Question Generation in an Educational Context

Kean Leung KEANL@LIVE.UNC.EDU
Ernesto Garcia Vazquez EGV@LIVE.UNC.EDU
Matthew Byrd BYRDOF@LIVE.UNC.EDU
Mary Newby SISYPHUS@EMAIL.UNC.EDU

Abstract

Question generation is a problem that is of great interest to the natural language processing community. If a model can reliably generate questions that meet some specification of quality, then datasets for sibling problems (namely question answering) can be automatically created. However, question generation has an important application in education. If a model can generate questions, then it frees up the time of an instructor to focus on other tasks. In this paper, we address this particular area of question generation by taking a state-of-the-art model in supervised question generation and applying it to an educational dataset.

1. Introduction

Question Generation, creating questions from text, is an important research area in natural language processing (NLP) with applications in question and answer systems and conversational bots such as Alexa, Siri and Google Assistant.

Question Generation (QG) also has many interesting applications in education: assisted learning and educational assessment. Additional practice questions, which are time-consuming for humans to produce, can aid students to improve their reading comprehension, focus on core material and practice recalling past information (G. Kurdi, 2020). Furthermore, QG can create large pools of questions as the basis for adaptive learning methodologies. In Education Assessment, QG protects question validity from overuse where over time questions are disseminated among students in a pre-assessment period. Additionally, QG standardizes the quality of questions and insulates question banks from educators ill-trained in assessments.

We will begin training with the SQuAD data-set like (Zhao et al., 2018); we will then transition to use the SciQ Dataset, which is a high-grade and science domain-specific data-set of 13.K multiple choice exam questions. The SciQ creators added additional questions to SciQ that were generated through an efficient crowd-sourcing method by producing recommendations for choosing documents and possible distractor answers (Welbl et al., 2017).

Two common frameworks for QG are a rule-based approach or a neural approach. The rule-based approach often utilizes linguistic instruments such as NLTK POS Tagger, or the Stanford NER for pre-processing modules; then verb and clause parsing occurs before rule application. Successful implementation of a rule-based approach is demonstrated by (Das et al., 2016). However, rule-based implementations often cause redundancies with information and occasionally have ungrammatical errors. In con-

University of North Carolina at Chapel Hill, COMP 562 Spring 2020, Course Project.

trast, neural approaches tend to highlight the most important information in a sentence from a semantic viewpoint and sound more natural linguistically (Zhao et al., 2018).

Instead of a rule-based approach, we will use an End to End Neural Network with a Sequence to Sequence (seq2seq) Framework which is also known as an Encoder-Decoder Framework as applied in (Zhao et al., 2018). We also frame this approach as a solution to an answer-aware problem: where inputs are a passage and answer and the output is a question to the given answer. A notable feature in this model is a Gated Self-Attention Mechanism to minimize the prevalent NLP difficulties with paragraphs and a Greedy Search Decoder.

In contrast to our methodology, similar research papers in neural models use beam search instead of greedy search as a decoder; beam search is a type of best-first graph search that explores the most promising node first and reduces memory requirements. However, since greedy search has simpler implementation, we chose to use it instead.

As mentioned, a Gated Self-Attention Mechanism used to deal paragraph level information. In human Question Generation, entire paragraphs are often required to develop quality questions due to their richness over sentence level text. However, due to the extraneous information contained in paragraphs, previous state of the art models for paragraphs performed worse at Question Generation than their sentence level equivalence. This approach is specially engineered to be robust with paragraph sized text inputs.

2. Model

2.1. Current State of the Art

The current state of the art in supervised question generation is a sequence to sequence recurrent neural network (RNN) model with a self-attention mechanism. The model (Figure 1) comes from (Zhao et al., 2018). There is another method that is better by some metrics at question generation (Zhang & Bansal, 2019), but this model relies on reinforcement learning and in particular it uses a question-answering model to estimate the reward function. Due to this reason the training time overall complexity is substantially increased for relatively small gains. Therefore, we have chosen to focus on Zaho's seq2seq model in order to test its effectiveness on the SciQ data set (Zhao et al., 2018).

