Question Generation for Question Answering

Nan Duan[†], Duyu Tang[†], Peng Chen[§], Ming Zhou[†]
Microsoft Research Asia[†]
Microsoft Research and AI Group[§]
{nanduan,dutang,peche,mingzhou}@microsoft.com

Abstract

This paper presents how to generate questions from given passages using neural networks, where large scale QA pairs are automatically crawled and processed from Community-QA website, and used as training data. The contribution of the paper is 2-fold: First, two types of question generation approaches are proposed, one is a retrieval-based method using convolution neural network (CNN), the other is a generation-based method using recurrent neural network (RNN); Second, we show how to leverage the generated questions to improve existing question answering systems. We evaluate our question generation method for the answer sentence selection task on three benchmark datasets, including SQuAD, MS MARCO, and WikiQA. Experimental results show that, by using generated questions as an extra signal, significant QA improvement can be achieved.

1 Introduction

Question Answering (or QA) is one of the core problems for AI, and consists of several typical tasks, i.e. community-based QA (Qiu and Huang, 2015), knowledge-based QA (Berant et al., 2013), text-based QA (Yu et al., 2014), and reading comprehension (Rajpurkar et al., 2016). Most of current QA systems, e.g. (Berant and Liang, 2014), (Qiu and Huang, 2015), (Xiong et al., 2017), (Yin and Schtze, 2017), need labeled QA pairs as training data. Although labeling efforts have been made, such as WebQuestions dataset (Berant et al., 2013) and SimpleQuestions dataset (Bordes et al., 2015) for knowledge-based QA, WikiQA dataset (Yang et al., 2015) for text-based QA, SQuAD dataset (Rajpurkar et al., 2016) and MS MARCO dataset (Nguyen et al., 2016) for reading comprehension, these datasets are still with limited sizes, as labeling is very expensive.

Motivated by this, we explore how to generate questions from given passages using neural networks, with three expected goals: (1) the training data should need few or no human efforts and reflect commonly-asked question intentions; (2) the questions are generated based on natural language passages, and should have good quality; (3) the generated questions should be helpful to QA tasks.

To achieve the 1^{st} goal, we propose to acquire large scale high-quality training data from Community-QA (CQA) website. The motivation of using CQA website for training data collection is that, such websites (e.g., YahooAnswers, Quora, etc.) contain large scale QA pairs generated by real users, and these questions reflect the most common user intentions, and therefore are useful to search, QA, and chatbot scenarios.

To achieve the 2^{nd} goal, we explore two ways to generate questions for a given passage, one is a retrieval-based method using convolution neural network (CNN), the other is a generation-based method using recurrent neural network (RNN). We evaluate the generation quality by BLEU score (Papineni et al., 2002) and human annotations, and discuss their pros and cons in Section 9.

To achieve the 3^{rd} goal, we integrate our question generation approach into an end-to-end QA task, i.e., answer sentence selection, and evaluate its impact on three popular benchmark datasets, SQuAD, MS MARCO, and WikiQA. Experimental results show that, the generated questions can improve the QA quality on all these three datasets.

2 Question Generation

Formally, given a passage S, question generation (**QG**) engine generates a set of questions $\{Q_i\}$, where each generated $\mathcal{Q}_i$ can be answered by $\mathcal{S}$. There are four components in our QG engine:

1. *Question Pattern Mining*, which extracts the frequently-asked question patterns from large scale CQA questions, without any human annotation effort;

2. *Question Pattern Prediction*, which predicts top-N question patterns $\mathcal{Q}_p^1, ..., \mathcal{Q}_p^N$ given $\mathcal{S}$, by a retrieval-based method or a generation-based method. Therefore, "*Prediction*" has two different meanings here: in retrieval-based method, it means to rank existing question patterns and select the highest ranked ones, while in generation-based method, it means to generate question patterns based on $\mathcal{S}$ in a sequence-to-sequence manner, each of which could be a totally new question pattern beyond the existing question pattern set;

3. *Question Topic Selection*, which selects a phrase $\mathcal{Q}_t$ from $\mathcal{S}$ as the question topic, based on a predicted question pattern $\mathcal{Q}_p$. $\mathcal{Q}_t$ will be filled into $\mathcal{Q}_p$ to form a complete question;

4. *Question Ranking*, which ranks all generated questions by a set of features. Here, multiple questions with different intentions could be generated, as $\mathcal{S}$ could contain multiple facts.

3 Question Pattern Mining

A question pattern (or **QP**) is defined as a word sequence $\mathcal{Q}_p = \{w_1, w_2, ..., w_L\}$. Each QP should contain one and only one special word "#" as the placeholder, and all the other $L-1$ words are terminal words. For example, "*who founded # ?*" is a question pattern, and "#" denotes the placeholder, which could be an organization name. We generate questions only using *frequently-asked* question patterns, where "frequently-asked" denotes that each question pattern should be extracted from a large scale question set more than T times, where T denotes a pre-defined threshold.

In this paper, a question cluster (or **QC**) based approach is proposed to mine frequently-asked question patterns from large scale CQA questions.

First, a set of question clusters is collected from CQA webpages, and each question cluster consists of questions that are grouped as related questions[1] by the CQA website. For example, when the query "*what is the population of nyc*" is issued to YahooAnswers[2], the returned page contains a list of related questions including "*population of nyc*", "*nyc population*", "*nyc census*", and etc.

