Question Generation System using Seq2Seq

Rajat Agarwal rajat.agarwal@nyu.edu
Tushar Jain tushar@nyu.edu
Kumar Mehta kjm627@nyu.edu

Dept. of Computer Science
Courant Institute of Mathematics
New York University
New York, NY 10011

Abstract

We propose an end-to-end model based on sequence to sequence natural language models to address the task of question generation conditioned on a given reading comprehension and answer. We propose a new architecture involving two encoders and a decoder along with an attention mechanism. This architecture is used to find probable answers and subsequently generate questions conditioned on the context and answers found. We train our model on the recently assembled SQuAD dataset.

1 Introduction

Asking questions is the simplest yet the most effective way of learning and actively processing information. Question generation has been defined (Rus and Arthur, 2009) as the task of automatically generating questions based on a natural language text input. The task of question generation is an essential component for many NLP problems and systems such as chat-bots, artificial intelligent applications and learning environments where seeking information is the key. A major portion of the human knowledge is recorded in the form of written content and thus generating questions based on text can have multi-fold applications.

Question generation is a complex task and involves several components. Firstly, it requires the understanding of the corresponding context by the system, also known as the task of Machine Comprehension. Secondly, it requires identification of potential answers in the context based upon which the question can be framed. Thirdly, it may require knowledge external to the given context in order to frame a good question. And finally, it requires knowledge of the language semantics and syntax in order to generate a valid interrogative sentence.

Several good datasets have come up in the recent past, which comprise of question-answer pairs that are based on a reading comprehension and articles. These include the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016), the MAchine Reading COmprehension (MS MARCO) dataset (Nguyen et al., 2016), Hermann et al., etc. We train our model on the SQuAD dataset, consisting of question-answer pairs for a number of Wikipedia articles.

Context:
New York University (NYU) is a private non-profit research university based in New York City. Founded in 1831, NYU's main campus is centered in Manhattan, located with its core in Greenwich Village, and campuses based throughout New York City.
Questions:
- What type of university is NYU?
- When was New York university founded?
- In what borough is the university located?

Figure 1: Given an unknown context, our model identifies interesting answers and generates questions conditioned on the context and answers. Sample sentence from the first paragraph of the wikipedia article on New York University.

Cho et al. proposed a sequence to sequence model consisting of 2 parts, the encoder that trains to encode a sequence of words into a fixed length vector representation, and the decoder that trains to decode this vector into another sequence of words. We use a slight modification of this model along with an attention layer introduced by (Luong et al., 2016) which makes the decoder target different parts of the encoder outputs. We use GloVe embedding (Pennington et al., 2014) to represent words as vectors and capture semantic parsing and syntax information. Thus, our model is

completely data driven and there are no manually written rules or assumptions about the input reading comprehension.

2 Related Work

In the last few years, a lot of new research has been done on the question generation task with (Rus et al., 2010) being one of the first ones. The initial attempts were focused on rule-based approaches (Mostow and Chen, 2009) where the context is converted into its syntactic representation, which is then used for question generation. Heilman and Smith introduced a ranking system for questions. They generated large number of questions by manually writing rules to perform syntactic transformations from declarative sentences to interrogative sentences and then used the ranking system to evaluate them. Mostafazadeh et al. automatically generated questions based on a given image and thus attempted to make connections between language and vision. Yuan et al. (2017) proposes one of the first attempts at an end to end recurrent neural model for natural language question generation from documents, conditioned on answers. They use supervised learning, an adapted variant of seq2seq model with softmax-pointer formulation by (Vinyals et al., 2015) in combination with reinforcement learning to improve the quality of the question generated. Du et al. attempts to introduce attention-based sequence to sequence learning model and investigate the effects of encoding on a paragraph and sentence level information.

Many datasets consisting of question-answer pairs based on a document have been recently released. Hermann et al. observed that paraphrase sentences that are associated with documents can be converted to context-query-answer triples. Using this approach, they have collected two new corpora of roughly a million news stories with associated queries from the CNN and Daily Mail websites. Nguyen et al. extracted anonymized user queries from real web documents and coupled them with human generated answers to generate the MS MARCO dataset consisting of 100k+ queries. Trischler et al. released the NewsQA dataset comprising of 100k+ human generated question answer pairs based on news articles from CNN. Rajpurkar et al. released the Stanford Question Answering Dataset (SQuAD), consisting of high quality questions posed by crowd workers, based on a set of Wikipedia articles.