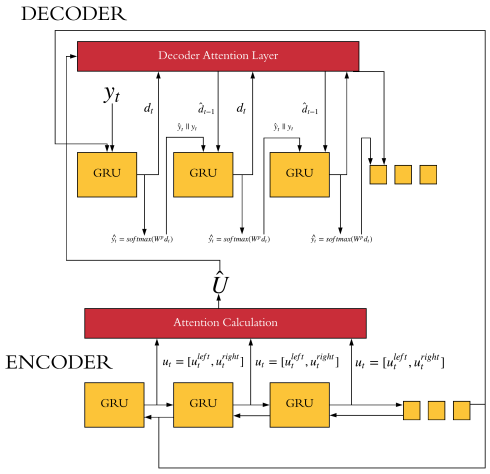

Figure 1. The Seq2Seq model

2.2. Gated Recurrent Units

The core of our model relies heavily on gated recurrent units. These units allow the model to choose what information is passed on at each time step. While a long short-term memory (LSTM) unit could also be used here, based on the findings of (Zhao et al., 2018), these do not actually do much for the model and introduce more parameters. The details of a gated recurrent unit are omitted here, as there are fairly standard and there are no novel additions to them here.

2.3. Attention Module

Attention is a relatively new idea in natural language processing. Attention works by allowing

seq2seq models to select what bits and pieces of the input sequence were most important at each time step during prediction of the output. This is done to address the issue of forgetfulness in longer input sequences, as information at the start of a sequence may be lost in the encoding for the sequence as a whole. Our particular type of attention is called self-attention and it works as follows, (Zhao et al., 2018).

2.3.1. Encoder Attention Calculation

For each word in the input sequence, we calculate a hidden state $\{u_t\}_{t=1}^{M}$. We represent this in the matrix U. For each of those hidden states, we let the model estimate how important it is in the overall sequence of inputs. Our hidden state space has a size of H. Note the use of row vectors instead of column vectors.

$U \in \mathbf{R}^{MxH}$, $u_t \in \mathbf{R}^{1xH}$, $W_a \in \mathbf{R}^{HxH}$

$a_t = softmax(u_t W^a U^T) \in \mathbf{R}^{1xM}$

$a_t := a_t U \in \mathbf{R}^{1xH}$

$g_{tt} = tanh(W^i[a_t, u_t]) \in \mathbf{R}^{1xH}$

$g_{st} = sigmoid(W^{ai}[a_t, u_t]) \in \mathbf{R}^{1xH}$

$\hat{u}_t = g_{tt} \bigodot g_{st} + (1 - g_{st}) \bigodot u_t \in \mathbf{R}^{1xH}$

This is taking all the hidden states, creating a mask with the softmax based on the current hidden state, and applying that mask. Finally, it gives the model a way to learn when to zero out attention states or otherwise modify it. All of the attention states are put into $\hat{U}$.

2.3.2. Decoder Attention Calculation

Finally, the decoder needs a method to look into the attention and determine what parts of the input are important in the current state to generate the next word. To do this, we perform the following calculation (d_t is of course the decoder state at time step t as calculated by a GRU):

$c_t = softmax(d_t W^d \hat{U}^T)$

$c_t := c_t \hat{U}$

$\hat{d}_t = tanh(W^{ds}[c_t, d_t])$

2.4. Prediction

Each time step has to output a word as per the definition of the problem. The paper that this model is based on uses a copy-mechanism that allows the model to copy words directly from the input (Zhao et al., 2018). We have refrained from implementing this as the paper uses an abnormal implementation that lets the model copy or generate at each time step without having to first decide if it wants to copy or generate.

$W^p \in \mathbf{R}^{|V|xH}$

$y_t = argmax(softmax(W^p d_t)$

3. Methods

3.1. Pre-processing

Our model uses BERT word-vectors for the context embeddings. For BIO tags, our model learns an encoding and takes a list of [0, 1, 2] as input with 0 representing O, 1 representing B, and 2 representing I. One important quirk of BERT is that the number of words that can be encoded at once is 512, including the start and end tokens. To get by this, we simply indexed context from the first word to the 510th word. Intuitively, this makes sense, we typically do not need every word in a paragraph to ask a simple question about it.

3.2. Training

Initially, to test the model, we trained it on SQuAD. It seemed to learn quite well as the loss decreased at a reasonable pace. We performed no rigorous tests on this version of the model, and instead moved to try training it on the original problem motivation: the SciQ dataset. The first time training on this dataset, the model seemed to perform nearly as well as it did while training on SQuAD, the loss was decreasing and its output began to match well with the target. However, during this training session something was very wrong. The problem turned out to be that

