

# Structure-Grounded Pretraining for Text-to-SQL

Xiang Deng<sup>\*1</sup>, Ahmed Hassan Awadallah<sup>2</sup>, Christopher Meek<sup>2</sup>,  
Oleksandr Polozov<sup>2</sup>, Huan Sun<sup>1</sup>, and Matthew Richardson<sup>2</sup>

<sup>1</sup>The Ohio State University  
{deng.595, sun.397}@osu.edu

<sup>2</sup>Microsoft Research, Redmond  
{hassanam, meek, polozov, matttri}@microsoft.com

## Abstract

Learning to capture text-table alignment is essential for table related tasks like text-to-SQL. The model needs to correctly recognize natural language references to columns and values and to ground them in the given database schema. In this paper, we present a novel weakly supervised **Structure-Grounded** pretraining framework (STRUG) for text-to-SQL that can effectively learn to capture text-table alignment based on a parallel text-table corpus. We identify a set of novel prediction tasks: column grounding, value grounding and column-value mapping, and train them using weak supervision without requiring complex SQL annotation. Additionally, to evaluate the model under a more realistic setting, we create a new evaluation set Spider-Realistic based on Spider with explicit mentions of column names removed, and adopt two existing single-database text-to-SQL datasets. STRUG significantly outperforms BERT<sub>LARGE</sub> on Spider and the realistic evaluation sets, while bringing consistent improvement on the large-scale WikiSQL benchmark.

## 1 Introduction

Semantic parsing is the task of mapping a natural language (NL) utterance to a machine-understandable meaning representation such as lambda calculus, abstract meaning representation, and structured query language (SQL). In this paper, we focus on the task of translating NL questions to executable SQL queries (text-to-SQL). This is a fundamental task for building natural language interfaces for databases, which can enable non-expert users to effortlessly query databases.

One of the key challenges in text-to-SQL is text-table alignment, that is, *to correctly recognize natural language references to columns and values*

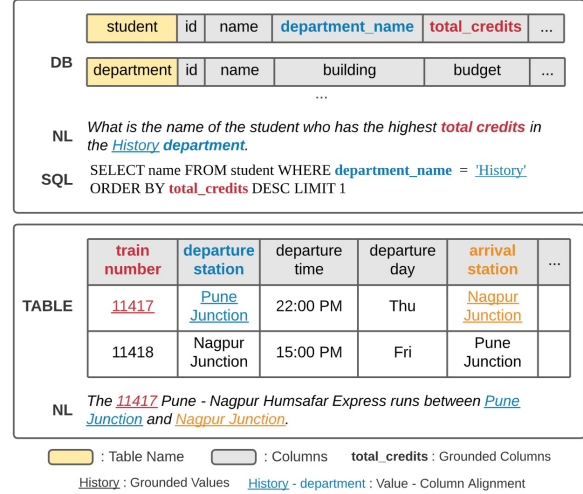


Figure 1: Illustration of our motivation. The top half shows an example of text-to-SQL while the bottom half is an example from a parallel text-table corpus. In both examples, there is association between tokens in the NL utterance and columns in the table. In this paper, we aim to leverage the text-table alignment knowledge in the parallel text-table corpus to help text-to-SQL.

and to ground them in the given database schema. Consider the example in the top half of Fig. 1, the model needs to first identify the column mentions *total credits*, *department*, and value mention *History*, then ground them to the given schema. This is challenging for three reasons. First, the model needs to jointly understand the NL utterance and database schema, as the user may refer to the column using various expressions which usually differ from the original column name. Second, the model needs to be able to generalize to new database schemas and referential language that is not seen in training. Finally, in the case that using cell values is not possible, the model still needs to identify potential value mentions and link them to the correct columns without exhaustively searching and matching over the database.

<sup>\*</sup>Work done during an internship at Microsoft Research.

Text-table alignment also naturally exists in parallel text-table corpora, e.g., web tables with context (Lehmberg et al., 2016), table-to-text generation datasets (Parikh et al., 2020; Chen et al., 2020a), table-based question answering datasets (Pasupat and Liang, 2015; Chen et al., 2020b), which are much easier to collect compared with text-to-SQL datasets. As shown in the bottom half of Fig. 1, there are three value mentions 11417, Pune Junction and Nagpur Jnction, which can be grounded to the train number, departure station and arrival station columns respectively. Such alignment information can be easily obtained by leveraging the table contents or using some human annotation. In this work, we aim to take advantage of the text-table alignment knowledge in the parallel corpus via pretraining and use it to help the downstream text-to-SQL task.