Second, for each question cluster $\mathcal{QC} = \{\mathcal{Q}_1, ..., \mathcal{Q}_K\}$, we enumerate all valid continuous n-grams, each of which should contain at least one content word and its order should be equal or less then 7, as question topic candidates $\{\mathcal{Q}_t^1, ..., \mathcal{Q}_t^M\}$. We then assign an importance score $Impts(\cdot)$ to each question topic candidate $\mathcal{Q}_t^m$:

$$Impts(\mathcal{Q}_t^m) = \sum_{\mathcal{Q}_k \in \mathcal{QC}} \delta(\mathcal{Q}_t^m, \mathcal{Q}_k) \cdot |\mathcal{Q}_t^m|$$

$\mathcal{Q}_t^m$ denotes the m^{th} question topic candidate, $\delta(\mathcal{Q}_t^m, \mathcal{Q}_k)$ equals to 1 when $\mathcal{Q}_t^m$ occurs in $\mathcal{Q}_k$, and 0 otherwise, $|\mathcal{Q}_t^m|$ denotes $\mathcal{Q}_t^m$'s word count, which boosts longer question topic candidates.

For each $\mathcal{QC}$, we select $\mathcal{Q}_t$ with the highest importance score as the question topic, and remove it from each question to form a question pattern. We call each removed question topic as a **historical question topic** of its corresponding question pattern. If $\mathcal{Q}_t$ doesn't exist in a question, ignore it.

Mining question patterns based on question clusters is motivated by the observation that, all questions within a QC tend to ask questions about an identical "*question topic*" (e.g., *nyc* in the above example) from different aspects, or the same aspect but using different expressions. Thus, we can leverage the consensus information among questions to detect the boundary of the question topic: *the more times an n-gram occurs in different questions within a question cluster, the more likely it is the question topic of the current question cluster*.

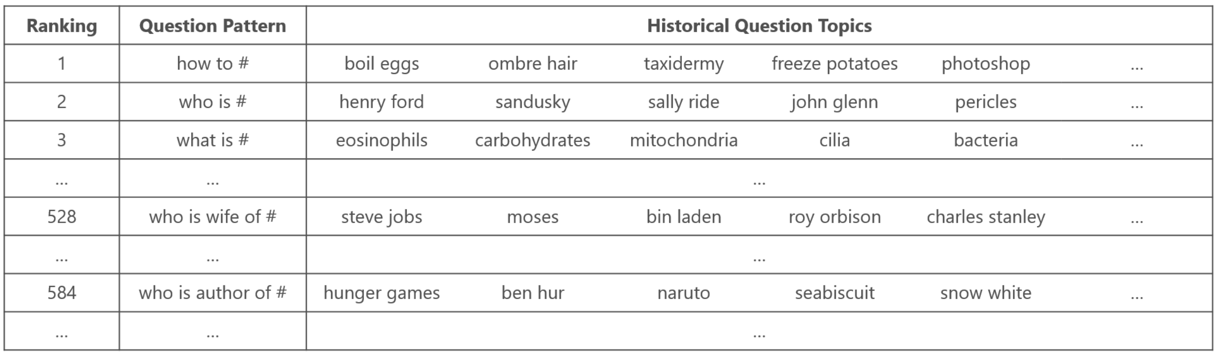
Although the question pattern mining approach described above is simple, it works surprisingly well. Table 1 shows statistics of question patterns mined from YahooAnswers, and Figure 1 gives examples of frequently-asked question patterns with their corresponding historical question topics. We have two interesting observations:

1. Most frequently-asked question patterns (frequency $>=$10,000) are with high quality, and reflect the most common user intentions;

2. Most historical question topics extracted are entities. This is achieved without using any prior semantic knowledge base or dictionary.

[1]Usually, each question page of a CQA website contains a field that shows a list of related questions.

[2]https://answers.yahoo.com/



Ranking	Question Pattern	Historical Question Topics					
1	how to #	boil eggs	ombre hair	taxidermy	freeze potatoes	photoshop	...
2	who is #	henry ford	sandusky	sally ride	john glenn	pericles	...
3	what is #	eosinophils	carbohydrates	mitochondria	cilia	bacteria	...
...	...	...					
528	who is wife of #	steve jobs	moses	bin laden	roy orbison	charles stanley	...
...	...	...					
584	who is author of #	hunger games	ben hur	naruto	seabiscuit	snow white	...
...	...	...					

Figure 1: Examples of frequently-asked question patterns with corresponding historical question topics.

# of QCs	67,330,506
# of QPs	20,818,569
# of QPs (freq. >= 10,000)	105,623

Table 1: Statistics of question patterns mined from YahooAnswers, where QC denotes Question Cluster and QP denotes Question Pattern.

4 Question Pattern Prediction

Given a passage $\mathcal{S}$, question pattern prediction predicts $\mathcal{S}$'s most related question patterns, and then use them to generate questions. For example, given $\mathcal{S}$ as "*Tesla Motors is an American automaker and energy storage company co-founded by Elon Musk, Martin Eberhard, Marc Tarpenning, JB Straubel and Ian Wright, and is based in Palo Alto.*", two question patterns can be derived from $\mathcal{S}$, including: (1) *who founded # ?*, which can be inferred by the context around "*co-founded by*", and (2) *where is # located ?*, which can be inferred by the context around "*is based in*". Based on these two question patterns, we can generate two questions, "*who founded Tesla Motors ?*" and "*where is Tesla Motors located ?*" respectively.

