Data-Anonymous Encoding for Text-to-SQL Generation

Zhen Dong[1]*, **Shizhao Sun**[2], **Hongzhi Liu**[1], **Jian-Guang Lou**[2] and **Dongmei Zhang**[2]
[1]Peking University [2]Microsoft Research
{zhendong,liuhz}@pku.edu.cn
{shizsu,jlou,dongmeiz}@microsoft.com

Abstract

On text-to-SQL generation, the input utterance usually contains lots of tokens that are related to column names or cells in the table, called *table-related tokens*. These table-related tokens are troublesome for the downstream neural semantic parser because it brings complex semantics and hinders the sharing across the training examples. However, existing approaches either ignore handling these tokens before the semantic parser or simply use deterministic approaches based on string-match or word embedding similarity. In this work, we propose a more efficient approach to handle table-related tokens before the semantic parser. First, we formulate it as a sequential tagging problem and propose a two-stage anonymization model to learn the semantic relationship between tables and input utterances. Then, we leverage the implicit supervision from SQL queries by policy gradient to guide the training. Experiments demonstrate that our approach consistently improves performances of different neural semantic parsers and significantly outperforms deterministic approaches.

1 Introduction

Interacting with relational databases or tables through natural language is an important and challenging problem (Androutsopoulos et al., 1995; Li and Jagadish, 2014; Pasupat and Liang, 2015; Zhong et al., 2017; Yu et al., 2018c). To tackle this problem, the neural semantic parser has been widely studied (Dong and Lapata, 2016; Jia and Liang, 2016; Herzig and Berant, 2017; Dong and Lapata, 2018), which automatically maps the input natural language into the logical form (e.g., SQL query) following the typical encoder-decoder structure (Hochreiter and Schmidhuber, 1997; Bahdanau et al., 2015; Vinyals et al., 2015).

Semantic parsing usually tackles two kinds of problems (Goldman et al., 2018; Herzig and Berant, 2018), i.e., the *lexical problem* and the *structural problem*. On text-to-SQL generation, the lexical problem refers to mapping tokens in input utterances to constants in SQL queries, e.g., the column names or cells in SQL queries. The structural problem refers to mapping intentions conveyed by input utterances to operators in SQL queries, e.g., the aggregators or the existence of WHERE clause in SQL queries. Intuitively, the lexical problem can be formulated as a sequential tagging problem, called *anonymization*, where each token in the input utterance will be tagged as being related to a column name, a cell or nothing. For ease of reference, we call the tokens in input utterances related to column names or cells *table-related tokens*, and the tagged utterance *anonymous utterance*.

If the lexical problem can be reduced before the neural semantic parsing, the training difficulties will be greatly alleviated. The reason is two-fold. First, by anonymizing table-related tokens in the input utterance before the neural semantic parser, we can conceal the complex semantics of table-related tokens. Second, different input utterances, which have different table-related tokens but the same structure, can be reduced to the same anonymous utterance before they are fed into the parser. This will result in sharing across training data and thus alleviate training difficulties (Goldman et al., 2018; Herzig and Berant, 2018). For example, in Figure 1a and 1b, input utterances (denoted as x) seemingly ask different questions about country and college team, but they can be reduced to a similar anonymous utterance (denoted as $\tilde{x}$) by replacing the token related to the column name as c and the token related to the cell as v.

However, reducing the lexical problem before the neural semantic parser is far from being well-studied on text-to-SQL generation although it

* This work was done when the author was visiting Microsoft Research Asia.

x	What country does the goaltender come from?
$\tilde{x}$	What c does the v come from?
y	SELECT Nationality WHERE Position=Goaltender

Pick #	Player	Position	Nationality	...
83	Wayne Wood	Goaltender	Canada	...
...	...	...	...	...

(a) Example A

x	What college team did Ryan Golden come from?
$\tilde{x}$	What c did v come from?
y	SELECT College/junior/club team WHERE Player=Ryan Golden

Pick	Player	Position	...	College/junior /club team
181	Ryan Golden	Left Wing	...	HS-Massachusetts
...	...	...	...	...

(b) Example B

x	What school did player number 6 come from?
$\tilde{x}$	What c did player c v come from?
y	SELECT School/Club Team WHERE No.=6

Player	No.	Nationality	...	School/Club Team
Alan Anderson	6	United States	...	Michigan State
...	...	...	...	...

(c) Example C

Figure 1: Examples for anonymizing table-related tokens on WikiSQL. For each subfigure, the upper box shows the utterance and the SQL query, while the lower box shows the table. In the upper box, x denotes the input utterance, $\tilde{x}$ denotes the anonymous utterance and y denotes the SQL query. In anonymous utterance $\tilde{x}$, we use c and v to replace the tokens that are related to column names and cells respectively. In addition, we use the color to indicate the correspondence of column names or cells among x, $\tilde{x}$ and the table.

has been demonstrated to be helpful on other tasks (Goldman et al., 2018; Herzig and Berant, 2018). First, most approaches ignores reducing the lexical problem (Zhong et al., 2017; Xu et al., 2017; Dong and Lapata, 2018; Shi et al., 2018; Wang et al., 2018a; Hwang et al., 2019). Without conducting anonymization before the neural semantic parser, these approaches cannot receive the aforementioned benefit, although they may have modules to predict the presence of column names and cells inside the parser. Second, a few studies have tried the anonymization by deterministic approaches, which compare the similarity based on string-match or word embedding between the input utterance and the table (Wang et al., 2018b; Yu et al., 2018a). These approaches cannot fully understand the semantic relationship between input utterances and tables, and ignore the relationship with components of SQL queries. For example, in Figure 1c, although the token 'player' is similar to the column name 'Player', it should not be tagged as a table-related token because the token "player" is just a non-sense demonstrative pronoun and the column 'Player' is not related to the SQL query.