We present a novel weakly supervised structure-grounded pretraining framework for text-to-SQL. We design a set of prediction tasks and optimize them leveraging a parallel corpus containing both NL sentence and tabular data to encourage the encoded representation to capture information required to support tasks that require table grounding. More specifically, we identify three critical tasks for aligning text with table: column grounding, value grounding and column-value mapping. We repurpose an existing large-scale table-to-text generation dataset ToTTo (Parikh et al., 2020) for pretraining and gain labels for the three tasks via weak supervision. We experiment under two settings, with or without human annotation: (1) human assisted setting, using the revised description together with the cell annotations; (2) automatic setting, using the raw sentences and infer the cell correspondence leveraging the table contents via string matching.

As pointed out by Suhr et al. (2020), existing text-to-SQL benchmarks like Spider (Yu et al., 2018) ease the text-table alignment challenge by explicitly mentioning exact column names in NL utterances, while in realistic users may refer to the column using various expressions. Suhr et al. (2020) propose a new cross-database setting that uses Spider for training and includes eight other single-domain text-to-SQL datasets for evaluation. In addition to partly adopt their setting, we create a new evaluation set called Spider-Realistic from the original Spider development set by removing the explicit mention of column names in the utterance.

We pretrain STRUG using 120k text-table pairs from ToTTo. In experiments, we notice that our structure-grounded pretraining objectives are very efficient and usually converge with around 5 epochs. This dramatically reduces the training cost compared with previous pretraining methods (Herzig et al., 2020; Yin et al., 2020). We adopt the same model architecture as BERT (Devlin et al., 2018), with simple classification layer on top for pretraining. For downstream tasks, STRUG can be used as a text-table encoder and easily integrated with any existing state-of-the-art models.

We conduct extensive experiments and show that:

(1) Combined with state-of-the-art text-to-SQL model RAT-SQL (Wang et al., 2020), using STRUG as encoder significantly outperforms directly adopting pretrained BERT<sub>LARGE</sub> on the widely used Spider benchmark. We also notice greater improvements on Spider-Realistic and the datasets adopted from Suhr et al. (2020), which further demonstrate the superiority of our pretraining framework in solving the text-table alignment challenge.

(2) STRUG also helps reduce the need for large amount of costly supervised training data. We experiment with the WikiSQL benchmark using limited training data, and show that our pretraining method can boost the model performance by a large margin and consistently outperform existing pretraining methods.

## 2 Related Work

**Cross-Database Text-to-SQL.** Remarkable progress has been made in text-to-SQL within the past few years. With sufficient in-domain training data, existing models already achieve nearly 90% accuracy on single-domain benchmarks like ATIS (Hemphill et al., 1990; Dahl et al., 1994) and GeoQuery (Zelle and Mooney, 1996). However, annotating NL questions with SQL queries is expensive and it is cost-prohibitive to collect training examples for all possible databases. A model that can generalize across domains and databases is desired. In light of this, Yu et al. (2018) present Spider, a cross-database text-to-SQL benchmark that trains and evaluates a system using different databases. More recently, Suhr et al. (2020) provide a holistic analysis of the challenges introduced in cross-database text-to-SQL and propose to include single-domain datasets in evaluation. Their study uncovers the

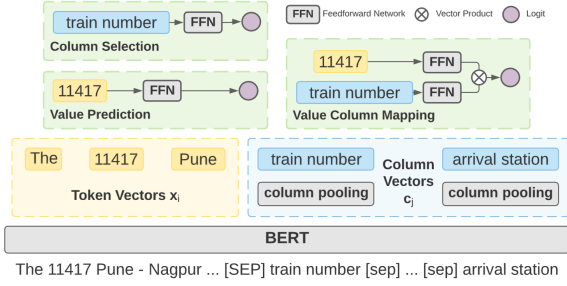


Figure 2: Overview of our model architecture and three pretraining objectives.

limitations of current text-to-SQL models, and demonstrates the need for models that can better handle the generalization challenges.

**Pretraining for Unstructured Text.** Language model (LM) pretraining has shown promising results on improving the generalization ability of natural language processing (NLP) models on many tasks (Devlin et al., 2018; Peters et al., 2018; Liu et al., 2019b; Lewis et al., 2019; Guu et al., 2020). Pretrained LMs like BERT (Devlin et al., 2018) can learn complex characteristics of word use across contexts from large-scale text corpus and apply the learned knowledge to downstream tasks. Aside from language modeling, multi-task learning (MTL) is another popular technique to pre-train a good text representation (Liu et al., 2015, 2019a). Liu et al. (2019a) demonstrate that MTL provides an effective way of leveraging data from many related tasks and is complementary to LM pretraining. However, as these models are usually pretrained with only unstructured text, they do not have knowledge about encoding structured data like database schema, or more importantly how to align them with the NL utterances.

