An Exploration in Neural Question Generation

Xiaoyong Jin
Department of Computer Science
University of California, Santa Barbara
x_jin@cs.ucsb.edu

Abstract

We experiment with a generative model based on sequence-to-sequence method to handle automatic question generation task. This generative model approximate a distribution over a pre-defined vocabulary and context words conditioned on a document, which plays a role of context, and an answer as a text span in that document, which can answer the generated question. In addition, we explore some critic models to assist generator based on REINFORCE algorithm. We use some tricks that have been proved significantly effective to train our model on SQUAD dataset. Some preliminary results are shown and many alternatives can produce more impressive questions in future work.

1 Introduction

Automatic question generation seeks to generate questions in natural languages that can be answered by specific answers based on given corpus illustrating certain facts. The corpus can either come from structured knowledge graph like database or be unstructured text.

Questions generation is considered as a critical problem in intelligent language system. There are many motivations people pay attention to question generation task. First, in machine reading task, posing question on certain information in the context is a good indicator showing the system fully understand the reading materials. While answering given questions in most factoid QA problems is an extractive task - it attempts to select a certain text span within the documents that the human-generated questions seek, asking questions based on documents with respect to given answers is an abstractive task - it summarizes the related content and the question words may not exist in the original corpus. Second, there is usually a unique answer being able to answer a given question according to a specific document. However, there might be a couple of questions that refer to distinct pieces of information but all can be answered by the given answer. For example, in figure1, given the article about super bowl and an answer "Denvor Broncos", human annotators make a question "Which NFL team represented the AFC at Super Bowl 50?", but there are many questions related to the article and answered. Some of them are logically equivalent to the ground-truth question, such as "Which tea was the champion of AFC?", while others shares nothing but the answer with ground-truth, such as "Which team defeated Carolina Panthers?". Third, QG is almost a revered task of popular text-based question answering problem. By generating more questions in combination with answers, much more labelled data would be of great help to neural-network based QA models. Finally, a mechanism to ask informative questions about documents has many potential applications, e.g. synthesizing FAQ documents and intelligent tutoring systems.

Previous works for question generation focus on rigid heuristic rules to transform a statement into a related question. However, these methods heavily rely on human-designed rules that cannot be adapted to different domains. Instead of spending much time learning linguistic rules, some works has applied deep neural networks to learn the features and generate questions as word sequences in an end-to-end manner.

Spring 2017 Deep Learning for Natural Language Processing (CS 292F), UCSB, CA, USA.

In this work, we extend a method introduced by (6) to question generation task, using a hierarchical encoder to collect and integrate information in document and answer. A decoder based on beam search leverages the output of the encoder and generate a question based on beam search. A critic tries to evaluate the quality of each generation step and feed a reward value back to guide the generator.

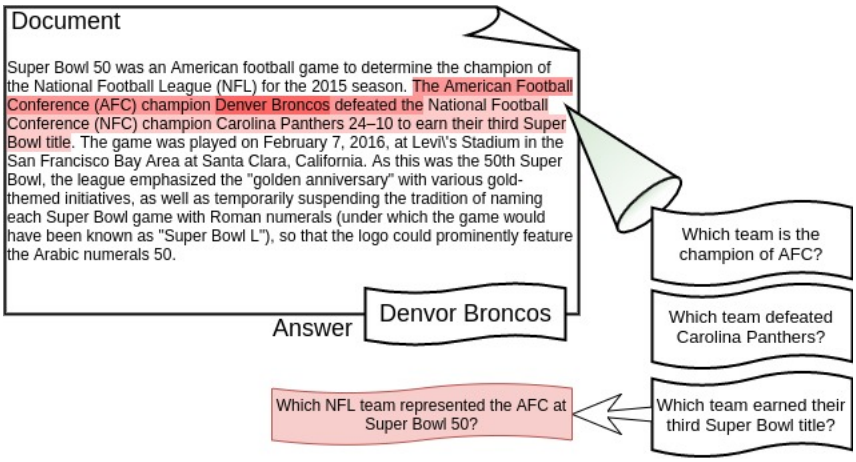

Figure 1: An example of question generation

2 Related work

Recently, automatic question generation has received increasing attention from computational linguistic and machine learning research community. Several earlier works process documents as individual sentences using syntactic [(5), (1)] or semantic parsing [(8), (9)], then reformulate questions using hand-crafted rules on parse tree. These conventional approaches generate questions with high word co-occurrences by rearranging parse trees.