4.1 Training Data Construction

We collect QA pairs from YahooAnswers. For each QA pair $< \mathcal{Q}, A >$, if (1) $\mathcal{Q}$ can be matched by a frequently-asked question pattern $\mathcal{Q}_p$, and (2) the matched question topic $\mathcal{Q}_t$ of $\mathcal{Q}$ based on $\mathcal{Q}_p$ exists in A, then we create a training instance as $< A, \mathcal{Q}_p, \mathcal{Q}_t >$. $\mathcal{Q}_t \in A$ makes sure that the question topic occurs in both $\mathcal{Q}$ and A. If the matched question topic only exists in $\mathcal{Q}$, we just discard it. By doing so, we collect a total of 1,984,401 training instances as training data for QP prediction.

Two neural network-based question pattern prediction approaches are explored in this paper:

- *Retrieval-based QP Prediction*, which considers QP prediction as a ranking task;
- *Generation-based QP Prediction*, which considers QP prediction as a generation task.

4.2 Retrieval-based QP Prediction

The retrieval-based QP prediction is done based on an attention-based convolution neural network. It takes a passage and a question pattern as input, and outputs their corresponding vector representations. We denote each input pair as $\langle \mathcal{S}, \mathcal{Q}_p \rangle$, where $\mathcal{S}$ is a passage, and $\mathcal{Q}_p$ is a question pattern.

In the input layer, given an input pair $\langle \mathcal{S}, \mathcal{Q}_p \rangle$, an attention matrix $\mathcal{A}tt \in \Re^{|\mathcal{S}| \times |\mathcal{Q}_p|}$ is generated by pre-trained word embeddings of $\mathcal{S}$ and $\mathcal{Q}_p$, where each element $\mathcal{A}tt_{i,j} \in \mathcal{A}$tt is computed as:

$$\mathcal{A}tt_{i,j} = cosine(v_i^{\mathcal{S}}, v_j^{\mathcal{Q}_p})$$

where $v_i^{\mathcal{S}}$ (or $v_j^{\mathcal{Q}_p}$) denotes the embedding vector of the ith (or jth) word in $\mathcal{S}$ (or $\mathcal{Q}_p$).

Then, column-wise and row-wise max-pooling are applied to $\mathcal{A}tt$ to generate two attention vectors $V^{\mathcal{S}} \in \Re^{|\mathcal{S}|}$ and $V^{\mathcal{Q}_p} \in \Re^{|\mathcal{Q}_p|}$, where the kth elements of $V^{\mathcal{S}}$ and $V^{\mathcal{Q}_p}$ are computed as:

$$V_k^{\mathcal{S}} = \max_{1<l<|\mathcal{Q}_p|} \{\mathcal{A}tt_{k,l}\} \; and \; V_k^{\mathcal{Q}_p} = \max_{1<l<|\mathcal{S}|} \{\mathcal{A}tt_{l,k}\}$$

$V_k^{\mathcal{S}}$ (or $V_k^{\mathcal{Q}_p}$) can be interpreted as the *attention score* of the kth word in $\mathcal{S}$ (or $\mathcal{Q}_p$) with regard to all words in $\mathcal{Q}_p$ (or $\mathcal{S}$).

Next, two *attention distributions* $D^{\mathcal{S}} \in \Re^{|\mathcal{S}|}$ and $D^{\mathcal{Q}_p} \in \Re^{|\mathcal{Q}_p|}$ are generated for $\mathcal{S}$ and $\mathcal{Q}_p$ based on $V^{\mathcal{S}}$ and $V^{\mathcal{Q}_p}$ respectively, where the kth elements of $D^{\mathcal{S}}$ and $D^{\mathcal{Q}_p}$ are computed as:

$$D_k^{\mathcal{S}} = \frac{e^{V_k^{\mathcal{S}}}}{\sum_{l=1}^{|\mathcal{S}|} e^{V_l^{\mathcal{S}}}} \; and \; D_k^{\mathcal{S}y} = \frac{e^{V_k^{\mathcal{Q}_p}}}{\sum_{l=1}^{|\mathcal{Q}_p|} e^{V_l^{\mathcal{Q}_p}}}$$

$D_k^{\mathcal{S}x}$ (or $D_k^{\mathcal{S}y}$) can be interpreted as the *normalized attention score* of the k^{th} word in $\mathcal{S}$ (or $\mathcal{Q}_p$) with regard to all words in $\mathcal{Q}_p$ (or $\mathcal{S}$).

Last, we update each pre-trained word embedding $v_k^{\mathcal{S}}$ (or $v_k^{\mathcal{Q}_p}$) to $\hat{v}_k^{\mathcal{S}}$ (or $\hat{v}_k^{\mathcal{Q}_p}$), by multiplying every value in $v_k^{\mathcal{S}}$ (or $v_k^{\mathcal{Q}_p}$) with $D_k^{\mathcal{S}}$ (or $D_k^{\mathcal{Q}_p}$). The underlying intuition of updating pre-trained word embeddings is to *re-weight the importance of each word in $\mathcal{S}$ (or $\mathcal{Q}_p$) based on $\mathcal{Q}_p$ (or $\mathcal{S}$), instead of treating them in an equal manner*.

