Learning to Collaborate for Question Answering and Asking

Duyu Tang[‡], **Nan Duan**[‡], **Zhao Yan**[†], **Zhirui Zhang**[♭], **Yibo Sun**[§],
Shujie Liu[‡], **Yuanhua Lv**[♮], **Ming Zhou**[‡]
[‡]Microsoft Research Asia, Beijing, China
[♮]Microsoft AI and Research, Sunnyvale CA, USA
[†]Beihang University, Beijing, China
[♭]University of Science and Technology of China, Anhui, China
[§]Harbin Institute of Technology, Harbin, China
{dutang,nanduan,v-zhaoya,v-zhirzhi,v-yibsu,shujliu,yuanhual,mingzhou}@microsoft.com

Abstract

Question answering (QA) and question generation (QG) are closely related tasks that could improve each other; however, the connection of these two tasks is not well explored in literature. In this paper, we give a systematic study that seeks to leverage the connection to improve both QA and QG. We present a training algorithm that generalizes both Generative Adversarial Network (GAN) and Generative Domain-Adaptive Nets (GDAN) under the question answering scenario. The two key ideas are improving the QG model with QA through incorporating additional QA-specific signal as the loss function, and improving the QA model with QG through adding artificially generated training instances. We conduct experiments on both document based and knowledge based question answering tasks. We have two main findings. Firstly, the performance of a QG model (e.g in terms of BLEU score) could be easily improved by a QA model via a policy gradient. Secondly, directly applying GAN that regards all the generated questions as negative instances could not improve the accuracy of the QA model. Learning when to regard generated questions as positive instances could bring performance boost.

1 Introduction

In this work, we consider the task of joint learning of question answering and question generation. Question answering (QA) and question generation (QG) are closely related natural language processing tasks. The goal of QA is to obtain an answer given a question. The goal of QG is almost reverse which is to generate a question from the answer. In this work, we consider answer selection (Yang et al., 2015; Balakrishnan et al., 2015) as the QA task, which assigns a numeric score to each candidate answer, and selects the top ranked one as the answer. We consider QG as a generation problem and exploit sequence-to-sequence learning (Seq2Seq) (Du et al., 2017; Zhou et al., 2017) as the backbone of the QG model.

The key idea of this work is that QA and QG are two closely tasks and we seek to leverage the connection between these two tasks to improve both QA and QG. Our primary motivations are twofolds. **On one hand**, the Seq2Seq based QG model is trained by maximizing the literal similarity between the generated sentence and the ground truth sentence with maximum-likelihood estimation objective function (Du et al., 2017). However, there is no signal indicating whether or not the generated sentence could be correctly answered by the input. This problem could be precisely mitigated through incorporating QA-specific signal into the QG loss function. **On the other hand**, the capacity of a statistical model depends on the quality and the amount of the training data (Sun et al., 2017). In our scenario, the capacity of the QA model depends on the difference between the positive and negative patterns embodied in the training examples. A desirable training dataset should contain the question-answer pairs that are literally similar yet have different category labels, i.e. some question-answer pairs are correct and some are wrong. However, this kind of dataset is hard to obtain in most situations because of the lack of manual annotation efforts. From this perspective, the QA model could exactly benefit from the QG model through incorporating additional question-answer pairs whose questions are automatically generated by the QG model[1].

To achieve this goal, we present a training algorithm that improves the QA model and the

[1]An alternative way is to automatically generate answers for each question. Solving the problem in this condition requires an answer generation model (He et al., 2017), which is out of the focus of this work. Our algorithm could also be adapted to this scenario.

Proceedings of NAACL-HLT 2018, pages 1564–1574
New Orleans, Louisiana, June 1 - 6, 2018.

QG model in a loop. The QA model improves QG through introducing an additional QA-specific loss function, the objective of which is to maximize the expectation of the QA scores of the generated question-answer pairs. Policy gradient method (Williams, 1992; Yu et al., 2017) is used to update the QG model. In turn, the QG model improves QA through incorporating additional training instances. Here the key problem is how to label the generated question-answer pair. The application of Generative Adversarial Network (GAN) (Goodfellow et al., 2014; Wang et al., 2017) in this scenario regards every generated question-answer pair as a negative instance. On the contrary, Generative Domain-Adaptive Nets (GDAN) (Yang et al., 2017) regards every generated question-answer pair appended with special domain tag as a positive instance. However, it is non-trivial to label the generated question-answer pairs because some of which are good paraphrases of the ground truth yet some might be negative instances with similar utterances. To address this, we bring in a collaboration detector, which takes two question-answer pairs as the input and determines their relation as collaborative or competitive. The output of the collaboration detector is regarded as the label of the generated question-answer pair.