As neural nets become more and more popular in many natural language tasks, many end-to-end models have been proposed.(12) introduces a neural system to convert knowledge base triples to factoid questions in natural language, where the subject and the relation from a context make up a question and the object serves as an answer. (15) and (4) use sequence-to-sequence-based models with attention mechanisms to generate questions based on given context. Both use Stanford Question Answering Dataset(11), while the former model uses word-level encoder and the latter one leverages the sentence-level information. In contrast, (3) propose to answer questions based on wikipedia documents by language model with attention mechanism. We can reverse the process by specifying a certain context and generate questions using similar method in this paper.

The concept of critic comes from adversarial training and its application to NLP. Generative Adversarial Nets (GANs) jointly train a generator and a discriminator to force the generator to learn the distribution of given dataset. The discriminator tries to tell generated samples from real samples in dataset. (14) and (7) extend GANs to discrete outputs i.e. linguistic tokens by rewards produced by the discriminator or so-called critic, so that the feedback from the critic can be back propagated to the generator.

3 Proposed model

3.1 Task Definition

Suppose we are given a single-paragraph document $D = (s_1, s_2, \cdots, s_N)$, where s_i is a sentence in that document paragraph, and each s_i consists of some words, $s_i = (x_1^i, x_2^i, \cdots, x_{N_i}^i)$. And a specific word or phrase in the document is highlighted as the wanted answer, say $A = (a_1, a_2, \cdots, a_m)$, a_j is one of the tokens in the answer. Note that $\{a_j\}_{j=1}^m$ is a text span in $D = \{x_j^i\}_{i=1,\cdots,N, j=1,\cdots,N_i}$. We attempt to generate a sequence Q, so that

$$Q = \arg\max_y P(y|D, A)$$

where y stands for all valid questions that can be answered by A based on context of D.

3.2 Encoder

Intuitively, since A is a small text span in D, there are only a few words around A that matter in the QA process, and only the sentence where A locates and those around it are important to build correspondence between Q and A. So instead of building a model on every word in the document, we pay much more attention to sentence-level encoding. Hence we use a hierarchical encoder on the document in order to extract semantic information in different levels.

In the first stage of encoding, we use a Bidirectional LSTM network on every single sentence in the document. For example, for sentence $s = (x_1, \cdots, x_N)$, we embed each token into continuous space by GloVe vectors. (10) In addition, we augment each word embedding with a binary feature indicating if the word belongs to the answer. We use the augmented embedding as the input to the BiLSTM, producing hidden vectors $\overleftarrow{h^w} = [\overleftarrow{h_1^w}, \overleftarrow{h_2^w}, \cdots, \overleftarrow{h_N^w}]$ and $\overrightarrow{h^w} = [\overrightarrow{h_1^w}, \overrightarrow{h_2^w}, \cdots, \overrightarrow{h_N^w}]$. We concatenate the each hidden state in both direction as the word-level annotation vector $h_i^w = [\overleftarrow{h_i^w}; \overrightarrow{h_i^w}]$. Meanwhile, a representation of current sentence is produced by concatenating the last hidden state in each direction, i.e. $h^s = [\overleftarrow{h_1^w}; \overrightarrow{h_N^w}]$ Similarly, we use another BiLSTM to encode all the sentences in the paragraph on all sentence representations, producing $\overleftarrow{h^d} = [\overleftarrow{h_1^s}, \overleftarrow{h_2^s}, \cdots, \overleftarrow{h_N^s}]$ and $\overrightarrow{h^d} = [\overrightarrow{h_1^s}, \overrightarrow{h_2^s}, \cdots, \overrightarrow{h_N^s}]$. The concatenation of sentence-level hidden states are used as sentence-level annotation vectors, i.e. $h_i^d = [\overleftarrow{h_i^d}; \overrightarrow{h_i^d}]$.

