ST-SQL: Semi-Supervised Self-Training for Text-to-SQL via Column Specificity Meta-Learning

Anonymous ACL submission

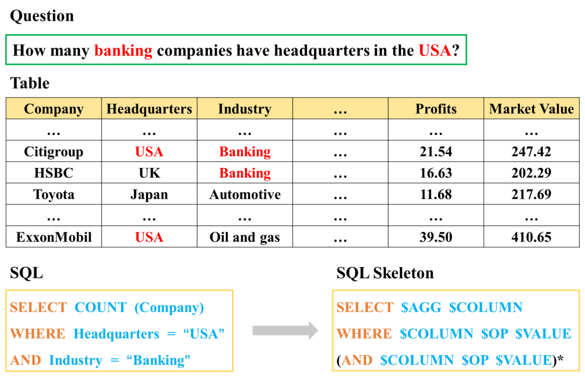
Abstract

The few-shot problem is an urgent challenge for the generalization capability of the single-table text-to-SQL task. Current few-shot methods neglect the potential information of unlabeled data and have a domain bias due to the same weight of samples. Motivated by this, this paper proposes a Self-Training text-to-SQL (ST-SQL) method which handles the problem from both views of data and algorithms. At the data level, ST-SQL performs data expansion by using an iterative framework to attach pseudo-labels to unlabeled data. The expanded data are sampled to reversely train the model. At the algorithm level, ST-SQL defines a column specificity to perform a more fine-grained gradient update during meta-training. The common samples are attached more weight to eliminate the domain bias. ST-SQL achieves state-of-the-art results on both open-domain and domain-specific benchmarks and brings more significant improvements on few-shot tests.

1 Introduction

Single-table text-to-SQL is a widely-studied semantic parsing task, which aims to translate natural language questions (NLQ) into SQL programs to access a single table. WikiSQL (Zhong et al., 2017) is the currently largest single-table text-to-SQL dataset, each example of which consists of a table, a question, and a target SQL program to be obtained. Figure 1 shows an example in WikiSQL.

Recently, benefiting from tabular-specific pre-trained models (Yin et al., 2020; Herzig et al., 2020; Zhao et al., 2021; Yu et al., 2020) and well-designed strategies (Hui et al., 2021; Chen et al., 2021a; Deng et al., 2021), state-of-the-art methods have been able to achieve more than 90% SQL execution accuracy on full-set WikiSQL. It brings a view that the single-table task seems to be solved (Suhr et al., 2020), so more and more works begin to challenge harder multi-table tasks (Yu et al., 2018b; Shi et al., 2021; Chen et al., 2021b).


Question

How many banking companies have headquarters in the USA?

Table

Company	Headquarters	Industry	...	Profits	Market Value
...	...	...	...	...	...
Citigroup	USA	Banking	...	21.54	247.42
HSBC	UK	Banking	...	16.63	202.29
Toyota	Japan	Automotive	...	11.68	217.69
...	...	...	...	...	...
ExxonMobil	USA	Oil and gas	...	39.50	410.65

SQL

```
SELECT COUNT (Company)
WHERE Headquarters = "USA"
AND Industry = "Banking"
```

SQL Skeleton

```
SELECT $AGG $COLUMN
WHERE $COLUMN $OP $VALUE
(AND $COLUMN $OP $VALUE)*
```

Figure 1: An example of single-table text-to-SQL.

However, is this really the case? We observe that most existing text-to-SQL models depend on sufficient pairs of NLQ and SQL annotations, which are regarded as the supervision training signal. Unfortunately, for each table in the actual scenario, high-quality SQL annotations require a high manual cost, consequently, there are only a few NLQ-SQL pairs that can be used for training. Some works (Chang et al., 2020; Chen et al., 2021a) call it a few-shot table challenge and demonstrate that it directly hinders the single-table text-to-SQL technology from theory to application.

On the one hand, although pre-trained models designed for tabular data (Yin et al., 2020; Herzig et al., 2020; Zhao et al., 2021; Yu et al., 2020) can bring a wealth of prior knowledge, they require millions of parallel corpus of text and tables for pre-training. It is obviously difficult to implement in some specific domains or non-English scenes. On the other hand, Meta-Learning (ML) is also a natural way to handle few-shot tables. Regrettably, the current ML-based framework (Chen et al., 2021a) is one-size-fits-all for all sub-tasks of text-to-SQL while ignoring their potential priorities. More importantly, a fact is neglected that different column samples typically have different domain relevance. For example in Figure 2, compared to column header "Market Value", which is almost only found in financial tables, "Company" is more

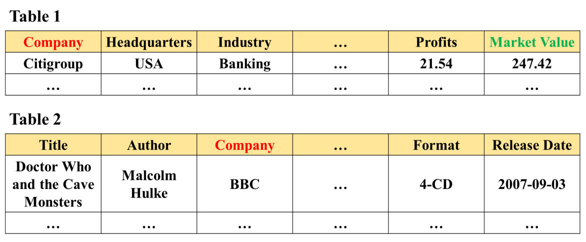


Table 1

Company	Headquarters	Industry	...	Profits	Market Value
Citigroup	USA	Banking	...	21.54	247.42
...	...	...	...	...	...

Table 2

Title	Author	Company	...	Format	Release Date
Doctor Who and the Cave Monsters	Malcolm Hulke	BBC	...	4-CD	2007-09-03
...	...	...	...	...	...

Figure 2: Column “Company” with low specificity can appear in tables with different topics, while column “Market Value” with high specificity may only exist in the financial tables.

common for all the tables. Intuitively, in order to make the model general to fast adapt to few-shot tables, the training process should pay more attention to the common columns, which are at the center of the sample space.

In this paper, we propose a new method called Self-Training text-to-SQL (ST-SQL) that overcomes the limitations by integrating the views of data and algorithms. We select Hydranet (Lyu et al., 2020), which is a strong text-to-SQL baseline, as our basic model and follow existing work to wrap it with content enhancement (Chen et al., 2021a). In order to improve its few-shot performance from the view of data, we propose a semi-supervised learning framework called ST-Semi to promote it by utilizing the potential information of unlabeled data. Specifically, in each training iteration, the basic model first predicts the SQL program of unlabeled data to obtain the pseudo labeled data, and then trains itself using the original labeled data and sampled pseudo labeled data. To obtain few-shot improvements from the view of algorithms, we propose a new column specificity meta-learning algorithm to assist the ST-Semi framework and optimize the model with an extra training object, which is only relevant to column prediction and is fundamental of all the sub-tasks. Considering the domain relevance of columns, we define a column specificity to weigh samples to make samples with a common column have a greater weight during loss calculation.