We conduct experiments on both document based (Yang et al., 2015) and knowledge (e.g. web table) based question answering tasks (Balakrishnan et al., 2015). Results show that the performance of a QG model (e.g in terms of BLEU score) could be consistently improved by a QA model via policy gradient. However, regarding all the generated questions as negative instances (*competitive*) could not improve the accuracy of the QA model. Learning when to regard generated questions as positive instances (*collaborative*) could improve the accuracy of the QA model.

2 Related Work

Our work connects to existing works on question answering (QA), question generation (QG), and the use of generative adversarial nets in question answering and text generation.

We consider two kinds of answer selection tasks in this work, one is table as the answer (Balakrishnan et al., 2015) and another is sentence as the answer (Yang et al., 2015). In natural language processing community, there are also other types of QA tasks including knowledge based QA (Berant et al., 2013), community based QA (Qiu and Huang, 2015) and reading comprehension (Rajpurkar et al., 2016). We believe that our algorithm is generic and could also be applied to these tasks with dedicated QA and QG model architectures. Despite the use of sophisticated features could learn a more accurate QA model, in this work we implement a simple yet effective neural network based QA model, which could be conventionally jointly learned with the QG model through back propagation.

Question generation draws a lot of attentions recently, which is partly influenced by the remarkable success of neural networks in natural language generation. There are several works on generating questions from different resources, including a sentence(Heilman, 2011), a topic (Chali and Hasan, 2015), a fact from knowledge base (Serban et al., 2016), etc. In computer vision community, there are also recent studies on generating questions from an image (Mostafazadeh et al., 2016). Our QG model belongs to sentence-based question generation.

GAN has been successfully applied in computer vision tasks (Goodfellow et al., 2014). There are also some recent trials that adapt GAN to text generation (Yu et al., 2017), question answering (Wang et al., 2017), dialogue generation (Li et al., 2016), etc. The relationship of the discriminator and the generator in GAN are competitive. The key finding of this work is that, directly applying the idea of "competitive" in GAN does not improve the accuracy of a QA model. We contribute a generative collaborative network that learns when to collaborate and yields empirical improvements on two QA tasks.

This work relates to recent studies which attempt to improve the performance of a discriminative QA model with generative models (Wang et al., 2017; Yang et al., 2017; Dong et al., 2017; Duan et al., 2017). These works regard QA as the primary task and use auxiliary task, such as question generation and question paraphrasing, to improve the primary task. This is one part of our goal and our another goal is to improve the QG model with the QA system and further to increasingly improve both tasks in a loop.

In terms of assigning category label to the generated question, Generative Adversarial Network (GAN) (Goodfellow et al., 2014; Wang et al.,

2017) and Generative Domain-Adaptive Nets (GDAN) (Yang et al., 2017) could be viewed as special cases of our algorithm. Our algorithm learns when to assign positive or negative labels, while GAN always assigns negative labels and GDAN always assigns positive labels. Besides, our work differs from (Wang et al., 2017) in that our question generation model is a generative model while theirs is actually a discriminative matching model. The approach of (Dong et al., 2017) learns to generate question from question via paraphrasing, and use the generated questions in the inference process. In this work, the QA model and the QG model are applied separately in the inference process. This inspires us to jointly conduct QA and QG in the inference process, which we leave as a future work.

3 Generative Collaborative Network

In this section, we first formulate the task of QA and QG, and then present our algorithm that jointly trains the QA and QG models.

3.1 Task Formulation

This work involves two tasks: question answering (QA) and question generation (QG).

There are different kinds of QA tasks in the natural language processing area. To verify the scalability of our algorithm, we consider two types of answer selection tasks, both of which are fundamental QA tasks in research community and of great importance in industrial applications including web search and chatbot. Both tasks take a question q and a list of candidate answers $A = \{a_1, a_2, ..., a_n\}$ as input, and outputs an answer a_i which has the largest probability to correctly answer the question. The only difference is that the answer in the task of answer sentence selection (Yang et al., 2015) is a natural language sentence, while the answer in table search (Balakrishnan et al., 2015) is a structured table consisting of caption, attributes and cells. Our QA model is abbreviated as $P_{qa}(a, q; \theta_{qa})$, whose output is the probability that q and a being a correct question-answer pair, and the parameter is denoted as θ_{qa}.

The task of QG takes an answer a which is a natural language sentence or a structured table, and outputs a natural language question q which could be answered by a. Inspired by the remarkable progress of sequence-to-sequence (Seq2Seq) learning in natural language generation, we deal with QG in an end-to-end fashion and develop a generative model based on Seq2Seq learning. Our QG model is abbreviated as $P_{qg}(q|a; \theta_{qg})$, whose output is the probability of generating q from a and the parameter is denoted as θ_{qg}.

