Question Generation from SQL Queries Improves Neural Semantic Parsing

Daya Guo[1*], Yibo Sun[3*], Duyu Tang[2], Nan Duan[2], Jian Yin[1], Hong Chi[2], James Cao[2], Peng Chen[2], and Ming Zhou[2]
[1] The School of Data and Computer Science, Sun Yat-sen University.
Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, P.R.China
[2] Microsoft Research [3] Harbin Institute of Technology
{guody5@mail2,issjyin@mail}.sysu.edu.cn
{dutang,nanduan,hongchi,jcao,peche,mingzhou}@microsoft.com
ybsun@ir.hit.edu.cn

Abstract

We study how to learn a semantic parser of state-of-the-art accuracy with less supervised training data. We conduct our study on WikiSQL, the largest hand-annotated semantic parsing dataset to date. First, we demonstrate that question generation is an effective method that empowers us to learn a state-of-the-art neural network based semantic parser with thirty percent of the supervised training data. Second, we show that applying question generation to the full supervised training data further improves the state-of-the-art model. In addition, we observe that there is a logarithmic relationship between the accuracy of a semantic parser and the amount of training data.

1 Introduction

Semantic parsing aims to map a natural language utterance to an executable program (logical form) (Zelle and Mooney, 1996; Wong and Mooney, 2007; Zettlemoyer and Collins, 2007). Recently, neural network based approaches (Dong and Lapata, 2016; Jia and Liang, 2016; Xiao et al., 2016; Guu et al., 2017; Dong et al., 2018) have achieved promising performance in semantic parsing. However, neural network approaches are data hungry, which performances closely correlate with the volume of training data. In this work, we study the influence of training data on the accuracy of neural semantic parsing, and how to train a state-of-the-art model with less training data.

We conduct the study on WikiSQL (Zhong et al., 2017), the largest hand-annotated semantic parsing dataset which is larger than other datasets in terms of both the number of logical forms and the number of schemata. The task is to map a natural language question to a SQL query. We use a state-of-the-art end-to-end semantic parser based on neural networks (detailed in Section 3), and vary the number of supervised training instances. Results show that there is a logarithmic relationship between accuracy and the amount of training data, which is consistent with the observations in computer vision tasks (Sun et al., 2017).

We further study how to achieve state-of-the-art parsing accuracy with less supervised data, since annotating a large scale semantic parsing dataset requires funds and domain expertise. We achieve this through question generation, which generates natural language questions from SQL queries. Our question generation model is based on sequence-to-sequence learning. Latent variables (Cao and Clark, 2017) are introduced to increase the diversity of generated questions. The artificially generated question-SQL pairs can be viewed as pseudo-labeled data, which can be combined with a small amount of human-labeled data to train the semantic parser.

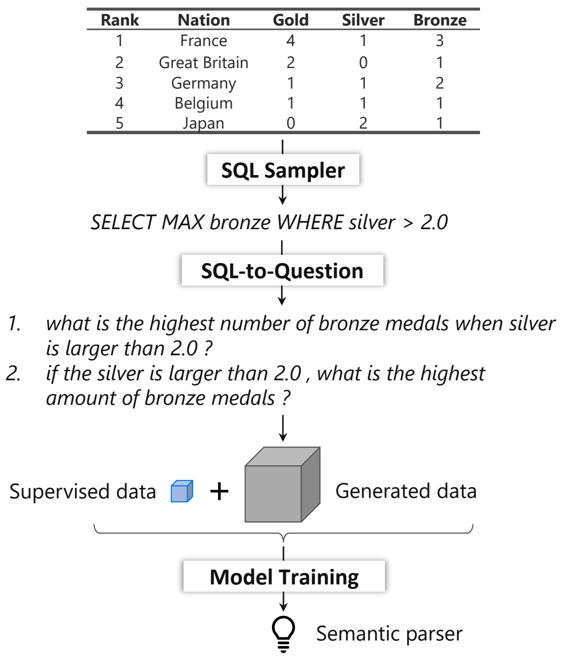
Results on WikiSQL show that the state-of-the-art logical form accuracy drops from 60.7% to 53.7% with only thirty percent of training data, while increasing to 61.0% when we combine the pseudo-labeled data generated from the question generation model. Applying the question generation model to full training data brings further improvements with 3.0% absolute gain. We further conduct a transfer learning experiment that applies our approach trained on WikiSQL to WikiTableQuestions (Pasupat and Liang, 2015). Results show that incorporating generated instances improves the state-of-the-art neural semantic parser (Krishnamurthy et al., 2017).