In the convolution layer, we first derive an input matrix $Z_{\mathcal{S}} = \{l_1, ..., l_{|\mathcal{S}|}\}$, where l_t is the concatenation of a sequence of $m = 2d - 1$[3] updated word embeddings $[\hat{v}_{t-d}^{\mathcal{S}}, ..., \hat{v}_t^{\mathcal{S}}, ..., \hat{v}_{t+d}^{\mathcal{S}}]$, centralized in the t^{th} word in $\mathcal{S}$. Then, the convolution layer performs sliding window-based feature extraction to project each vector representation $l_t \in Z_{\mathcal{S}}$ to a contextual feature vector $h_t^{\mathcal{S}}$:

$$h_t^{\mathcal{S}} = tanh(W_c \cdot l_t)$$

where W_c is the convolution matrix, $tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$ is the activation function. The same operation is performed to $\mathcal{Q}_p$ as well.

In the pooling layer, we aggregate local features extracted by the convolution layer from $\mathcal{S}$, and form a sentence-level global feature vector with a fixed size independent of the length of the input sentence. Here, *max-pooling* is used to force the network to retain the most useful local features by $l_p^{\mathcal{S}} = [v_1^{\mathcal{S}}, ..., v_K^{\mathcal{S}}]$, where:

$$v_i^{\mathcal{S}} = \max_{t=1,...,|\mathcal{S}|} \{h_t^{\mathcal{S}}(i)\}$$

$h_t^{\mathcal{S}}(i)$ denotes the i^{th} value in the vector $h_t^{\mathcal{S}}$. The same operation are performed to $\mathcal{Q}_p$ as well.

In the output layer, one more non-linear transformation is applied to $l_p^{\mathcal{S}}$:

$$y(\mathcal{S}) = tanh(W_s \cdot l_p^{\mathcal{S}})$$

W_s is the semantic projection matrix, $y(\mathcal{S})$ is the final sentence embedding of $\mathcal{S}$. The same operation is performed to $\mathcal{Q}_p$ to obtain $y(\mathcal{Q}_p)$.

We train model parameters W_c and W_s by minimizing the following ranking loss function:

$$\mathcal{L} = \max\{0, M - cosine(y(\mathcal{S}), y(\mathcal{Q}_p)) + cosine(y(\mathcal{S}), y(\mathcal{Q}_p^-))\}$$

where M is a constant, $\mathcal{Q}_p^-$ is a negative instance.

[3]In this paper, m is set to 3.

4.3 Generation-based QP Prediction

The generation-based QP prediction is done based on an sequence-to-sequence BiGRU (Bahdanau et al., 2015) that is commonly used in the neural machine translation field.

The encoder reads a word sequence of an input passage $\mathcal{S} = (x_1, ..., x_{|\mathcal{S}|})$, and the decoder predicts a word sequence of an output question pattern $\mathcal{Q}_p = (y_1, ..., y_{|\mathcal{Q}_t|})$. The probability of generating a question pattern $\mathcal{Q}_p$ is computed as:

$$p(\mathcal{Q}_p) = \prod_{i=1}^{|\mathcal{Q}_p|} p(y_i|y_1, ..., y_{i-1}, c_i)$$

where each conditional probability is defined as:

$$p(y_i|y_1, ..., y_{i-1}, c_i) = g(y_{i-1}, s_i, c_i)$$

$g(\cdot)$ denotes a nonlinear function that outputs the probability of generating y_i. s_i denotes the hidden state of time t in decoder, which is computed as:

$$s_i = (1 - z_i) \circ s_{i-1} + z_i \circ \tilde{s}_i$$

where

$$\tilde{s}_i = \tanh(WE_{y_{i-1}} + U[r_i \circ s_{i-1}] + Cc_i)$$

$$z_i = \delta(W_z E_{y_{i-1}} + U_z s_{i-1} + C_z c_i)$$

$$r_i = \delta(W_r E_{y_{i-1}} + U_r s_{i-1} + C_r c_i)$$

$\delta(\cdot)$ is the sigmoid function, $\circ$ represents element-wise multiplication, $E_w \in \Re^{m\times 1}$ denotes the word embedding of a word w, W, W_z, $W_r \in \Re^{n\times m}$, U, U_z, $U_r \in \Re^{n\times n}$, and C, C_z, $C_r \in \Re^{n\times 2n}$ are weights. c_i denotes the context vector, which is computed as:

$$c_i = \sum_{j=1}^{|\mathcal{S}|} \alpha_{ij} h_j$$

where

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{|\mathcal{S}|} exp(e_{ik})}$$

$$e_{ij} = v_a^T \tanh(W_a s_{i-1} + U_a h_j)$$

$v_a \in \Re^{n'}$, $W_a \in \Re^{n'\times n}$, and $U_a \in \Re^{n'\times 2n}$ are weights. h_j denotes the j^{th} hidden state from encoder, which is the concatenation of the forward hidden state $\overrightarrow{h}_j$ and the back forward state $\overleftarrow{h}_j$.