**Pretraining for Text-Table Data.** Inspired by the success of pretrained LMs, some recent work has tried to apply similar pretraining objectives to text-table data. TaBERT (Yin et al., 2020) and TAPAS (Herzig et al., 2020) jointly learn text-table representations by leveraging a large amount of web tables and their textual context. They flatten the tables and use special embeddings to model the structure information. A masked language model (MLM) objective is then used to learn knowledge from the text-table data. However, MLM is originally designed for modeling single NL sentences, making it weak at capturing the association between a pair of sequences, like the alignment between text and tables. More recently, Grappa (Yu

et al., 2020) explores a different direction for pre-training which shares some similarity with existing work on data augmentation for semantic parsing. Grappa first constructs synthetic question-SQL pairs via a synchronous context free grammar induced from existing text-to-SQL datasets, a SQL semantic prediction objective is then used to learn compositional inductive bias from the synthetic data. However, the strength of Grappa mainly comes from the synthetic data. There still lack models that can fully utilize the alignment knowledge naturally present in parallel text-table corpora, which is the aim of this work.

### 3 Weakly Supervised Multitask Pretraining

#### 3.1 Motivation

One of the critical generalization challenges in cross-database text-to-SQL is text-table alignment, i.e., the model needs to understand language and database schemas unseen in training, including value mentions, novel columns and to correctly map between them. Similar generalization challenges have been studied for a long time in NLP field. Recently, pretrained language models have achieved great success in tackling the challenges by learning syntactic and semantic information of words across contexts from a large text corpus. Inspired by this, in this work we aim to leverage a large parallel text-table corpus and learn the use of words in both unstructured text and structured tabular data, as well as the token and cell level association knowledge.

Unlike previous works that use the pretraining corpus to optimize unsupervised objectives like MLM (Herzig et al., 2020; Yin et al., 2020), we aim to directly capture the text-table alignment knowledge via three structure-grounded tasks: column grounding, value grounding and column-value mapping. Our tasks are designed for finding the association between the unstructured text and structured table schema and are closely related to text-to-SQL. As a result, the learned alignment knowledge can be effectively transferred to the downstream task and improve the final performance.

#### 3.2 Pretraining Objectives

We use the same model architecture as BERT, and add simple classification layers on top for the three structure-grounded tasks. For downstream tasks, our model can be easily integrated into existing

Year	Competition	Venue	Position	Event	Notes
1993	European Junior Championships	San Sebastian, Spain	7th	100m	11.74
			3rd	4x100 m relay	44.60
1994	World Junior Championships	Lisbon, Portugal	12th (semis)	100m	11.66 (wind: +1.3 m/s)
			2nd	4x100 m relay	44.78
1995	World Championships	Gothenburg, Sweden	7th (q-finals)	100m	11.54
			3rd	4x100 m relay	43.01

	Human Assisted Setting	Automatic Setting
NL Utterance	Gabriele Becker competed at the <a href="#">1995 World Championships</a> both individually and in the relay.	After winning the German under-23 100 m title, she was selected to run at the <a href="#">1995 World Championships</a> in Athletics both individually and in the relay.
Column Grounding	Year, Competition, Event	Year, Competition
Value Grounding	1995, World Championships	1995, World Championships
Column-Value Mapping	1995-Year, World Championships-Competition	1995-Year, World Championships-Competition

Figure 3: Illustration of the parallel corpus ToTTo (Parikh et al., 2020) and our two weakly supervised pretraining settings. Cell highlighted with yellow are the cell annotations provided by ToTTo, and cell highlighted with dashed lines are cell annotations obtained via string matching in automatic setting.

models as text-table encoder. Following previous work (Hwang et al., 2019; Wang et al., 2020; Guo et al., 2019; Zhang et al., 2019), we linearize the input by concatenating the NL utterance and column headers, with `<sep>` token in between for separation.

Formally, given a pair of NL utterance  $\{x_i\}$  and table with a list of column headers  $\{c_j\}$ , we first obtain the contextualized representation  $\mathbf{x}_i$  of each token in the utterance and  $\mathbf{c}_j$  for each column using the last layer output of the BERT encoder. Here each column header  $c_j$  may contain multiple tokens  $c_{j,0}, \dots, c_{j,|c_j|}$ . We obtain a single vector representation for each column using column pooling. More specifically, we take the output of the first and last token of the header, the column representation is then calculated as  $\mathbf{c}_j = (\mathbf{c}_{j,0} + \mathbf{c}_{j,|c_j|})/2$ .  $\{\mathbf{x}_i\}$  and  $\{\mathbf{c}_j\}$  are then used to compute losses for the three tasks. An overview of our model architecture and pretraining objectives is shown in Fig. 2.