2 Overview of the Approach

Our task aims to map a question to a SQL query, which is executable over a table to yield the an-

[*] Work done while this author was an intern at Microsoft Research.

swer. Formally, the task takes a question q and a table t consisting of n column names and $n \times m$ cells as the input, and outputs a SQL query y. In this section, we describe an overview of our approach, which is composed of several components.



SQL Sampler
SELECT MAX bronze WHERE silver > 2.0
SQL-to-Question
1. what is the highest number of bronze medals when silver is larger than 2.0 ?
2. if the silver is larger than 2.0 , what is the highest amount of bronze medals ?
Supervised data
+
Generated data
Model Training
Semantic parser

Rank	Nation	Gold	Silver	Bronze
1	France	4	1	3
2	Great Britain	2	0	1
3	Germany	1	1	2
4	Belgium	1	1	1
5	Japan	0	2	1

Figure 1: An overview of our approach that improves semantic parsing with question generation.

Figure 1 gives an overview of our approach. First, given a table, a SQL query sampler is used to sample valid, realistic, and representative SQL queries. Second, a question generation component takes SQL queries as inputs to obtain natural language questions. Here, the question generation model is learnt from a small-scale supervised training data that consists of SQL-question pairs. Lastly, the generated question-SQL pairs are viewed as the pseudo-labeled data, which are combined with the supervised training data to train the semantic parser.

Since we conduct the experiment on WikiSQL dataset, we follow Zhong et al. (2017) and use the same template-based SQL sampler, as summarized in Table 1. The details about the semantic parser and the question generation model will be introduced in Sections 3 and Section 4, respectively.

3 Semantic Parsing Model

We use a state-of-the-art end-to-end semantic parser (Sun et al., 2018) that takes a natural language question as the input and outputs a SQL query, which is executed on a table to obtain the answer. To make the paper self-contained, we briefly describe the approach in this section.

Format of a Sampled SQL Query	
SELECT *agg_op* *agg_col* From *table* WHERE *cond1_col* *cond1_op* *cond1* AND *cond2_col* ...	
Sampling Rules	
Variable	Sampling range
agg_col or *cond_col*	The aggregation column *agg_col* and the condition column *cond_col* can be one of columns in the table.
agg_op	The aggregation operator *agg_op* can be empty or *COUNT*. If the type of *agg_col* is numeric, *agg_op* can additionally be one of *MAX* and *MIN*.
cond_op	The condition operator *cond_op* is $=$. If the type of *cond_col* is numeric, *cond_op* can additionally be one of $>$ and $<$.
cond	The condition value *cond* can be any cell value under the *cond_col*. If the type of *cond_col* is numeric, *cond* can be numerical value sampled from minimum value to maximum value in the *cond_col*.
Filter Rules	
1.The condition will be removed if doing the action does not change the execution result. 2.We only save the sampled SQL queries that produce non-empty result set.	

Table 1: The SQL sampler of (Zhong et al., 2017).

The semantic parser is abbreviated as **STAMP**, which is short for Syntax- and Table- Aware seMantic Parser. Based on the encoder-decoder framework, STAMP takes a question as the input and generates a SQL query. It extends pointer networks (Zhong et al., 2017; Vinyals et al., 2015) by incorporating three "channels" in the decoder, in which the column channel predicts column names, the value channel predicts table cells and the SQL channel predicts SQL keywords. An additional switching gate selects which channel to be used for generation. In STAMP, the probability of a token to be generated is calculated as Equation 1, where $p_z(\cdot)$ is the probability of the channel z_t to be chosen, and $p_w(\cdot)$ is the probability distribution of generating a word y_t from the selected channel.

$$p(y_t|y_{<t}, x) = \sum_{z_t} p_w(y_t|z_t, y_{<t}, x) p_z(z_t|y_{<t}, x) \tag{1}$$

Specifically, the encoder takes a question as the input, uses bidirectional RNN with GRU cells to compute the hidden states, and feeds the concatenation of both ends as the initial state of the decoder. The decoder has another GRU to calculate the hidden states.