3.2 Algorithm Description

We describe the joint learning algorithm in this part. The end goal is to leverage the connection of QA and QG to improve the performances on both QA and QG tasks. A brief illustration of the training progress is given in Figure 1 , which includes a QA model, a QG model and a collaboration detector (CD). A formal description of the algorithm is given in Algorithm 1. We can see that the QA model and the QG model both have two training objectives. One part is the standard supervised learning objective based on task-specific supervisions. Another part of the objective is obtained by leveraging auxiliary tasks.

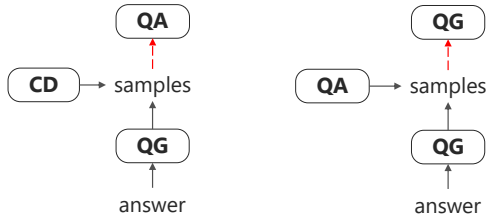

Figure 1: An brief illustrating of the joint training process. The red dashed line stands for the model being updated. QA, QG and CD stand for question answering, question generation and collaboration detection, respectively.

The supervised objective of the QA model is to maximize the probability of the correct category label, and the supervised objective of the QG model is to maximize the probability of the correct question sequence. Since the goal of QA is to predict whether a question-answer pair is correct or not, training the QA model requires negative QA pairs whose labels are zero. But these negative QA pairs are not used for training the QG model as the goal of QG is to generate the correct question.

The main contribution of this work is to explore effective learning objectives that leverage auxiliary tasks. In order to improve the QA model, we generate additional training instances, each of which is composed of a question, an answer and a category label. In this work, we clamp the answer part and feed the answer to the QG model to

Algorithm 1 Generative Collaborative Network for QA and QG

Input: training data D; the batch size for QG training m; the beam size for QG inference k; hyper parameters λ_{qg} and λ_{qg}; hyper parameters b_{qa} and b_{qg}; pretrained collaboration detector $P_{cd}(q, q')$
Output: QA model $P_{qa}(a, q)$ parameterized by θ_{qa}; QG model $P_{qg}(q|a)$ parameterized by θ_{qg}

pretrain $P_{qa}(a, q)$ and $P_{qg}(q|a)$ separately on D
repeat
  get a minibatch of positive QA pairs $PD = \{q_i^p, a_i^p\} \in D, 1 \leq i \leq m$, in which a_i^p is the answer of q_i^p
  get a minibatch of negative QA pairs $ND = \{q_i^n, a_i^n\} \in D, 1 \leq i \leq m$, in which a_i^n is not the answer of q_i^n
  apply $P_{qg}(q|a)$ on PD to generate in a list of question-answer beams $GD = \{q_{ij}^g, a_i^g\}, 1 \leq i \leq m, 1 \leq j \leq k$
  apply $P_{qa}(a, q)$ on GD to assign a QA-specific score to each generated instance
  choose the top ranked result from each beam in GD, and then apply $P_{cd}(q, q')$ on the selected instance
  update θ_{qa} by maximizing the following objective

$$\sum_{i=1}^{m} \Big(logP_{qa}(a_i^p, q_i^p) + log\big(1 - P_{qa}(a_i^n, q_i^n)\big) \Big) + \lambda_{qa} \sum_{i=1}^{m} \Big(\mathbb{I}_{b_{qa}}[P_{cd}(q_i^p, q_{i0}^g)] logP_{qa}(a_i^p, q_{i0}^g) \Big) + \lambda_{qa} \sum_{i=1}^{m} \Big(\big(1 - \mathbb{I}_{b_{qa}}[P_{cd}(q_i^p, q_{i0}^g)]\big) log\big(1 - P_{qa}(a_i^p, q_{i0}^g)\big) \Big) \quad (1)$$

  update θ_{qg} by maximizing the following objective

$$\sum_{i=1}^{m} logP_{qa}(q_i^p|a_i^p) + \lambda_{qg} \sum_{i=1}^{m} \sum_{j=1}^{k} P_{qa}(a_i^p, q_{ij}^g) logP_{qg}(q_{ij}^g|a_i^p) \quad (2)$$

until models converge

generate the question. We use beam search and select the top ranked result as the question.[2] Here the issue is how to infer the label of the generated instance. We believe that heuristically assigning the label as 0 or 1 is problematic. For instance, let us suppose the answer sentence is *"Microsoft was founded by Paul Allen and Bill Gates on April 4, 1975."*, and the ground truth question is *"who founded Microsoft"*. In this case, the generated question *"who is the founder of Microsoft"* is a good one yet *"who is the founder of Google"* and *"how old is Bill Gates"* are both bad cases. To address this, we introduce an additional collaboration detector (CD) to infer the label of the generated instance. Intuitively, the CD acts as a discriminative paraphrase model, which measures the semantic similarity between the ground truth question and generated question. In equation (1), the $\mathbb{I}_{b_{qa}}(x)$ is an indicator function whose value is 1 if the value of x is larger than a threshold b_{qa}, such as 0.5 or 0.3. The hyper parameter λ_{qa} controls the weight of the auxiliary objective to the QA model.