Next, we need to explicitly give the answer encoding, pointing out the part the generator needs to focus on. Since answer A consists of some words or phrases in the paragraph, we use the word-level annotation vectors and word embeddings as input to the third BiLSTM. To be specific, we make a new sequence with all word embeddings in combination with corresponding annotation vectors, i.e. $[h_i^w; e_i]$, then apply a new BiLSTM over the sequence and form answer representaion h^A in a way similar to sentence representation.

In addition to annotation vectors, we also compute an initial state s_0 for the decoder with the representation of answer and document. Like h^s and h^A, we concatenate the last sentence-level hidden states in both directions to obtain a representation of the whole document h^D. Then we re-weight the answer and document by a linear layer

$$s_0 = tanh(W_0^D h^D + W_0^A h^A + b_0)$$

where W_0^D, W_0^A, b_0 are parameters.

3.3 Decoder

Our decoder is a neural model that generates outputs y_t sequentially. At each time step t, the decoder models a conditional distribution parametrized by θ,

$$p(y_t | y_{<t}, D, A; \theta)$$

where y_t is a discrete sample from a vocabulary, and $y_{<t}$ represents the previous utterances.

In more detail, the decoder is a recurrent neural network. Its internal states are updated by an input y and a context vector c, i.e.

$$s_t = LSTM(s_{t-1}, y_{t-1}, c_t)$$

At every time step, the model computes a soft-alignment score over the document sentences to decide which sentences are more relevant to the question being generated. We use additive attention mechanism (2) to compute the score. A slight modification includes the answer representation into the memory and context, so that the score is derived by

$$e_{tj} = v^T tanh(W[h_j^s; h^A] + U s_t)$$

$$\alpha_{tj} = \frac{\exp(e_{tj})}{\sum_{j=1}^{N} \exp(e_{tj})}$$

and the context vector

$$c_t = [\sum_{j=1}^{N} \alpha_{tj} h_j^s; h^A]$$

In this way, at each step we reiterate the answer to force the semantic meaning of generated tokens staying close to answer.

The difference between question generation in our setting and neural machine translation to which sequence-to-sequence models are first applied is that the generated question needs to copy some relevant words or phrases from the original document to better describe the wanted answer. Therefore, in addition to an output vocabulary, which comes from words in human-generated questions, we need to sample target words from word lists in the corresponding documents. Therefore, we apply two distinct output layers to derive two distributions from each hidden state of the decoder. One is a softmax layer over a predefined vocabulary, which constitutes the question words in the training dataset. It produces a log-likelihood vector o_t. The other is the attention score α_t.

Furthermore, a scalar weight variable z_t is learned to enable the model to interpolate between document copying and generation from question vocabulary. It's computed by an one-layer MLP with $sigmoid$ activation. Thus we eventually get a fully defined distribution

$$p_t = [z_t o_t; (1 - z_t)\alpha_t]$$

to approximate the conditional distribution $p(y_t|y_{<t}, D, A)$

3.4 Critic

As described above, we train the encoder-decoder model by maximizing the likelihood of ground-truth. However, we want to generate not only questions similar to ground-truth, but brand new questions that current document and answer are able to handle. So we need a critic model to encourage those utterances that are different from ground-truth but promising to form a good question.

There are many possible choices for the critic model

1. A question answering model $F_{QA}(\hat{q})$. It receives a generated question $\hat{q}$ as input and give an answer $\hat{a}$ based on the document. The real answer a is then compared with $\hat{a}$. If they are similar, it means $\hat{q}$ is a good question
2. We can compare the attention focus of two questions on the document. Theoretically questions share the same answer should pay attention to the similar part in the context
3. Paraphrasing question detector. This model is inspired by Quora duplicate question detection challenge. If the model judges the generated question is a paraphrase of ground-truth, then it's also a good question

