A Comparative Study of Neural Question Generation Models

Stanford CS224N Custom Project

Ankit Dwivedi
`ankitd@stanford.edu`

Arunothia Marappan
`arunothi@stanford.edu`

Abstract

In this project, we aim to do a quantitative and qualitative comparison of the performance of different neural network architectures on the task of generating questions (QG) from text passages using the SQuAD dataset [1].Specifically, we plan to 1) reproduce the sequence to sequence attention based LSTM model detailed in our reference paper [2], 2) implement a transformer model that uses multi-headed attention as described in [3], and 3) fine-tune a pre-trained language model like BERT [4] or GPT-2 [5] on the QG task. We present some promising results from (1), and briefly discuss the current status and future plans for (2) and (3).

1 Key Information to include

- Mentor: Hugh Zhang

2 Introduction

This project is motivated by the task of question generation (QG) from text passages, which has several key applications, especially in the field of education. (Xinya Du et al., 2017) [2] and several following works have shown that neural models help in generating natural questions from text passages that, when compared to traditional rule-based approaches, are more grammatical, fluent, and challenging in terms of syntactic divergence from the original reading passage and the reasoning needed to answer.

Encouraged by the successful application of neural models to Question Generation, we pursue a comparative study of the performance of different neural network architectures on the QG task, to gain insights into the nuances of applying deep neural networks to a challenging real world NLP problem. In this project, we have implemented and trained a multi-layer vanilla seq2seq LSTM network (without attention) to obtain a baseline, and attention based seq2seq networks with different hyperparameter configurations to try and reproduce the results presented in our reference paper [2]. We describe the details of these models in the next 2 sections.

We implemented the models from scratch using PyTorch. The code is available at (Github https://github.com/ankitdwivedi23/cs224n-project). We used the starter code provided at (Github https://github.com/minggg/squad) for the overall project scaffolding and adapted it to our requirements, and also referred to the coding part of assignments 4 and 5. We trained and evaluated both the models on the full SQuAD 1.0 dataset.

3 Related Work

Before the progress made in deep learning and neural networks, QG had been mostly tackled in the past via rule-based approaches. The state-of-the-art system at the time was an overgenerate-and-rank approach that generates multiple questions from an input sentence using a rule-based approach and then ranks them using a supervised learning-based ranker (Heilman et. al, 2009) [6].

The rule-based approaches rely heavily on a manually crafted feature set, and the questions generated often overlap word for word with the tokens in the input sentence, making them very easy to answer. A good question should consist of more than syntactic transformation of a declarative sentence, and the authors [2] proceed to propose a neural model that achieves this objective.

Our reference paper [2] was one of the first applications of a neural model to the QG task.

4 Approach

4.1 Baseline

QG is the task of generating a natural question y related to information in a given input context passage x. Formally, the QG task is defined as finding $\bar{y}$, such that:

$$\bar{y} = \arg\max_{y} P(y|x)$$

where $P(y|x)$ is conditional log-likelihood of the predicted question sequence y, given the input x.

Our baseline models this conditional probability using a vanialla encoder-decoder architecture without attention mechanism. The input sequence is represented using an embedding layer, whose weights are trained along with the rest of the model. The embeddings are passed to 2 layer bidirectional LSTM encoder. A unidirectional double layer LSTM decoder decodes the final encoder hidden state by applying a linear projection layer and a softmax layer to the LSTM cell output at each step, predicting an output word for that step. The decoder repeats this process until the $< EOS >$ token is predicted.

The model's training objective is to minimize the negative log-likelihood of the training data with respect to all the parameters (denoted by θ). Given a training corpus containing S sentence-question pairs, the loss function is defined as:

$$\mathcal{L} = -\sum_{i=1}^{S} \log P(y^{(i)}|x^{(i)};\theta)$$

At train time, we use teacher forcing. At test time, inference is done with beam search using a beam width of 3.

4.2 LSTM with Attention

In our implementation of the model described in [2], we basically implement the same network as the baseline, but add an attention layer that uses multiplicative attention to make the decoder focus on certain elements of the input when generating each output question word. The attention layer comprises of a linear projection layer with weight matrix W of shape $\mathbb{R}^{h*2h}$, where h is the hidden size of the LSTM layer, followed by a softmax layer and dot product with each encoder hidden state to get the attention weights. This is illustrated in Figure 1.