In turn, the QA model is used to assign a QA-specific score $P_{qa}(a, q')$ to each generated question q'. We follow the recent reinforcement learning based approach for dialogue prediction (Li et al., 2016), and define simple heuristic rewards that characterize good questions. The goodness of the generated question is measured by the prediction of the QA model. Similar to the strategy adopted by (Zaremba and Sutskever, 2015), we use a baseline strategy b_{qg} (e.g. 0.5) to decrease the learning variance. The expected reward (Williams, 1992; Yu et al., 2017) for a generated question is given in Equation (2). In this way, the parameters of the QG model could be conventionally updated with stochastic gradient descent.

We pretrain the QA model and the QG model before the joint learning process. The main reason is that a randomized QA model will provide unreliable rewards to the QG model, and a randomized QG model will generate bad questions.

4 The Neural Architecture of Each Module

Our algorithm includes a question answer (QA) model, a question generation (QG) model and a collaboration detector (CD) model. We implement these models with dedicated neural networks.

As we have mentioned before, our training algorithm is applied to both document-based and web table based question answer tasks. In this section, we take table based QA and QG tasks to describe the neural architecture of each module. A question/query q is a natural language expression consisting of a list of words. A table t has fixed

[2] We also implemented using all the beam search results or sampling one result from the beam. However, these tricks do not bring performance boost.

schema including one or more $headers$, one or more $cells$, and a $caption$. A header indicates the property of a column, and a cell is a unit where a row and a column intersects. The caption is typically an explanatory text about the table.

4.1 The Question Answer (QA) Model

We develop a neural network to match a natural language question/query to a structured table. Since a table has multiple aspects including headers, cells and the caption, the model is developed to capture the semantic relevance between a query and a table at different levels.

As the meaning of a query is sensitive to the word order, i.e. the intentions of "*list of flights london to berlin*" and "*list of flights berlin to london*" are totally different, we represent a query with a sequential model. In this work, recurrent neural network (RNN) is used to map a query of variable length to a fixed-length vector. We use gated recurrent unit (GRU) (Cho et al., 2014) as the basic computation unit, which adaptively forgets the history and remembers the input.

$$z_i = \sigma(W_z e_i^q + U_z h_{i-1}) \quad (3)$$

$$r_i = \sigma(W_r e_i^q + U_r h_{i-1}) \quad (4)$$

$$\widetilde{h}_i = \tanh(W_h e_i^q + U_h(r_i \odot h_{i-1})) \quad (5)$$

$$h_i = z_i \odot \widetilde{h}_i + (1 - z_i) \odot h_{i-1} \quad (6)$$

where z_i and r_i are update and reset gates of GRU. We use bi-directional RNN to get the meaning of a query from forward and backward directions, and concatenate two last hidden states as the query vector.

An important property of a table is that exchanging two rows or two columns does not change its meaning. To satisfy this condition, we develop an attention based approach, in which the header and cells are regarded as the external memory. Each header/cell is represented as a vector. Given a query vector, the model calculates the weight of each memory unit and then output a query-specific header/cell representation through weighted average (Bahdanau et al., 2015; Sukhbaatar et al., 2015). This process could be repeated executed for several times, so that more abstractive evidences could be retrieved and composed to support the final decision. Similar techniques have been successfully applied in table-based question answering (Yin et al., 2015b; Neelakantan et al., 2015).

We represent the table caption with RNN, the same strategy we have adopted to represent the query. Element-wised multiplication is used to compose the query vector and the caption vector. Furthermore, since the number of co-occurred words directly reflect the relatedness between the question and the answer, we incorporate the embedding of co-occurred word count as additional features. Finally, we feed the concatenation of all the vectors to a $softmax$ layer whose output length is 2. We have implemented a ranking based loss function $l_{qa} = max(0, 1 - P_{qa}(a, q) + P_{qa}(a', q))$ and a negative log-likelihood (NLL) base loss function $l_{qa} = -\log(P_{qa}(a, q))$. We find that NLL works better and use it in the following experiments.

4.2 The Question Generation (QG) Model

Inspired by the notable progress that sequence-to-sequence learning (Seq2Seq) (Sutskever et al., 2014) has made in natural language generation, we implement a table-to-sequence (Table2Seq) approach that generates natural language question from a structured table.