To this end, we propose to use the learning-based approach to reduce the lexical problem, i.e., anonymize the table-related tokens. First, we propose a two-stage anonymization model to learn the semantic relationship between input utterances and tables. Then, considering that there is no labeled data for the anonymization, we propose to extract the set of column names and cells appearing in the SQL query and use this set as the supervision. Another benefit of leveraging such implicit supervision is that we can make the anonymization model consider the relationship with the components of the SQL query when it learns the semantic relationship, and thus avoid suffering the same disadvantage as deterministic approaches. Furthermore, to bridge the gap that our model is a sequential tagging model while the supervision extracted from the SQL query is an unordered set, we leverage the policy gradient (Williams, 1992) to train the model. Moreover, we train the anonymization model and the neural semantic parser as whole by a varational inference-like framework with the anonymous utterance as the hidden variable.

Experimental results demonstrate that our approach can consistently improve performances of different neural semantic parsers and significantly outperform typical deterministic approach on the anonymization problem.

2 Related Work

Semantic Parsing. Semantic parsing aims to map natural languages into executable programs. In the area of semantic parsing, the programs could be in various types, e.g., λ-calculus (Zettlemoyer and Collins, 2005), Python (Oda et al., 2015), SQL (Zhong et al., 2017), etc; the source of the knowledge can also be different, e.g., the knowledge base, the table, the image (Suhr et al., 2017), etc; and the supervision can take different forms as well, e.g., question-denotation pairs (Pasupat and Liang, 2015), question-program pairs, etc. In this work, we focus on text-to-SQL generation with the table as the source of the knowledge and with question-SQL query pairs as supervision.

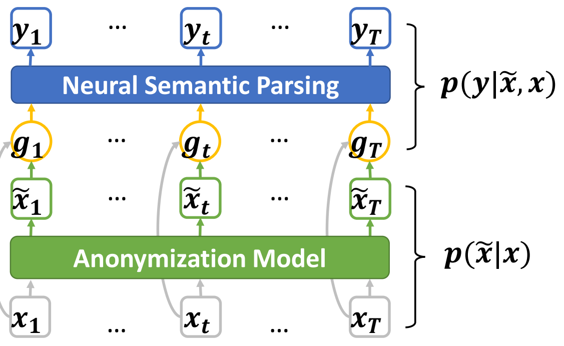
Reducing Lexical Problem. On tasks other than text-to-SQL generation, some studies have tried to reduce lexical problem by anonymizing tokens that are related to the program constants. Goldman et al. (2018) lifts tokens in utterances to an abstract form by referring to fixed mappings on visual reasoning task. Herzig and Berant (2018) employs a rule-based method to transform content words in utterances to an abstract representation. Although these studies inspire us to consider reducing lexical problem on text-to-SQL generation, the rules they employed cannot be directly applied to text-to-SQL generation.

On text-to-SQL generation, a few studies tried to anonymize table-related tokens although they do not aim at reducing lexical problem. To better understand rare entities, Yu et al. (2018a) used string match to recognize column names. To learn a domain-transferable semantic parser, Wang et al. (2018b) detected column names by measuring the closeness of edit distance and word embedding. As discussed in Section 1, these deterministic approaches can not fully understand the semantic relationship between input utterances and tables, and ignore the relationship with components of SQL queries.

Entity Linking. Our approach can be regarded as an implementation of entity linking on the concrete task, because general entity linking (Shen et al., 2015; Kolitsas et al., 2018) approaches fail to handle particular challenges in our scenario. On the one hand, there are lots of cases cannot be handled by the deterministic entity linking method which only relies on measuring the similarity; on the other hand, there is no labeled data for the learning-based entity linking method.

3 Problem Formulation

Denote the input utterance as $x = x_1 \dots x_{|x|}$ and the corresponding SQL query as $y = y_1 \dots y_{|y|}$. We formulate the lexical problem as a sequential tagging problem. Formally, for each token x_t in the input utterance, we give it a tag $\tilde{x}_t \in \{\mathrm{COL}_1, \dots, \mathrm{COL}_K, \mathrm{CELL}, \mathrm{UNK}\}$, where COL_k represents that x_t is related to the k-th column name in the table, CELL represents that x_t is related to a cell in the table and UNK represents that



y_1 ... y_t ... y_T
Neural Semantic Parsing
$p(y|\tilde{x},x)$
g_1 ... g_t ... g_T
$\tilde{x}_1$ ... $\tilde{x}_t$ ... $\tilde{x}_T$
Anonymization Model
$p(\tilde{x}|x)$
x_1 ... x_t ... x_T

Figure 2: Framework.

x_t is not related to the table. Here K is the number of column names in the table. Note that indexes of column names cannot be ignored in the algorithm to better leverage the implicit supervision from SQL queries (see Section 4.2) although it is ignored when giving examples in Figure 1 and Table 4 for ease of read. For ease of reference, we call this sequential tagging problem *anonymization* and the tagged sequence $\tilde{x} = \tilde{x}_1 \dots \tilde{x}_{|x|}$ *anonymous utterance*.