In this project, it's too expensive to jointly train the critic with the generator. As a preliminary study, we use a CNN-based text classifier to tell whether the input question comes from the real dataset, just like a discriminator in GANs. In this case, at every step, we maximize a reformulated likelihood

$$l(\theta) = p(y_t|t_{<t}, D, A; \theta) V_{G_\theta}^{D_\phi}(y_t|y_{<t}, D, A)$$

where D is the critic, θ and ϕ are parameters of generator and critic, respectively, and V is the reward estimation

$$V_{G_\theta}^{D_\phi}(y_t|y_{<t}, D, A) = \mathbb{E}_y \left[p(y|y_{\leq t}, D, A; \theta) D(y|D, A, \phi)\right]$$

As explained in (14), the discriminator can only evaluate a finished question, but we need rewards for every step during generation. So at each intermediate time step, we use Monte Carlo search to fast complete the sentence according to current policy (parameter setting) by simple greedy search. Practically we cannot fully evaluate the expectation of reward, so we only sample a number of finished questions and take their average as an approximation of value function. As likelihood, we scale the reward value into $[0, 1]$, so that finished questions are always better than meaningless extensions, which is important in decoding process.

During the rewarded training, sometimes the generator might go beyond the expected track and get low reward. It cannot escape from this distraction without ground-truth. So we adopt teacher forcing to encourage generator to stay close to real data by giving maximum reward to ground-truth.

4 Training and Inference

At first, we train the model by purely maximizing ground-truth's likelihood. We train the generator with scheduled sampling, that is, at a number of steps, we use the previous output token as current input rather than the word in the ground-truth. This method can mitigate the discrepancy between training phase and inference phase and the resulting failure. After generator is trained, it generates some sample questions. The critic is then trained to classify generated samples and those from training set.

In addition, we add a regularization item $L_p = p_t \log p_t$ to avoid p_t's degeneration to a single frequently seen word. And we avoid any answer word in question during decoding.

It's hard to determine when to stop pre-training critic. A perfect critic will always give negative comment on generated questions at the beginning of joint training, so we only train the critic for one epoch while train the generator for two epochs.

After pre-training both module, we follow generator's policy to obtain a sequence, apply critic and MC search to evaluate each step and update generator according to the reformulated likelihood. Then critic is trained one more time to keep sharp.

During both joint training and final test, we perform inference by beam search. The generator stops once an <EOS> token ranks first among all candidates.

Since a paragraph is quite long, we set a relatively small batch size 16, and a beam size of 32. The regularization weight is set to 1.0. For word embedding, we use pre-trained GloVe vectors keep them fixed for all found tokens, and initialize random vectors and train them for out-of-vocabulary words.

5 Experiments

5.1 Dataset

We do experiments on Stanford Question Answering Dataset. We first preprocess the noisy dataset by

1. Tokenize paragraphs into sentences and words
2. Check spelling to reduce OOV words
3. Choose only one answer for a question since there are multiple plausible answers in the original development set.

5.2 Baseline

We use a vanilla seq2seq model as our baseline. This model use only one BiLSTM to encode word-level information in document and encode answer by averaging the answer words' hidden vectors. It also uses greedy decoding instead of beam search.

5.3 Results

Here is one great example the generator produces We can see that the model can replace the subject

Table 1: A good generated example

Paragraph	In May 2005, Carrie Underwood was announced the winner, with Bice the runner-up. Both Underwood and Bice released the coronation song "Inside Your Heaven". Underwood has since sold 65 million records worldwide, and become the most successful Idol contestant in the U.S., selling over 14 million albums copies in the U.S...
Answer	Inside Your Heaven
Human-generated	What song did two finalists released?
Machine-generated	What song did Carrie Underwood released?

in the question. According to the article, "Carrie Underwood" is one of "the finalists", so this two questions should share the same answer, but the generated one is different from the real one. This

indicates our model pay attention to the actual meaning of "finalists" and find "Carrie Underwood" in the first sentence as an example.

