CS230

Question Generator
Natural Language Processing

Lupe Hernandez
lupeh
lupeh@stanford.edu

Sam Randall
samuellr
samrandall@stanford.edu

Ahmad Nazeri
an1965
ahmad.nazeri@stanford.edu

Abstract

Being able to ask the right questions is a key skill for any person. We aim to train a machine learning algorithm to generate questions using an article as the input. In this paper, we construct a sequence-to-sequence architecture with attention, which consists of an encoder and decoder aimed at generating coherent and relevant questions from a source file. We chose to build our model from scratch to better understand the end to end process of leveraging machine learning. To test our results, we leveraged human level cognisance and evaluation to measure the quality of our generated questions in terms of understand-ability and relevance. We were able to generate questions that captured many of the nuances of English so they could be understood by a human but unfortunately, some of the questions were unrelated to the article or had some additional words appended to the question.

1 Introduction

In much of the world, and especially in the United States, the demand for teachers and teaching resources has increased dramatically (U.S Department of Education, 2019) . With this in mind we still see that the burden of developing quizzes or creating relevant questions from an article is still very much a human-centric task. Manually building these sets of questions about a text passage requires experience, resources, time, and often training. Therefore, this latter approach oftentimes is not feasible due to time constraints, budget cuts, or insufficient instructor training. Advancements in question generation would aid in cases such as those described above.

Despite a large number of studies on Neural Question Generation, it remains a challenge to create high-quality questions from unstructured data in large quantities (Bang et al., 2020). The authors of this paper have chosen to adopt the conventional approach of formulating this task as a Sequence-to-Sequence problem using various encoders, decoder, attention layers, and GLoVE word embedding to improve the quality of the questions being generated. In this way, the authors are able to inject some level of understanding and structure into the process of formulating a question.

To ensure the quality of generated questions this paper uses human proxies to decide whether or not a given output question makes sense, is relevant to the article or subject at hand, and is useful in an educational environment. Since human level understanding and cognition approximates Bayes' error sufficiently well for this reading comprehension task, we want to make sure that our model outputs results consistent with what a human would ask.

CS230: Deep Learning, Fall 2020, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

2 Related work

More modern approaches have relied on **multi-task Question Generation**. Mrinmaya (2018) achieved comparable or better performance over all the previous Natural Language Processing models by jointly training the question generation and answering models. More recently, Bang et al. (2019) explicitly made use of a multi-tasked labeling strategy to identify whether a question should be copied from the input passage or be generated instead. This approach by Bang et al. further helps to guide the model to learn the accurate boundaries between copying and generation. This distinction between copying and generation is very similar to the learning processes children undergo when they are developing the ability to formulate questions.

For our project, the **Learning to Ask: Neural Question Generation for Reading Comprehension** paper (Du, Shao, Cardie) played a major influence on our project. The paper describes a RNN encoder-decoder model with attention in order to focus on important aspects of the input data. While the paper explores two variations of the model; one with encoding sentences and another encoding both sentences and paragraphs, for our project, we focused on sentence level encoding. With respect to attention, the **Attention Is All You Need** paper really helped us understand the attention mechanism. The attention function maps the query and key-value pairs to the output. This would allow us to better generate questions using the model described in the Learning to Ask paper.

A final series of papers seeks to improve the quality of generated questions by feeding the encoder with extra information. Yifan et al. (2018) proposes an end-to-end framework to generate questions of designated difficulty levels to better mimic the questions several types of people might ask. In this example, we expect there to be a high level of machine reading comprehension in order to produce questions of varying difficulty.

Deepak and Kaheer (2019) analyzed the effect of incorporating "world knowledge" into neural question generation models (Deepak and Kasheer 2019). More specifically, the authors of the paper *Improving Neural Question Generation using World Knowledge* use linked entities to the Wikipedia knowledge base and "fine grained entity types" (Deepak and Kasheer 2019). Feeding this knowledge base to the encoder helps the model to generate the correct word or phrase instead of an incorrect word or phrase.