Each channel is implemented with an attentional neural network. In the SQL channel, the input of the attention module includes the decoder hidden state and the embedding of the SQL keyword to be calculated (i.e. e_i^{sql}).

$$p_w^{sql}(i) \propto exp(W_{sql}[h_t^{dec}; e_i^{sql}]) \tag{2}$$

In the column channel, the vector of a column name includes two parts, as given in Equation 3. The first vector (h_i^{col}) is calculated with a bidirectional GRU because a column name might contain multiple words. The second vector is a question-aware cell vector, which is weighted averaged over the cell vectors belonging to the column. Cell vectors (h_i^{cell}) are also obtained by a bidirectional GRU. The importance of a cell is measured by the number of co-occurred question words, which is further normalized through a $softmax$ function to yield the final weight $\alpha_j^{cell} \in [0, 1]$.

$$p_w^{col}(i) \propto exp(W_{col}[h_t^{dec}; h_i^{col}; \sum_{j \in col_i} \alpha_j^{cell} h_j^{cell}]) \tag{3}$$

In the value channel, the model has two distributions and weighted average them as Equation 4. Similar to $p^{sql}(\cdot)$, a standard cell distribution $\hat{p}_w^{cell}(\cdot)$ is calculated over the cells belonging to the last predicted column name. They incorporate an additional probability distribution $\alpha^{cell}(\cdot)$ based on the aforementioned word co-occurrence. The hyper parameter λ is tuned on the dev set.

$$p_w^{cell}(j) = \lambda \hat{p}_w^{cell}(j) + (1 - \lambda)\alpha_j^{cell} \tag{4}$$

Please see more details on model training and inference in Sun et al. (2018).

4 Question Generation Model

In this section, we present our SQL-to-question generation approach, which takes a SQL query as the input and outputs a natural language question. Our approach is based on sequence-to-sequence learning (Sutskever et al., 2014; Bahdanau et al., 2015). In order to replicate rare words from SQL queries, we adopt the copying mechanism. In addition, we incorporate latent variables to increase the diversity of generated questions.

4.1 Encoder-Decoder

Encoder: A bidirectional RNN with gated recurrent unit (GRU) (Cho et al., 2014) is used as the encoder to read a SQL query $x = (x_1, ..., x_T)$. The forward RNN reads a SQL query in a left-to-right direction, obtaining hidden states $(\overrightarrow{h_1}, ..., \overrightarrow{h_T})$. The backward RNN reads reversely and outputs $(\overleftarrow{h_1}, ..., \overleftarrow{h_T})$. We then get the final representation $(h_1, ..., h_T)$ for each word in the query, where $h_j = [\overrightarrow{h_j}; \overleftarrow{h_j}]$. The representation of the source sentence $h_x = ([\overrightarrow{h_T}; \overleftarrow{h_1}])$ is used as initial hidden state of the decoder.

Decoder: We use a GRU with an attention mechanism as the decoder. At each time-step t, the attention mechanism obtains the context vector c_t that is computed same as the multiplicative attention (Luong et al., 2015). Afterwards, the concatenation of the context vector, the embedding of the previously predicted word y_{t-1}, and the last hidden state s_{t-1} is fed to the next step.

$$s_t = GRU(s_{t-1}, y_{t-1}, c_t) \tag{5}$$

After obtaining hidden states s_t, we adopt the copying mechanism that predicts a word from the target vocabulary or from the source sentence (detailed in Subsection 4.2).

4.2 Incorporating Copying Mechanism

In our task, the generated question utterances typically include informative yet low-frequency words such as named entities or numbers. Usually, these words are not included in the target vocabulary but come from SQL queries. To address this, we follow CopyNet (Gu et al., 2016) and incorporate a copying mechanism to select whether to generate from the vocabulary or copy from SQL queries.

The probability distribution of generating the t-th word is calculated as Equation 6, where $\psi_g(\cdot)$ and $\psi_c(\cdot)$ are scoring functions for generating from the vocabulary $\boldsymbol{\nu}$ and copying from the source sentence x, respectively.

$$p(y_t|y_{<t}, x) = \frac{e^{\psi_g(y_t)} + e^{\psi_c(y_t)}}{\sum_{v \in \boldsymbol{\nu}} e^{\psi_g(v)} + \sum_{w \in \mathbf{x}} e^{\psi_c(w)}} \tag{6}$$

The two scoring functions are calculated as follows, where W_g and W_c are model parameters, v_i is the one-hot indicator vector for y_i and h_i is the hidden state of word y_i in the source sentence.

$$\begin{aligned} \psi_g(y_i) &= v_i^T W_g s_t \\ \psi_c(y_i) &= tanh({h_i}^T W_c) s_t \end{aligned} \tag{7}$$

4.3 Incorporating Latent Variable

Increasing the diversity of generated questions is very important to improve accuracy, generalization, and stability of the semantic parser, since this increases the mount of training data and produces more diverse questions for the same intent. In this work, we incorporate stochastic latent variables (Cao and Clark, 2017; Serban et al., 2017) to the sequence-to-sequence model in order to increase question diversity.