**Column grounding** In text-to-SQL, an important task is to identify grounded columns and fill them into the SQL query. With a parallel text-table corpus, this is similar to select the columns that is mentioned in the associated NL sentence. This task requires the model to understand the semantic meaning of a column based on the its header alone, and infer its relation with the NL sentence based on the contextualized representation. We formulate it as a binary classification task. For each column  $c_j$ , we use a one layer feedforward network  $f(\cdot)$  to get prediction  $p_j^c = f(\mathbf{c}_j)$  of whether  $c_j$  is mentioned in the sentence or not. The column grounding loss

$\mathcal{L}_c$  is then calculated using the binary cross entropy loss with ground truth labels  $y_j^c \in \{0, 1\}$ . Note this task requires the model to identify the meaning of a column without access to any of its values. Hence, it is suitable for the typical text-to-SQL setting where the model only has access to the database schema.

**Value grounding** For clauses like `WHERE` and `HAVING`, to generate an executable SQL query, the model also needs to extract the value to be compared with the grounded column from the NL utterance. This can be transformed to the task of finding cell mentions in the NL sentence with a parallel text-table corpus. The model needs to first identify phrases that are entities and then keep only those related to the associate table. Since the contents of the table is not available, it is necessary for the model to infer the possible value types based on the table schema and use it to filter the values. Similar to column grounding, we also view this as a classification task. For each token  $x_i$ , we get prediction of  $x_i$  being part of a grounded value as  $p_i^v = f(\mathbf{x}_i)$ . The value grounding loss  $\mathcal{L}_v$  is then calculated using the binary cross entropy loss with ground truth labels  $y_i^v \in \{0, 1\}$ .

**Column-Value mapping** As there may be multiple columns and values used in the SQL query, a text-to-SQL model also needs to correctly map the grounded columns and values. This is used to further strengthen the model’s ability to capture the correlation between the two input sequences by learning to align the columns and values grounded in previous two tasks. We formulate this as a match-



ing task between the tokens in the NL sentence and the columns. For every token  $x_i$ , we pair it with each column  $c_j$  and calculate the probability of  $x_i$  matching  $c_j$  as  $p_{i,j}^{cv} = f([\mathbf{x}_i, \mathbf{c}_j])$ . Here  $[\cdot, \cdot]$  is the vector concatenation operation. We then apply a softmax layer over the predictions for each token  $p_i^{cv} = \{p_{i,j}^{cv}\}_{j=1}^{|c|}$ , and the final column-value mapping loss  $\mathcal{L}_{cv}$  is then calculated as  $\mathcal{L}_{cv} = \text{CrossEntropy}(\text{softmax}(p_i^{cv}), y_i^{cv})$ , where  $y_i^{cv} \in \{0, 1\}^{|c|}$  is the ground truth label.

The final loss  $\mathcal{L}$  is the sum of all three losses. We experimented with different weights for the individual losses but did not notice significant improvement on the results, so we only report results with equally weighted losses.

$$\mathcal{L} = \mathcal{L}_c + \mathcal{L}_v + \mathcal{L}_{cv} \quad (1)$$

### 3.3 Pretraining Data

We obtain ground truth labels  $y_j^c$ ,  $y_i^v$  and  $y_i^{cv}$  from a parallel text-table corpus based on a simple intuition: given a column in the table, if any of its cell values can be matched to a phrase in the sentence, this column is likely mentioned in the sentence, and the matched phrase is the value aligned with the column. To ensure high quality text-table alignment information in the pretraining corpus, unlike previous work that use loosely connected web tables and their surrounding text, here we leverage an existing large-scale table-to-text generation dataset ToTTo (Parikh et al., 2020). ToTTo contains 120,761 NL descriptions and corresponding web tables automatically collected from Wikipedia using heuristics. Additionally, it provides cell level annotation that highlights cells mentioned in the description and revised version of the descriptions with irrelevant or ambiguous phrases removed.

We experiment with two pretraining settings, with or without human annotation. In the human assisted setting, we use the cell annotations along with the revised description. More specifically, we first label all the columns  $c_j$  that contain at least one highlighted cell as positive ( $y_j^c = 1$ ). We then iterate through all the values of the highlighted cells and match them with the NL description to extract value mentions. If a phrase is matched to a highlighted cell, we select all the tokens  $x_i$  in that phrase and align them with the corresponding columns  $c_j$  ( $y_i^v = 1$ ,  $y_{i,j}^{cv} = 1$ ). In the automatic setting, we use only the tables and the raw sentences, and obtain cell annotations by comparing each cell

with the NL sentence using exact string matching. Note that in both settings, the cell values are used only for pretraining data construction, not as inputs to the pretraining model.

To make the pretraining more effective and to achieve a better generalization performance, we also incorporate two data augmentation techniques. First, since the original parallel corpus only contains one table for each training example, we random sample  $K_{neg}$  tables as negative samples and append them to the input sequence. This simulates a database with multiple tables and potentially hundreds of columns, which is common in text-to-SQL. Second, we random replace the matched phrases in the NL sentences with values of cells from the same column. This way we can better leverage the contents of the table and improve the model’s generalization ability by exposing it to more cell values.