Table2Seq is composed of an encoder and a decoder. The encoder maps the caption, headers, and cells into continuous vectors, which will be fed to the decoder to generate a question in a sequential way. Similar with the way we have adopted in the QA model, we represent the caption with bidirectional GRU based RNN. The vector of each word in the caption is the concatenation of hidden states from both directions. The vectors of headers and cells are regarded as additional hidden states of the encoder. The representation of each cell is also mixed with the corresponding header representation. The initial vector of the decoder is the average of the caption vector, header vector, and the cell vector.

The backbone of the decoder is an attention based GRU RNN, which generates a word at each time step and repeats the process until generating the end-of-sentence symbol. We made two modifications to adapt the decoder to the table structure. The first modification is that the attention model is calculated over the headers, cells and the caption of a table. Ideally, the decoder should learn to focus on a region of the table when generating a word. The second modification is a table based copying mechanism. It has been proven that the copying mechanism (Gulcehre et al., 2016; Gu

et al., 2016) is an effective way to replicate low-frequent words from the source to the target sequence in sequence-to-sequence learning. In the decoding process, a word is generated either from the target vocabulary via standard $softmax$ or from a table via the copy mechanism. A neural gate g_t is used to trade-off between generating from the target vocabulary and copying from the table. The probability of generating a word y calculated as follows, where $\alpha_t(y)$ is the attention probability of the word y from the table at time step t and $\beta_t(y)$ is the probability of predicting the word y from the $softmax$ at time step t.

$$p_t(y) = g_t \odot \alpha_t(y) + (1 - g_t) \odot \beta_t(y) \quad (7)$$

Since every component of the Table2Seq is differentiable, the parameters could be optimized in an end-to-end fashion with back-propagation. Given a question-answer pair (x, y), the supervised training objective is to maximize the probability of question word at each time step. In the inference process, beam search is used to generate top-k confident results, where k is the beam size.

4.3 The Collaboration Detector (CD)

The goal of a collaboration detector is to determine the label of the instance generated by the QG model. The positive prediction, namely the predicted value is equals to 1, stands for the collaborative relationship between the generated instance and the ground truth, while the negative prediction stands for the competitive relationship.

We consider this task as predicting the category of the given two question-answer pairs, one of which is the ground truth, and another is the generated question-answer pair. Since the answer part is the same, we simplify the problem as classifying two questions as related or not, which is a binary classification problem.

The neural architecture of the collaboration detector (CD) is exactly the same as the caption component in the QA model. We represent two questions with bidirectional RNN, and use element-wise multiplication to do the composition. The result is further concatenated with a co-occurred word count embedding, followed by a $softmax$ layer. The model is trained by minimizing the negative log-likelihood label, which is provided in the training data.

The training data of the CD model includes two parts. The first part is from Quora dataset[3], which is built for detecting if pairs of question text are semantically equivalent. The Quora dataset has 345,989 positive question pairs and 255,027 negative pairs. We further obtain the second part of the training data from the web queries, which are more similar to the web queries in our two QA task. We obtain the query dataset from query logs through clustering the web queries that click the same web page. In this way, we obtain 6,118,023 positive query pairs. We use a heuristic rule to generate the negative instances for the query dataset. For each pair of query text, we clamp the first query and retrieve a query that is mostly similar to the second query. To improve the efficiency of this process, we randomly sample 10,000 queries and define the "similarity" as the number of co-occurred words in two questions. In this way we collect another 6,118,023 negative pairs of query text. We initialize the values of word embeddings with $300d$ Glove vectors[4], which is learned on Wikipedia texts. We use a held-out data consisting of 20K query pairs to check the performance of the CD model. The accuracy of the CD model on the held-out dataset is 83%. In the joint training process, we clamp the parameters of the CD model and use its outputs to facilitate the learning of the QA model.

5 Experiment

We conduct experiments on table-based QA and document-based QA tasks. We will describe experimental settings and report results on these two tasks in this section.

5.1 Table based QA and QG

Setting We take table retrieval (Balakrishnan et al., 2015) as the table-based QA task. Given a query and a collection of candidate table answers, the task aims to return a table that is most relevant to the query. Figure 2 gives an example of this task, in which a query matches to different aspects of a table. We regard document-based QA tasks as a special case of the table-based QA task, in which the cells and the headers are both empty.