Specifically, we introduce a latent variable $z \sim p(z)$, which is a standard Gaussian distribution $\mathcal{N}(0, I_n)$ in our case, and calculate the likelihood of a target sentence y as follows:

$$p(y|x) = \int_z p(y|z,x)p(z)\,dz \tag{8}$$

We maximize the evidence lower bound (ELBO), which decomposes the loss into two parts, including (1) the KL divergence between the posterior distribution and the prior distribution, and (2) a cross-entropy loss between the generated question and the ground truth.

$$\begin{aligned} logp(y|x) \geq &-D_{KL}(Q(z|x,y)||p(z)) \\ &+E_{z\sim Q}logp(y|z,x) \end{aligned} \tag{9}$$

The KL divergence in Equation 9 is calculated as follow, where n is the dimensionality of z.

$$\begin{aligned} &D_{KL}(Q(z|x,y)||p(z)) = \\ &-\frac{1}{2}\sum_{j=1}^{n}(1 + log(\sigma_j^2) - \mu_j^2 - \sigma_j^2) \end{aligned} \tag{10}$$

$Q(z|x,y)$ is a posterior distribution with Gaussian distribution. The mean μ and standard deviation σ are calculated as follows, where h_x and h_y are representations of source and target sentences in the encoder, respectively. Similar to h_x, h_y is obtained by encoding the target sentence.

$$\begin{aligned} \mu &= W_\mu[h_x; h_y] + b_\mu \\ log(\sigma^2) &= W_\sigma[h_x; h_y] + b_\sigma \end{aligned} \tag{11}$$

4.4 Training and Inference

At the training phase, we sample z from $Q(z|x,y)$ using the re-parametrization trick (Kingma and Welling, 2014), and concatenate the source last hidden state h_x and z as the initial state of the decoder. Since the model tends to ignore the latent variables by forcing the KL divergence to 0 (Bowman et al., 2016), we add a variable weight to the KL term during training. At the inference phase, the model will generate different questions by first sampling z from $p(z)$, concatenating h_x and z as the initial state of the decoder, and then decoding deterministically for each sample.

Here, we list our training details. We set the dimension of the encoder hidden state as 300, and the dimension of the latent variable z as 64. We use dropout with a rate of 0.5, which is applied to the inputs of RNN. Model parameters are initialized with uniform distribution, and updated with stochastic gradient decent. Word embedding values are initialized with Glove vectors (Pennington et al., 2014). We set the learning rate as 0.1 and the batch size as 32. We tune hyper parameters on the development, and use beam search in the inference process.

5 Experiment

We conduct experiments on the WikiSQL dataset[1] (Zhong et al., 2017). WikiSQL is the largest hand-annotated semantic parsing dataset which is an order of magnitude larger than other datasets in terms of both the number of logical forms and the number of schemata (tables). WikiSQL is built by crowd-sourcing on Amazon Mechanical Turk, including 61,297 examples for training, and 9,145/17,284 examples for development/testing. Each instance consists of a natural language question, a SQL query, a table and a result. Here, we follow Zhong et al. (2017) to use two evaluation metrics. One is logical form accuracy (Acc$_{lf}$), which measures the percentage of exact string match between the generated SQL queries and the ground truth SQL queries. Since different logical forms might obtain the same result, another metric is execution accuracy (Acc$_{ex}$), which is the percentage of the generated SQL queries that result in the correct answer.

5.1 Impact of Data Size

We study how the number of training instances affects the accuracy of semantic parsing.

In this experiment, we randomly sample 20 subsets of examples from the WikiSQL training data, incrementally increased by 3K examples (about 1/20 of the full WikiSQL training data). We use the same training protocol and report the accuracy of the STAMP model on the dev set. Results are given in Figure 2. It is not surprising that more training examples bring higher accuracy. Interestingly, we observe that both accuracies of the neural network based semantic parser grow logarith-

[1]`https://github.com/salesforce/WikiSQL`

Methods	Training Data	Dev		Test	
		Acc_{lf}	Acc_{ex}	Acc_{lf}	Acc_{ex}
Attentional Seq2Seq	100%	23.3%	37.0%	23.4%	35.9%
Aug.PntNet (Zhong et al., 2017)	100%	44.1%	53.8%	43.3%	53.3%
Aug.PntNet (re-implemented by us)	100%	51.5%	58.9%	52.1%	59.2%
Seq2SQL (Zhong et al., 2017)	100%	49.5%	60.8%	48.3%	59.4%
SQLNet (Xu et al., 2017)	100%	–	69.8%	–	68.0%
STAMP	30%	54.6%	69.7%	53.7%	68.9%
STAMP + QG	30%	61.6%	74.4%	61.2%	73.9%
STAMP	100%	61.5%	74.8%	60.7%	74.4%
STAMP + QG	100%	64.3%	76.5%	63.7%	75.5%

Table 2: Performance of different approaches on the WikiSQL dataset. The two evaluation metrics are logical form accuracy (Acc_{lf}) and execution accuracy (Acc_{ex}). The settings of the training data represent the proportion of supervised data we use.