3 Model

We take inspiration from the sequence to sequence model proposed by Cho et al. We propose an architecture which consists of 2 encoders namely document encoder and answer encoder, coupled with decoder. We use an attention layer between answer encoder and decoder. We make no assumptions about the nature of the input and train the model end to end using log-likelihood loss.

3.1 Document encoder

We use 100-dimensional GloVe, introduced by Pennington et al. to embed each context word into a 100-dimensional vector giving us G = (g^1, g^2, ..., g^D) a D x 100 matrix, representing the entire context paragraph.

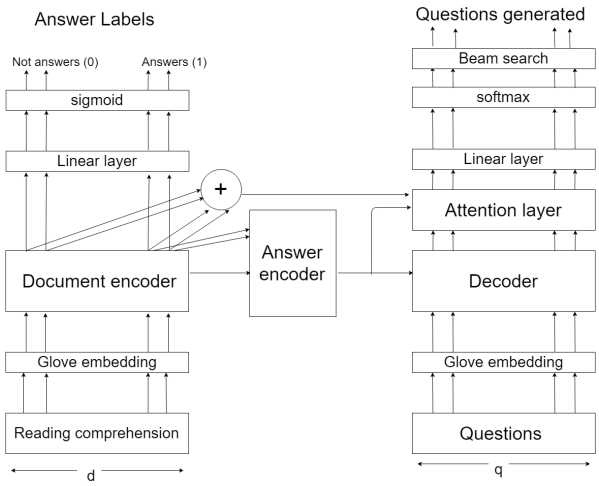

Figure 2: Architecture of the proposed network. The network consists of 4 main modules: a) Seq2Seq Encoder and Decoder b) Answer Generation module c) Attention Mechanism d) Beam Search

We then use a bi-directional long short-term memory network (LSTM) (Hochreiter and Schmidhuber, 1997), to encode the sequential document context into a single fixed length vector representation. The input to the encoder at timestep t would be the vector g^t and the hidden state of the previous timestep t-1. For a LSTM network with hidden size h, the document encoder gives a h dimensional forward and backward hidden state, which are then concatenated to give a final 2h dimensional vector representation V of the context document.

$$\vec{h}_t^1 = LSTM(g^t, \vec{h}_{t-1}^1)$$

$$\overleftarrow{h}_t^1 = LSTM(g^t, \overleftarrow{h}_{t+1}^1)$$

$$h_f^1 = (h^d; h^0)$$

$$h^1 = (\vec{h}^1; \overleftarrow{h}^1)$$

The vector representation is the initial state to the answer encoder while the output of each timestep is projected onto a binary plane and is used to identify potential subsets of answers around which the final question will be generated. The outputs of bi-LSTM are also used as an input to the attention mechanism to focus on words surrounding the answer based on which the question generated is conditioned upon.

3.2 Answer generation

The output of the document encoder is then passed through a linear layer, followed by a sigmoid activation function to output a binary feature, which signifies whether a word is an answer or not. The answers are passed to the answer encoder and thus used in the attention mechanism to focus on words related to the probable answers while generating questions.

$$answer\ labels = sigmoid(w_a * h^1 + b_a)$$

The answer labels generated through the above equation are used during inference when the actual answers are not provided. During training, we train the model to generate these answer labels and directly use the answer labels provided in the training data.

3.3 Answer encoder

The answer encoder adds another encoding layer for context words in the paragraph that are identified as a part of the answer. Without any loss of generality, we assume that the answers are a continuous subset of the document (Ds, ..., De) where $1 \leq s \leq e \leq D$. The answer encoder takes the outputs from the document encoder that are answers to questions (while training) or that are identified as potential answers (during inference), and the hidden state output of the document encoder. The hidden state is then passed to the attention layer.

$$A = (h_s^1, ..., h_e^1)$$

$$h_0^2 = h_f^1$$

$$h_t^2 = LSTM(A_t, h_{t-1}^2)$$

3.4 Attention layer

The attention layer enables the decoder to target different sub-parts of the encoder output. Our attention mechanism is inspired by the attention model introduced by Luong et al.. The hidden state from the answer encoder is compared to all the outputs of the document encoder to compute attention weights ts for all context words. We then compute a context vector as the weighted average of these outputs based on the attention weights computed. Finally, we combine this context vector ct with the answer encoders hidden state, to yield the attention vector at. This vector is combined with the output vector of the decoder to generate the final output vector.