The typical neural semantic parser aims to estimate $p(y|x)$. In our work, we decompose $p(y|x)$ into two processes (as shown in Figure 2), i.e.,

$$p(y|x) = \sum_{\tilde{x}} p(y|x,\tilde{x})\, p(\tilde{x}|x)\,. \tag{1}$$

Specifically, one process is to learn the anonymous utterance $\tilde{x}$ given the input utterance x for the purpose of reducing lexical problem, i.e.,

$$p(\tilde{x}|x) = \prod_t p(\tilde{x}_t|x)\,, \tag{2}$$

while the other process is to learn the neural semantic parser with anonymous utterance $\tilde{x}$ as the additional input, i.e.,

$$p(y|x,\tilde{x}) = \prod_t p(y_t|y_{<t},x,\tilde{x})\,. \tag{3}$$

In the following, we will discuss how to learn the anonymous utterance (Eqn (2)) and how to learn the neural semantic parser with anonymous utterance as additional input (Eqn (1)) in detail.

4 Approach

4.1 Anonymization Model

The anonymization is a structural problem by nature. First, we need to differentiate whether the token is related column names, cells or not related to

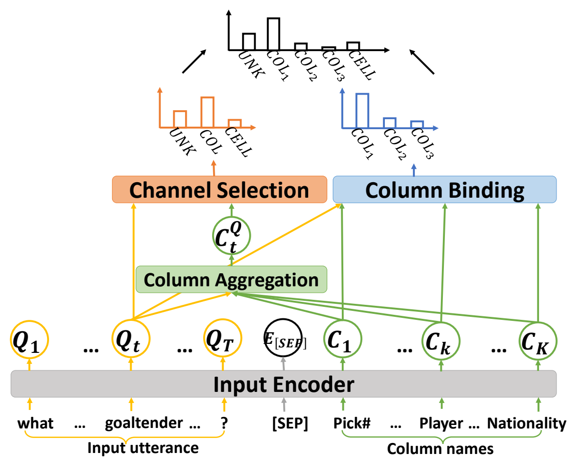


Figure 3: Illustration of the anonymization model when the table content is not available. We take the t-th token in the input utterance as the example.

the table. Second, we need to further recognize the concrete column/cell that the token is related to. Therefore, we design a model that uses two stages, i.e., *channel selection* stage and the *column/cell binding* stage, to tackle these two subproblems respectively. The probability distributions produced by these two stages together determine the result,

$$p\left(\tilde{x}_t|x\right) = \sum_{a_t} p_{\text{channel}}\left(a_t|x\right) p_{\text{binding}}\left(\tilde{x}_t|a_t,x\right), \tag{4}$$

where $a_t \in \{\text{COL}, \text{CELL}, \text{UNK}\}$ is the selected channel indicating that the token is related to column names, cells or nothing respectively, $p_{\text{channel}}\left(\cdot\right)$ is the probability produced by the channel selection stage, and $p_{\text{binding}}\left(\cdot\right)$ is the probability produced by the column/cell binding stage.

For ease of reference, we call the two-stage model *anonymization model*, and illustrate it in Figure 3. In the following, we introduce details of its different components.

Input Encoder. We leverage BiLSTM or BERT (Devlin et al., 2018) to encode both the input utterance and the table. Due to the privacy problem, the table content is not allowed to be used on most text-to-SQL tasks (Zhong et al., 2017; Yu et al., 2018c). Therefore, we use the concatenation of the embedding of the tokens in the input utterance, the embedding of the separator and the embedding of the column names in the table as the input for BiLSTM or BERT. We denote the output of input encoder as $\{Q_1, \ldots, Q_T, E_{[\text{SEP}]}, C_1, \ldots, C_K\}$, where $Q_t, t \in \{1, \ldots, T\}$ is the vector for the encoding of the t-th token in the input utterance, $E_{[\text{SEP}]}$ is the vector for the encoding of the separator, and $C_k, k \in \{1, \ldots, K\}$ is the vector for the encoding of the k-th column name in the table. Note that we run a BiLSTM or BERT between column names instead of over each column name. The reason is that Yu et al. (2018a) have shown that although the order of column names does not matter, running a BiLSTM or BERT between colomn names can capture relationships between them, which will benefit the accuracy and the training time.

Channel Selection. We implement a linear gate with encodings of the input utterance and aggregated encodings of column names as input:

$$p_{\text{channel}}\left(a_t|x\right) = \text{softmax}\left(W_{\text{gate}}\left[Q_t; C_t^Q\right]\right), \tag{5}$$

where W_{gate} stands for learnable parameters, C_t^Q is the aggregated encoding of column names, and $[Q_t; C_t^Q]$ is the concatenation of Q_t and C_t^Q.