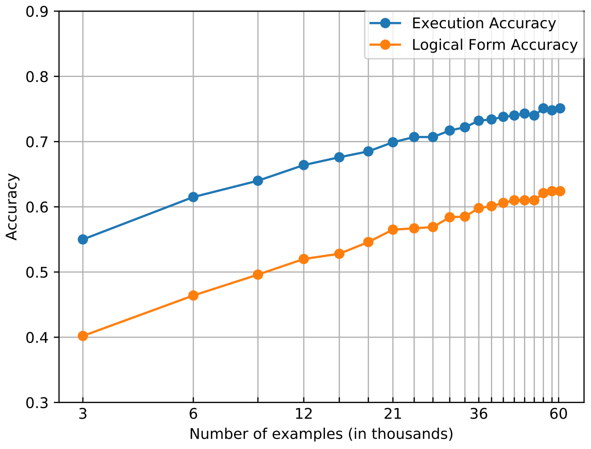


Figure 2: Semantic parsing accuracies of the STAMP model on WikiSQL. The x-axis is the training data size in log-scale, and the y-axis includes two evaluation metrics Acc_{lf} and Acc_{ex}.

mically as training data expands, which is consistent with the observations in computer vision tasks (Sun et al., 2017).

5.2 Model Comparisons

We report the results of existing methods on WikiSQL, and demonstrate that question generation is an effective way to improve the accuracy of semantic parsing. Zhong et al. (2017) implement several methods, including **Attentional Seq2Seq**, which is a basic attentional sequence-to-sequence learning baseline; **Aug.PntNet**, which is an augmented pointer network in which words of the target sequence come from the source sequence; and **Seq2SQL** which extends Aug.PntNet by further learning two separate classifiers for SELECT aggregator and SELECT column. Xu et al. (2017) develop **SQLNet**, which uses two separate models to predict SELECT and WHERE clauses, respectively, and introduce a sequence-to-set neural network to predict the WHERE clause. **STAMP** stands for the semantic parser which has been described in Section 3.

From Table 2, we can see that STAMP performs better than existing systems when trained on the full WikiSQL training dataset, achieving state-of-the-art execution accuracy and logical form accuracy on WikiSQL. We further conduct experiments to demonstrate the effectiveness of our question generation driven approach. We run the entire pipeline (STAMP+QG) with different percentages of training data. The second column "Training Data" in Table 2 and the x-axis in Figure 3 represent the proportion of WikiSQL training data we use for training the QG model and semantic parser. That is to say, STAMP +QG with 30% means that we sample 30% WikiSQL training data to train the QG model, and then combine QG generated data and exactly the same 30% WikiSQL training data we sampled before to train the semantic parser. In this experiment, we sample five SQL queries for each table in the training data, resulting in 43.5K SQL queries. Applying the QG model on these SQL queries, we get 92.8K SQL-question pairs. From Figure 3, we see that accuracy increases as the amount of supervised training data expands. Results show that QG empowers the STAMP model to achieve the same accuracy on WikiSQL dataset with 30% of the training data. Applying QG to the STAMP model under the full setting brings further improvements, resulting in new state-of-the-art accuracies.

Methods	Dev			Test		
	Acc_{sel}	Acc_{agg}	Acc_{where}	Acc_{sel}	Acc_{agg}	Acc_{where}
Aug.PntNet (re-implemented by us)	80.9%	89.3%	62.1%	81.3%	89.7%	62.1%
Seq2SQL (Zhong et al., 2017)	89.6%	90.0%	62.1%	88.9%	90.1%	60.2%
SQLNet (Xu et al., 2017)	91.5%	90.1%	74.1%	90.9%	90.3%	71.9%
STAMP	89.4%	89.5%	77.1%	88.9%	89.7%	76.0%
STAMP+QG	89.7%	90.1%	79.8%	89.1%	90.2%	79.0%

Table 3: Fine-grained accuracies on the WikiSQL dev and test sets. Logical form accuracy (Acc_{lf}) is evaluated on SELECT column (Acc_{sel}) , SELECT aggregator (Acc_{agg}), and WHERE clause (Acc_{where}), respectively. All these models are trained on the full WikiSQL training data.