Our contributions can be summarized as:

- We propose a semi-supervised learning text-to-SQL framework that can automatically annotate unlabeled NLQ-table pairs and performs self-training. To the best of our knowledge, it is the first time to leverage semi-supervised learning to enhance text-to-SQL in the few-shot scenario.
- We propose a column specificity meta-learning (CSML) algorithm to make the more general samples have the more significant contribution for gradient calculation, in order to make the model easier for fast adaptation.
- We conduct comprehensive experiments on the open-domain dataset WikiSQL and domain-specific dataset ESQL (Chen et al., 2021a). Our method outperforms all baselines on few-shot tests and achieves a competitive result on the full set of WikiSQL.

2 Preliminaries

Given an NLQ q and a table $\mathcal{T} = (\mathcal{H}, \mathcal{C})$, the goal of the single-table text-to-SQL task is to output a SQL query y, denoted by $y = \mathcal{M}(q, \mathcal{T})$. Here, $\mathcal{M}$ is the target mapping (i.e., model) to be learned, $h_i \in \mathcal{H}$ is the i-th column header and $c_i \in \mathcal{C}$ is the set of cells belong to $\mathcal{H}_i$. In addition, each y follows a unified skeleton in Figure 1. The tokens with $ indicate a slot that needs to be filled and * represents that the bracket can be repeated zero or more times. In recent work, the task has been simplified into the following six sub-tasks.

- **Select-Column(SC)**: Selecting the column for $COLUMN in the SELECT clause.
- **Select-Aggregation(SA)**: Choosing the aggregation function for $AGG in the SELECT clause.
- **Where-Number(WN)**: Judging the number of conditions in the WHERE clause.
- **Where-Column(WC)**: Selecting columns for $COLUMN of conditions in the WHERE clause.
- **Where-Operator(WO)**: Choosing the operators for $OP of each condition in the WHERE clause.
- **Where-Value(WV)**: Extracting the mentions from the question as the value for $VALUE of each condition in the WHERE clause.

Some works (Lyu et al., 2020; Yin et al., 2020) have proved that SC and WC, which are directly related to columns, have the most important role in the final performance.

In this paper, the training data is denoted by $\mathcal{D} = (\mathcal{A}, \mathcal{U})$, where $\mathcal{A} = \{a_1, a_2, ..., a_{|\mathcal{A}|}\}$ is the set of labeled data and $\mathcal{U} = \{u_1, u_2, ..., u_{|\mathcal{U}|}\}$ is the set of unlabeled data. Here, $a_i = \{(q_i, y_i, \mathcal{T}_i)\}$ and $u_j = \{(q_j, \mathcal{T}_j)\}$. y_i is the gold SQL for q_i. Generally, $|\mathcal{A}| < |\mathcal{U}|$.

3 Method

Figure 3 illustrates the overview of our proposed ST-SQL. ST-SQL consists of a basic model $\mathcal{M}$ and a training framework $\mathcal{F}$. For $\mathcal{M}$ with parameter

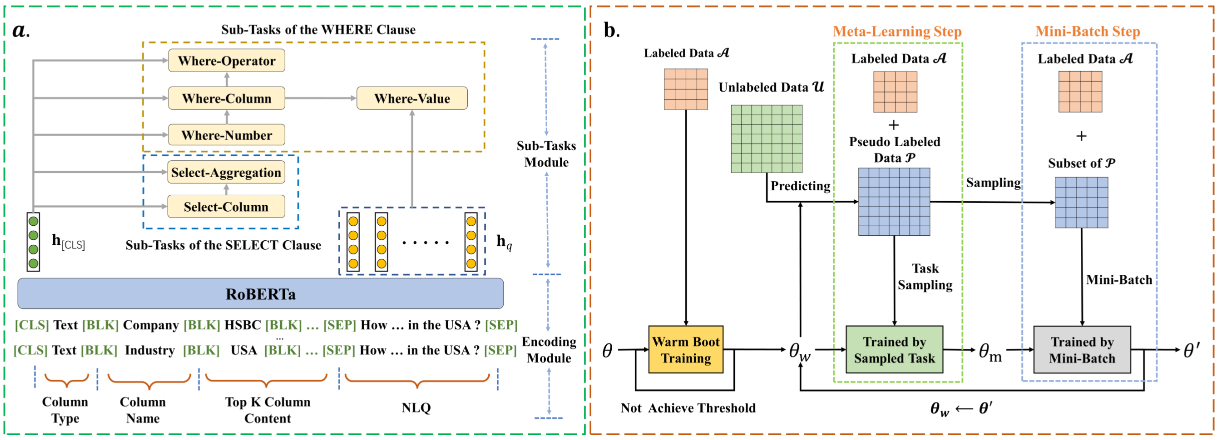


Figure 3: An overview of ST-SQL. a) The architecture of the basic model. b) The training procedure of ST-Semi.

θ, the framework $\mathcal{F}$ is used to obtain a new set of parameters $\theta^{'}$ as follows.

$$\mathcal{M}_{\theta^{'}} = \mathcal{F}(\mathcal{M}_{\theta}, \mathcal{A}, \mathcal{U}) \tag{1}$$

where $\theta^{'}$ has few-shot adaptation capability. Next, we will detail the basic model $\mathcal{M}$ and the training framework $\mathcal{F}$.

3.1 Basic Model

As mentioned in Section 2, the column prediction is critical for the total task. Therefore, we adopt Hydranet (Lyu et al., 2020) as our basic model $\mathcal{M}$. In addition, in view of the proven improvement of the table content to the text-to-SQL task (Yu et al., 2020; Chen et al., 2021a), we apply a content enhancement (Chen et al., 2021a) for $\mathcal{M}$. Specifically, the basic model consists of an encoding module and sub-tasks module.

3.1.1 Encoding Module

The encoding module is set to a pre-trained RoBERTa. For the purpose of use the semantic information of each NLQ-column pair (q, h_i), the input format is represented by

$$\texttt{[CLS]}, x_{t_i}, \texttt{[BLK]}, x^1_{h_i}, ..., x^m_{h_i}, \texttt{[BLK]}, x^1_{c^1_i}, ..., x^{l^1}_{c^1_i}, ..., \texttt{[BLK]}, x^1_{c^k_i}, ..., x^{l^k}_{c^k_i}, \texttt{[SEP]}, x^1_q, ..., x^n_q,$$

where `[CLS]`, `[BLK]` and `[SEP]` are the user-defined placeholders, x_{t_i} denotes the type of h_i and x^i_z denotes the i-th token of z. c^j_i denotes one cell of the column contents under h_i, used to enhance the potential semantic information. In this paper, top-k cells are obtained by calculating a literal score (Chen et al., 2021a) with q. The inputs are converted by RoBERTa into the following vectors.

$$[\mathbf{h}^i_{\texttt{[CLS]}}, \mathbf{h}_{t_i}, \mathbf{h}_{\texttt{[BLK]}}, \mathbf{h}^1_{h_i}, ..., \mathbf{h}^m_{h_i}, \mathbf{h}_{\texttt{[BLK]}}, \mathbf{h}^1_{c^1_i}, ..., \mathbf{h}^{l^1}_{c^1_i}, ..., \mathbf{h}_{\texttt{[BLK]}}, \mathbf{h}^1_{c^k_i}, ..., \mathbf{h}^{l^k}_{c^k_i}, \mathbf{h}_{\texttt{[SEP]}}, \mathbf{h}^1_q, ..., \mathbf{h}^n_q],$$

where each $\mathbf{h} \in \mathbb{R}^d$ is the hidden state of the input token x with the context information, and $\mathbf{h}^i_{\texttt{[CLS]}}$ is regarded as the semantic vector of (q, h_i).