We conduct experiments on the web data. The queries come from real-world user queries which we obtain from the search log of a commercial

[3] https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

[4] https://nlp.stanford.edu/projects/glove/

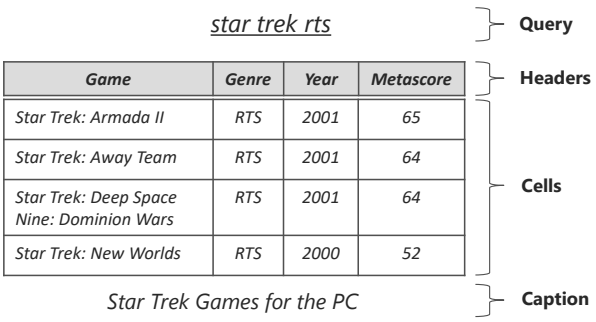

Figure 2: An example illustrating the table-based QA task.

search engine. We filter them down to only those that are directly answered by a table. In this way, we collect 1.49M query-table pairs. An example of the data is given in Figure 2. We randomly select 1.29M as the training set, 0.1M as the dev set and 0.1M as the test set.

We evaluate the performance on table-based QA with *Mean Average Precision (MAP)* and *Precision@1 (P@1)* (Manning et al., 2008). We use the same candidate retrieval adopted in (Yan et al., 2017), namely representing a table as bag-of-words, to guarantee the efficiency of the approach. Each query has 50 candidate tables on average. It is still an open problem to automatically evaluate the performance of a natural language generation system (Lowe et al., 2017). In this work, we use BLEU-4 (Papineni et al., 2002) score as the evaluation metric, which measures the overlap between the generated question and the referenced question. The hyper parameters are tuned on the validation set and the performance is reported on the test set.

Results and Analysis We report the results and our analysis on table-based QA and QG respectively in this part.

We first report the results of single systems on table-based QA. We compare to four single systems implemented by (Yan et al., 2017). In **BM25**, each table is represented as a flattened vector, and the similarity between a query and a table is calculated with the BM25 algorithm. **WordCnt** uses the number of co-occurred words in query-caption pair, query-header pair, and query-cell pair, respectively. **MT based PP** is a phrase-level feature. The features come from a phrase table which is extracted from bilingual corpus via statistical machine translation approach (Koehn et al., 2003). LambdaMART (Burges, 2010) is used to train the ranker. **CNN** uses convolutional neural network to measure the similarity between the query and table caption, table headers, and table cells, respectively. **TQNN** is the table-based QA model implemented in this work, which is regard as the baseline for the joint learning algorithm. Results of single systems are given in Table 1. We can see that BM25 is a simple yet very effective baseline method. Our basic model performs better than all the single models in terms of MAP.

Method	MAP	Acc@1
BM25	0.429	0.294
WordCnt	0.318	0.190
MT based PP	0.327	0.213
CNN	0.359	0.238
TQNN (baseline)	0.439	0.285
Seq2SeqPara	0.437	0.283
GCN (competitive)	0.436	0.282
GCN (collaborative)	0.446	0.292
GCN (final)	**0.456**	**0.301**

Table 1: The performances of single systems on table-based question answering (p-value < 0.01 with t-test between TQNN and GCN).

We also implement four different joint learning settings. In these settings, the QA model and the QG model are all pretrained, and the same way (policy gradient) is used to improve the QG model via the QA predictions. The only difference is how the QA model benefits from the QG model. As we use external resources to train a CD model, we also implement **Seq2SeqPara** for comparison. We train a question generator with a Seq2Seq model on the CD training data, and regard the generated questions as positive instances. Our generative collaborative network is abbreviated as **GCN**. **GCN (competitive)** is analogous to (Goodfellow et al., 2014), where all the generated questions are regarded as negative instances (with label as zero). On the contrary, **GCN (collaborative)** is analogous to (Yang et al., 2017), where the generated questions are regard as positive instances. Our main observation from Table 1 is that simply regarding all the generated questions as negative instances ("competitive") could not bring performance boost. On the contrary, regarding the generated questions as positive ones ("collaborative") improves the QA model. Our algorithm (GCN) significantly improves the TQNN model. Based on these results, we believe that the relationship

between the QA model and the QG model should not be always competitive. Learning when to collaborate through leveraging a CD model is a practical way to improve the performance on question answering.

As described in Equation (1), the influence of the CD model on the QA model also depends on the value of the hyper parameter b_{qa}. A small value of b_{qa} stands for a preference of "collaborative", while a large value of b_{qa} represents a preference of "competitive". Results are given in Figure 3. The GCN model performs better when b_{qa} is in the range $[0.3, 0.5]$, in which the model prefers "collaborative".