For training, stochastic gradient descent (SGD) algorithm is used, and Adadelta (Zeiler, 2012) is used to adapt the learning rate of each parameter. Given a batch of $D = \{< \mathcal{S}, \mathcal{Q}_p >\}_{i=1}^{M}$ pairs with size M instances, the objective function is to minimize the negative log-likelihood:

$$\mathcal{L} = -\frac{1}{M} \sum_{<\mathcal{S},\mathcal{Q}_p>} \sum_{t=1}^{T} log(p(y_t|y_{<t}, \mathcal{S}))$$

For prediction, beam search is used to output the top-N question pattern predictions.

Retrieval-based approach can only find existing question patterns for each passage, but it makes sure that each question pattern comes from real questions and is in a good grammatical form; Generation-based approach, on the other hand, can generate totally new question patterns beyond existing question pattern set. We will compare both of them in the experimental part (Section 8).

5 Question Topic Selection

Given a passage $\mathcal{S}$ and a predicted question pattern $\mathcal{Q}_p$, question topic selection selects an n-gram (or a phrase) $\mathcal{Q}_t$ from $\mathcal{S}$, which can be then filled into $\mathcal{Q}_p$ to form a complete question. Since we have two question pattern prediction methods, we have two ways to select the question topic $\mathcal{Q}_t$ as well.

For $\mathcal{Q}_p$ from retrieval-based method, two types of prior knowledge are used to extract question topic candidates from $\mathcal{S}$, including:

- Entities as question topic candidates, which are detected based on Freebase[4] entities;
- Noun phrases as question topic candidates, which are detected based on the Stanford parser (Klein and Manning, 2003).

Once a question topic candidate $\mathcal{Q}_t$ is extracted from $\mathcal{S}$, we then measure how $\mathcal{Q}_t$ can fit $\mathcal{Q}_p$ by:

$$s(\mathcal{Q}_t, \mathcal{Q}_p) = \frac{1}{N} \cdot \sum_k \#(\mathcal{Q}_p^{t_k}) \cdot dist(v_{\mathcal{Q}_t}, v_{\mathcal{Q}_p^{t_k}})$$

$s(\mathcal{Q}_t, \mathcal{Q}_p)$ denotes the confidence that $\mathcal{Q}_t$ can be filled into $\mathcal{Q}_p$ to generate a reasonable question. $\mathcal{Q}_p^{t_k}$ denotes the kth historical question topic of $\mathcal{Q}_p$. $\#(\mathcal{Q}_p^{t_k})$ denotes the number of times that $\mathcal{Q}_p^{t_k}$ is extracted from different question clusters to generate $\mathcal{Q}_p$. v_p denotes the question topic embedding of p, which is computed as the average of word embeddings in p. $dist(\cdot)$ denotes the cosine distance between two question topic embeddings. $N = \sum_k \#(\mathcal{Q}_p^{t_k})$ denotes the total number of historical question topics of $\mathcal{Q}_p$. The basic principle of the above equation is that, *the historical question topics of a given question pattern can help on measuring how possible a question topic candidate can be filled into this question pattern to form a reasonable question*. For example, as most historical question topics of "*who founded # ?*" are organization names, then it is very unlikely a date or a film name is suitable for this question pattern to generate a reasonable question.

For $\mathcal{Q}_p$ from generation-based method, suppose the placeholder # is the i^{th} word in $\mathcal{Q}_p$, then we select the j^{th} word $w_j \in \mathcal{S}$ as the question topic, which satisfies the following constraint:

$$w_j = \arg\max_{w_{j'} \in \mathcal{S}} \alpha_{ij'} = \arg\max_{w_{j'} \in \mathcal{S}} \frac{exp(e_{ij'})}{\sum_{k=1}^{|\mathcal{S}|} exp(e_{ik})}$$

This question topic selection strategy leverages the attention scores between $\mathcal{S}$ and $\mathcal{Q}_p$, and can be considered as a *COPY* mechanism.

6 Question Ranking

Given a predicted question pattern $\mathcal{Q}_p$ and a selected question topic $\mathcal{Q}_t$ of an input passage $\mathcal{S}$, a complete question $\mathcal{Q}$ can be simply generated by replacing $\#$ in $\mathcal{Q}_p$ with $\mathcal{Q}_t$. We use a set of features to rank generated question candidates:

- *question pattern prediction score*, which is the prediction score by either retrieval-based approach or generation-based approach;
- *question topic selection score*, for retrieval-based approach, this score is computed as $s(\mathcal{Q}_t, \mathcal{Q}_p)$, while for generation-based approach, this score is the attention score;
- *QA matching score*, which measures relevance between generated question $\mathcal{Q}$ and $\mathcal{S}$.
- *word overlap between $\mathcal{Q}$ and $\mathcal{S}$*, which counts number of words that co-occur in $\mathcal{Q}$ and $\mathcal{S}$;
- *question pattern frequency*, which equals to the extraction frequency of $\mathcal{Q}_p$, if $\mathcal{Q}$ is generated from or matched by $\mathcal{Q}_p$, and 0 otherwise.

All features are combined by a linear model as:

$$p(\mathcal{Q}|\mathcal{S}) = \sum_i \lambda_i \cdot h_i(\mathcal{Q}, \mathcal{S}, \mathcal{Q}_p, \mathcal{Q}_t)$$

[4]https://developers.google.com/freebase/

where $h_i(\mathcal{Q}, \mathcal{S}, \mathcal{Q}_p, \mathcal{Q}_t)$ is one of the features described above, and λ_i is the corresponding weight.

