Neural Question Generation: Transformed

Stanford CS224N Custom Project

Jeffrey Hu
Department of Computer Science
Stanford University
jeffhu@stanford.edu

Abstract

We focus on the task of neural question generation for which the goal is to generate fluent sentences from a given sentence or paragraph. This task could have important implications for the field of education as questions can automatically be generated for reading comprehension materials, thus saving educators time from this tedious task. Furthermore, this task could inform and improve the inverse task of question answering. For our baseline model, we examine and implement a vanilla sequence to sequence model based on the work of Xingya Du et. al who pioneered the first major publication on automatic question generation. We also focus on utilizing the transformer architecture for this task. According to a recent survey article covering the field of question generation, scarce attention has been paid to utilizing transformers and pre-trained contextual embeddings, so our work provides interesting and new insights [1].

1 Key Information to include

- Mentor: Peiyu Liao
- External Collaborators (if you have any): None
- Sharing project: None

2 Introduction

Question generation (QG) is defined as the task of "automatically generating questions from various inputs such as raw text, database, or semantic representation" [1]. As people have the ability to ask rich and creative questions, we see that QG requires a comprehensive understanding of the source material. The task of QG is beyond simple syntactic transformations of declarative sentences. Constructing a natural question often involves text summarization, using synonyms for terms, and using entities from previous sentences or clauses.

Beyond presenting a challenging and unique problem in the field of natural language processing, the task of QG also offers many practical applications. In the field of education, automatically generated questions from learning materials can save educators time and can help better standardize educational materials and tests. Furthermore, QG systems can serve as important components to chatbot applications, as asking natural and directed questions can improve user experience. For chatbots used to gauge and support mental health conditions, QG can be especially important. Furthermore, QG systems can be used to develop data sets for NLP research particularly in the field of question answering (QA). This is important because it reduces the need and cost of human labor that are currently needed to produce such data sets. Researchers have even begun to frame QG and QA as dual tasks that are trained simultaneously, and primary results have shown that this training framework improves both QG and QA tasks [1].
3 Related Work

For decades, QG research has primarily dealt with two major facets: the identification of key concepts in a context to form questions upon ("What to ask") and incorporating these concepts into a natural and syntactically correct question ("How to ask"). In the past before neural approaches to QG, research separately considered these two aspects through context selection (the problem of "what") and question construction (the problem of "how"). For instance, given a sentence as context, content selection finds a topic through semantic and syntactic parsing and given the topic and context, question construction determines the question type ("Who", "What", "When", etc) and constructs the question using pre-defined question templates or transforming the input sentence. One of the first papers to detail this rule-based approach was written by Rus et al [2]. However, both context selection and question construction require well-crafted rules based on expert linguistic knowledge. And this architecture lacks true semantic awareness of the context and produces questions that are not robust since question generation is limited to templates and generic transformations.

Neural models revolutionized QG research since they provide an end-to-end framework in which content selection and question construction are jointly optimized. The neural approach is fully data-driven and requires no external rules that require deep linguistic knowledge. Furthermore, these models largely provide more robust and flexible questions. The majority of neural models in the literature follow the sequence-to-sequence (Seq2Seq) framework [3]. The first major publication to utilize a neural framework to QG is the Learning to Ask paper by Du et al. Subsequent publications largely cite Du et al. one way or another, and thus, we decided to set out baseline model to be similar to the model described in Du’s paper.

[CLS] The Super Bowl 50 was played at Santa Clara California. [SEP] Santa Clara, California. [SEP]

Figure 1: Example of an input in an answer-aware question generation setting.

The field of question generation has predominantly used towards answer-aware tasks for generation [4]. An example of this is figure 1. In this setting, instead of feeding in only context, the target answer is demarcated or attached to the context. Under this setting Song et al. proposes a nonparametric generative model, jointly trained for question generation and answering [5]. Zhao et al. obtained state of the art results on this task by using a gated self-attention encoder and a maxout pointer decoder [6]. The advantage of using such a framework is that there is much more signal to be able to model the exact question that should be generated. We attempt to solve the answer-agnostic version of the task, where the target answer is not available in the input to the model. This makes the task much more difficult, since a particular context could have many valid questions, but the model would incur high loss as long as the question did not match the target question. The ability to generate questions without a constraint on an answer is very attractive, as it increases the model’s degrees of freedom. One application of such a model might be a chat bot interacting with a human user.