Specifically, to obtain the most relevant column names for the t-th token in the input utterance, we leverage attention mechanism (Bahdanau et al., 2015) towards column names $\{C_1, \ldots, C_K\}$ to compute the aggregated encoding of the column names C_t^Q, i.e., $C_t^Q = \sum_{k=1}^K \alpha_t^k C_k$, where α_t^k is obtained through $u_t^k = v_1 \tanh\left(W_1 Q_t + W_2 C_k\right)$ and $\alpha_t^k = \text{softmax}\left(u_t^k\right)$, $k \in \{1, \ldots, K\}$. Here, v_1, W_1 and W_2 are learnable parameters.

Column/Cell Binding[1]**.** The probability distribution generated by this stage, i.e., $p_{\text{binding}}(\cdot)$, can be categorized into three types, i.e., $p_{\text{binding}}^{\text{COL}}(\cdot)$, $p_{\text{binding}}^{\text{CELL}}(\cdot)$ and $p_{\text{binding}}^{\text{UNK}}(\cdot)$, corresponding to the three channels in the first stage respectively. Obviously, $p_{\text{binding}}^{\text{UNK}}\left(\tilde{x}_t|a_t,x\right) = 1$.

For column names, we produce a probability distribution over the K columns in the table through measuring the relevance between the t-th token and the k-th column name:

$$p_{\text{binding}}^{\text{COL}_k}\left(\tilde{x}_t|a_t,x\right) = \text{softmax}\left(z_t^k\right), \tag{6}$$
$$z_t^k = v_2 \tanh\left(W_3 Q_t + W_4 C_k\right), \; k \in \{1, \ldots, K\},$$

where v_2, W_3 and W_4 are learnable parameters.

[1]For the situation that the input utterance contains table names or the table content is allowed to be used, we can handle table names or cells by the similar way of column names.

For cells, considering that table content is not available due to the privacy problem, we simply predict a substring from the input utterance. Specifically, we follow the longest match principle to merge consecutive tokens in the input utterance, which are labeled as CELL by the channel selection stage, to one cell. Therefore, the generated by column/cell binding stage is simply set as 1, i.e., $p^{\text{CELL}}_{\text{binding}}(\tilde{x}_t|a_t, x) = 1$.

4.2 Implicit Supervision from SQL Queries

To train the anonymization model, the ground truth for the anonymous utterance is indispensable. However, there is no such labeled data, and manually labeling the whole training data for each text-to-SQL task is unrealistic, especially when the amount of training data is tremendous.

To tackle this problem, we propose to extract the set of column names and cells appearing in the SQL query and use this set as the supervision to guide the training of the anonymization model. Another benefit of such approach is that we can make our model consider the relationship with components of the SQL query, and thus avoid suffering the same trouble as deterministic approaches. Concretely, for each SQL query y, we denote the set of column names and cells appearing in it as S_{SQL}, which consists of three parts, i.e., $S_{\text{SQL}} = S_{\text{sel\_col}} \cup S_{\text{other\_col}} \cup S_{\text{cell}}$:
1. $S_{\text{sel\_col}}$ is the set of column names appearing in the SELECT clause;
2. $S_{\text{other\_col}}$ is the set of column names appearing in other clauses;
3. S_{cell} is the set of cells appearing in the whole SQL query.

4.3 Policy Gradient for Sequential Tagging

Maximizing Expected Reward. Ideally, if we have the ground truth, we can train the anonymization model by minimizing the gap (e.g., KL divergence) between the predicted probabilities and the ground truth. Unfortunately, it is infeasible because the implicit supervision from SQL queries (i.e., S_{SQL}) takes a form of the unordered set while the anonymization model faces a sequential tagging task. To address this problem, we propose to encourage the set of column names and cells appearing in the predicted anonymous utterance, denoted as S_{pred}, to be similar to that extracted from the SQL query, i.e., S_{SQL}. To this end, we define a reward of the predicted anonymous utterance $r(\tilde{x})$ as the similarity between S_{pred} and S_{SQL}, and then train the anonymization model by maximizing expected reward,

$$J_{\text{ER}}(\theta) = \sum_{(x,\tilde{x})\in\mathcal{D}} \mathbb{E}_{\tilde{x}\sim p_\theta(\cdot|x)} r(\tilde{x}), \quad (7)$$

where $\mathcal{D}$ represents the set of training pairs.

However, directly computing expected reward requires traversing all the possible anonymous utterance $\tilde{x}$, which is unrealistic. Therefore, we leverage Monte Carlo estimate as the approximation of the expected reward. Concretely, we sample N anonymous utterances following probability distribution generated by the anonymization model, and then average the reward of the samples to estimate the expected reward,

$$J_{\text{ER}}(\theta) \approx \sum_{(x,\tilde{x})\in\mathcal{D}} \sum_{j=1}^{N} \frac{1}{N} r(\tilde{x}^j), \quad (8)$$

where $\tilde{x}^j$ denotes the j-th sample and $\tilde{x}^j \sim p_\theta(\cdot|x)$.