7 Question Generation for QA

This section describes how question generation can improve existing QA systems. There are several types of QA systems, i.e. knowledge-based QA, community-based QA, text-based QA, etc, and in this paper, we focus on text-based QA task (a.k.a. answer sentence selection), which aims to select one or multiple answer sentences from a text given an input question. We select this task as it can be considered as a dual task of QG.

A typical answer sentence selection method, such as (Yin et al., 2016; Santos et al., 2016; Miller et al., 2016; Tymoshenko et al., 2016), computes the relevance score between input question $\mathcal{Q}$ and each answer candidate $\mathcal{A}$, and selects the one with the highest relevance score as the final answer:

$$\widehat{\mathcal{A}} = \arg\max_{\mathcal{A}} P(\mathcal{A}|\mathcal{Q})$$

Motivated by Dual Learning (He et al., 2016), we integrate question generation into answer ranking procedure, by changing the above formula to:

$$\widehat{\mathcal{A}} = \arg\max_{\mathcal{A}} \{P(\mathcal{A}|\mathcal{Q}) + \lambda \cdot QQ(\mathcal{Q}, \mathcal{Q}^{gen}_{max})\}$$

λ is hyper-parameter, and in order to compute $QQ(\mathcal{Q}, \mathcal{Q}^{gen}_{max})$, we generate top-10 questions $\{\mathcal{Q}^{gen}_1, ..., \mathcal{Q}^{gen}_{10}\}$ for current answer candidate $\mathcal{A}$, and then compute the *question-to-generated question* matching score $QQ(\mathcal{Q}, \mathcal{Q}^{gen}_{max})$, by computing the similarity between input question $\mathcal{Q}$ and generated questions $\{\mathcal{Q}^{gen}_1, ..., \mathcal{Q}^{gen}_{10}\}$ as:

$$QQ(\mathcal{Q}, \mathcal{Q}^{gen}_{max}) = \arg\max_{i=1,...,10} sim(\mathcal{Q}, \mathcal{Q}^{gen}_i) \cdot p(\mathcal{Q}^{gen}_i)$$

$sim(\mathcal{Q}, \mathcal{Q}^{gen}_i)$ is the similarity between the input question $\mathcal{Q}$ and the i^{th} generated question $\mathcal{Q}^{gen}_i$, and computed as the cosine distance between averaged word embedding of $\mathcal{Q}$ and averaged word embedding of $\mathcal{Q}^{gen}_i$, $p(\mathcal{Q}^{gen}_i)$ denotes the posterior probability that is computed based on the generation score of each generated question:

$$p(\mathcal{Q}^{gen}_i) = \frac{p(\mathcal{Q}^{gen}_i|\mathcal{A})}{\sum_{i'=1}^{10} p(\mathcal{Q}^{gen}_{i'}|\mathcal{A})}$$

$p(\mathcal{Q}^{gen}_i|\mathcal{A})$ is output by the question generation model described in Section 6. The underlying motivation is that, *the questions generated from correct answers are more likely to be similar to labeled questions than questions generated from wrong answers*.

8 Related Work

Yao et al. (2012) proposed a semantic-based question generation approach, which first parses the input sentence into its corresponding Minimal Recursion Semantics (MRS) representation, and then generates a question guided by the English Resource Grammar that includes a large scale handcrafted lexicon and grammar rules.

Labutov et al. (2015) proposed an 'ontology-crowd-relevance' method for question generation. First, Freebase types and Wikipedia session names are used as semantic tags to understand texts. Question are then generated based on question templates that are aligned with types/session names and labeled by crowdsourcing. All generated questions are ranked by a relevance model.

Chali and Hasan (2015) proposed a topic-to-question method, which uses about 350 general-purpose rules to transform the semantic-role labeled sentences into corresponding questions.

Serban et al. (2016) used the encoder-decoder framework to generate 30M QA pairs, but their inputs are knowledge triples, instead of passages.

Song and Zhao (2016) proposed a question generation method using question template seeds and used search engine to do question expansion.

Du et al. (2017) proposed a neural question generation method using a vanilla sequence-to-sequence RNN model, which is most-related to our work. But this method is still based on labeled dataset, and tried RNN only.

Comparing to all these related work mentioned above, our question generation approach has two uniqueness: (1) all question patterns, that are used as training data for question generation, are automatically extracted from a large scale CQA question set without any crowdsourcing effort. Such question patterns reflect the most common user intentions, and therefore are useful to search, QA, and chatbot engines; (2) it is also the first time question generation is integrated and evaluated in an end-to-end QA task directly, and shows significant improvements.

9 Experiment

9.1 Dataset

As described in Section 4.1, we collect $1,984,401$ $< A, \mathcal{Q}_p, \mathcal{Q}_t >$ pairs from YahooAnswers, and use them as the training set of the question pattern prediction model. We re-use the dev sets and test sets of SQuAD, MS MARCO, and WikiQA, to evaluate the quality of generated questions. The dataset statistics are in Table 2.