4 Approach

4.1 Task Definition

In this section we define the question generation task, as modeled in Du et al. Given an input sentence \( x \) we attempt to generate an output sentence \( y \) that is a syntactically and semantically valid question, pertaining to the input sentence. The output \( y \) can be a sequence of arbitrary length: \([y_1, ..., y_y]\). For a given sentence, there are many possible valid sentences that could be generated. However, to create a clear target label to calculate loss from, we only pretend there is only one "correct" question that we are attempting to generate. Suppose the length of the input sentence is \( M \), then we can split \( x \) into a series of tokens \([x_1, ..., x_M]\) that is fed into the model.

4.2 Du et al

As a reference model, we reimplement the model from Du et al in PyTorch.
They use the RNN encoder-decoder architecture with a global attention mechanism (mechanism described in the paper by Luong and Manning). In particular, they use multiplicative attention. Additionally, they use fixed GloVe vectors with 300 dimensions for word embeddings.

Given some input sequence \( x \), the task is to generate a natural question sequence \( y \). The authors aim to maximize \( y = \arg \max P(y|x) \) where \( P(y|x) \) is the conditional log-likelihood of the predicted question sequence \( y \) given the input sequence \( x \).

They examine two variations of models: one that encodes sentence level information and another that encodes paragraph level information. In the sentence model, the encoding of the sentence is the initial hidden state for the decoder. In the paragraph model, they use the concatenation of the sentence encoder’s output and the paragraph encoder’s output as the initial hidden state for the decoder.

In the sentence level encoder, they use a bidirectional LSTM and concatenate the last hidden state of the forward and backward pass. In the paragraph level encoder they set a length threshold \( L \) and truncate the paragraph at the \( L \)th token before using the bidirectional LSTM to encode the truncated paragraph. To decode, they use an LSTM and use beam search with \( k = 3 \). To handle rare words, when they encounter a decoded \(<UNK>\) token at some time step \( t \), they replace it with a token from the input sentence that has the highest attention score at that time step.

4.3 BERT-fused NMT

For the second model we implement a BERT-fused NMT described in the recent paper of Zhu et al that achieves state of the art results on neural translation tasks [7].

While BERT is more conventionally fine-tuned for particular tasks, the authors find that using BERT purely for the contextual embeddings allows it to perform very well on translation tasks. Broadly, this model is a sequence to sequence model with contextual embeddings generated by BERT. However, instead of only using it as input embeddings, the representation generated by BERT is fed into all layers. A BERT-encoder attention module is used to control how the various layers respond to the BERT inputs. The decoder uses both the encoder hidden states combined with the BERT inputs.
to make predictions. The weighing of this combination is controlled by a BERT-decoder attention model. Technical details can be found in the paper.

4.4 GPT-2

Since question generation is a content generation task, we fine-tune GPT-2 to generate input sentences \[^{[8]}\]. We use the HuggingFace implementation of GPT-2, and construct a dataset for it to train on. To do so, we take the input sentences \(x\) and concatenate the output sentence \(y\), yielding strings of \([x_1, ..., x_M, "Question" = "y_1, ..., y_N]\), where each element in the array is a token created from byte-pair encoding (BPE). Then, each sentence-question pair \(s_i\) is joined with a newline token, creating a file \([s_1, \backslash n, s_2, \backslash n, ..., s_D]\) where \(D\) is the size of the dataset. At test time, GPT-2 is seeded with a context sentence, and the "Question: " prompt. It then generates tokens autoregressively until the question mark character is generated and all of the tokens generated up until that point are used as the question.

the writings of samuel pepys describe the pub as the heart of england.

Question: who said that pubs are the heart of england? \n
they are typically located in the country or along a highway.

Question: where is an inn typically located? \n
Figure 4: First two lines of input to GPT-2 finetuning

5 Experiments

5.1 Data

We utilize the Stanford Question Answering Dataset (SQuAD) which is a reading comprehension dataset consisting over 100,000 questions produced by Amazon Mechanical Turk workers from approximately 536 Wikipedia articles \[^{[9]}\]. For any question in the dataset, the answer is a segment of text in the context associated with the question. Du et. al made their train, development, and test splits publicly available, and to aid comparison between our new methods, we use the same splits in our work throughout. The training set contains 70,484 examples, the dev set 10,570 examples, and the test set 11,877 examples.

| Sentence: | Oxygen is used in cellular respiration and released by photosynthesis, which uses the energy of sunlight to produce oxygen from water. |
| Questions: | - What life process produces oxygen in the presence of light?  
photosynthesis  
- Photosynthesis uses which energy to form oxygen from water?  
sunlight  
- From what does photosynthesis get oxygen?  
water |