$$score(h^1, h_{t-1}^3) = (h^1)^T W_s h_{t-1}^3)$$

$$a_{ts} = \frac{exp(score(h^1, h_{t-1}^3))}{\sum_{t=1}^{Q} exp(score(h^1, h_t^3))}$$

$$c_t = \sum_{Q} \alpha_{ts} h^1)$$

$$a^t = f(c_t, h_{t-1}^3) = tanh(W_c[c_t, h_{t-1}^3] + b_c)$$

3.5 Decoder

The decoder converts the single vector representation of the input sequence given by the encoder and with activations from the attention layer and sequentially gives out the generated question words. The decoder is a neural model consisting of single-directional LSTM that generates outputs sequentially. At each timestep, the network generates a probability distribution over the output vocabulary |V| conditioned on parameters Θ which are encoded document D and answer labels A.

We use a single-directional LSTM network to decode the output sequentially for each timestep t. This takes as input the question label for the current time-step and the hidden-state vector of the previous timestep. The decoder's first cell's hidden state is initiliazed by the hidden state of the answer encoder. The output of the decoder are passed on to the attention layer, which generates the attention-based activations for each timestep. These 2h-dimensional attention-based vector activations are then projected on to the dimension of the vocabulary |V| and are passed through a softmax layer to generate word probabilities for each time-step t.

$$h_t^3 = LSTM(y_{t-1}, h_{t-1}^3)$$
$$y_t = softmax(w_0 * h_t^3 + b_0)$$
$$p_\theta = (y_t | y_{<t}, D, A)$$

4 Experimental setup

In this section, we first describe the training corpus, explain our training and inference setups and give implementation details about our model.

4.1 Dataset

The SQuAD dataset consists of 536 high-PageRank Wikipedia articles with human-generated corpus of (document, question, answer) triples covering a variety of subjects. Questions are formulated by crowdworkers in natural language and answers are spans of text in the related paragraph highlighted by the same crowdworkers. We divide the training set into 80-20% as shown in Table 1 to use as training and development sets respectively and use the development set provided by them as our test set.

Number of pairs (Train)	86080
Number of pairs (Validation)	8000
Number of pairs (Test)	9040
Avg. number of tokens in context	157
Avg. number of tokens in question	11

Table 1: Distribution of data in train, test and validation sets along with average token lengths of contexts and questions in the entire dataset.

4.2 Training and inference

While training, we use the outputs of the document encoder for only answer labels which are provided to the answer encoder. Also, we use teacher forcing where the outputs are compared to the actual question labels at each timestep and the input to next timestep is the actual ground truth question label. The model is trained to reduce the answer loss given by the cross-entropy loss and negative log likelihood loss (NLL loss) for the question generator given respectively by:

$$L_A = -\sum_{d \in D} y_d * log(\hat{y}_d) + (1 - y_d) * log(1 - \hat{y}_d)$$

$$L_Q = -\sum_t (log p_\theta(y_d | y_{<t}, D, A))$$

During inference, if the answer labels are not provided, then we first find answers from the outputs of the document encoder, which are then provided to the answer encoder. For identifying answers in the context, we assume that an answer is a continuous subset of the context, and we don't try to find optimal answers by considering subsets of the answer string found considering the performance of our current system and operational efficiency as explained in Figure 3. The output of each timestep of the decoder is passed as input for the next timestep. We also use beam search (Graves, 2012) to optimize the decoders performance. The idea of using beam search is to widen our search for the most optimal translation by considering a subset of translations (with a beam width of 10) instead of going forward with just the most optimal translation at each step.

4.3 Implementation Details

We implement our models in Tensorflow using the seq2seq system and libraries. We use nltk parser to tokenize sentences in the context as well as the question labels. We thus extract context-question answer pairs from the training dataset and convert lengths of all contexts to the maximum length context D in the data by adding <eos> words to remaining contexts of lower lengths. During inference, we clip the context lengths to this maximum length D if any context length is greater than D. Similarly, we convert the lengths of all questions to match the length Q of the question with maximum words during training as well as inference. In the first phase of the inference, first we find out the probable answers in the context. For this, we set a threshold of λ = 0.35. For each cluster of probable answers, we then condition our decoder to generate the questions based on the answers. For context and question encoding, we use word embedding of 100 dimensions initialized by glove.6B.100d pre-trained word embeddings.