	# of QA Pairs
training set	1,984,401
dev set (SQuAD)	26,442
test set (SQuAD)	26,604
dev set (MS MARCO)	39,510
test set (MS MARCO)	42,850
dev set (WikiQA)	2,733
test set (WikiQA)	6,165

Table 2: Dataset Statistics.

Besides, an answer sentence selection model (Yin et al., 2016) is trained based on the 1,984,401 QA pairs from the training set as well, and used to compute the QA matching score for question ranking, as we described in Section 6. Feature weights for question ranking are optimized on dev set.

9.2 Evaluation on Question Generation

We first perform a vanilla sequence-to-sequence method (Du et al., 2017) using the original training sets of these three datasets, and show QG results in Table 3, where BLEU 4 score is used as the metric.

BLEU 4	Seq2Seq-QG
SQuAD	12.28
MS MARCO	10.46
WikiQA	2.04

Table 3: QG results using original training sets.

We then evaluate the quality of the generated questions based on auto-extracted training set. For each passage in the test set, we generate two top-1 questions based on retrieval-based method and generation-based method respectively, and then compare them with labeled questions using *BLEU 4* as the metric. Results are listed in Table 4.

BLEU 4	R-QG	G-QG
SQuAD (crawled)	9.87	12.39
MS MARCO (crawled)	9.86	11.46
WikiQA (crawled)	11.38	13.57

Table 4: QG results using auto-extracted training set, where **R-QG** denotes results from Retrieval-based QG method, **G-QG** denotes results from Generation-based QG method.

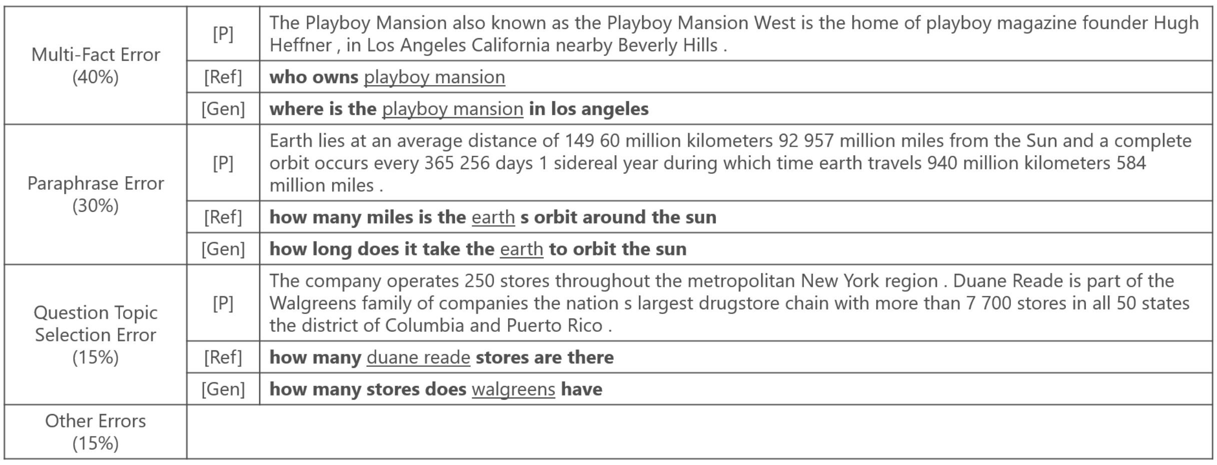
From Table 3 and 4 we can see two findings: (1) Comparing to QG results based on original labeled training sets, G-QG achieves comparable or better results. We think this is due to two facts: first, the size of the automatically constructed training set is much larger than the labeled training sets, and second, as the QA pairs from CQA websites are generated by real users, the quality is good. (2) Generation-based QG performs better than Retrieval-based QG. By analyzing outputs we find that, for question pattern prediction, both retrieval-based and generation-based methods perform similarly. However, Generation-based QG performs better than Retrieval-based QG on question topic selection. This could be caused by the fact that, in Generation-based QG, question topic selection is based on the attention mechanism, which is optimized together with question pattern prediction in an end-to-end way; while in Retrieval-based QG, question topic selection is a separate task, and based on the similarity between each question topic candidate and historical question topics of a given question pattern. The embedding of each question topic is pre-trained, which is not directly related to the question generation task. So such method cannot handle unseen question topics very well. Another disadvantage of Retrieval-based QG is that, each time, we have to compute the similarity between the input passage and each question pattern. When question pattern size is large, the computation is very expensive.

In order to better understand the question generation quality, we manually check a set of sampled outputs, and list the main errors in Figure 2:

- *Multi-Fact Error* (40%). Most input passages include more than one fact. For such a question, it is reasonable to generate different questions from different aspects, all of which can be answered by the input passage. For each passage in QAGen, we only label one question as ground truth. In the future, we will extend QAGen to be a more comprehensive dataset, by labeling multiple questions to each passage for more reasonable evaluation;

- *Paraphrase Error* (30%). The same question can be expressed by different ways. Labeling more paraphrased questions for a passage can alleviate this issue as well;

- *Question Topic Selection Error* (15%). This error is caused by selecting either a total-