3.1.2 Sub-Tasks Module

This module is used to make predictions for the sub-tasks in Section 2. First, for the SC task and WC task, the ranking scores of each column $\mathcal{H}_i$ are calculated by

$$P_{sc}(\mathcal{H}^i = \mathrm{SC}|\mathrm{q}) = \mathrm{sigmoid}(\mathrm{W_{sc}} \cdot \mathbf{h}^{\mathrm{i}}_{\texttt{[CLS]}}), \tag{2}$$

$$P_{wc}(\mathcal{H}^i \in \mathrm{WC}|\mathrm{q}, \mathcal{H}^{\mathrm{i}}) = \mathrm{sigmoid}(\mathrm{W_{wc}} \cdot \mathbf{h}^{\mathrm{i}}_{\texttt{[CLS]}}), \tag{3}$$

where $W_{sc} \in \mathbb{R}^d$ and $W_{wc} \in \mathbb{R}^d$ are trainable parameter matrices. These two scores are the basis of the whole SQL generation procedure. Then, the result of SC can be obtained by selecting the column with the highest P_{sc} score. Before getting the result of WC, WN needs to be calculated the first which can be calculated by a weighted sum of the conditional probability of each candidate number as follows.

$$P_{wn}(n_j|q, \mathcal{H}^i) = \mathrm{softmax}(W_{wn}[j:] \cdot \mathbf{h}^i_{\texttt{[CLS]}}), \tag{4}$$

$$\omega^i_j = \mathrm{softmax}(W_{\omega}[j:] \cdot \mathbf{h}^i_{\texttt{[CLS]}}), \tag{5}$$

$$n^{'} = \arg\max_{j}(\sum_{i} \omega^i_j \cdot P(n_j|q, \mathcal{H}^i)), \tag{6}$$

where $W_{wn} \in \mathbb{R}^{\hat{n} \times d}$, $W_{\omega} \in \mathbb{R}^d$ are affine transformations. $P(n_j|q, h_i)$ denotes the probability of n_j that i-th column predicts, $\hat{n}$ denotes the maximum of candidate n, and $n^{'}$ is the result for WN. Subsequently, the top n columns with the highest P_{wc} scores are taken as the result of WC. Finally, the remaining tasks can be predicted based on the columns selected by SC and WC as follows.

$$P_{sa}(a_j|q, \mathcal{H}^i) = \mathrm{softmax}(W_{sa}[j,:] \cdot \mathbf{h}^i_{\texttt{[CLS]}}), \tag{7}$$

$$P_{wo}(o_j|q, \mathcal{H}^i) = \mathrm{softmax}(W_{wo}[j,:] \cdot \mathbf{h}^i_{\texttt{[CLS]}}), \tag{8}$$

$$P^s_{wv}(x^i_j = s|q, \mathcal{H}^i) = \mathrm{softmax}(\mathbf{h}^i_q[j,:] \cdot W^s_{wv}), \tag{9}$$

$$P^e_{wv}(x^i_j = e|q, \mathcal{H}^i) = \mathrm{softmax}(\mathbf{h}^i_q[j,:] \cdot W^e_{wv}), \tag{10}$$

where $W_{sa} \in \mathbb{R}^{n_a \times d}$, $W_{wo} \in \mathbb{R}^{n_o \times d}$, $W_{wv}^s \in \mathbb{R}^d$ and $W_{wv}^e \in \mathbb{R}^d$ are weight matrices, n_a and n_o denote the number of candidate aggregation functions and operators, $\mathbf{h}_q^i \in \mathbb{R}^{l_q \times d}$ represent the question embedding for i-th column, l_q denotes the length of the question. The aggregation function and operators with the highest P_{sa} and P_{wo} scores calculated according to the columns of SC and WC, are chosen as the result of AGG and OP respectively. Mentions for WV are extracted by the start and end indexes with the highest P_{wv}^s and P_{wv}^e scores. Finally, a SQL program is obtained by filling the results of the six sub-tasks into the SQL skeleton.

3.2 Self-Training Framework

Our proposed self-training framework ST-Semi, i.e., $\mathcal{F}$ in (1), integrates the meta-learning technique into a semi-supervised training process. Algorithm 1 shows the detailed procedure.

The entire process can be divided into a warm boot stage and an iterative adaptation stage.

In the warm boot stage (line 2-9), random initialized $\mathcal{M}$ is trained by a conventional mini-batch strategy on labeled data $\mathcal{A}$ (line 5-6) while its best performance s on the validation set is updated (line 8). When s reaches threshold λ, the procedure is completed and parameters θ_w are obtained. Here, we use LF accuracy (detailed in 4.1.2), which is a widely-used metric for text-to-SQL, to evaluate s.

In the iterative adaptation stage, $\mathcal{M}$ is trained by using both $\mathcal{A}$ and unlabeled data $\mathcal{U}$. Concretely, at the first step of each epoch, $\mathcal{M}$ predicts SQL program $\hat{y}$ as a pseudo label for each unlabeled sample $(q, \mathcal{T})$ in $\mathcal{U}$ (Line 12-17). The pseudo labeled data is used to reversely promote the parameter update of $\mathcal{M}$. Intuitively, pseudo-label samples with higher model prediction confidence can have a greater impact on parameter updating. Therefore, we design a confidence score for each sample as follows.

$$\psi_i = \sqrt[\tau]{\prod_{z_i} \max(P(x_{z_i}|q_i, \mathcal{T}_i), \zeta)}, \tag{11}$$

where ζ is a threshold and τ is a hyper-parameter. $P(x_{z_i}|q_i, \mathcal{T}_i)$ is the confidence in the current task according to the model's output, where $z_i \in \{\mathrm{SC}, \mathrm{WC}, \mathrm{SA}\}$. Note that we only calculate ψ_i for SC, WC, and SA because the first two tasks are very basic and important, and SA generally has a poor performance.