We define our vocabulary |V| as the 200k most frequently occurring words in our dataset, including the stop-words, the start of sentence symbol <sos>, end of sentence symbol <eos> and padding symbol <pad>. The remaining words are categorized as unknown words and have the representation <unk>. We set the hidden size for our document encoders and decoder to 500 and number of layers of LSTMs to 2 in the question decoder and the answer encoder, keeping number of

Context:
NYU is also a worldwide university, operating NYU Abu Dhabi and NYU Shanghai, and centers in **Accra, Berlin, Buenos Aires, Florence, London, Madrid, Paris, Prague, Sydney, Tel Aviv, and Washington, D.C.**

Figure 3: Given an unknown context, our model identifies interesting answers and identifies the complete cluster of adjacent tokens as one answer, although a subset of these clusters can also be valid answers.

layers 1 in the intermediate encoder and setting the dropout probability between the layers to 0.2. We train with mini-batch size 128 and learning rate = 0.004 using Adam Optimizer, coupled with gradient clipping to avoid exploding gradient problem. We set the number of hidden units in the attention mechanism to 500 as well and feed it with the outputs of the document encoder and hidden state to generate hidden state with attention for the decoder. Our code is available on GitHub.[1]

4.4 Evaluation metrics

We use automatic machine translation evaluation metrics like BLEU score, vector semantic cosine similarity score (using SpaCy framework by Honnibal and Montani and hereby called SpaCy for brevity) and perplexity, along with human evaluation to evaluate the questions generated by our model and compare it to the questions generated by the baseline sequence to sequence model. BLEU score calculates the average n-gram precision on a batch of reference sentences. SpaCy computes the cosine similarity at the sentence level, using the 300 dimensional GloVe embeddings to represent individual words. Perplexity is a measurement of how well a probability distribution or probability model predicts a sample. We find perplexity using the formula :

$$PPL = e^{-\sum_x p(x) \ln p(x)}$$

For human evaluation, we ask professional English speakers to evaluate generated questions by our model, on a scale of 1-5, after providing them the context and the answer based on which the question was generated.

We use these metrics to compare results from 3 different question generation models - A baseline sequence to sequence model consisting of a basic encoder decoder system, the model architecture presented above using beam search and the same architecture, not using beam search.

[1] https://github.com/tshrjn/qgen

Model	BLEU	SpaCy	PPL	Human
Seq2Seq	7.7	0.438	442.4	–
Our Model (with beam)	13.3	0.953	158.3	4.6
Our Model (w/o beam)	11.1	0.918	139.3	4.3

Table 2: Table showing performance of our model compared to the baseline seq2seq model. PPL is the perplexity measure (lower is better).

5 Results and Analysis

Table 2 shows the summary of results across the three models, using various evaluation metrics. We see that using our model architecture with beam search performs better across all the given metrics. The bleu scores are low for all three models as it only considers n-gram precision, while there can be valid questions generated for the same context-answer pair which have a very different semantic structure. The question generated by our model received an average rating of 4.4/5 in the human evaluation. The questions were judged on the basis of 4 criterion namely : 1. Understandability, 2. Clarity, 3. Ambiguity and 4. Ease of answering the question by 30 professional English speakers.

The baseline model performance is not upto the mark on any evaluation criterion used. This is because the model tends to learn the questions seen in the training set. Hence, the questions generated on the test set does not pick words from the context, and simply repeats questions seen in the train set. Our model outperforms the baseline model on all evaluation criterion. The reason is the attention mechanism and a separate layer of answer encoding, which forces the decoder to pay heed to context around the answer while generating questions. Using beam search further improves the quality of the generated questions. The reason is that beam search does not just consider the word with the highest probability, but considers top k (k is the beam width) optimal translations for every time step, hence generating a subset of k most optimal questions from the entire set of questions which

can be generated. Also, our model first predicts probable answers, and then captures information of the importance of other words around the answer before generating the question, which helps in improving the quality of questions formed. We

Context 1: The classic case of a corrupt, exploitive dictator often given is the regime of *Marshal Mobutu Sese Seko*, who ruled the Democratic Republic of the Congo (which he renamed Zaire) from 1965 to 1997.
Ground truth question: Who ruled nigeria until he died in 1998 ?
Generated question: Who was the president of zaire ?

Context 2: India (IAST: Bhrat), also called the Republic of India (IAST: Bhrat Gaarjya), is a country in South Asia. It is the seventh-largest country by area, the second-most populous country (with over *1.2 billion people*), and the most populous democracy in the world.
Ground truth question: How many people live in India ?
Generated question: What is the population of Republic Of India?

Context 3: Tuvalu consists of *three* reef islands and six true atolls. Its small, scattered group of atolls have poor soil and a total land area of only about 26 square kilometers.
Ground Truth Question: How many reef islands does the tuvalu group have ?
Generated Question: How many atolls are in the atolls of ?