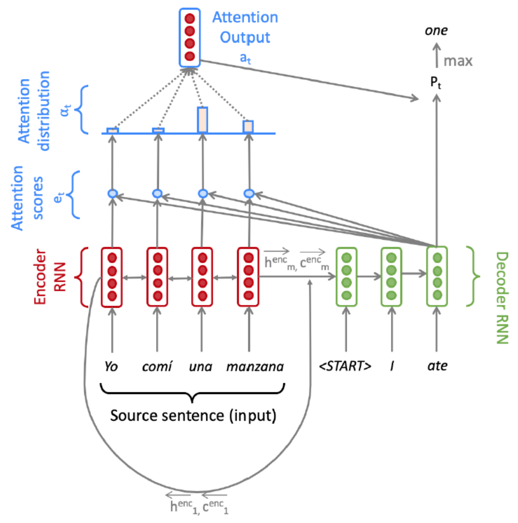


Figure 1: LSTM with Attention as illustrated in Assignment 4

4.3 Transformer

We also implemented a Transformer model that uses multi-headed attention as described in [3]. For this, we used the Transformer modules available in the torch.nn PyTorch package. We also used the implementation provided in [7] to debug and fix our own implementation, by fitting it on a toy dataset. Unfortunately, we ran out of time to try and tune this model on the full dataset, but will continue working on it.

5 Experiments

5.1 Data

We used SQuAD, (Rajpurkar et al, 2016) [1] dataset which contains $100,000+$ question-answer pairs on $500+$ articles. We extracted the questions asked on each context and built a context-question pair dataset for our task of question generation, and filtered out unanswerable questions. Additionally, we truncated the context to contain only the tokens within a +/- 10 span of the answer tokens. While this is not ideal and leads to removing important context in many cases, we applied this pre-processing to simplify the task for the initial versions.

The train set consists of 86579 context-question pairs, while the validation set consists of 2848 pairs. Since the test set did not have the answer span available for context truncation, we use the validation set to report our results. We have not used the validation set for any hyperparameter tuning, as we use the hyperparameter configuration provided in our reference paper.

5.2 Evaluation method

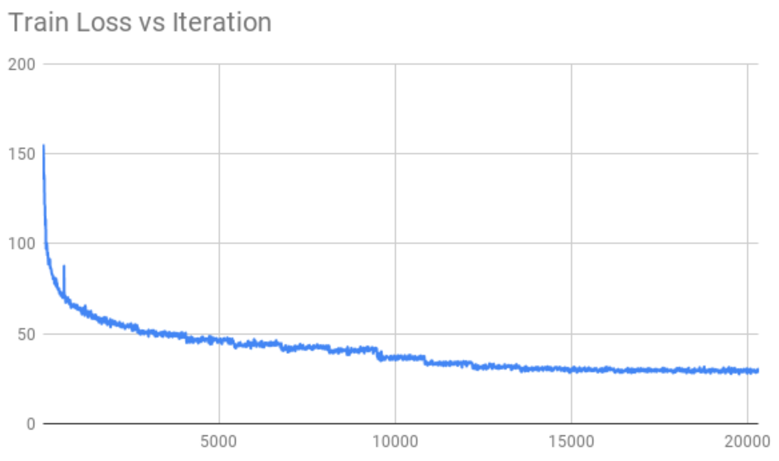
We used BLEU score for model evaluation. We built the gold reference corpus by incorporating all questions linked with the context of a candidate question.