## 4 Curating a More Realistic Evaluation Set

As one of the first datasets to study cross-database text-to-SQL, Spider has been a widely used benchmark in accessing model’s ability to generalize to unseen programs and databases. However, as pointed out by Suhr et al. (2020), Spider eases the task by using utterances that closely match their paired SQL queries, for example by explicitly mentioning the column names in the question. While in practice, the NL references to columns usually differ from the original column name. To cope with the problem, Suhr et al. (2020) propose to train the model with cross-domain dataset like Spider, and add another eight single-domain datasets like ATIS (Hemphill et al., 1990; Dahl et al., 1994) and Geo-Query (Zelle and Mooney, 1996) for evaluation. However, some of the datasets differ a lot from Spider, introducing many novel query structures and dataset conventions. As we can see from Table 1, the model (Suhr et al., 2020) has very poor performance in some datasets. In light of this, we present a new realistic and challenging evaluation set based on Spider. We first select a complex subset from the Spider development set where there are columns compared against values or used in clauses like ORDER BY. We then manually modify the NL questions in the subset to remove the explicit mention of columns names, except for the columns in the SELECT clause, while keep the SQL queries unchanged. This way we do not introduce extra

Dataset	# Examples (Suhr et al., 2020)	Exec Acc (Suhr et al., 2020)	# Examples Our Filtering	% Col Mentioned
ATIS (Hemphill et al., 1990; Dahl et al., 1994)	289	0.8	275	0.0
Restaurants (Tang and Mooney, 2000)	27	3.7	39	0.0
Academic(Li and Jagadish, 2014)	180	8.2	179	5.2
Yelp(Yaghmazadeh et al., 2017)	54	19.8	68	4.2
Scholar(Iyer et al., 2017)	394	0.5	396	0.0
Advising(Finegan-Dollak et al., 2018)	309	2.3	281	4.0
Spider (Yu et al., 2018)	1034	69.0	1034	39.0
Spider-Realistic	-	-	508	1.8
IMDB(Yaghmazadeh et al., 2017)	107	24.6	111	1.6
GeoQuery(Zelle and Mooney, 1996)	532	41.6	525	3.8

Table 1: Data statistic of existing dataset. Here we show the execution accuracy reported in Suhr et al. (2020). Suhr et al. (2020) filter the original datasets based on rules. We follow the descriptions in Suhr et al. (2020) and report our reproduced results here. % Col Mentioned measures the proportion of examples in the evaluation set where all columns compared against entities in the gold query are explicitly mentioned in the NL utterance.<sup>1</sup>

Example	Type
Show name, country, age for all singers <b>ordered</b> <del>by age</del> from the oldest to the youngest.	Remove
Find the number of concerts happened in the stadium <b>with the highest capacity</b> that can accommodate the most people.	paraphrase
How many pets <b>have a greater weight than 10</b> are over 10 lbs?	

Table 2: Examples of how we create Spider-Realistic from Spider. Phrases shown in bold are exact match to column names.

challenges like adapting to new query structures but make it possible to fairly assess the model’s capability in aligning text and tables. To make a more comprehensive comparison, we also include two datasets from Suhr et al. (2020), IMDB (Yaghmazadeh et al., 2017) and GeoQuery, on which existing models can achieve fair performance without finetuning with in-domain data.

## 5 Experiments

### 5.1 Benchmarks and Base Models

**Spider and the realistic evaluation sets.** Spider (Yu et al., 2018) is a complex cross-database text-to-SQL dataset. It contains 10k complex question-query pairs grounded on 200 databases where multiple tables are joined via foreign keys. In addition, we create a new realistic evaluation set based on the original Spider development set as described in Section 4. We also include IMDB and GeoQuery,

<sup>1</sup>Unlike Suhr et al. (2020), here we do not consider examples where there is no column compared against entity.

two of the eight single-domain datasets used in Suhr et al. (2020), for a more comprehensive comparison. For the baseline model, we use RAT-SQL (Wang et al., 2020) which is the state-of-the-art model according to the official leaderboard. To generate executable SQL queries, we modify the pointer generator in RAT-SQL to enable it to copy values from the question. We use the same trained model for evaluation on the Spider development set and the realistic evaluation sets. Yu et al. (2018) includes some single-domain text-to-SQL datasets like GeoQuery as extra training data for Spider. Here we use IMDB and GeoQuery for evaluation and thus train the model with only the original Spider data, and discard the additional training data used by some previous works. We use both the set match accuracy and execution accuracy from the official Spider evaluation script.