Figure 4: Comparison between generated and ground truth questions for different contexts with answers highlighted in the paragraph

also observe that training the model with decaying teacher forcing ratio generates questions with higher evaluation scores. By reducing teacher-forcing after every n-epochs, the model learns from its previous mistakes, and makes more careful selections at a given time-step, thus generating questions which are more fluent and valid for the given context and answer. We use a teacher-forcing decay ratio of 0.9, and decay the metric after every 5 epochs. We train our system on entire dataset for 40 epochs. We observe that training the model for more than 40 epochs results in a degraded performance on test-data set, despite of reduced loss for training-set, which implies that the model starts to overfit the training data.

Our system does not always give a perfectly generated question. In example 3 of Figure 4, we see that our system generates questions with words which are in the given context, but the overall question itself doesn't make logical sense.

6 Conclusion

We propose a end to end sequence to sequence model for question generation, conditioned on potential answers from a given context. We show how to train this model by using beam search and varying th teacher forcing ratio. We then see that our model used with beam search during inference performs significantly better than the baseline sequence to sequence model using the automatic evaluation metrics. We also conduct human evaluation and see that the questions generated by our model are evaluated better.

For future work, we would like to investigate our findings further and try to explore use of policy gradient optimization strategies to help reduce the out-of-context words in generated questions and increase the fluency (PPL) and accuracy (BLEU and SpaCy similarity) measures. We would also like to investigate the effects of using question-answer pairs generated as supplementary training data to Machine Comprehension tasks, and see the impact of these on their performance.

7 Contributions

- Rajat Agarwal - Worked on implementing variations of seq2seq model, built data parser for SQuAD dataset, and worked on GAN as an alternative to seq2seq model.
- Tushar Jain (Audit) - Implementation of seq2seq model, setup up code base for hyper-parameter tuning and worked on Language Modeling as a baseline model.
- Kumar Mehta - Worked on data parser for SQuAD dataset, implemented attention mechanism in the main model, created module for testing generated questions correctness using Spacy Similarity Score and Bleu Score.

References

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR* abs/1406.1078. http://arxiv.org/abs/1406.1078.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. *arXiv preprint arXiv:1705.00106* .

Alex Graves. 2012. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711* .

Michael Heilman and Noah A. Smith. 2010. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, HLT '10, pages 609–617. http://dl.acm.org/citation.cfm?id=1857999.1858085.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *CoRR* abs/1506.03340. http://arxiv.org/abs/1506.03340.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear* .

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP), Volume 1: Long Papers*. Association for Computational Linguistics, Beijing, China, pages 11–19. http://www.aclweb.org/anthology/P15-1002.

Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Larry Zitnick, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. 2016. Generating natural questions about an image. *CoRR* abs/1603.06059. http://arxiv.org/abs/1603.06059.

Jack Mostow and Wei Chen. 2009. Generating instruction automatically for the reading strategy of self-questioning. In *Proceedings of the 2009 Conference on Artificial Intelligence in Education: Building Learning Systems That Care: From Knowledge Representation to Affective Modelling*. IOS Press, Amsterdam, The Netherlands, The Netherlands, pages 465–472. http://dl.acm.org/citation.cfm?id=1659450.1659520.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. *CoRR* abs/1611.09268. http://arxiv.org/abs/1611.09268.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, pages 2383–2392. https://aclweb.org/anthology/D16-1264.

Vasile Rus and C Graesser Arthur. 2009. The question generation shared task and evaluation challenge. In *The University of Memphis. National Science Foundation*. Citeseer.

Vasile Rus, Brendan Wyse, Paul Piwek, Mihai Lintean, Svetlana Stoyanchev, and Cristian Moldovan. 2010. The first question generation shared task evaluation challenge. In *Proceedings of the 6th International Natural Language Generation Conference*. Association for Computational Linguistics, Stroudsburg, PA, USA, INLG '10, pages 251–257. http://dl.acm.org/citation.cfm?id=1873738.1873777.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. *CoRR* abs/1611.09830. http://arxiv.org/abs/1611.09830.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*. pages 2692–2700.

Xingdi Yuan, Tong Wang, Çaglar Gülçehre, Alessandro Sordoni, Philip Bachman, Sandeep Subramanian, Saizheng Zhang, and Adam Trischler. 2017. Machine comprehension by text-to-text neural question generation. *CoRR* abs/1705.02012. http://arxiv.org/abs/1705.02012.