Figure 5: An example of a sample context (sentence) and three questions from SQuAD. Source: \[^{[10]}\]

5.2 Evaluation method

It is hard to quantify "good" questions since good questions tend to be significant, syntactically correct, semantically sound, and natural. As a result, recent QG research tend to utilize human evaluation by randomly sampling a few hundred generated questions and asking human judges to rate
them. However, human evaluation can be labor intensive, time consuming, inconsistent, and hard to reproduce, so researchers still use automatic evaluation metrics even though studies have shown that automatic evaluation metrics do not correlate well with fluency and coherence [1]. In our work, we use BLEU, ROGUE, and METEOR.

BLEU measures the average $n$-gram precision on a set of references. A BLEU-$n$ score indicates a BLEU score calculated using up to $n$-grams. METEOR is an improvement upon BLEU as it calculates similarity between generated questions and references by considering stemming, synonyms, and paraphrases. ROUGE focuses on recall – how much words in gold standard references appear in generated questions.

### 5.3 Experimental details

All of the following models were trained on a Tesla K80 GPU, through Azure.

#### 5.3.1 Du et al

These match the implementation details as described in the paper. Instead of using stochastic gradient descent with a learning rate of 1, Adam optimizer was used with a learning rate of 0.01 as it was found to converge faster. Additionally, GLoVe pretrained embeddings were used [11]. The total training time for a run took about 1 to 2 hours, matching the time described in the Du paper. The original authors used Lua in their implementation whereas we used PyTorch.

#### 5.3.2 BERT-fused NMT

For the BERT model, Adam optimizer was also used with a learning rate of 0.0005. The model was run for 15 epochs, and the best model was chosen by validation loss. The loss function was a label smoothed cross entropy, with 0.1 label smoothing. Gradients were clipped when the norm exceeded 25. The BERT model used was "bert-base-uncased". The BERT model was frozen (not training along with the encoder and decoder). ReLU activation functions were used. The model trained for about 4 hours. Mini-batch size was 64. The input datasets were tokenized and byte-pair encoded with the Moses NLP library before being fed into the model. Decoding is done with beam search with a width of 5.

#### 5.3.3 GPT-2

Since our dataset is relatively small (10MB), the HuggingFace medium size GPT-2 was used, with the corresponding tokenizer. Once again Adam optimizer was used with an epsilon of 1e-8, and no weight decay. Gradients were clipped at 1. Learning rate of 5e-5. Only one training epoch was performed over the training set to prevent overfitting. The total training time took about 1 hour. The training batch size was 4 samples. During decoding time, the GPT-2 temperature is set to 1, with no repeats allowed. Decoding is done with a beam search with a width of 5.

### 5.4 Results

Note that IR$_{BM25}$, IR$_{Edit Distance}$, MOSES+, DirectIn, and H&S all refer to rule based question generation models that predate neural models.

As expected, the reimplementation of Du et al. performs roughly as well as the original Du et al with pretrained embeddings. Disappointingly and perhaps unexpectedly, both transformer based models perform worse than Du. It appears that the BLEU1 scores of the transformer based models are relatively competitive compared to their BLEU 4. We believe that the transformer based models we implemented tend to perform worse because of large amount of pretrained context. This is investigated more in the qualitative analysis section. The transformer models display relatively high BLEU-1 scores, implying that many of the words overlap. However, comparatively the BLEU-4 scores are relatively low, implying that structurally the sentences generated are not very similar to the target sentence. It is unlikely that the models are not learning, as the loss and perplexity drop very fast. One technical possibility for why the models are failing is that the models overfit too quickly making it difficult to discover a good model through validation. It is also possible that our approach was flawed, and requires precise hyper parameter tuning to properly unlock the power of these models.
Table 1: Automatic evaluation results of different systems by BLEU 1-4, METEOR and ROUGE\(_L\)