There are many other good examples, shown in table2

Table 2: Some good generated examples

Human generated	Machine Generated
What two empires did armenia belong to in the beginning of the 5th century?	What is the name of empire armenia belong to?
In what buddhism is the goal a state of nirvana?	What is the ultimate goal of nirvana?
How long is the growing season in the seattle area?	How long did the growing season last in seattle?
At what street address is the lord mayor 's residence located?	What is the lord mayor 's official residence?
When was the soundtrack for series 5 released?	When did the soundtrack special soundtrack released?

There are also some bad examples that suffer from various malfunction shown in table 3. We found there are mainly 5 kinds of bad questions.

1. Stop prematurely. An <EOS> token occurs too early before informative words. A possible explanation is that the last words in such cases are frequently seen in the end of normal questions, which confuses the model
2. Repeated words. As we mentioned, the output softmax degenerates to a singular point. The regularization cannot completely avoid such cases
3. Oversimplified. The question might make sense but a lot of information is lost, and the answer is likely to change
4. Mismatch. In some rare cases, a wrong-picked keyword similar to the correct one might significantly change the meaning of whole question and cause mismatch between question and answer
5. Nonsense. In these cases, an irrelevant word is chosen and the sentence becomes completely nonsense

Table 3: Some bad examples

Stop Prematurely	During what span were the marshall islands first settled? When did the marshall islands start?
Repeat words	Who started the great migration of the slavs? What country did the slavs of germans believe to be be be be be be be be be be ?
Oversimplified	Where do the feynman lectures on physics pull information from? What is the feynman lectures?
Mismatch	When was the second satellite for the beidou-1 system launched? When did the first satellite of beidou-1b open?
Nonsense	How many animals of the poultry variety are raised for consumtion each year? How many birds are raised to pork year?

5.4 Quantitative Result

We use BLEU as our automatic evaluation metric. BLEU compares the co-occurrence of n-grams in two sentences to measure the similarity. We show the results of our baseline, a generator without critic and a full model in figure2

We can find our model performs better than baseline for sure. But since the critic we use is quite simple and not intuitively effective in classification of long range semantic information, it doesn't improve a lot. We guess the reason why critic shows worse performance on BLEU-3 and BLUE-4 score is that the critic gives rewards to the next utterance at each step. Due to the nature of recurrent model, it gives priority to generating a token that is consistent with the current token, and the