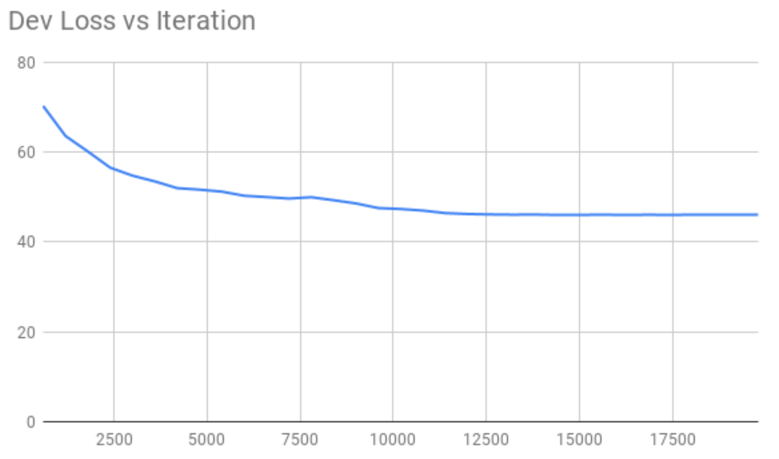
5.3 Experimental details

We use the same hyperparameter settings as provided in our reference paper.

- **Embeddings**: We trained our model with glove.840B.300d word vectors.
- **Optimizer and Learning Rate**: SGD, with learning rate initialized to 1.0 and decreased by 0.5 on each epoch starting at epoch 8.

- **Training Time**: Models were trained for 15 epochs, and take between 2 to 3.5 hours to train on the entire train set.





5.4 Results

The below table presents the BLEU scores on the validation set for our baseline and attention models for different configurations of encoder-decoder hidden size and dropout rate. It also compares them with the scores presented in the reference paper [2] (evaluated on the full SQuAD set).

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4
Xinya Du et al, 17 [2]	43.09	25.96	17.50	12.28
Baseline (vanilla seq2seq) (hidden size = 300, dropout = 0.2)	20.78	10.57	5.81	3.19
Seq2Seq with attention (hidden size = 100, dropout = 0.2)	21.56	11.02	6.26	3.79
Seq2Seq with attention (hidden size = 300, dropout = 0.2)	25.22	14.11	8.70	5.63
Seq2Seq with attention (hidden size = 600, dropout = 0.2)	24.32	13.46	8.24	5.21
Seq2Seq with attention (hidden size = 600, dropout = 0.3)	22.81	12.37	7.44	4.73

6 Analysis

The questions generated by the best performing model are far from perfect, but there are some promising signs for future improvements. We present an analysis of the model outputs, along with some interesting examples, in the next few subsections.

6.1 Well Formed Questions

These are hand-picked examples, and do not reflect the general performance of the model, but these examples show the effectiveness of a simple encoder-decoder model powered by the attention mechanism.

Well formed questions

Context: ... The Centre currently realizes about 500 projects a year. <unk> National Gallery of Art, the oldest exhibition site in Warsaw , with a ...
Reference Question: What is the oldest exhibition site in Warsaw ?
Generated Question: What is the oldest art site in Warsaw ?

Context: ... leftist in democratic competition with Islamists, had announced banning alcohol and nightclubs within six months, shortly before he was overthrown ...
Reference Question: What had <unk> planned on banning within six months , before he was overthrown ?
Generated Question: How long did it take to ban alcohol ?

Context: ... 2,000 professors . Other institutions for higher education include the Medical University of Warsaw, the largest medical school in Poland and one of the most prestigious, the National Defence University, highest military academic institution in Poland , the Fryderyk ...
Reference Question: What is the largest medical school in Poland ?
Generated Question: What is the largest medical school in Poland ?

Context: ... disputed Scotland 's southern borders. William invaded Scotland in 1072, riding as far as <unk> where he met up ...
Reference Question: When was Scotland invaded by William ?
Generated Question: What year did William invade Scotland ?

Context: ... The next major step occurred when James Watt developed (<unk>) an improved version of <unk>'s engine, with ...
Reference Question: When did Watt finish the development of his improvements to <unk>'s engine ?
Generated Question: Who developed an improved version of <unk> ?

6.2 Out of Vocabulary Tokens

As evident from the examples in the previous subsection, there are quite a few out-of-vocabulary tokens (<unk>) in the input contexts and generated questions. Below are some more examples of instances where the model generated a <unk> despite the corresponding word being present in the input context. This problem can be overcome to an extent by using character embeddings or byte-pair encodings.

Out of vocabulary tokens

Context: ... principal advantages the Rankine cycle holds over others is that during the compression stage relatively little work is required to drive the pump, the working fluid being in its liquid phase at this ...
Reference Question: What is a main advantage of the Rankine cycle ?
Generated Question: What is the main benefit of the <unk> cycle ?