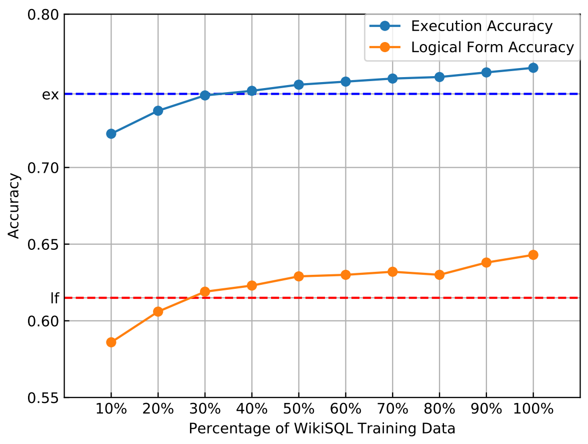


Figure 3: Accuracies of STAMP+QG with different portions of supervised data. Dashed lines are Acc_{lf} and Acc_{ex} of STAMP on the full training data.

5.3 Fine-grained Accuracies

Since SQL queries in WikiSQL consist of SELECT column, SELECT aggregator, and WHERE clause, we report fine-grained accuracies with regard to these aspects, respectively.

From Table 3, we observe that the main advantage of STAMP+QG over STAMP comes from the prediction of the WHERE clause, which is also the main challenge of the WikiSQL dataset. We further analyze STAMP and STAMP+QG on the WHERE clause by splitting the dev and test sets into three groups according to the number of conditions in the WHERE clause. From Table 4, we see that combining QG is helpful when the number of WHERE conditions is more than one. The main reason is that dominant instances in the WikiSQL training set have only one WHERE condition, as shown in Table 5, thus the model might not have memorized enough patterns for the other two limited-data groups. Therefore, the pseudo-labeled instances generated by our SQL sampler and QG approach are more precious to the limited-data groups (i.e #where =2 and #where≥3).

#where	STAMP		STAMP+QG	
	dev	test	dev	test
$= 1$	80.9%	80.2%	81.5%	80.9%
$= 2$	65.1%	65.4%	68.3%	66.9%
≥ 3	44.1%	48.2%	53.4%	51.9%

Table 4: Execution accuracy (Acc_{ex}) on different groups of WikiSQL dev and test sets.

#where	supervised data	generated data
$= 1$	69.1%	55.4%
$= 2$	24.1%	33.0%
≥ 3	6.1%	11.4%

Table 5: Distribution of the number of WHERE conditions in supervised and generated data.

5.4 Influences of Different QG Variations

To better understand how various components in our QG model impact the overall performance, we study different QG model variations. We use three evaluation metrics, including two accuracies and BLEU score (Papineni et al., 2002). The BLEU score evaluates the question generation.

Methods	Scale	BLEU	Acc_{lf}	Acc_{ex}
s2s	30%	20.6	59.0%	72.1%
s2s+lv	30%	22.1	60.0%	72.3%
s2s+cp	30%	29.6	60.8%	73.5%
s2s+cp+lv	30%	29.5	61.2%	73.9%
s2s	100%	26.0	62.6%	74.9%
s2s+lv	100%	26.3	63.0%	75.3%
s2s+cp	100%	31.5	63.2%	75.6%
s2s+cp+lv	100%	31.6	63.7%	75.5%

Table 6: Performances of different question generation variations.

SQL	SELECT COUNT 2nd leg WHERE aggregate = 7-2
Question (ground truth)	what is the total number of 2nd leg where aggregate is 7-2
Question (s2s + cp)	how many 2nd leg with aggregate being 7-2
Question (s2s + cp + lv)	(1) what is the total number of 2nd leg when the aggregate is 7-2 ? (2) how many 2nd leg with aggregate being 7-2 (3) name the number of 2nd leg for 7-2

Table 7: Generated examples from different question generation model variations.

Results are shown in Table 6, in which **s2s** represents the basic attentional sequence-to-sequence learning model (Luong et al., 2015), **cp** means the copying mechanism, and **lv** stands for the latent variable. We can see that incorporating a latent variable improves QG model performance, especially in limit-supervision scenarios. This is consistent with our intuition that the performance of the QG model is improved by incorporating the copying mechanism, since rare words of great importance mainly come from the input sequence.

To better understand the impact of incorporating a latent variable, we show examples generated by different QG variations in Table 7. We can see that incorporating a latent variable empowers the model to generate diverse questions for the same intent.