To maximize the above objective function by gradient descent, we employ REINFORCE (Williams, 1992) method, i.e,

$$\nabla_\theta J_{\text{ER}}(\theta) \approx \sum_{(x,\tilde{x})\in\mathcal{D}} \sum_{j=1}^{N} \frac{1}{N} r(\tilde{x}^j) \nabla_\theta \log p_\theta\left(\tilde{x}^j|x\right). \quad (9)$$

Measurement of Reward. For ease of reference, we decompose S_{pred} into two parts, i.e., $S_{\text{pred}} = S_{\text{pred\_col}} \cup S_{\text{pred\_cell}}$, where $S_{\text{pred\_col}}$ and $S_{\text{pred\_cell}}$ represent the set of column names and the set of cells appearing in the predicted anonymous utterance respectively.

The measurement of the reward $r(\tilde{x})$ is designed based on the similarity between S_{pred} and S_{SQL} by referring to the following principles:

1. The predicted anonymous utterance should contain the column names in the SELECT clause of the SQL query, i.e., $S_{\text{sel\_col}} \subset S_{\text{pred\_col}}$. The anonymization model will be punished when it misses the column names in $S_{\text{sel\_col}}$. This is motivated by our preliminary analysis that almost every column name in the SELECT clause has the corresponding token in the input utterance.

2. The column names in the predicted anonymous utterance should be at least a subset of the column names appearing in the SQL query, i.e., $S_{\text{pred\_col}} \subseteq S_{\text{sel\_col}} \cup S_{\text{other\_col}}$. Unlike $S_{\text{sel\_col}}$, the column names appearing in other clauses of the

SQL query may possible not have corresponding tokens in the input utterance. For example, in Figure 1a, the column name 'Position' does not have corresponding tokens in the input utterance. Therefore, it is unreasonable to strictly force column names in the predicted anonymous utterance to be the same as that appearing in the SQL query. Instead, it is better to punish the anonymization model when it predicts column names outside the set of column names appearing in the SQL query.

3. The cell names in the predicted anonymous utterance should be the same as that appearing in the SQL query, i.e., $S_{\text{pred\_cell}} = S_{\text{cell}}$. If there is any missing or extra cell in $S_{\text{pred\_cell}}$, the anonymization model will be punished.

According to above principles, we design the reward $r(\tilde{x})$ as,
1. if $S_{\text{sel\_col}} \subseteq S_{\text{pred\_col}}$, $S_{\text{pred\_cell}} = S_{\text{cell}}$ and $S_{\text{pred\_col}} \subseteq S_{\text{sel\_col}} \cup S_{\text{other\_col}}$ are all true, $r(\tilde{x}) = 1$;
2. if one of $S_{\text{sel\_col}} \nsubseteq S_{\text{pred\_col}}$, $S_{\text{pred\_cell}} \neq S_{\text{cell}}$ and $S_{\text{pred\_col}} \nsubseteq S_{\text{sel\_col}} \cup S_{\text{other\_col}}$ is true, $r(\tilde{x}) = -1$.

4.4 Training and Inference

To train the anonymization model and the parser as a whole, we regard the anonymous utterance $\tilde{x}$ as the hidden variable for the neural semantic parser $p(y|x)$. Then, maximizing the log likelihood of $p(y|x)$ is equivalent to maximizing its Evidence Lower BOund (ELBO) (Kim et al., 2018), i.e.,

$$\text{ELBO}\,(\varphi, \theta) = \mathbb{E}_{q_\theta(\tilde{x}|x)}\left[\log \frac{p_\varphi(y, \tilde{x}|x)}{q_\theta(\tilde{x}|x)}\right] \quad (10)$$

One popular strategy to maximizing ELBO is the coordinate ascent. Specifically, it iteratively executes following two steps (Neal and Hinton, 1998): 1) *variational E-step*, which maximizes ELBO w.r.t. θ keeping φ fixed, i.e., $\theta^\star = \arg\min_\theta \mathbb{KL}\left(q_\theta(\tilde{x}|x) \| p_\varphi(\tilde{x}|y, x)\right)$; and 2) *variational M-step*, which maximizes ELBO w.r.t. φ keeping θ fixed, i.e., $\varphi^\star = \arg\max_\varphi \mathbb{E}_{q_\theta(\tilde{x}|x)}\left[\log p_\varphi(y, \tilde{x}|x)\right]$.

In our scenario, φ and θ refer to the learnable parameters in the neural semantic parser and the anonymization model respectively. For variational E-step, it usually finds the best variational approximation to the true posterior (Neal and Hinton, 1998). As discussed in Section 4.2, the true posterior we can obtain is in the form of the unordered set. Thus, we actually minimize the the expected reward, i.e., $J_{\text{ER}}(\theta)$, instead of the KL divergence (see Section 4.3). For variational M-step, to save training time, we simply sample one example greedily, which approximates to maximizing log likelihood of $p_\varphi(y, \tilde{x}|x)$, i.e., $J_{\text{MLE}}(\varphi)$. Moreover, since performing coordinate ascent on the entire dataset is too expensive, the variational E-step and M-step are usually performed over mini-batches (Hoffman et al., 2013). To further improve time efficiency, we optimize objectives of the variational E-step and M-step simultaneously instead of alternatively. Thus, the actual objective is,