At the second step, θ_w is updated by the mixed data of the labeled data and pseudo labeled data. According to the characteristics of different sub-tasks, we update θ_w in two ways. For SC and WC, tasks $\mathcal{D}_T$ are first sampled from the mixed data $\mathcal{D}_M = \mathcal{A} \bigcup \mathcal{P}$. θ_m is obtained by updating θ_w with column specificity meta-learning $\mathcal{F}_m$ on $\mathcal{D}_T$ (Line 19-20). Furthermore, in order to stabilize the parameters and adapt $\mathcal{M}$ into the target dataset, for all sub-tasks, a mini-batch is performed after the meta-learning. Here, $\mathcal{M}$ is trained by $\mathcal{D}_m = \mathcal{A} \bigcup \mathcal{P}'$ in mini-batch form, where $\mathcal{P}'$ is a subset sampled from $\mathcal{P}$ (Line 21-25).

After multiple epochs of iterative adaptation, θ', which is the set of parameters with the best performance on validation data, is returned as the final parameters.

Algorithm 1 Self-Training Framework

```
Require: Labeled set $\mathcal{A}$, unlabeled set $\mathcal{U}$, validation set $\mathcal{D}_v$,
    basic model $\mathcal{M}$, hyper-parameters $\alpha, \beta, \gamma, \delta, \eta$.
 1: Initialize $\mathcal{M}$ with random parameters $\theta$
 2: $s \leftarrow 0$
 3: while $s < \lambda$ do
 4:     for batch $\mathcal{B} \subseteq \mathcal{A}$ do
 5:         Evaluate $\mathcal{B} = \{(q_i, \mathcal{T}_i, y_i)\}$,
            $\nabla_\theta \mathcal{L}_w = \nabla_\theta \sum_i \mathcal{L}(\mathcal{M}(q_i, \mathcal{T}_i, \theta), y_i)$
 6:         Update parameters with gradient descent:
            $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_w$
 7:     end for
 8:     $s \leftarrow \max(\mathrm{score}(\mathcal{M}, \theta_w, \mathcal{D}_v), s)$
 9: end while
10: $\theta_w \leftarrow \theta$
11: while not done do
12:     Pseudo labeled set $\mathcal{P} \leftarrow \emptyset$
13:     for $q_j, \mathcal{T}_j \in \mathcal{U}$ do
14:         Predict pseudo label $\hat{y}_j \leftarrow \mathcal{M}(q_j, \mathcal{T}_j, \theta_w)$
15:         calculate confidence score $\psi_i$ for $\hat{y}_i$
16:         $\mathcal{P} \leftarrow \mathcal{P} \cup (q_j, \mathcal{T}_j, \hat{y}_j, \psi_j)$
17:     end for
18:     Set $\psi_i = 1$ for $y_i \in \mathcal{A}$
19:     Sample $\mathcal{D}_T \subseteq \mathcal{D}_M = \mathcal{A} \cup \mathcal{P}$
20:     Meta learning training step:
        $\theta_m \leftarrow \mathcal{F}_m(\mathcal{D}_T, \theta_w, \gamma, \delta, \eta)$
21:     Sample $\mathcal{P}' \subseteq \mathcal{P}$
22:     for batch $\mathcal{B} \subseteq \mathcal{D}_m = \mathcal{A} \cup \mathcal{P}'$ do
23:         Evaluate $\mathcal{B} = \{(q_k, \mathcal{T}_k, y_k, \psi_k)\}$,
            $\nabla_{\theta_m} \mathcal{L}_b \leftarrow \nabla_{\theta_m} \sum_k \psi_k \mathcal{L}_b(\mathcal{M}(q_k, \mathcal{T}_k, \theta_m), y_k)$
24:         Update parameters with gradient descent:
            $\theta_m \leftarrow \theta_m - \beta \nabla_{\theta_m} \mathcal{L}_b$
25:     end for
26:     $\theta_w \leftarrow \theta_m$
27: end while
28: $\theta' \leftarrow \theta_w$
29: return $\theta'$
```

3.3 Column Specificity Meta-Learning

In ST-Semi, $\mathcal{F}_m$ is the column specificity meta-learning (CSML) algorithm used to update θ_w for few-shot tables. Motivated by the objects of pre-trained text-to-SQL models, for $(q, \mathcal{H}^i)$, we define two new training objects to adopt meta-learning procedure:

- Predict whether $\mathcal{H}^i$ exists in the SELECT clause

Algorithm 2 Column Specificity Meta-Learning

```
Require: Task set $\mathcal{D}_T$, parameters $\theta_w$ of the basic model $\mathcal{M}$,
    hyper-parameters $\gamma$, $\delta$, and $\eta$
1: for all tasks do
2:     Evaluate support set $\mathcal{S} = \{(q_j, \mathcal{H}_j, \psi_j, \mu_j, y_j)\}$,
       $\nabla_{\theta_w} \mathcal{L}_{\mathcal{S}} = \nabla_{\theta_w} \Sigma_j \psi_j \mu_j \mathcal{L}(\mathcal{M}(q_j, \mathcal{H}_j, \theta_w), y_j)$
3:     Update parameters with gradient descent:
       $\theta_w^{'} \leftarrow \theta_w - \gamma \nabla_{\theta_w} \mathcal{L}_{\mathcal{S}}$
4:     Evaluate query set $\mathcal{Q} = \{(q_k, \mathcal{H}_k, \psi_k, \mu_k, y_k)\}$,
       $\nabla_{\theta_w^{'}} \mathcal{L}_{\mathcal{Q}} = \nabla_{\theta_w^{'}} \Sigma_k \psi_k \mu_k \mathcal{L}(\mathcal{M}(q_k, \mathcal{H}_k, \theta_w^{'}), y_k)$
5:     Update parameters with gradient descent:
       $\mathcal{L}_{\mathcal{SQ}} = \eta \mathcal{L}_{\mathcal{S}} + (1-\eta)\mathcal{L}_{\mathcal{Q}}, \theta_w \leftarrow \theta_w^{'} - \delta \nabla_{\theta_w^{'}} \mathcal{L}_{\mathcal{SQ}}$
6: end for
7: $\theta_m \leftarrow \theta_w$
8: return $\theta_m$
```

of y corresponding to q_i.

- Predict whether $\mathcal{H}^i$ exists in the WHERE clause of y corresponding to q_i.

There are two reasons for using these two objects: 1) First, they are relevant to SC and WC tasks, which are fundamental for text-to-SQL in our basic model $\mathcal{M}$. 2) More importantly, their results are very dependent and sensitive to the table which means that they are susceptible to the few-shot tables. For each sample $(q, \mathcal{H}^i)$, the loss is calculated by a binary cross-entropy.

$$\begin{aligned}\mathcal{L} =& y_{sc} \cdot \log(P_{sc}) + (1-y_{sc}) \cdot \log(1-P_{sc}) \\ &+ y_{wc} \cdot \log(P_{wc}) + (1-y_{wc}) \cdot \log(1-P_{wc})\end{aligned} \tag{12}$$

where $y_{sc} \in \{0,1\}$ and $y_{wc} \in \{0,1\}$ are the gold labels. P_{sc} and P_{wc} are calculated by (2) and (3).