5.5 Transfer Learning on WikiTableQuestions

In this part, we conduct an extensional experiment on WikiTableQuestions[2] (Pasupat and Liang, 2015) in a transfer learning scenario to verify the effectiveness of our approach. WikiTableQuestions contains 22,033 complex questions on 2,108 Wikipedia tables. Each instance consists of a natural language question, a table and an answer. Following Pasupat and Liang (2015), we report development accuracy which is averaged over the first three 80-20 training data splits. Test accuracy is reported on the train-test data.

In this experiment, we apply the QG model learnt from WikiSQL to improve the state-of-the-art semantic parser (Krishnamurthy et al., 2017) on this dataset. Different from WikiSQL, this dataset requires question-answer pairs for training. Thus, we generate question-answer pairs by follow steps. We first sample SQL queries on the tables from WikiTableQuestions, and then use our QG model to generate question-SQL pairs. Afterwards, we obtain question-answer pairs by executing SQL queries. The generated question-answer pairs will be combined with the original WikiTableQuestions training data to train the model.

	Dev	Test
Pasupat and Liang (2015)	37.0%	37.1%
Neelakantan et al. (2016)	37.5%	37.7%
Haug et al. (2017)	-	38.7%
Zhang et al. (2017)	40.4%	43.7%
STAMP (WikiSQL)	-	14.5%
STAMP (WikiSQL) + QG	-	15.2%
NSP	41.9%	43.8%
NSP + QG	42.2%	44.2%

Table 8: Accuracy (Acc_{ex}) of different approaches on WikiTableQuestion dev and test sets.

Results are shown in Table 8, in which **NSP** is short for the state-of-the-art neural semantic parser (Krishnamurthy et al., 2017). Since the train-test data used in NSP is different from others, we retrain the NSP under the same protocol. **STAMP (WikiSQL)** means that the STAMP model trained on WikiSQL is directly tested on WikiTableQuestions. Despite applying QG slightly improves STAMP in this setting, the low accuracy reflects the different question distribution between these two datasets. In the supervised learning setting, we can see that incorporating QG further improves the accuracy of NSP from 43.8% to 44.2%.

5.6 Discussion

To better understand the limitations of our QG model, we analyze a randomly selected set of 100 questions. We observe that 27% examples do not correctly express the meanings of SQL queries, among which the majority of them miss information from the WHERE clause. This problem might be mitigated by incorporating a dedicated encoder/decoder that takes into account the SQL structure. Among the other 73% of examples that correctly express SQL queries, there are two po-

[2]https://nlp.stanford.edu/software/sempre/wikitable/

tential directions to make further improvements. The first direction is to leverage table information such as the type of a column name or column-cell correlations. For instance, without knowing that cells under the column name "*built*" are all building years, the model hardly predicts a question "*what is the average building year for superb?*" for "*SELECT AVG built WHERE name = superb*". The second direction is to incorporate common knowledge, which would help the model to predict *the earliest week* rather than *the lowest week*.

6 Related Work

Semantic Parsing. Semantic parsing is a fundamental problem in NLP that maps natural language utterances to logical forms, which could be executed to obtain the answer (denotation) (Zettlemoyer and Collins, 2005; Liang et al., 2011; Berant et al., 2013; Krishnamurthy and Kollar, 2013; Pasupat and Liang, 2016; Iyer et al., 2017). Existing works can be classified into three areas, including (1) the language of the logical form, e.g. first-order logic, lambda calculus, lambda dependency-based compositional semantics (lambda DCS) and structured query language (SQL); (2) the form of the knowledge base, e.g. facts from large collaborative knowledge bases, semi-structured tables and images; and (3) the supervision used for learning the semantic parser, e.g. question-denotation pairs and question-logical form pairs. In this work, we regard the table as the knowledge base, which is critical for accessing relational databases with natural language, and also for serving information retrieval for structured data. We use SQL as the logical form, which has a broad acceptance to the public. In terms of supervision, this work uses a small portion of question-logical form pairs to initialize the QA model and train the QG model, and incorporate more generated question-logical form pairs to further improve the QA model.