$$J(\varphi, \theta) = \underbrace{\sum_{(x,y)\in\mathcal{D}} \log p_\varphi(y|\tilde{x}, x)}_{J_{\text{MLE}}(\varphi)} + \underbrace{\sum_{(x,\tilde{x})\in\mathcal{D}} \mathbb{E}_{\tilde{x}\sim q_\theta(\cdot|x)} r(\tilde{x})}_{J_{\text{ER}}(\theta)}, \quad (11)$$

where $J_{\text{MLE}}(\varphi)$ is the log likelihood of the generated SQL query given the input utterance and the anonymous utterance, and $J_{\text{ER}}(\theta)$ is the expected reward of the anonymous utterance given the input utterance (see Eqn (8) for details.)

Specifically, when optimizing $J_{\text{MLE}}(\varphi)$, we use the concatenation of the encoding of the input utterance (denoted as $e(x_t)$) and the encoding of the anonymous utterance (denoted as $h(\tilde{x}_t)$) as the input for the neural semantic parser, i.e., $g_t = [e(x_t); h(\tilde{x}_t)]$, $t \in \{1, \dots, T\}$ (see Figure 2). For $e(x_t)$, it is determined by the parser itself. For $h(\tilde{x}_t)$, it is concatenated by two parts: 1) embedding of the channel name, i.e., COL, CELL and UNK, 2) embedding of index k when the channel is COL, indicating that the t-th token is related to the k-th column name in the table.

At test time, the prediction for input utterance x is obtained by $\hat{x} = \arg\max_{\tilde{x}'} p\left(\tilde{x}'|x\right)$ and $\hat{y} = \arg\max_{y'} p\left(y'|\hat{x}, x\right)$.

5 Experiments

5.1 Experimental Setup

We conduct experiments on the WikiSQL (Zhong et al., 2017) and Spider (Yu et al., 2018c). Each example consists of a natural language question, a SQL query and a table. On WikiSQL, one question is only related to one table; while on Spider, one question is usually related to multiple tables.

Our model is implemented in PyTorch (Paszke et al., 2017). The type of the input encoder in the anonymization model (i.e., BiLSTM or BERT) is set the same as that in the concrete parser. Embedding vectors of the anonymous utterance are initiated by GloVe (Pennington et al., 2014). We

Method	Dev (%)		Test (%)	
	ACC_{QM}	ACC_{EX}	ACC_{QM}	ACC_{EX}
Seq2SQL (Zhong et al., 2017)[1]	53.5	62.1	51.6	60.4
Seq2SQL + DAE	$\mathbf{60.0}(\mathbf{+6.5})$	$\mathbf{67.5}(\mathbf{+5.4})$	$\mathbf{58.0}(\mathbf{+6.4})$	$\mathbf{66.1}(\mathbf{+5.7})$
SQLNet (Xu et al., 2017)	63.2	69.8	61.3	68.0
SQLNet + DAE	$\mathbf{64.6}(\mathbf{+1.4})$	$\mathbf{71.1}(\mathbf{+1.3})$	$\mathbf{63.4}(\mathbf{+2.1})$	$\mathbf{70.0}(\mathbf{+2.0})$
TypeSQL (Yu et al., 2018a)	68.0	74.5	66.7	73.5
TypeSQL + DAE	$\mathbf{68.6}(\mathbf{+0.6})$	74.5	$\mathbf{67.8}(\mathbf{+1.1})$	$\mathbf{74.1}(\mathbf{+0.6})$
SQLova (Hwang et al., 2019)[2]	80.3	85.8	79.4	85.2
SQLova + DAE	$\mathbf{81.3}(\mathbf{+1.0})$	$\mathbf{86.4}(\mathbf{+0.6})$	$\mathbf{80.5}(\mathbf{+1.1})$	$\mathbf{86.1}(\mathbf{+0.9})$
– Supervision	81.0	86.1	80.0	85.4
– Co-training	80.6	86.0	79.8	85.3

[1] For Seq2SQL, we use Xu et al. (2017)'s version because it achieves better performance.
[2] For SQLova, we use BERT-Base as the input encoder due to limited computation resources.

Table 1: Performance improvement of the neural semantic parser on WikiSQL.

Method	Easy (%)	Medium (%)	Hard (%)	Extra Hard (%)	All (%)
SyntaxSQLNet (Yu et al., 2018b)	38.4	15.0	16.1	3.5	18.9
SyntaxSQLNet + DAE	$\mathbf{39.6}(\mathbf{+1.2})$	$\mathbf{18.2}(\mathbf{+3.2})$	$\mathbf{20.7}(\mathbf{+4.6})$	$\mathbf{7.6}(\mathbf{+4.1})$	$\mathbf{22.1}(\mathbf{+3.2})$
SyntaxSQLNetAug (Yu et al., 2018b)	44.4	23.0	23.0	2.9	24.9
SyntaxSQLNetAug + DAE	$\mathbf{44.8}(\mathbf{+0.4})$	$\mathbf{27.0}(\mathbf{+4.0})$	$\mathbf{24.1}(\mathbf{+1.1})$	$\mathbf{5.9}(\mathbf{+3.0})$	$\mathbf{27.4}(\mathbf{+2.5})$

Table 2: Performance improvement of the neural semantic parser on Spider with different hardness levels.