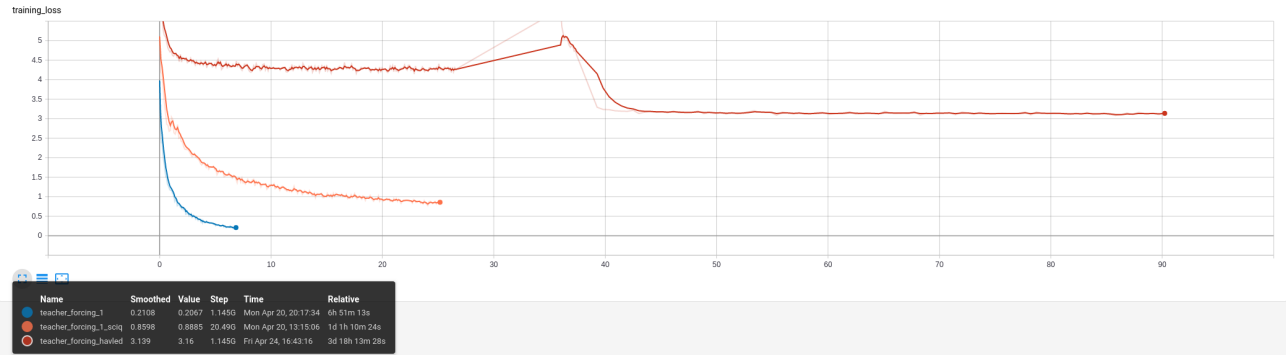

Figure 2. Training loss with varying hyperparameters

we were using teacher forcing at each time step during training. This caused the model to have nearly no generative abilities once teacher forcing was unable to be applied. We changed the teacher forcing ratio to .5 and noticed the model training slower, more sporadically, and it did not reach a very good minimum. One reason for this could be a small batch size. After increasing the batch size, we found that the loss found a new minimum area but the overall performance of the model did not change. Figure 2 shows these results, with the blue representing teacher_forcing equal to one on the SQuAD dataset, and the orange representing teacher_forcing equal to 1 on the SciQ dataset. The red line shows teacher forcing halved with a variation on the batch size during training.

4. Results and Analysis

After training, we tried testing our model using the SciQ dataset with a teacher forcing ratio of 0.5, but we were not able to obtain any sensible results. The problem was that the model, at every time step, was outputting the same exact sentence: [‘CLS’, ‘How’, ‘many’, ‘different’, ‘types’, ‘of’, ‘winds’, ‘[SEP]’]. The model seemed to be not sensitive to changes in the input. We think the problem is that the model is only sensitive to the teacher forcing. Furthermore, we hypothesize that due to batch size or regularization, the encoder is using the entire paragraph, which can also explain why the model outputs the same sentence at every time step. One possible solution to this problem is to split the paragraph into sections and have each section train parallel to each other. However, we did not implement this solution to our model as it would make our program much more complex and run much longer.

5. Conclusion and Future Work

Overall we have implemented a sequence to sequence RNN model with a self-attention mechanism to generate questions using sentences as the given inputs. We have trained our model with the SQuAD dataset and the SciQ dataset. In training, we found that the model worked well with the SQuAD dataset as the training loss decreased at a reasonable pace over time. However, during the model’s training session using the SciQ dataset, we found that at first the model seemed to perform as well as SQUaD, but about halfway the model had nearly no generative abilities. We then tried changing the teacher forcing ratio to 0.5 and increasing the batch size and found that the model’s performance was similar to when the model was training on the SQuaAD dataset. In testing using the SciQ dataset and teacher forcing ratio of 0.5, we found we were not obtaining any sensible results, as the model outputted the same sentence at every time step. One way that might solve this problem would be the split the paragraph into sections and have each section train parallel to each other. However, we did not have

enough time to implement this solution and will be implemented into our model in future works.

In future works, we will generate more sentences so as to be able to use the BLEU and ROUGE-L metric used for automatic machine translation evaluation to better evaluate the performance and accuracy of our model. We did not implement BLEU and ROUGE-L in our model as both metrics require multiple sentences to be able to output a score. In our case, our output is just one sentence.

References

Das, R., Ray, A., Mondal, S., and Das, D. A rule based question generation framework to deal with simple and complex sentences. In *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 542–548, 2016.

G. Kurdi, J. Leo, B.Parsia. A systematic review of automatic question generation for educational purposes. *Int J Artif Intell Educ 30*, 3 (3):121–204, 2020.

Welbl, Johannes, Liu, Nelson F., and Gardner, Matt. Crowdsourcing multiple choice science questions. *ArXiv*, abs/1707.06209, 2017.

Zhang, Shiyue and Bansal, Mohit. Addressing semantic drift in question generation for semi-supervised question answering. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019. doi: 10.18653/v1/d19-1253.

Zhao, Yao, Ni, Xiaochuan, Ding, Yuanyuan, and Ke, Qifa. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3901–3910, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1424. URL https://www.aclweb.org/anthology/D18-1424.