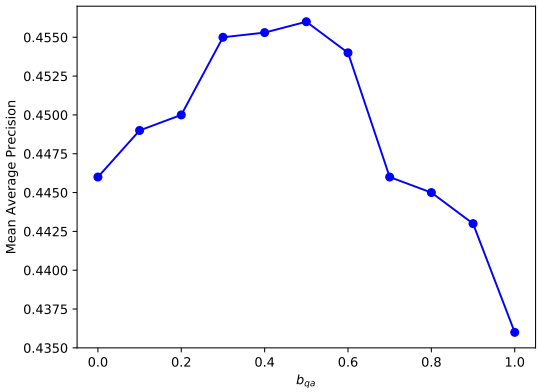

Figure 3: The performances of GCN on table-based QA with different values of b_{qa}. GCN falls back to "competitive" when b_{qa} equals to one. GCN is totally collaborative when b_{qa} equals to zero.

We conduct an additional experiment to test whether our algorithm could improve an existing system. We take BM25 as the baseline, and incorporate one of the five joint models as an additional feature. LambdaMART is used to train the combined ranker. Results are given in Table 2. We can see that the baseline system could be dramatically improved by our system, despite the improvements of different approaches are on par.

Method	MAP	Acc@1
BM25	0.429	0.294
BM25 + TQNN (baseline)	0.650	0.513
BM25 + GAN	0.654	0.519
BM25 + Seq2SeqPara	0.650	0.513
BM25 + GDAN	0.658	0.523
BM25 + GCN (this work)	**0.660**	**0.526**

Table 2: The performances of combined systems on table-based question answering..

We have reported the results on table-based QA. Here we show the performances of different approaches on table-based QG. Results in terms of BLEU-4 are given in Table 3. Different from the trends on QA, "competitive" performs better than "collaborative" on QG. This is reasonable because as the joint training progresses, the QA model in "collaborative" keeps telling the QG model that the generated instances are good enough. On the contrary, the "competitive" model is more critical, which tells the QG model how wrong the generated questions are. In this way, the QG model could be increasingly improved by the QA signal. The QG model is easier to be improved compared to the QA model. Our GCN approach obtains a significant improvement over the baseline model on this task.

Method	Dev	Test
Table2Seq (baseline)	15.71	15.54
Seq2SeqPara	17.01	16.95
GCN (competitive)	17.28	17.34
GCN (collaborative)	16.77	16.66
GCN (final)	**17.59**	**17.61**

Table 3: The performances on table-based question generation. Evaluation metric is BLEU-4 score.

We also report the learning curve of the GCN model as the joint training progresses. The performance on the dev set is given in Figure 4.

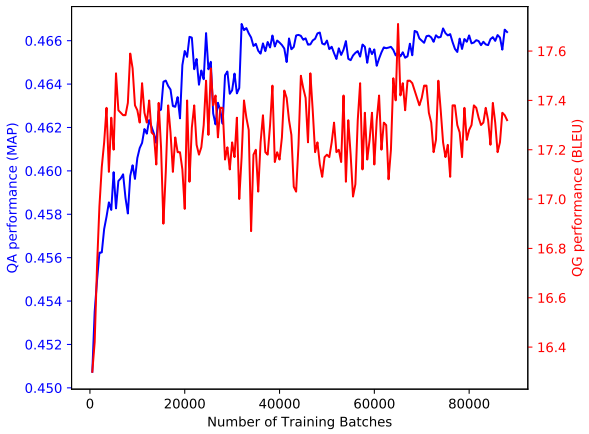

Figure 4: The learning curve of GCN on the dev data. The evaluation metrics are MAP (for QA) and BLEU (for QG).

5.2 Document based QA and QG

To test the scalability of the algorithm, we also apply it to document based QA and QG tasks. The QA task is answer sentence selection (Yang et al.,

Method	MAP	P@1
WordCnt	0.395	0.179
CDSSM (Shen et al., 2014)	0.442	0.228
ABCNN (Yin et al., 2015a)	0.469	0.263
DSL (Tang et al., 2017)	0.484	0.275
DQNN (baseline)	0.471	0.263
Seq2SeqPara	0.470	0.260
GCN (competitive)	0.468	0.257
GCN (collaborative)	0.476	0.272
GCN (final)	**0.492**	**0.282**

Table 4: The performance on document-based QA task (p-value < 0.05 with t-test between DQNN and GCN).

2015). Given a question and a list of candidate answer sentences from a document, the goal is to find a most relevant answer sentence as the answer. Since the WikiQA dataset (Yang et al., 2015) is too small to learn a powerful question generator, we use the MARCO dataset (Nguyen et al., 2016), which is originally designed for reading comprehension yet also has manually annotated labels for sentence/passage selection. A characteristic of MARCO dataset is that the ground truth of the test is invisible to the public. Therefore, we follow (Tang et al., 2017) and split the original validation set into the dev set and the test set. The results on QA and QG are given in Table 4 and Table 5. We can see that the results are almost consistent with the results on table-based QA and QG tasks. Our GCN algorithms achieves promising performances compared to strong baseline methods.