3 Dataset

The **S**tanford **Qu**estion **A**nswering **D**ataset1.1 (SQuAD v1.1) provides two distinct files, one for training and another for development. The SQuAD v1.1 dataset is a reading comprehension dataset, consisting of questions posed by crowd-workers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage. Not to be confused with the SQuAD v2.0 dataset, the 1.1 version does not contain unanswerable questions. This was a deliberate choice by the authors since building a machine learning model that determines when no answer is supported is beyond the scope of our overall objective. In all, the SQuAD v1.1 dataset contains over 500 Wikipedia articles with over 100,000 pairs of questions and answers. Having this nicely defined data allowed our team to easily ingest and execute our code. Below is an example of

Example:

- **Article**: As of the 2010 census, there were 579,999 people, 230,233 households, and 144,120 families residing in the city. The population density was 956.4 inhabitants per square mile (321.9/km^2). There were 256,930 housing units at an average density of 375.9 per square mile (145.1/km^2).
- **Question**: What was the density of the population per square mile?
- **Answer**: 956.4

A significant portion of time was spent formatting the SQuAD dataset such that it would work well in our model architecture. For preprocessing step we separated the data into their component sentences, questions, and answers into a CSV file. These component parts were then segmented (tokenized) into individual words and punctuation marks which was subsequently used to build our corpus of vocabulary terms. We leveraged GLoVE with 300d, which has a corpus of 400,000 words. Simply, put GLoVE allowed us to take a corpus of text and transform each word in that corpus into a

position in a high-dimensional space. This means that similar words will be placed together, therefore expanding our vocabulary.

4 Methods

We have trained two models using different model architectures, one with just an encoder and one with an encoder and a decoder. We found lackluster results with just the encoder model, but it was a good first exercise. We found that without the decoder, a sequence of words that made sense was an unlikely output from our model.

The decoder and encoder Recurrent Neural Network model was first implemented with only a single layer to get the general framework down. Then the model was amended to run with two layers and then ultimately 10 layers. Although this model produced results that exceeded our previous attempts, it too was insufficient. A transition was made to a sequence-to-sequence model that employed an attention layer. In our model, the attention layer would take in the previous hidden states of the decoder and all the stacked forward and backward hidden states of the encoder to output an attention vector. This attention vector will be the length of the source sentence. The idea behind the attention layer is to try and capture how well the encoder hidden state matches the previous decoder hidden states. This model, by far, has achieved the best results. The basic architecture can be seen below in Figure 1. Additionally, we increased the number of layers for both the encoder and decoder which did not improve our results in the way we were hoping. The sentences being generated were comprised of a question word followed by several iterations of the word "the." Once we added the attention layer, our results improved significantly. Multiple word sentences, including nouns, adjectives, and subjects were finally being generated, and more than that, they actually made sense. This breakthrough put us in a good position to start the process of hyper-parameter tuning.

The process that this Sequence-to Sequence model implements is as follows. First we start off by creating vocabulary dictionaries, then padding and splitting this dictionary of vocabulary terms into inputs/labels. Our model then assigns word indices to each vocabulary term and feeds these indices into an embedding layer, the output of which goes into GRU with dropout set to zero for single layer GRU. This output gets fed into the decoder, which itself consists of an embedding layer and a fully connected layer. Our model runs 30 epochs, each with batch learning applied, and after each epoch we validate our model on the validation set and save the model if it improves the validation loss. We use a batch size of 128 and a training set size of 18000 questions. We experimented with multi-layer networks but settled on a 1-layer gated recurrent network with 512 hidden units. Additionally, we use the Adam optimizer as that is generally considered to be best practice in the deep learning literature.

We also experimented with various sized embedding layer and found the more detailed information the better, as we imported an embedding layer from GLoVE. However, we found that our model yielded questions with similar words to what we wanted (e.g. if an article was about Beyonce, it would ask about Rihanna), so we reasoned a 300 dimension embedding space would better differentiate between similar words. The results, indeed, proved us correct, as in this final version we saw far fewer "close to correct" nouns.

We trained the model using cross-entropy loss. We decided to use cross entropy loss for our baseline model because it is a good loss function for minimizing the distance between two probability distributions; the predicted and the actual (Zhuang, 2020). We noticed that good performance in our loss function did not necessarily indicate it was asking reasonable questions so we propose that improving upon this function will optimize our results to include the learned understanding of articles better.

4.1 Hyper-tuning

During the process of tuning our models, we did focus on tuning the learning rate, epochs, dropout, and the hidden layers for encoder and decoder, and experimented with a randomized grid search of combinations of various values.