	#Total	#Distinct x	#Distinct $\tilde{x}$
Dev	8421	8387	5488
Test	15878	15828	9680

Table 3: The number of distinct input utterances and distinct anonymous utterances on WikiSQL.

use the manually labeled training data (which is 10% of the whole training data) to initiate our anonymization model and then optimize the entire framework on the whole training data. All the other hyperparameters, e.g., the learning rate, hyperparameters in ADAM optimizer, the number of training epochs, etc., are tuned on the dev set[2].

5.2 Performance of Neural Semantic Parsing

First, we show that reducing the lexical problem through the anonymization model can improve the performance of neural semantic parser.

To this end, we add the anonymization model to typical neural semantic parsers as presented in Section 4. We use '[A]+DAE' to denote the neural semantic parser with the anonymization model, where A stands for the original name of the concrete parser and **DAE** is the abbreviated name of our approach, i.e., data-anonymous encoding.

[2]For Spider, we tune hyperparameters on training set and test the model on dev set.

On WikiSQL, the performance is evaluated by query-match accuracy (ACC_{QM}) and execution accuracy (ACC_{EX}), which measure accuracies of canonical representation and execution result matches between the predicted SQL and the ground truth respectively (Yu et al., 2018a). Table 1 shows the results. First, we can observe that query match accuracy on test data can be improved by 6.4% at most and 1.1% at least. Furthermore, for TypeSQL, query match accuracy can be further improved by 1.1% although it has used string-match based approach to anonymize table-related tokens. Moreover, we perform ablation studies by 1) removing the supervision for the anonymization model (denoted as '-Supervision' in Table 1), and 2) simply using the output of the trained anonymization model as the input for the parser without training them as a whole (denoted as '-Co-training' in Table 1). We can observe that the performance improvement is limited without supervision and co-training, indicating that both of them are indispensable.

On Spider, the performance is evaluated by exact-match accuracy on different difficulty levels of SQL queries, i.e., easy, medium, hard and extra hard. (Yu et al., 2018c). Table 2 shows the results. First, the overall accuracy can be improved by 3.2% and 2.5% respectively. Furthermore, per-

$\tilde{x}$	which *column* have *column* of *cell*?
x	which place [*column*] has a rank [*column*] of 71 [*cell*]? which county [*column*] has a median household income [*column*] of \$98,090 [*cell*] ?
$\tilde{x}$	what be *column* when *column* be *cell*?
x	what is the inclination [*column*] when the alt name [*column*] is ops-1584 [*cell*]? what was the district [*column*] when the reason for change [*column*] was died january 1, 1974 [*cell*]?
$\tilde{x}$	name *column* for *cell*
x	name the candidates [*column*] for georgia 8 [*cell*] name the party [*column*] for jack thomas brinkley [*cell*]

Table 4: Top frequent anonymous utterances on dev set of WikiSQL. The tokens with underlines are table-related tokens. We use '[*column*]' and '[*cell*]' to differentiate whether the token is related to a column name or a cell.

Method	Dev (%)			Test (%)		
	$\mathrm{ACC_{SC}}$	$\mathrm{ACC_{OC}}$	$\mathrm{ACC_{CE}}$	$\mathrm{ACC_{SC}}$	$\mathrm{ACC_{OC}}$	$\mathrm{ACC_{CE}}$
TypeSQL (Yu et al., 2018a)	75.9	92.9	–	76.0	92.9	–
AnnotatedSeq2Seq (Wang et al., 2018b)	88.8	64.6	–	88.8	63.6	–
DAE	**92.6**	**93.6**	**86.7**	**92.0**	**93.7**	**86.2**

Table 5: Performances of different anonymization models on WikiSQL.

formances on medium, hard and extra hard SQL queries achieve more improvement than that on easy SQL queries, indicating that our approach is more helpful for solving complicated cases.

5.3 Performance of Reducing Input Utterances

To further demonstrate the effectiveness of reducing the lexical problem, we show that different input utterance can be reduced to the same anonymous utterance. To this end, we process input utterances and anonymous utterances by 1) converting the characters to the lowercase, 2) lemmatizing the tokens and 3) removing the articles.

Table 3 shows that by anonymizing table-related tokens, the number of distinct utterances is reduced from 8387 to 5488 on dev set and from 15828 to 9680 on test set. Furthermore, Table 4 shows three anonymous utterances that are most frequent on dev set and examples of corresponding input utterances. All of these indicate that although input utterances can hardly be identical, they often share the same anonymous utterance.

5.4 Performance of Anonymization Methods

In addition, we compare performances of different anonymization methods, including 1) TypeSQL, which uses exact string match to detect column names, 2) AnnotatedSeq2Seq, which detects column names by setting a threshold for the edit distance and the cosine similarity of word embedding, and 3) DAE, our learning-based approach[3].