Method	BLEU-4
Seq2Seq (baseline)	8.87
DSL (Tang et al., 2017)	9.31
Seq2SeqPara	9.16
GCN (competitive)	9.22
GCN (collaborative)	9.04
GCN (final)	**9.89**

Table 5: The performance on document-based QG task.

6 Conclusion

We present an algorithm dubbed generative collaborative network for jointly training the question answering (QA) model and the question generation (QG) model. Different from standard GAN, the relationship between QA model (discriminator) and the QG model (generator) in our algorithm is not always competitive. We show that "collaborative" performs better than "competitive" in terms of QA accuracy, and our algorithm that learns when to collaborate obtains further improvement on both QA and QG tasks.

This work could be further improved from several directions. Our current algorithm focuses on the joint training of QA and QG models, while the inferences of these two models are independent. How to conduct joint inference is an interesting future work. Besides, the samples are currently generated from the QG model via beam search. Improving the diversity of the samples requires different sampling mechanisms. Another potential direction is to jointly learn the collaboration detector together with the QA and QG models.

References

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

Sreeram Balakrishnan, Alon Y Halevy, Boulos Harb, Hongrae Lee, Jayant Madhavan, Afshin Rostamizadeh, Warren Shen, Kenneth Wilder, Fei Wu, and Cong Yu. 2015. Applying webtables in practice. In *CIDR*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*. volume 2, page 6.

Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Microsoft Research Technical Report MSR-TR-2010-82* 11(23-581):81.

Yllias Chali and Sadid A Hasan. 2015. Towards topic-to-question generation. *Computational Linguistics* .

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of EMNLP*. pages 1724–1734.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. *arXiv preprint arXiv:1708.06022* .

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

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. *Proceeding of EMNLP* .

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. pages 2672–2680.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393* .

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *arXiv preprint arXiv:1603.08148* .

Shizhu He, Cao Liu, Kang Liu, and Jun Zhao. 2017. Generating natural answers by incorporating copying and retrieving mechanisms in sequence-to-sequence learning. In *Proceedings of ACL*. pages 199–208.

Michael Heilman. 2011. *Automatic factual question generation from text*. Ph.D. thesis, Carnegie Mellon University.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. *Proceedings of NAACL-HLT* 1:48–54.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541* .

Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. In *Proceedings of ACL*. pages 1116–1126.

Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. 2008. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge.

Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. 2016. Generating natural questions about an image. *arXiv preprint arXiv:1603.06059* .

Arvind Neelakantan, Quoc V Le, and Ilya Sutskever. 2015. Neural programmer: Inducing latent programs with gradient descent. *arXiv preprint arXiv:1511.04834* .

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. *arXiv preprint arXiv:1611.09268* .

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*. pages 311–318.

Xipeng Qiu and Xuanjing Huang. 2015. Convolutional neural tensor network architecture for community-based question answering. In *IJCAI*. pages 1305–1311.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of EMNLP*. pages 2383–2392.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In *ACL*. pages 588–598.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the Conference on Information and Knowledge Management (CIKM)*. pages 101–110.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems (NIPS)*. pages 2431–2439.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. 2017. Revisiting unreasonable effectiveness of data in deep learning era. *arXiv preprint arXiv:1707.02968* .

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Duyu Tang, Nan Duan, Tao Qin, and Ming Zhou. 2017. Question answering and question generation as dual tasks. *arXiv preprint arXiv:1706.02027* .

Jun Wang, Lantao Yu, Weinan Zhang, Yu Gong, Yinghui Xu, Benyou Wang, Peng Zhang, and Dell Zhang. 2017. Irgan: A minimax game for unifying generative and discriminative information retrieval models. *arXiv preprint arXiv:1705.10513* .

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

Zhao Yan, Duyu Tang, Nan Duan, Junwei Bao, Yuanhua Lv, Ming Zhou, and Zhoujun Li. 2017. Content-based table retrieval for web queries. *arXiv preprint arXiv:1706.02427* .

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP*. Citeseer, pages 2013–2018.

Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William W Cohen. 2017. Semi-supervised qa with generative domain-adaptive nets. *arXiv preprint arXiv:1702.02206* .

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2015a. Neural generative question answering. *arXiv preprint arXiv:1512.01337* .

Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2015b. Neural enquirer: Learning to query tables with natural language. *arXiv preprint arXiv:1512.00965* .

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*. pages 2852–2858.

Wojciech Zaremba and Ilya Sutskever. 2015. Reinforcement learning neural turing machines. *arXiv preprint arXiv:1505.00521* 419.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. *arXiv preprint arXiv:1704.01792* .