**WikiSQL.** WikiSQL (Zhong et al., 2017) is a large-scale text-to-SQL dataset consists of over 80k question-query pairs grounded on over 30k Wikipedia tables. Although existing models are already reaching the upper-bound performance on this dataset (Hwang et al., 2019; Yavuz et al., 2018), mainly because of the simplicity of the SQL queries and large amount of data available for training. Previous works have used this dataset to demonstrate the model’s generalization ability with limited training data (Yu et al., 2020; Yao et al., 2020). For the baseline model, we use SQLova (Hwang et al., 2019) without execution-guided decoding. Following the official leaderboard, we report both logical form accuracy and execution accuracy. To better demonstrate how our pretraining objectives help with the text-to-SQL task, we also show break-

	Models	Exact	Exec
Schema Only	EditSQL (Zhang et al., 2019) w. BERT	57.6	-
	IRNET (Guo et al., 2019) w. BERT	61.9	-
	RYANSQL (Choi et al., 2020) w. BERT	70.6	-
	Suhr et al. (2020) w. BERT <sub>LARGE</sub> +	65.0	69.0
	RAT-SQL (Wang et al., 2020) w/o value linking		
	w. BERT <sub>LARGE</sub>	67.0	69.8
	w. STRUG (Human Assisted)	70.5	73.3
	w. STRUG (Automatic)	69.8	74.2
Content Used	Global-GNN (Bogin et al., 2019)	52.7	-
	TranX w. TaBERT (Yin et al., 2020)	64.5	-
	RAT-SQL		
	w. BERT <sub>LARGE</sub>	69.8	72.3
	w. GRAPPA	73.4	-
	w. STRUG (Human Assisted)	72.7	75.5
	w. STRUG (Automatic)	72.6	74.9

Table 3: Results on Spider development set. The top half shows models using only database schema, the bottom half shows models using the database content. We train our model three times with different random seeds and report the average performance.

down accuracy for several subtasks.

## 5.2 Implementation Details

For all experiments, we use the BERT implementation from Huggingface (Wolf et al., 2019) and the pretrained BERT<sub>LARGE</sub> model from Google<sup>2</sup>. For pretraining, we use Adam optimizer (Kingma and Ba, 2014) with a initial learning rate of 2e-5 and batch size of 48. In both settings, we use  $K_{neg} = 1$  and pretrains our model for 5 epochs.

For Spider and the realistic evaluation sets, we use the official implementation of RAT-SQL<sup>3</sup> and modify it to generate executable SQL queries. We follow the original settings and do hyperparameter search for learning rate (3e-4, 7.44e-4) and warmup step (5k, 10k). We use the same linear learning rate scheduler with warmup and train for 40,000 steps with batch size of 24. The learning rate for the pretrained encoder (e.g. BERT) is 3e-6 and is frozen during warmup.

For WikiSQL, we use the official SQLova implementation<sup>4</sup>. We use the default setting with learning rate of 1e-3 for the main model and learning rate of 1e-5 for the pretrained encoder. We train the model for up to 50 epochs and select the best model using the development set.

<sup>2</sup>We use the BERT-Large, Uncased (Whole Word Masking) model from [https://storage.googleapis.com/bert\\_models/2019\\_05\\_30/wwm\\_uncased\\_L-24\\_H-1024\\_A-16.zip](https://storage.googleapis.com/bert_models/2019_05_30/wwm_uncased_L-24_H-1024_A-16.zip)

<sup>3</sup><https://github.com/microsoft/rat-sql>

<sup>4</sup><https://github.com/naver/sqlova>

Models	Spider-Realistic		IMDB		Geo	
	Exact	Exec	Exact	Exec	Exact	Exec
RAT-SQL w/o value linking						
w. BERT <sub>LARGE</sub>	46.9	52.4	21.8	26.1	12.4	40.1
w. STRUG (Human Assisted)	53.3	57.8	31.0	35.2	14.4	44.0
w. STRUG (Automatic)	54.9	60.3	30.4	34.5	17.4	48.5
RAT-SQL						
w. BERT <sub>LARGE</sub>	58.1	62.1	14.7	22.4	20.0	46.5
w. STRUG (Human Assisted)	62.2	65.7	18.4	26.4	23.0	57.5
w. STRUG (Automatic)	62.2	65.3	24.2	30.0	23.7	55.7

Table 4: Results on the realistic evaluation sets.