Moreover, in CSML, different $(q, \mathcal{H}^i)$ have different contributions when calculating the total loss according to the column specificity of $\mathcal{H}^i$. The quantitative calculation is as follows.

$$\mu_{\mathcal{H}^i} = \frac{N_{total}}{N_{distinct} \cdot N_{h_i}}, \tag{13}$$

where N_{total} denotes the total frequency of all the columns and $N_{distinct}$ denotes the distinct number of columns. N_{h_i} denotes the frequency of the header h_i. Here, N_{total} divided by $N_{distinct}$ represents the average frequency of all columns. The greater the $\mu_{\mathcal{H}^i}$, the more special of $\mathcal{H}^i$.

In task sampling, the mixed data $\mathcal{D}_M = \{(q_i, \mathcal{H}_i, \psi_i, \mu_i)\}$ is shuffled, where ψ_i is derived from the score of the sample corresponding to q. From $\mathcal{D}_M$, each task randomly samples support set $\mathcal{S}$ and query set $\mathcal{Q}$ in n_w-way k_s/k_q-shot settings, where the samples related to the same table are treated as "a way".

The training procedure is shown in Algorithm 2. For each task T, the parameter update is divided into two steps. In the first step, loss $\mathcal{L}_{\mathcal{S}}$ is calculated by evaluating the support set $\mathcal{S}$ using ψ and μ as weights (Line 3). $\theta_w^{'}$ can be obtained by gradient update from θ_w with learning rate γ (Line 4). In the second step, loss $\mathcal{L}_{\mathcal{Q}}$ is calculated by evaluating the query set $\mathcal{Q}$ on $\theta_c^{'}$ (Line 5). $\mathcal{L}_{\mathcal{S}}$ and $\mathcal{L}_{\mathcal{Q}}$ are joined to calculate the gradient which is used to update $\theta_w^{'}$ to new θ_w with learning rate δ (Line 5). After all tasks are iterated, the final parameters θ_m are used as the output of the meta-learning step in ST-Semi.

4 Experiments

In this section, we compare ST-SQL to other existing methods to analyze the impact of different factors on few-shot performance.

4.1 Experiment Setup

4.1.1 Dataset

We evaluate the model performance on the following two benchmarks.

WikiSQL (Zhong et al., 2017) is currently the largest open-domain single-table text-to-SQL dataset which contains more than 20k tables collected from Wikipedia. Here, 80k questions are divided into 56,355 training questions, 8,421 development questions, and 15,878 test questions.

ESQL (Chen et al., 2021a) is a Chinese domain-specific single-table text-to-SQL dataset containing 17 tables, 10k training questions, 1,000 development questions, and 2,000 test questions. ESQL also provides zero-shot development and test set which contain 494 questions and 972 questions separately.

We also utilize WikiTableQuestions (Pasupat and Liang, 2015) as an available unlabeled data source. It is an open-domain single-table QA dataset that provides more than 2k tables and 20k question-answer pairs. In our experiments, when using the full training data of WikiSQL, all the NLQ-table pairs in WikiTableQuestions are used.

4.1.2 Evaluation Metrics

Logical Form(LF) accuracy is a general metric for text-to-SQL which evaluates the exact matching results between the predicted SQL and the gold SQL. Execution accuracy(EX) is another metric that evaluates whether query results of the predicted SQL and gold SQL are identical.

4.1.3 Implementation Details

We performed the experiments on Tesla V100 Super GPU. We used AdamW as the optimizer for training. In the experiments on WikiSQL, We did not limit the number of column content k because all tables in WikiSQL only have a small amount of data. On the contrary, the scale of the tables in ESQL is very large, so we set k equal to 5 when experimenting on ESQL. The settings of the other hyper-parameters are in Table 1. Our implementation is publicly available[1]

Params	Value	Description
α, β	3e-5	learning rate of ST-Semi
γ, δ	3e-5, 1e-5	learning rate of CSML
η	0.5	the weight of loss calculation in CSML
$\lvert\mathcal{D}_T\rvert$	100/300	the number of sampled tasks at each iteration
n_w, k_s, k_q	4, 15, 5	the specific settings for each task
λ	80.0/65.0	the threshold of warm boot
σ	20%/5%	sampling ratio of the unlabeled data
ζ, τ	0.1, 2.5	the settings for confidence calculation

Table 1: Hyper-parameters settings."X/X" denotes the experiments that use full training data or in few-shot settings separately.

4.1.4 Methods for Comparison

We compared with notable work that has reported results on WikiSQL task, including Seq2SQL (Zhong et al., 2017), Coarse2Fine (Dong and Lapata, 2018), SQLNet (Xu et al., 2017), TypeSQL (Yu et al., 2018a), X-SQL (He et al., 2019), IE-SQL (Ma et al., 2020), and SDSQL (Hui et al., 2021), SQLova (Hwang et al., 2019), HydraNet (Lyu et al., 2020), BRIDGE (Lin et al., 2020), TaBERT (Yin et al., 2020), GRAPPA (Yu et al., 2020), TAPAS (Herzig et al., 2020), and MC-SQL (Chen et al., 2021a).

4.2 Overall Results on WikiSQL

We first evaluated the performance of ST-SQL on the full set of WikiSQL. The experimental results are shown in Table 2. ST-SQL ranks the third best on both LF and EX on the test set of WikiSQL while the SSL-only setting achieves state-of-the-art results. Here, the "SSL-only" setting represents the ST-SQL that is removed the meta-learning step from ST-Semi. It proves that self-training on additional unlabeled data is useful to enhance the model, and can even perform better than using additional annotation. Moreover, the middle of Table 2 shows that semi-supervised learning is competitive with tabular pre-trained models. Here, a possible reason the drop brought by CSML is that mini-batch can optimize parameters more stably than meta-learning in the case of sufficient training data

[1]https://github.com/ygxw0909/ST-SQL

Method	Dev.LF	Dev.EX	Test.LF	Test.EX
Seq2SQL	49.5	60.8	48.3	59.4
SQLNet	63.2	69.8	61.3	68.0
TypeSQL	68.0	74.5	66.7	73.5
Coarse2Fine	72.5	79.0	71.7	78.5
SQLova	81.6	87.2	80.7	86.2
X-SQL	83.8	89.5	83.3	88.7
HydraNet	83.6	89.1	83.8	89.2
MC-SQL*-	84.1	89.7	83.7	89.4
IE-SQL+	84.6	88.7	84.6	88.8
SDSQL+	86.0	91.8	85.6	91.4
BRIDGE*	86.2	91.7	85.7	91.1
TaBERT*	84.0	89.6	83.7	89.1
GRAPPA*	85.9	-	84.7	-
TAPAS*	85.1	-	83.6	-
ST-SQL*	85.7	90.8	85.4	90.3
ST-SQL(SSL-only)*	**86.4**	**91.9**	**85.8**	**91.6**

Table 2: Overall results on WikiSQL. "*" denotes the model using column content. "+" denotes the model is trained with additional labeling features. "-" denotes the result is based on the base version of the pre-training model.