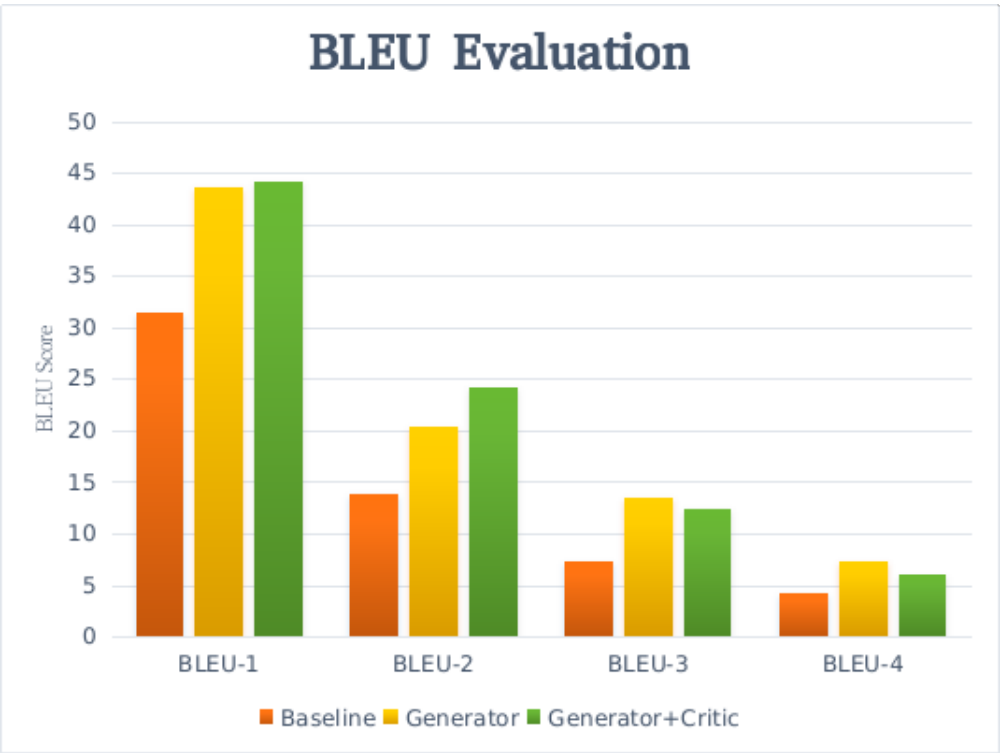

Figure 2: Quantitative results

consistence with previous tokens are partially ignored because of memory loss. Thus bigrams are much more likely to have high quality than longer units.

Compared with other recent works like (4), our result is far behind the state of the art according to BLEU metric. However, BLEU cannot perfectly measure the quality of generated natural language and it's still a very challenging topic to find good automatic evaluation methods in natural language research.

6 Conclusion and Future Work

In this report we introduce our exploration in question generation in natural language based on specified document and answer. We use a sequence-to-sequence based model and make some modification to adapt to this specific scenario. We adopt the concept of critic from adversarial training and try to leverage external reward to boost the performance of generator.

In the future, we can improve our model in many aspects, for instance,

1. Try more sophisticated critic model to produce higher quality reward signals
2. Explore better mechanism of reward propagation to avoid expensive Monte Carlo search. A promising alternative has been introduced by (7)
3. Integrate QA task into QG system might be a new direction in machine reading and comprehension. Some works (13) published this week might be a good exploration.(13)

References

[1] ALI, H., CHALI, Y., AND HASAN, S. A. Automation of question generation from sentences. In *Proceedings of QG2010: The Third Workshop on Question Generation* (2010), pp. 58–67.

[2] BAHDANAU, D., CHO, K., AND BENGIO, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[3] CHEN, D., FISCH, A., WESTON, J., AND BORDES, A. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051* (2017).

[4] DU, X., SHAO, J., AND CARDIE, C. Learning to ask: Neural question generation for reading comprehension. *arXiv preprint arXiv:1705.00106* (2017).

[5] HEILMAN, M., AND SMITH, N. A. 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* (2010), Association for Computational Linguistics, pp. 609–617.

[6] LI, J., LUONG, M.-T., AND JURAFSKY, D. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057* (2015).

[7] LI, J., MONROE, W., SHI, T., RITTER, A., AND JURAFSKY, D. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547* (2017).

[8] LINDBERG, D., POPOWICH, F., NESBIT, J., AND WINNE, P. Generating natural language questions to support learning on-line.

[9] MANNEM, P., PRASAD, R., AND JOSHI, A. Question generation from paragraphs at upenn: Qgstec system description. In *Proceedings of QG2010: The Third Workshop on Question Generation* (2010), pp. 84–91.

[10] PENNINGTON, J., SOCHER, R., AND MANNING, C. D. Glove: Global vectors for word representation. In *EMNLP* (2014), vol. 14, pp. 1532–1543.

[11] RAJPURKAR, P., ZHANG, J., LOPYREV, K., AND LIANG, P. Squad: 100, 000+ questions for machine comprehension of text. *CoRR abs/1606.05250* (2016).

[12] SERBAN, I. V., GARCÍA-DURÁN, A., GÜLÇEHRE, Ç., AHN, S., CHANDAR, S., COURVILLE, A. C., AND BENGIO, Y. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. *CoRR abs/1603.06807* (2016).

[13] TANG, D., DUAN, N., QIN, T., AND ZHOU, M. Question answering and question generation as dual tasks. *arXiv preprint arXiv:1706.02027* (2017).

[14] YU, L., ZHANG, W., WANG, J., AND YU, Y. Seqgan: sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence* (2017).

[15] ZHOU, Q., YANG, N., WEI, F., TAN, C., BAO, H., AND ZHOU, M. Neural question generation from text: A preliminary study. *arXiv preprint arXiv:1704.01792* (2017).