## 5.3 Main Results

**Spider and the realistic evaluation sets.** We first show results on Spider development set in Table 3. The original Spider setting assumes only the schema information about the target database is known in both training and evaluation phase, as the content of the database may not be accessible to the system due to privacy concern. More recently, some works have tried to using the database content to help understand the columns and link with the NL utterance. Here we show results for both settings. In the first setting where only schema information is known, we disable the value-based linking module in RAT-SQL. As we can see from Table 3, replacing BERT<sub>LARGE</sub> with STRUG consistently improves the model performance in both settings. Under the setting where content is available, using STRUG achieves similar performance as GRAPPA and outperforms all other models. GRAPPA uses both synthetic data and larger text-table corpus for pretraining. However, it mainly learns inductive bias from the synthetic data while our model focuses on learning text-table association knowledge from the parallel text-table data. The two pretraining techniques are complementary and we expect combining them can lead to further performance improvement.

Results on the realistic evaluation sets are summarized in Table 4. Firstly, we notice the absolute performance of the model drops significantly on all three datasets, demonstrate that inferring columns without explicit hint is a challenging task and there is much room for improvement. Secondly, comparing results in Table 3 and Table 4, using STRUG brings larger improvement over BERT<sub>LARGE</sub> in the realistic evaluation sets. As shown in Table 1, the original Spider dataset has a high column mention ratio, so the models can use exact match for column grounding without really understanding the utterance and database schema. The realistic evaluation sets simulate the real world scenario and

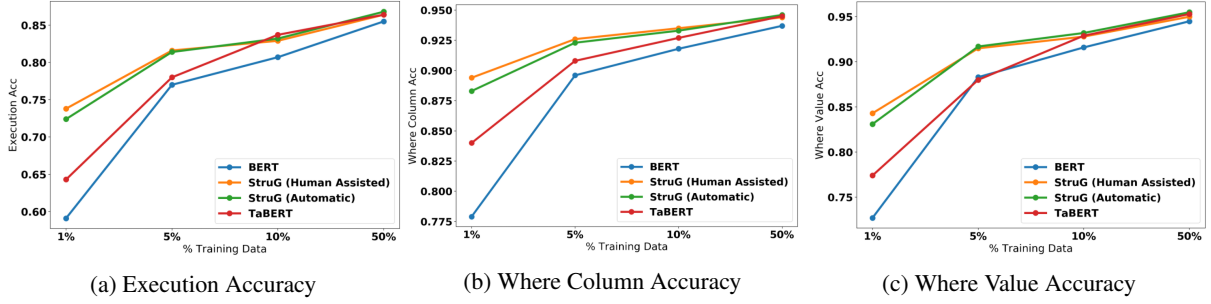


Figure 4: Model performance on the test set with different fractions of training data.

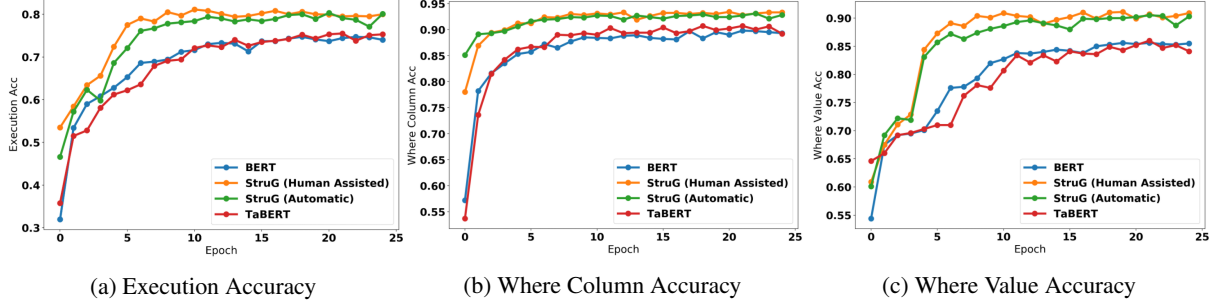


Figure 5: Model performance on the development set during training with 5% of training data.

contain much less such explicit hint, making the text-table alignment knowledge learned by STRUG more valuable. Interestingly, we also notice that using value linking actually hurts the performance on IMDB. One possible explanation is that in IMDB value linking may provide many noisy links, e.g. a person’s name can be linked to multiple columns including actor\_name, writer\_name, character\_name and even film\_title, this may confuse the model and leads to wrong prediction.

In all benchmarks, we notice that STRUG pretrained using the automatic setting actually performs similarly as setting where cell annotations are used. This indicates the effectiveness of our heuristic for cell annotation and the potential to pretrain STRUG with more unsupervised parallel text-table data.