4.3 Few-Shot Test

To evaluate the performance of ST-SQL on a few-shot scenario, we establish the following settings on WikiSQL and ESQL.

Data Settings For the training set of WikiSQL and ESQL, we sampled 1%(563) and %5(500) NLQs-table pairs, respectively, to build the few-shot training sets. The remaining data of the training set is regarded as unlabeled data. We also consider the zero-shot settings, which is an extreme case of the few-shot setting. For WikiSQL, we sampled 2%(359) tables and all the related NLQs (1148) from the training set and the development/test set has 2281/4148 unseen tables. For ESQL, we followed the few-shot setting and directly used its released zero-shot development and test set which contain 7 unseen tables.

Ablation Settings To explore the contribution of each component of ST-SQL under the few-shot scenario, we applied the following settings.

- **Basic** We only employed the basic model described in Section 3.1.
- **Basic+SSL** We applied ST-Semi to the basic model but removed the meta-learning step.
- **Basic+SSL+ML** We replaced CSML with an existing meta-learning algorithm(Chen et al., 2021a) of the text-to-SQL field.

Baseline Settings We select three categories of methods from Table 2 for comparison in the few-shot/zero-shot test. First, SQLova (Hwang et al., 2019), HydraNet (Lyu et al., 2020), and BRIDGE (Lin et al., 2020) area supervised

Method	FS on WikiSQL Dev.LF	FS on WikiSQL Test.LF	ZS on WikiSQL Dev.LF	ZS on WikiSQL Test.LF	FS on ESQL Dev.LF	FS on ESQL Test.LF	ZS on ESQL Dev.LF	ZS on ESQL Test.LF
SQLova	35.7	35.1	46.2	45.7	59.1	54.9	52.0	53.3
MC-SQL*	52.8	53.7	62.8	62.8	62.3	62.6	53.0	53.9
HydraNet	66.3	66.0	68.7	68.9	66.8	68.8	63.6	66.8
BRIDGE*	64.4	64.0	70.3	70.5	-	-	-	-
TaBERT+SQLova*	56.1	55.9	72.9	72.9	-	-	-	-
TAPAS+basic*	43.4	42.2	63.1	62.8	-	-	-	-
GRAPPA+basic*	74.1	74	76.2	76.2	-	-	-	-
ST-SQL*	**78.1**	**78.9**	**79.1**	**79.0**	**77.1**	**76.1**	**71.1**	**71.1**
Basic*	71.5	71.5	74.1	73.5	67.2	70.5	66.8	68.6
Basic+SSL*	77.2	77.7	78.3	77.9	69.5	69.7	67.6	69.4
Basic+SSL+ML*	76.2	76.2	77.7	77.1	72.2	72.8	68.2	69.2

Table 3: Results of the few-shot and zero-shot tests. "*" denotes the model using column content.

learning-based methods that are representative due to their good performance. Second, TaBERT (Yin et al., 2020), Grappa (Yu et al., 2020), and TAPAS (Herzig et al., 2020) are pre-trained text-to-SQL models with rich prior knowledge. Third, MC-SQL (Chen et al., 2021a) is a specifically-designed method for zero-shot tables, which also leverages the meta-learning. For the above methods, we directly ran its public source code to obtained experiments. We did not consider the other methods in Table 1 because they had not released the source code or were not advanced enough.

The results in Table 3 show that our method outperforms all the baselines and gains significant improvements on few-shot and zero-shot tests on both WikiSQL and ESQL. These representative end-to-end methods can not handle few-shot and zero-shot tables very well. For the pre-trained models, we set different basic models according to their output characteristics. However, the Chinese version of these pre-training models had not been released yet, so that they were only tested on WikiSQL. Since they involve a wealth of prior knowledge and do not require many samples to fine-tune, they got better performance on WikiSQL, especially GRAPPA. Despite that, the proposed ST-Semi framework demonstrated a stronger ability to effectively using on additional unlabeled data and deal with few-shot and zero-shot tables. Amazingly, the accuracy of ST-SQL on WikiSQL with only a 1% labeling rate is close to some methods using full labeled training data.

In the ablation tests, first, the basic ST-SQL model showed the significance of table content according to the comparison with HydraNet. The semi-supervised learning in ST-Semi also improved the accuracy greatly on WikiSQL. However, it did not contribute obviously to ESQL, the reason we supposed is that, as a domain-specific dataset, ESQL is more difficult than WikiSQL to be adapted, and it leads to much noise of pseudo labels to semi-supervised learning without the assistance of meta-learning. While it was combined with CSML as ST-SQL, the promotion becomes remarkable. For comparing our CSML with other similar meta-learning algorithms, we replaced CSML with the training process in (Chen et al., 2021a), and the results showed CSML performed much better.

4.4 Detail Analysis

Effect of ST-Semi Figure 4 shows the impact of the adoption of ST-Semi on the trend of accuracy changes during the training process in the few-shot tests. After reaching the threshold and finishing warm boot, ST-Semi further boosts the increase of accuracy compared with the basic model. The self-training process mostly completed in 30 epochs.

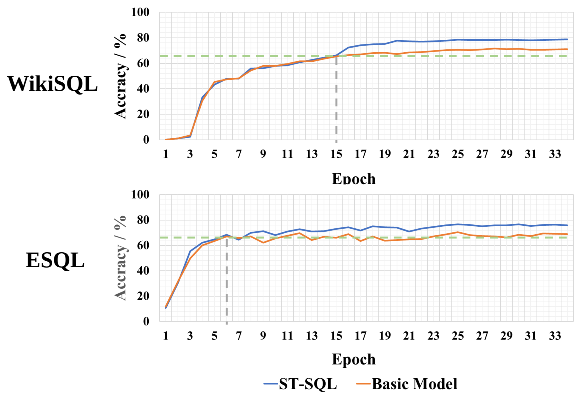


Figure 4: Performance trend with/without ST-Semi. The green line shows the threshold of the warm boot.

Moreover, we made a comparative experiment on the influence of the number of unlabeled data on ST-Semi, where the sampling ratio of mini-batch and the relevant settings of task sampling had been scaled suitably. The results in Figure 5 show that with proper sampling-related hyper-parameter settings to ensure that unlabeled data is not excessively added to each round of training, for ST-Semi, increasing the total number of additional unlabeled data can improve the effectiveness of the ST-SQL.