4.1.1 Learning Rate

Range : [0.001 1, distributed on a log scale.], For the learning rate, we did leverage static learning rates as well as using dynamic learning rate through the following formula: $L = l - l(e/E)$ where

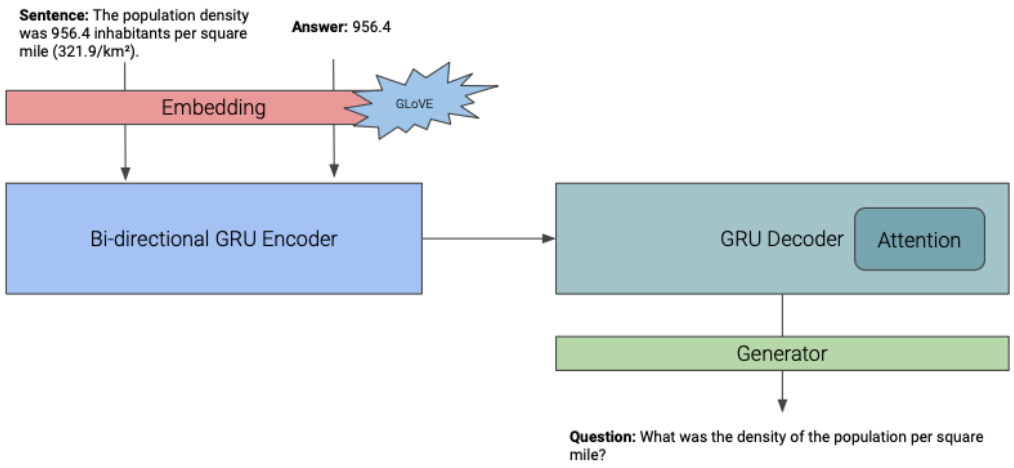

Figure 1: Model Architecture (*for detailed version, see appendix*)

L=learning rate, l=starting learning rate, e=the current epoch iteration, E=the total number of epochs. Based on a grid search, we noticed that having a static rate of 0.001 gave the best results for our model. Thus, we started to leverage a 0.001 learning rate going forward.

4.1.2 Epochs

At first we started training our model with 10 epochs, but found that the algorithm was generating questions with multiple repeating words. It became clear that the model was not sufficiently learning the correct way to structure a question, so the number of epochs was increased by increments of 5. We would the best results when we trained our model with 30 epochs.

4.1.3 Dropout

For the dropout, we tried many numbers ranging from 0.0 to 0.5 and noticed that for a single GRU layer, having a dropout rate of 0.0 was best.

4.1.4 Hidden Layers

As previously mentioned, we attempted to increase the number of layers, both for the encoder and decoder and found that the time to train the model increased significantly. The results we obtained were just as good, or in many cases worse that the results we obtained from a single encoder and decoder model, so it was decided to remain with the latter architecture.

5 Experiments/Results/Discussion

5.1 Evaluation of Question Quality

Our system generated results varied in quality. We had clear examples of questions being generated that were identical to the human generated questions, as well as examples of nonsensical questions that were unanswerable based on the articles they were about. In an effort to weigh the potential of this project, we chose the questions that seemed to make the most sense to use in this empirical evaluation. This is excluding the questions that were identical, as that would not have made sense for the proposed questionnaire below.

1. Are questions that are generated using our model as understandable as human-generated questions?
2. Are questions that are generated using our model as relevant to a given article topic as human-generated questions?

3. Are questions that are generated using our model perceived as useful as human-generated questions?

In the evaluation phase, we invited eight volunteers to rate a set of questions using a scale from one to three (1 - NOT AT ALL) , (2 - SOMEWHAT), (3 - VERY) to answer the aforementioned categories.

For the sake of brevity we considered at most three Wikipedia articles from which to generate our questions. For each Wikipedia article, we mixed one human-generated question with one automatic or system generated question and asked our volunteers to use the scale described above to score each question. We are testing our research hypothesis that our our hand-chosen system-generated questions are just as understandable, relevant to a given article, and useful as those created by humans.

5.2 Results

The results were divided into the three research categories (Understand-ability, Relevance, and Usefulness), and by whether or not the question was generated by our model or by a human. A summary of the results can be found in Table 1. A two-tailed T-Test was then conducted with a significance level of 0.05 to test our hypothesis.