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU-1</th>
<th>BLEU-2</th>
<th>BLEU-3</th>
<th>BLEU-4</th>
<th>METEOR</th>
<th>ROUGE(_L)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRBM25</td>
<td>5.18</td>
<td>0.91</td>
<td>0.28</td>
<td>0.12</td>
<td>4.57</td>
<td>9.16</td>
</tr>
<tr>
<td>IR_Edit Distance</td>
<td>18.28</td>
<td>5.48</td>
<td>2.26</td>
<td>1.06</td>
<td>7.73</td>
<td>20.77</td>
</tr>
<tr>
<td>MOSES+</td>
<td>15.61</td>
<td>3.64</td>
<td>1.00</td>
<td>0.30</td>
<td>10.47</td>
<td>17.82</td>
</tr>
<tr>
<td>DirectIn</td>
<td>31.71</td>
<td>21.18</td>
<td>15.11</td>
<td>11.20</td>
<td>14.95</td>
<td>22.47</td>
</tr>
<tr>
<td>H&amp;S</td>
<td>38.50</td>
<td>22.80</td>
<td>15.52</td>
<td>11.18</td>
<td>15.95</td>
<td>30.98</td>
</tr>
<tr>
<td>Vanilla seq2seq</td>
<td>31.34</td>
<td>13.79</td>
<td>7.36</td>
<td>4.26</td>
<td>9.88</td>
<td>29.75</td>
</tr>
<tr>
<td>Du et al (no pre-trained)</td>
<td>41.00</td>
<td>23.78</td>
<td>15.71</td>
<td>10.80</td>
<td>15.17</td>
<td>37.95</td>
</tr>
<tr>
<td>Du et al (w/ pre-trained)</td>
<td>43.09</td>
<td>25.96</td>
<td>16.50</td>
<td>12.28</td>
<td>16.62</td>
<td>39.75</td>
</tr>
<tr>
<td>Our Du et al</td>
<td>43.03</td>
<td>24.74</td>
<td>16.60</td>
<td>10.91</td>
<td>16.08</td>
<td>38.97</td>
</tr>
<tr>
<td>BERT-fused NMT</td>
<td>26.39</td>
<td>8.61</td>
<td>3.85</td>
<td>1.98</td>
<td>7.06</td>
<td>24.72</td>
</tr>
<tr>
<td>GPT2</td>
<td>11.16</td>
<td>3.39</td>
<td>1.17</td>
<td>0.00</td>
<td>4.00</td>
<td>10.87</td>
</tr>
</tbody>
</table>

However, several permutations over hyper parameters and model structure were attempted to yield these results, making it less likely that there is a simple fix, if there is one at all.

6 Analysis

As noted in the quantitative analysis, the transformer models disappointingly did not perform as well as we expected. We qualitatively examine outputs of each of models below in an attempt to understand our results.

We see that Du’s model has the most coherent and natural question from the input even though the question is of type “Where” rather than the “Who” question posed by the target. However, Du’s model is still lacking since the question is not answerable or semantically proper. We see the output produced by BERT does not even have a relevant subject as “new haven” does not appear anywhere in the input sentence. This irrelevancy is seen in a lot of the other outputs produced by the BERT model. This is mirrored by the GPT-2 outputs, that mentions a “colony”. We suspect that these large pretrained models are bringing in context from their prior training, that is dominating the signal from these training examples. In particular, since the Du model is trained from scratch, it becomes forced to learn that the words that come from the sentence have a high likelihood of appearing in the question. However, BERT and GPT-2 may be trying to emulate the sentiment and context of the question, rather than the actual information inside of it. Although this particular example doesn’t demonstrate it well for GPT-2, it’s clear that the models have both learned to ask mostly syntactically and semantically valid questions that are complex. The part they fail on seems to be understanding that the questions are relevant to the passage.

Additionally BPE might play a role in the failure of these models to perform well, as Du learns to replace <unk>s with words with high attention values, effectively copying over words from the input. BERT and GPT-2 dice up works that they don’t understand, and fail to regenerate these diced up out of vocabulary words.
7 Conclusion

In this paper, we replicate the results of Du et al. for question generation, and attempt to exceed the performance through recent advancements in transformer architectures. We are able to successfully adapt these complex models to the task of question generation. Although the performance of a few baselines is exceeded, large pretrained models such as GPT-2 and BERT as used in this paper do not perform as well as Du et al. We speculate that this is due to the high number of parameters found in these pretrained models. Nevertheless, this paper hopefully gives insight into the effectiveness of these cutting-edge methods on a smaller subfield of natural language processing, and at the same time asks new questions. For future research, we would like to investigate whether or not different variations of GPT-2 and BERT would have better success, such as fine-tuning the BERT for Question Answering. Additionally, understanding why these models fail to perform as well as a seq2seq with attention (Du et al.) might provide insight into ways that the next generation of natural language processing models might be architected.

References