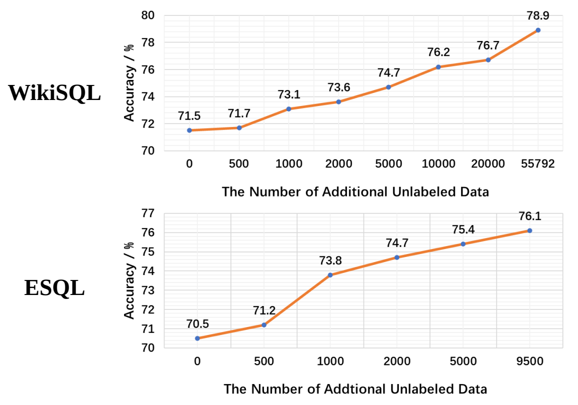


Figure 5: Results of using different sizes of additional unlabeled sets.

Error Analysis We randomly sampled 200 bad cases after the few-shot tests and analyzed the error reason. Here, we divide error into 5 categories and statistics the frequency of each type of error in sampled bad cases which is shown in Figure 6. According to the results, Column selection and aggregation function prediction are the tasks that still need to be promoted.

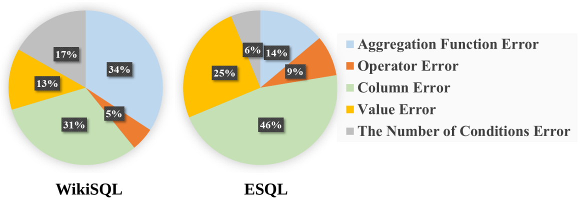


Figure 6: Percentage of error causes.

5 Related Work

Single table text-to-SQL task recently attract lots of attention since the release of WikiSQL (Zhong et al., 2017). Previous work can be mainly divided into two directions. The earlier methods (Dong and Lapata, 2018; Krishnamurthy et al., 2017; Sun et al., 2018; Zhong et al., 2017) use sequence-to-sequence models to generate SQL token by token, which is called the generation-based method. But the sketch-based methods first proposed by SQLNet (Xu et al., 2017) later show better performance. Based on that, TypeSQL (Yu et al., 2018a) adds data type of columns into representation. SQLova (Hwang et al., 2019) and X-SQL (He et al., 2019) leverage BERT (Devlin et al., 2019) and MT-DNN (Liu et al., 2019) to text-to-SQL models and achieve significant improvement. Furthermore, HydraNet (Lyu et al., 2020) proposes a column-wise model which performs better than all the former question-wise models. BRIDGE (Lin et al., 2020) designs a sequential text-DB encoder to enhance the representation which achieves outstanding performance. IE-SQL (Ma et al., 2020) transfers the task into an information extraction problem. In the latest work SDSQL (Hui et al., 2021) designs an assistant task to help model learning the dependency between questions and schema.

Recently, lots of customized pre-training models appear in different research areas, as well as text-to-SQL. TaBERT (Yin et al., 2020) is proposed for a joint understanding of textual and tabular data, where table content is leveraged into representation. TAPAS (Herzig et al., 2020) is proposed for Table QA, but the table encoding ability also can be used in text-to-SQL. GRAPPA (Yu et al., 2020) is proposed for table semantic parsing combined with a grammar-augmented pre-training framework.

Few-shot and zero-shot problem in the text-to-SQL task is first mentioned by (Chang et al., 2020), where the model is additionally trained to map the entity of questions to the columns of tables. But the limitation is that additional annotations are needed. MC-SQL (Chen et al., 2021a) leverages table content features into embedding and uses meta-learning for training. However, the experiment shows that the meta-learning algorithm is not greatly helpful.

6 Conclusion

In this paper, we presented a new self-training method on the single table text-to-SQL task. We leveraged column content into a column-wise model as the basic model. Base on that, we designed a self-training framework ST-Semi which combines semi-supervised learning and meta-learning technique to make effective use of unlabeled data and adopt self-training ability. Moreover, we proposed a column specificity meta-learning algorithm to handle few-shot tables. The experimental results indicated that our method outperforms the other baselines on multiple benchmarks in few-shot settings. Furthermore, our method also showed a good performance on zero-shot tables. In future work, we will try to optimize the design of meta-learning and expand the research area to multi-tables, complex text-to-SQL.

References

Shuaichen Chang, Pengfei Liu, Yun Tang, Jing Huang, Xiaodong He, and Bowen Zhou. 2020. Zero-shot text-to-sql learning with auxiliary task. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7488–7495. AAAI Press.

Yongrui Chen, Xinnan Guo, Chaojie Wang, Jian Qiu, Guilin Qi, Meng Wang, and Huiying Li. 2021a. Leveraging table content for zero-shot text-to-sql with meta-learning. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 3992–4000. AAAI Press.

Zhi Chen, Lu Chen, Yanbin Zhao, Ruisheng Cao, Zihan Xu, Su Zhu, and Kai Yu. 2021b. Shadowgnn: Graph projection neural network for text-to-sql parser. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 5567–5577. Association for Computational Linguistics.

Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. 2021. Structure-grounded pretraining for text-to-sql. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 1337–1350. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

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, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 731–742. Association for Computational Linguistics.

Pengcheng He, Yi Mao, Kaushik Chakrabarti, and Weizhu Chen. 2019. X-SQL: reinforce schema representation with context. *CoRR*, abs/1908.08113.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4320–4333. Association for Computational Linguistics.

Binyuan Hui, Xiang Shi, Ruiying Geng, Binhua Li, Yongbin Li, Jian Sun, and Xiaodan Zhu. 2021. Improving text-to-sql with schema dependency learning. *CoRR*, abs/2103.04399.

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

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, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 1516–1526. Association for Computational Linguistics.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020. Bridging textual and tabular data for cross-domain text-to-sql semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 4870–4888. Association for Computational Linguistics.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4487–4496. Association for Computational Linguistics.

Qin Lyu, Kaushik Chakrabarti, Shobhit Hathi, Souvik Kundu, Jianwen Zhang, and Zheng Chen. 2020. Hybrid ranking network for text-to-sql. *CoRR*, abs/2008.04759.

Jianqiang Ma, Zeyu Yan, Shuai Pang, Yang Zhang, and Jianping Shen. 2020. Mention extraction and linking for SQL query generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6936–6942. 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 of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers, pages 1470–1480. The Association for Computer Linguistics.

Peng Shi, Patrick Ng, Zhiguo Wang, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Cícero Nogueira dos Santos, and Bing Xiang. 2021. Learning contextual representations for semantic parsing with generation-augmented pre-training. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 13806–13814. AAAI Press.

Alane Suhr, Ming-Wei Chang, Peter Shaw, and Kenton Lee. 2020. Exploring unexplored generalization challenges for cross-database semantic parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8372–8388. Association for Computational Linguistics.

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. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 361–372. Association for Computational Linguistics.

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

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8413–8426. Association for Computational Linguistics.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir R. 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: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pages 588–594. Association for Computational Linguistics.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir R. Radev, Richard Socher, and Caiming Xiong. 2020. Grappa: Grammar-augmented pre-training for table semantic parsing. *CoRR*, abs/2009.13845.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. 2018b. 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, Brussels, Belgium, October 31 - November 4, 2018*, pages 3911–3921. Association for Computational Linguistics.