	Understand-ability	**Relevance**	**Usefulness**
	Mean (s.d.)	**Mean (s.d.)**	**Mean (s.d.)**
System-GQ	2.13 (0.86)	2.13 (0.68)	11.57 (0.62)
Human-GQ	2.90 (0.40	2.80 (0.41)	2.77 (0.43)
Difference	*t=0.77*	t=0.67	t=1.20
Significance	p = 0.0001	p = 0.74	p = 3.14E-11

Table 1: The results of our survey

In the context of the three Wikipedia articles given to our volunteers, Table 1 shows that the mean of understand-ability for human-generated questions (2.90) is a quite a bit higher than that of system-generated questions (2.13). This demonstrated that their difference is statistically significant, meaning that human-generated questions are superior in terms of understand-ability. With respect to the relevance of the questions to the given article, the mean of the score for human-generated questions (2.80) is also higher than of the system-generated questions (2.13). However, given the p value, we see that their difference is not significant enough to reject out hypothesis. Our system-generated questions are just as relevant as those generated by humans, at least within the scope of our empirical evaluation. However, in the context of the usefulness of questions for supporting a student's understanding the mean of human-generated questions (2.77) is significantly higher than of system-generated questions (1.57). It becomes clear that in terms of relevance our system-generated questions are just as relevant to the article as those created by humans, but in terms of understand-ability and usefulness our algorithm falls short. As stated above, the results of our program vary in efficiency and quality. Some are reasoned questions with clear answers near or identical to ones created by humans. Others are sentences in which the meaning was not always conveyed and the question not always answerable.

6 Conclusion/Future Work

Our model is able to generate questions that can be understood by humans in general but with some hiccups here and there. Since the beginning of the course, our goal has been to build the project from scratch in order to get a better understanding of the end-to-end pipeline for machine learning projects. Because of our focus on learning, our model requires some additional fine tuning in order to develop questions that make sense every time and are related to the particular article. We do have plans to continue working on this project after this quarter and hope to continue learning about what it takes to develop an end-to-end machine learning pipeline.

Possible future work would include even more fine tuning for data preprocessing as well as adding beam search into the model. The beam search would allow the model to determine the best sequence of words for the question by generating multiple possibilities. Additionally, another possible avenue for future work would include developing an answer generation component in order to completely meet the needs of the students for reading comprehension. We are really excited for our model and hopefully, its use will allow students improve their reading comprehension in the future.

7 Contributions

First of all, we want to thank Jo Chuang for being our advisor on this project and providing guidance to group.

- **Sam Randall**: Sam implemented the initial pre-processing step, the initial working model with LTSMs, added embedding to the decoder network, and then did the grid search to find optimal parameters for our finalized version.
- **Guadalupe Hernandez**: With the help of his teammates Guadalupe was able to incorporate the GLoVE word embeddings into the model. He also played a key role in the implementation of the attention layer. When the team was experiencing latency issues due to training, he transferred the working code to a Google Colab workbook with GPUs to help the process along. Additionally, Guadalupe created and conducted the empirical evaluation for our results.
- **Ahmad Nazeri**: Ahmad focused on identifying some research that we could leverage and as well as try to implement some of them. Since, the group figured out a working model, Ahmad refocused on helping the team to identify ways to improve it. First, Ahmad incorporated the nltk to split sentences more accurately as well as run some iterations with a dynamic learning rate. Ahmad helped identify some small mistakes within the code and identified ways to fix it. And finally, he focused on the paper, presentation, as well as cleaning up the Github.

References

[1] Barthold Albrecht & Yanzhuo Wang & Xiaofang Zhu "Who is Ernie and if so how many? A multitasking Bert for question answering with discrete reasoning". CS230 papers (Spring 2019).

[2] Li Dong & Nan Yang & Wenhui Wang & Furu Wei "Unified Language Model Pre-training forNatural Language Understanding and Generation" In: arXiv: 1905.03197v3 (2019).

[3] Roberto Frias & Pedro Azevedo. “Exploring NLP and Information Extraction to Jointly Address Question Generation and Answering”. In: Maglogiannis (2020).

[4] Pranav Rajpurkar & Jian Zhang. “SQuAD: 100,000+ Questions for Machine Comprehension of Text”. In: arXiv preprint: 1606.05250v3 (2016).