Context: ... The embargo caused an oil crisis , or " shock " , with many <unk> and long-term effects on global politics and the global economy. It was later called the "first oil shock", followed by the 1979 oil crisis, termed ...
Reference Question: What was another term used for the oil crisis ?
Generated Question: What was the cause of the <unk> ?

Context: ... On August 15, 1971, the United States unilaterally pulled out of the Bretton ...
Reference Question: When did the United States withdraw from the Bretton Woods Accord ?
Generated Question: What year did the US withdraw out of the <unk> ?

6.3 Irrelevant or Unanswerable Questions

The subject of some of the generated questions is a topic or entity that has not been mentioned in the input context. There are cases of unanswerable questions as well, where the fact or data need to answer the question is missing from the context. We suspect that overfitting might be a primary cause of the former behavior, while the short length of the truncated context might be a factor contributing to the latter.

Irrelevant or unanswerable questions

Context: ... drive the pump, the working fluid being in its liquid phase at this point . By Condensing the fluid, the ...
Reference Question: During the compression stage of the Rankine cycle, what state is the working fluid in ?
Generated Question: What is the main component of the brain ?

Context: ...in steam turbines, turbine entry temperatures are typically 565 C (the creep limit of stainless steel) and condenser ...
Reference Question: What is the turbine entry temperature of a steam turbine , in degrees Celsius ?
Generated Question: What is the temperature limit of copper ?

Context: ... nonexistence of the ultraviolet catastrophe , proved troublesome . Through the work of leading theoretical physicists, a new theory of electromagnetism was developed using quantum mechanics. This final modification to electromagnetic theory ultimately led to ...
Reference Question: What was used to create a new electromagnetic theory to reconcile the troubles with electromagnetic theory as it used to stand ?
Generated Question: What was the new theory of quantum mechanics ?

Context: ... The clinical section is located in a <unk> building with 700 beds , 10 operating theatres , an intensive care unit ...
Reference Question: How many beds does the Maria <unk> - Curie Institute of Oncology have ?
Generated Question: How many floors does the FBI section have ?

6.4 Overfitting

Another clear indication of the model over-fitting on the training set is the repeated use of certain phrases in the generated questions. For instance, 52% of the generated questions in the validation set start with "what is the name". There is also a suspiciously high count of questions starting with "what is the term". The current model applies dropout after the first LSTM layer in the encoder and the decoder, and after the attention layer. We can use some other regularization techniques, like dropout in more layers, L2 regularization, label smoothing etc., to reduce this over-fitting.

7 Conclusion & Future Work

The models we implemented in this project helped us go through and experience the nitty-gritty of the complete development and debugging cycle of implementing neural networks. A lot of our iterations went into debugging and fixing code issues. Given all of that, the results from the initial models look promising. However, there remains a lot of scope for future work, and we are certainly not done with this project. Some immediate next steps for us would be:

1. Try and reproduce the results of our reference paper by hyperparameter tuning and more effective context pre-processing. The truncated shorter contexts we used for our models removes a lot of important information that can help in generating more fluent, meaningful, and challenging questions.
2. Try more regularization techniques like controlled teacher forcing (teacher forcing with probability p at each decoder time step, where p is a hyperparameter), L2 regularization, dropout in more layers, label smoothing etc.
3. Train the transformer model on the full set.
4. Fine-tune a pretrained lanugage model like BERT.

8 Acknowledgments

We would like to extend the warmest gratitude to the entire teaching staff for not only a wonderfully enriching learning experience, but also for their thoughtful and empathetic adjustments in these uncertain times. We would also like to thank our mentor Hugh Zhang for patiently answering our questions and giving us very useful debugging tips that helped us fix our models.

References

[1] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics (ACL)*, 2018.

[2] Xinya Du, Junru Shao, and Claire Cardie. Learning to ask: Neural question generation for reading comprehension. *CoRR*, abs/1705.00106, 2017.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[5] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[6] Michael Heilman and Noah A. Smith. Question generation via overgenerating transformations and ranking. 2009.

[7] Alexander Rush. The annotated transformer. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 52–60, Melbourne, Australia, July 2018. Association for Computational Linguistics.