Question Generation Our work also relates to the area of question generation, which has drawn plenty of attention recently partly influenced by the remarkable success of neural networks in text generation. Studies in this area are classified based on the definition of the answer, including a sentence (Heilman, 2011), a topic word (Chali and Hasan, 2015), a fact (including a subject, a relation phrase and an object) from knowledge bases (Serban et al., 2016), an image (Mostafazadeh et al., 2016), etc. Recent studies in machine reading comprehension generate questions from an answer span and its context from the document (Du et al., 2017; Golub et al., 2017). Wang et al. (2015) first generate logical forms, and then use AMTurkers to paraphrase them to get natural language questions. Iyer et al. (2017) use a template-based approach based on the Paraphrase Database (Ganitkevitch et al., 2013) to generate questions from SQL. In this work, we generate questions from logical forms, in which the amount of information from two directions are almost identical. This differs from the majority of existing studies because a question typically conveys less semantic information than the answer.

Improving QA with QG This work also relates to recent studies that uses a QG model to improve the performance of a discriminative QA model (Wang et al., 2017; Yang et al., 2017; Duan et al., 2017; Konstas et al., 2017). The majority of these works generate a question from an answer, while there also exists a recent work (Dong et al., 2017) that generates a question from a question through paraphrasing. In addition, Tang et al. (2017) consider QA and QG as dual tasks, and further improve the QG model in a dual learning framework. These works fall into three categories: (1) regarding the artificially generated results as additional training instances (Yang et al., 2017; Golub et al., 2017); (2) using generated questions to calculate additional features (Duan et al., 2017; Dong et al., 2017); and (3) using the QG results as additional constraints in the training objectives (Tang et al., 2017). This work belongs to the first direction. Our QG approach takes a logical form as the input, and considers the diversity of generated questions by incorporating latent variables.

7 Conclusion

In this paper, we observe the logarithmic relationship between the accuracy of a semantic parser and the amount of training data, and present an approach that improves neural semantic parsing with question generation. We show that question generation helps us obtain a state-of-the-art neural semantic parser with less supervised data, and further improves the state-of-the-art model with full annotated data on WikiSQL and WikiTableQuesions datasets. In future work, we would like to make use of table information and external knowledge to improve our QG model. We also plan to apply the approach to other tasks.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (61472453, U1401256, U1501252, U1611264,U1711261,U1711262). Thanks to the anonymous reviewers for their helpful comments and suggestions.

References

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

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, 5, page 6.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. *conference on computational natural language learning*, pages 10–21.

Kris Cao and Stephen Clark. 2017. Latent variable dialogue models and their diversity. *arXiv preprint arXiv:1702.05962*.

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 the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43.

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

Li Dong, Chris Quirk, and Mirella Lapata. 2018. Confidence modeling for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 743–753. Association for Computational Linguistics.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1342–1352.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 866–874.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. Ppdb: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764.

David Golub, Po-Sen Huang, Xiaodong He, and Li Deng. 2017. Two-stage synthesis networks for transfer learning in machine comprehension. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 835–844.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640.

Kelvin Guu, Panupong Pasupat, Evan Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1051–1062.

Till Haug, Octavian-Eugen Ganea, and Paulina Grnarova. 2017. Neural multi-step reasoning for question answering on semi-structured tables. *arXiv preprint arXiv:1702.06589*.

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

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. *international conference on learning representations*.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural amr: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157, Vancouver, Canada. Association for Computational Linguistics.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526, Copenhagen, Denmark. Association for Computational Linguistics.

Jayant Krishnamurthy and Thomas Kollar. 2013. Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association for Computational Linguistics*, 1:193–206.

Percy Liang, Michael I Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 590–599.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *empirical methods in natural language processing*, pages 1412–1421.

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, Martin Abadi, Andrew McCallum, and Dario Amodei. 2016. Learning a natural language interface with neural programmer. *arXiv preprint arXiv:1611.08945*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480.

Panupong Pasupat and Percy Liang. 2016. Inferring logical forms from denotations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 23–32.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

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.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301.

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

Yibo Sun, Duyu Tang, Nan Duan, Jianshu Ji, Guihong Cao, Xiaocheng Feng, Bing Qin, Ting Liu, and Ming Zhou. 2018. Semantic parsing with syntax- and table-aware sql generation. *arXiv preprint arXiv:1804.08338*.

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*.

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

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*.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1332–1342.

Yuk Wah Wong and Raymond J Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Annual Meeting-Association for computational Linguistics*, 1, page 960.

Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1341–1350, Berlin, Germany. Association for Computational Linguistics.

Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

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

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*, pages 658–666.

Luke S Zettlemoyer and Michael Collins. 2007. Online learning of relaxed ccg grammars for parsing to logical form. In *EMNLP-CoNLL*, pages 678–687.

Yuchen Zhang, Panupong Pasupat, and Percy Liang. 2017. Macro grammars and holistic triggering for efficient semantic parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1225–1234, Copenhagen, Denmark. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.