[5] Bang Liu , Haojie Wei , Di Niu , Haolan Chen , Yancheng He. 2020. Asking Questions the Human Way: Scalable Question-Answer generation from text Corpus. arXiv:2002.00748v2 (2020)

[6] Bang Liu, Mingjun Zhao, Di Niu, Kunfeng Lai, Yancheng He, Haojie Wei, and Yu Xu. 2019. Learning to Generate Questions by Learning What not to Generate. In The World Wide Web Conference. ACM, 1106–1118.

[7] Carnegie Mellon University. "New AI enables teachers to rapidly develop intelligent tutoring systems." ScienceDaily. ScienceDaily, 30 April 2020.

[8] Deepak Gupta, Kaheer Suleman, Mahmoud Adada, Andrew McNamara, and Justin Harris. 2019. Improving Neural Question Generation using World Knowledge. arXiv preprint arXiv:1909.03716 (2019)

[9] Michael Heilman. 2011. Automatic factual question generation from text. Language Technologies Institute School of Computer Science Carnegie Mellon University 195 (2011).

[10] Mrinmaya Sachan and Eric Xing. 2018. Self-training for jointly learning to ask and answer questions. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). 629–640.

[11] U.S. Department of Education. 2019. “Office of Postsecondary Education: Teacher Shortage Areas” (webpage), accessed March 2019.Yllias Chali and Sadid A Hasan. 2015. Towards topic-to-question generation. Computational Linguistics 41, 1 (2015), 1–20.

[12] Yifan Gao, Jianan Wang, Lidong Bing, Irwin King, and Michael R Lyu. 2018. Difficulty Controllable Question Generation for Reading Comprehension. arXiv preprint arXiv:1807.03586 (2018).

[13] Zhuang Liu, Kaiyu Huang, Degen Huang, and Jun Zhao. Semantics_reinforced Networks for question Generation. 24th European Conference on Artificial Intelligence - ECAI 2020

[14] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need In: arXiv preprint: 1706.03762v5 (2017).

[16] Xinya Du, Junru Shao, Claire Cardie. Learning to Ask: Neural Question Generation for Reading Comprehension In: arXiv preprint: 1705.00106v1 (2017).

Appendix

The model architecture below captures the inner workings of our model.

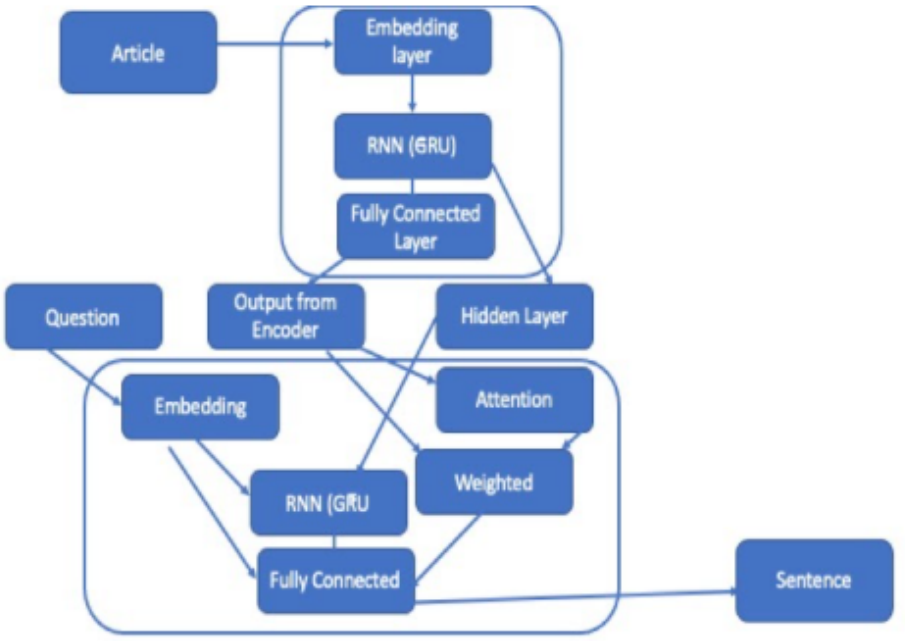

Figure 2: Detailed Model Architecture