Multi-Fact Error (40%)	[P]	The Playboy Mansion also known as the Playboy Mansion West is the home of playboy magazine founder Hugh Heffner , in Los Angeles California nearby Beverly Hills .
	[Ref]	**who owns** playboy mansion
	[Gen]	**where is the** playboy mansion **in los angeles**
Paraphrase Error (30%)	[P]	Earth lies at an average distance of 149 60 million kilometers 92 957 million miles from the Sun and a complete orbit occurs every 365 256 days 1 sidereal year during which time earth travels 940 million kilometers 584 million miles .
	[Ref]	**how many miles is the** earth **s orbit around the sun**
	[Gen]	**how long does it take the** earth **to orbit the sun**
Question Topic Selection Error (15%)	[P]	The company operates 250 stores throughout the metropolitan New York region . Duane Reade is part of the Walgreens family of companies the nation s largest drugstore chain with more than 7 700 stores in all 50 states the district of Columbia and Puerto Rico .
	[Ref]	**how many** duane reade **stores are there**
	[Gen]	**how many stores does** walgreens **have**
Other Errors (15%)		

Figure 2: List of error analysis examples. **[P]** denotes a passage, **[Ref]** denotes the labeled question of the passage, and **[Gen]** denotes the generated question of the passage.

ly wrong question topic, or a partially right question topic. In the future, we plan to develop an independent question topic selection model for the question generation task.

9.3 Evaluation on QA

As described in Section 7, we combine question generation into QA system for answer sentence selection task, and do evaluation on SQuAD, MS MARCO, and WikiQA. Evaluation results are shown in Table 5, 6, and 7, where **QA** denotes the result of our in-house implementation of a retrieval-based answer selection approach (DocChat) proposed by (Yan et al., 2016), **QA+QG** denotes result by combining question-to-generated question matching score with DocChat score.

SQuAD	MAP	MRR	ACC@1
QA	0.8843	0.8915	0.8160
QA+QG	0.8887	0.8963	0.8232

Table 5: Impact of QG on SQuAD.

MS MARCO	MAP	MRR	ACC@1
QA	0.5131	0.5195	0.3029
QA+QG	0.5230	0.5291	0.3153

Table 6: Impact of QG on MS MARCO.

The improvement on MS MARCO dataset is most significant. We think it due to the fact that, the questions from MS MARCO dataset are from Bing search log, which are generated naturally by real users. This is similar to the questions coming for CQA websites; while questions from the other datasets are labeled by crowd-sourcing.

WikiQA	MAP	MRR	ACC@1
QA	0.7703	0.7851	0.6540
QA+QG	0.7742	0.7893	0.6624

Table 7: Impact of QG on WikiQA.

In order to explain these improvements, two datasets, *WikiQG+* and *WikiQG-*, are built from WikiQA test set: given each document and its labeled question, we pair the question with its CORRECT answer sentence as a QA pair and add it to WikiQG+; we also pair the same question with a randomly selected WRONG answer sentence as a QA pair and add it to WikiQG-. Then, we generate questions for passages in WikiQG+ and WikiQG- respectively, and compare them with labeled questions. The BLEU 4 score is **0.2031** on WikiQG+, and 0.1301 on WikiQG-, which indicates that the questions generated from correct answers are more likely to be similar to labeled questions than questions generated from wrong answers.

10 Conslusion

This paper presents a neural question generation method that is based on training data collected from CQA questions. We integrate the QA pair generation task into an end-to-end QA task, and show significant improvements, which indicates that, QA task and QG are dual tasks that can boost each other. In the future, we will explore more ways to leverage QG for QA task.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *ACL*.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. In *arXiv*.

Yllias Chali and Sadid Hasan. 2015. Towards topic-to-question generation. In *CL*.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Association for Computational Linguistics (ACL)*.

Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tie-Yan Liu, and Wei-Ying Ma. 2016. Dual learning for machine translation. In *NIPS*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *ACL*.

Igor Labutov, Sumit Basu, and Lucy Vanderwende. 2015. Deep questions without deep understanding. In *ACL*.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *EMNLP*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jiangfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. In *NIPS*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Xipeng Qiu and Xuanjing Huang. 2015. Convolutional neural tensor network architecture for community based question answering. In *IJCAI*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. In *arXiv*.

Iulian Vlad Serban, Alberto Garca-Durn, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questionswith recurrent neural networks: The 30m factoid question-answer corpus. In *ACL*.

Linfeng Song and Lin Zhao. 2016. Domain-specific question generation from a knowledge base. In *arXiv*.

Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitt. 2016. Convolutional neural networks vs. convolution kernels: Feature engineering for answer sentence reranking. In *NAACL*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dynamic coattention networks for question answering. In *ICLR*.

Zhao Yan, Nan Duan, Junwei Bao, Peng Chen, Ming Zhou, Zhoujun Li, and Jianshe Zhou. 2016. Docchat: An information retrieval approach for chatbot engines using unstructured documents. In *ACL*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP*.

Xuchen Yao, Gosse Bouma, and Yi Zhang. 2012. Semantics-based question generation and implementation. In *Dialogue and Discourse*.

Wenpeng Yin and Hinrich Schtze. 2017. Task-specific attentive pooling of phrase alignments contributes to sentence matching. In *EACL*.

Wenpeng Yin, Hinrich Schtze, Bing Xiang, and Bowen Zhou. 2016. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. In *TACL*.

Lei Yu, Karl Morizt Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. In *NIPS Workshop*.

Matthew Zeiler. 2012. Adadelta: an adaptive learning rate method. In *CoRR abs*.