Anonymization methods are evaluated by following metrics: 1) whether the column names in SELECT clause is included in the prediction ($\mathrm{ACC_{SC}} = \frac{\sum_{i=1}^{Z} \mathbb{I}(S^i_{\text{sel\_col}} \subseteq S^i_{\text{pred\_col}})}{Z}$); 2) whether there is wrongly predicted column names ($\mathrm{ACC_{OC}} = \frac{\sum_{i=1}^{Z} \mathbb{I}(S^i_{\text{pred\_col}} \subseteq S^i_{\text{sel\_col}} \cup S^i_{\text{other\_col}})}{Z}$); and 3) whether all the cells in the SQL query are correctly predicted ($\mathrm{ACC_{CE}} = \frac{\sum_{i=1}^{Z} \mathbb{I}(S^i_{\text{pred\_cell}} = S^i_{\text{cell}})}{Z}$). Here, Z is the amount of test data, $\mathbb{I}(\cdot)$ is the indicator function, and superscript i is the index of data.

Table 5 shows that DAE significantly outperforms TypeSQL and AnnotatedSeq2Seq on all the evaluation metrics. First, for $\mathrm{ACC_{SC}}$, DAE outperforms TypeSQL and AnotatedSeq2Seq by 16% and 3.5% on test data; for $\mathrm{ACC_{OC}}$, DAE outperforms TypeSQL and AnnotatedSeq2Seq by 0.8% and 28% on test data. Moreover, DAE can achieve around 86% for $\mathrm{ACC_{CE}}$, while other methods fail to recognize cells when the table content is not available due to the privacy problem.

6 Conclusion

In this work, we propose a learning-based approach to reduce the lexical problem before the

[3]For experiments in this subsection, to make our anonymization model in the same settings as the others, we use BiLSTM as the input encoder and do not train the anonymization model with neural semantic parser as a whole.

neural semantic parser on text-to-SQL generation. Specifically, we propose a two-stage anonymization model and leverage implicit supervision from SQL queries by policy gradient to guide its training. In the future, we plan to improve the performance of the anonymization model by exploring more efficient expected reward. In addition, we also plan to extend our approach to the tasks with question-denotation pairs as supervision.

References

Ion Androutsopoulos, Graeme D Ritchie, and Peter Thanisch. 1995. Natural language interfaces to databases–an introduction. *Natural language engineering*, 1(1):29–81.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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*, pages 33–43.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 731–742.

Omer Goldman, Veronica Latcinnik, Ehud Nave, Amir Globerson, and Jonathan Berant. 2018. Weakly supervised semantic parsing with abstract examples. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 1809–1819.

Jonathan Herzig and Jonathan Berant. 2017. Neural semantic parsing over multiple knowledge-bases. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 623–628.

Jonathan Herzig and Jonathan Berant. 2018. Decoupling structure and lexicon for zero-shot semantic parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1619–1629.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. 2013. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347.

Wonseok Hwang, Jinyeung Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on wikisql with table-aware word contextualization. *arXiv preprint arXiv:1902.01069*.

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*, pages 12–22.

Yoon Kim, Sam Wiseman, and Alexander M Rush. 2018. A tutorial on deep latent variable models of natural language. *arXiv preprint arXiv:1812.06834*.

Nikolaos Kolitsas, Octavian-Eugen Ganea, and Thomas Hofmann. 2018. End-to-end neural entity linking. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 519–529.

Fei Li and HV Jagadish. 2014. Constructing an interactive natural language interface for relational databases. *Proceedings of the VLDB Endowment*, 8(1):73–84.

Radford M Neal and Geoffrey E Hinton. 1998. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer.

Yusuke Oda, Hiroyuki Fudaba, Graham Neubig, Hideaki Hata, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Learning to generate pseudo-code from source code using statistical machine translation. In *30th IEEE/ACM International Conference on Automated Software Engineering*, pages 574–584.

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*, pages 1470–1480.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. *NIPS Workshop*.

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*, pages 1532–1543.

Wei Shen, Jianyong Wang, and Jiawei Han. 2015. Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):443–460.

Tianze Shi, Kedar Tatwawadi, Kaushik Chakrabarti, Yi Mao, Oleksandr Polozov, and Weizhu Chen. 2018. Incsql: Training incremental text-to-sql parsers with non-deterministic oracles. *arXiv preprint arXiv:1809.05054*.

Alane Suhr, Mike Lewis, James Yeh, and Yoav Artzi. 2017. A corpus of natural language for visual reasoning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 217–223.

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

Chenglong Wang, Kedar Tatwawadi, Marc Brockschmidt, Po-Sen Huang, Yi Mao, Oleksandr Polozov, and Rishabh Singh. 2018a. Robust text-to-sql generation with execution-guided decoding. *arXiv preprint arXiv:1807.03100*.

Wenlu Wang, Yingtao Tian, Hongyu Xiong, Haixun Wang, and Wei-Shinn Ku. 2018b. A transfer-learnable natural language interface for databases. *arXiv preprint arXiv:1809.02649*.

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

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

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018a. Typesql: Knowledge-based type-aware neural text-to-sql generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, volume 2, pages 588–594.

Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. 2018b. Syntaxsqlnet: Syntax tree networks for complex and cross-domain text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1653–1663.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018c. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921.

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

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