Liang Zhao, Hexin Cao, and Yunsong Zhao. 2021. GP: context-free grammar pre-training for text-to-sql parsers. *CoRR*, abs/2101.09901.

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

	Few-Shot						Zero-Shot					
Method	SC	SA	WN	WC	WO	WV	SC	SA	WN	WC	WO	WV
SQLova	81.9 / 80.8	77.3 / 77.4	88.8 / 88.4	66.1 / 65.3	77.6 / 77.4	67.0 / 66.9	89.1 / 88.3	82.2 / 82.5	92.0 / 92.0	76.2 / 75.0	83.5 / 83.7	71.1 / 71.4
MC-SQL*	88.2 / 88.3	83.8 / 85.2	92.2 / 91.2	86.5 / 85.3	82.2 / 81.8	73.8 / 74.0	91.4 / 90.9	87.9 / 88.5	95.7 / 95.1	92.0 / 90.8	88.1 / 87.5	80.3 / 80.0
HydraNet	94.2 / 94.1	86.2 / 86.7	92.9 / 92.1	84.0 / 83.1	89.5 / 88.9	85.4 / 84.1	95.8 / 95.5	88.0 / 88.4	94.0 / 93.5	84.9 / 84.4	91.7 / 91.1	86.0 / 85.5
BRIDGE*	-	-	-	-	-	-	-	-	-	-	-	-
TaBERT+SQLova*	91.1 / 91.0	83.2 / 83.2	94.8 / 94.8	87.4 / 87.2	87.7 / 87.8	78.9 / 78.7	95.7 / 95.9	88.6 / 88.5	**97.2 / 97.3**	93.5 / 93.5	**95.7 / 95.8**	88.3 / 88.4
TAPAS+basic*	86.6 / 85.7	84.1 / 84.0	89.1 / 88.5	78.9 / 78.4	85.4 / 85.1	57.6 / 57.2	93.9 / 93.6	87.4 / 87.5	94.0 / 93.9	90.1 / 89.9	92.3 / 92.0	75.9 / 75.6
GRAPPA+basic*	95.0 / 94.8	87.1 / 88.1	93.8 / 93.6	90.4 / 89.9	92.0 / 91.9	89.3 / 88.8	96.2 / 95.9	88.7 / 88.9	94.8 / 94.6	91.1 / 90.7	93.0 / 92.8	90.4 / 90.3
ST-SQL*	**97.1 / 97.0**	87.6 / 88.5	**95.1 / 95.1**	**93.3 / 93.4**	**93.9 / 94.1**	**92.9 / 92.9**	**97.1 / 96.8**	88.7 / 89.0	96.0 / 95.5	**94.1 / 93.7**	94.8 / 94.3	**93.4 / 93.0**
Basic*	95.6 / 95.0	88.0 / 88.1	92.3 / 91.9	87.6 / 87.4	90.4 / 90.3	86.8 / 86.4	95.6 / 95.5	87.9 / 88.1	94.3 / 93.9	90.4 / 89.6	92.3 / 91.9	89.0 / 88.2
Basic+SSL*	96.3 / 96.5	**88.1 / 88.6**	94.5 / 94.5	92.1 / 92.2	93.1 / 93.1	91.7 / 91.6	96.0 / 95.8	**88.9 / 89.4**	95.8 / 95.3	93.2 / 92.9	94.6 / 93.1	93.1 / 92.5
Basic+SSL+ML*	96.8 / 96.5	86.8 / 87.4	94.1 / 93.6	91.9 / 91.6	92.7 / 92.2	91.6 / 91.0	96.9 / **96.8**	88.0 / 88.1	95.6 / 95.1	93.8 / 93.2	94.3 / 93.9	92.8 / 92.4

Table 4: The results of each sub-task in few-shot and zero-shot tests on WikiSQL.

	Few-Shot						Zero-Shot					
Method	SC	SA	WN	WC	WO	WV	SC	SA	WN	WC	WO	WV
SQLova	86.7 / 89.7	**96.8** / 95.9	**98.4** / 96.7	76.8 / 69.8	89.4 / 86.4	75.0 / 69.3	82.4 / 86.8	94.9 / 95.2	96.4 / 96.4	73.3 / 70.3	84.6 / 85.3	70.2 / 68.5
MC-SQL*	90.2 / 90.3	94.1 / 93.7	97.7 / 96.1	80.9 / 79.7	89.1 / 87.8	72.9 / 71.3	83.8 / 83.5	**98.2** / 95.9	96.8 / 94.1	81.2 / 81.9	**92.5** / 88.8	63.4 / 64.8
HydraNet	93.0 / 97.8	92.6 / 93.3	97.5 / 96.8	81.3 / 83.3	90.2 / 92.1	76.5 / 80.9	91.9 / 92.5	95.7 / 97.1	97.0 / 95.9	80.2 / 82.0	88.5 / 87.4	73.9 / 75.0
ST-SQL*	**98.9 / 98.6**	96.0 / **97.0**	98.1 / 97.0	**86.3 / 86.5**	**93.5 / 93.3**	**83.0 / 81.7**	**97.5 / 97.2**	**98.2 / 97.3**	97.8 / 96.6	**83.8 / 83.7**	**92.5 / 94.2**	**74.7** / 76.1
Basic*	95.6 / 97.2	92.4 / 93.8	97.9 / 97.0	82.4 / 84.6	89.4 / 89.6	77.4 / 80.2	95.4 / 95.4	97.8 / 97.2	**97.9 / 96.8**	76.3 / 80.9	90.5 / 85.8	73.9 / 75.1
Basic+SSL*	97.7 / 96.9	93.3 / 94.3	98.0 / 97.1	82.0 / 84.9	91.9 / 89.6	77.3 / 79.1	95.8 / 95.8	97.7 / 97.0	97.3 / 96.2	79.4 / 81.0	92.1 / 92.6	73.6 / 74.9
Basic+SSL+ML*	97.9 / 98.0	95.6 / 95.9	98.1 / **97.2**	80.4 / 82.6	91.1 / 89.5	80.0 / 77.8	96.6 / 96.6	93.5 / 96.7	97.2 / 95.9	77.3 / 80.0	91.5 / 90.9	73.1 / **77.5**

Table 5: The results of each sub-task in few-shot and zero-shot tests on ESQL.

A Appendices

Table 4 and Table 5 show the detailed results of the few-shot and zero-shot experiments on WikiSQL and ESQL. Here, we evaluated the performance of each sub-task. ST-SQL achieved the best performance for most of the sub-tasks, especially SC and WC. It reveals that CSML can bring improvements in column selection. Leveraging the column content also significantly improved the effect of WV. Benefiting from the enhancement of the ST-Semi using additional unlabeled data, all sub-tasks had a certain improvement compared to the basic model.