous hidden and cell states from their layer, (s_{t-1}^l, c_{t-1}^l). The equations for our decoder hence are as follows:

$$(s_t^1, c_t^1) = DecoderLSTM^1(d(y_t), (s_{t-1}^1, c_{t-1}^1))$$

and similarly for the second layer:

$$(s_t^2, c_t^2) = DecoderLSTM^2(s_t^1, (s_{t-1}^2, c_{t-1}^2))$$

At each time step, the decoder takes each annotation and feeds it to an alignment model to generate an attention score. This attention score is used to focus on the most relevant information contained in the source sentence.

The alignment model is parametrized as a feedforward neural network, and jointly trained with the remaining system components.

Subsequently, a softmax function is applied to each attention score to obtain the corresponding weight value

The application of the softmax function essentially normalizes the annotation values to a range between 0 and 1 and, hence, the resulting weights can be considered as probability values. Each probability (or weight) value reflects how important s_i and c_{t-1} are in generating the next state, c_t, and the next consequent output.

Model

Finally, having the encoder and decoder in place, we implement the model with the Seq2Seq Architecture where the following steps are undertaken:

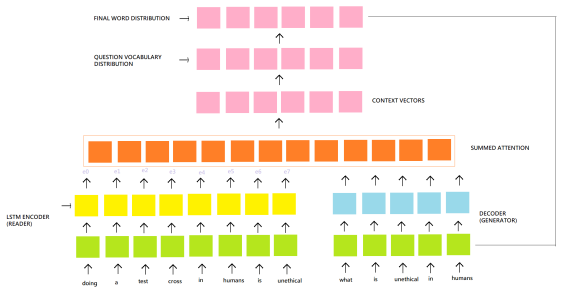
1. Source sentence is input
2. Input is parsed via the encoder to output context vectors after applying the attention weights
3. Context vectors is parsed by the decoder
4. Output Question is generated

The attention weights that are applied in our model utilises the Bahdanau Attention Mechanism. A softmax function is applied to the attention scores, output by our encoder effectively normalizing them into weight values in a range between 0 and 1. These weights together with the previously computed annotations are used to generate a context vector through a weighted sum of the annotations.

Proposed Question Generation System - Baseline++: QG Net (GLoVE)

Taking a look again at the paradigms defined by us before, QG-Net proposes to solve paradigms of type 3 and 4. Quoting from the paper, the authors study the problem of automatic quiz question generation from educational content.

We tried to implement the QG-Net model from our understanding with the following figure:



The model also consists of an encoder-decoder model which s described as Context Reader and Question Generator in the paper.

Context Reader

The context reader is basically an encoder with a unilayer BiLSTM. Each layer can be described similarly as in the previous model as follows:

$$(h_t^1, c_t^1) = EncoderLSTM(e(x_t), (h_{t-1}^1, c_{t-1}^1))$$

$$(h_t^1, c_t^1) = EncoderLSTM(e(x_t), (h_{t+1}^1, c_{t+1}^1))$$

The final hidden and cell state, now instead of directly being passed to the decoder is aggregated with the output of what we feed in the

decoder. Before explaining this step we understand the Question Generator part in the next subsection.

Question Generator

This is the decoder part, where the output question words are input. The layer receives the hidden and cell state from the previouos time step as follows:

$$(s_t^1, c_t^1) = DecoderLSTM(d(y_t), (s_{t-1}^1, c_{t-1}^1))$$

Now, a softmax function calculates a probability distribution over all words from a fixed question vocabulary.

Aggregating over the encoder output as described earlier and the decoder output, the attention weight matrix is calculated to finally generate the context vectors.

An important concept that the implementation of QG-Net relies upon is the inclusion of pointer networks. These pointer network calculates the output word probabilities as a mixture of two probabilities, one over the question vocabulary and the other over the input context vocabulary.

Observations and Insights

Both the models were Seq2Seq models, thus showing that question generation is considered as a sequence-to-sequence learning problem where the input is a sequence (a piece of context, e.g., a sentence or paragraph in a textbook), and the output is also a sequence (a question).

LSTM+Attention

The LSTM+Attention Model utilises attention to generate natural questions for any given sentence. The attention is used to focus on different parts of the context.

QG Net

The QG-Net has a novel approach where the authors say that because several distinct questions can be generated from the same context, depending where in the context you want to ask question, the input sequence also needs to encode the "answer" information, e.g., which part of the input sequence to ask question about.

Results and Analysis

Drawbacks and Challenges

There were certain parts of the QG-Net model which we could not understand perfectly, which might have led to certain gaps in the output. We can enlist them here as follows:

1.

Conclusion and Future Work

From the above observations and analysis, we can see that QG-Net outperforms exiiting work in the field of Question generation, as it successfully generates more "human-intuitive" and natural questions, by considering context and answer encodings as well as inputs.

Both of the models that we implemented can be enhanced further by

1. **Hyperparameter Tuning:** Inclusion of more training data and increasing epochs might lead to better results for each model. With the lack of processing power and train-time, the hyperparameters that were chosen for the purpose of this project were not the best overall.
2. **Pointer Networks:** Although we successfully passed pointer information from the context reader to the question generator in the QG-Net model, the mixing probability concept can be implemented in a more intricate way to encapsulate better context information.

*****