**WikiSQL.** Results on WikiSQL are summarized in Table 5. When using the full training corpus, we notice that using STRUG achieves near the same performance as BERT<sub>LARGE</sub>. This is probably because of the large size of training data and the simple SQL structure of WikiSQL. To better demonstrate that the knowledge learned in pretraining can be effectively transferred to text-to-SQL task and reduce the need for supervised training data, we also conduct experiments with random sampled training examples. From Fig. 4 we can see that with only 1% of training data (around 500 examples), mod-

Models	ACC <sub>lf</sub>	ACC <sub>ex</sub>
HydraNet (Lyu et al., 2020)	83.8	89.2
X-SQL (He et al., 2019)	83.3	88.7
SQLova (Hwang et al., 2019)		
w. BERT <sub>LARGE</sub>	82.1	87.3
w. TaBERT	82.5	87.9
w. STRUG (Human Assisted)	82.1	87.5
w. STRUG (Automatic)	82.4	87.8
SQLova (5%)		
w. BERT <sub>LARGE</sub>	70.7	77.0
w. TaBERT	71.5	78.0
w. STRUG (Human Assisted)	75.6	81.6
w. STRUG (Automatic)	75.6	81.4

Table 5: Performance on WikiSQL. Here we show logical form accuracy and execution accuracy on the test set. (5%) means random sample 5% of original training data for training.

els using STRUG can achieve over 0.70 accuracy, outperforming both BERT<sub>LARGE</sub> and TaBERT by a large margin. Using STRUG brings consist improvement over BERT<sub>LARGE</sub> till using half of the training data, where all models reach nearly the same performance as using the full training data. We also show the training progress using 5% of training data in Fig. 5. We can see that STRUG also helps speed up the training progress.



Models	ACC <sub>S-COL</sub>	ACC <sub>S-AGG</sub>	ACC <sub>W-COL</sub>	ACC <sub>W-VAL</sub>
SQLova (5%)				
w. BERT <sub>LARGE</sub>	95.2	88.4	89.6	88.3
w. TaBERT	95.4	88.4	90.8	88.0
w. STRUG (Human Assisted)	95.5	88.9	92.6	91.5
w. STRUG (Automatic)	95.8	88.9	92.3	91.7

Table 6: Subtask performance on WikiSQL. S-COL, S-AGG, W-COL and W-VAL stands for tasks of predicting SELECT column, aggregation operator, WHERE columns and WHERE values, respectively.

## 6 Conclusion

In this paper, we propose a novel while effective structure-grounded pretraining technique for text-to-SQL. Our approach to pretraining leverages a set of novel prediction tasks to learn knowledge from a parallel text-table corpus to help solve the text-table alignment challenge in text-to-SQL. We design two settings to obtain pretraining labels without requiring complex SQL query annotation: using human labeled cell association, or leveraging the table contents. In both settings, STRUG significantly outperforms BERT<sub>LARGE</sub> in all the evaluation sets. Meanwhile, although STRUG is surprisingly effective that performs in par with models like TaBERT and Grappa using only 120k text-table pairs for pretraining. We believe it is complementary with these existing text-table pretraining methods. In the future, we plan to further increase the size of the pretraining corpus, and explore to incorporate MLM and synthetic data.

## References

- Ben Bogin, Matt Gardner, and Jonathan Berant. 2019. Global reasoning over database structures for text-to-sql parsing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3650–3655.
- Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. 2020a. Logical natural language generation from open-domain tables. *arXiv preprint arXiv:2004.10404*.
- Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Wang. 2020b. Hybridqa: A dataset of multi-hop question answering over tabular and textual data. *arXiv preprint arXiv:2004.07347*.
- DongHyun Choi, Myeong Cheol Shin, EungGyun Kim, and Dong Ryeol Shin. 2020. Ryansql: Recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases. *arXiv preprint arXiv:2004.03125*.
- Deborah A Dahl, Madeleine Bates, Michael K Brown, William M Fisher, Kate Hunicke-Smith, David S Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the atis task: The atis-3 corpus. In *HUMAN LANGUAGE TECHNOLOGY: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.
- 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*.
- Catherine Finegan-Dollak, Jonathan K Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-sql evaluation methodology. *arXiv preprint arXiv:1806.09029*.
- Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-sql in cross-domain database with intermediate representation. *arXiv preprint arXiv:1905.08205*.
- Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*.
- Pengcheng He, Yi Mao, Kaushik Chakrabarti, and Weizhu Chen. 2019. X-sql: reinforce schema representation with context. *arXiv preprint arXiv:1908.08113*.
- Charles T Hemphill, John J Godfrey, and George R Doddington. 1990. The atis spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.
- Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. *arXiv preprint arXiv:2004.02349*.
- Wonseok Hwang, Jinyeong Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on wikisql with table-aware word contextualization. *arXiv preprint arXiv:1902.01069*.
- Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. *arXiv preprint arXiv:1704.08760*.
- Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Oliver Lehmberg, Dominique Ritze, Robert Meusel, and Christian Bizer. 2016. A large public corpus of web tables containing time and context metadata. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 75–76.

- Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. [arXiv preprint arXiv:1910.13461](#).
- Fei Li and HV Jagadish. 2014. Constructing an interactive natural language interface for relational databases. [Proceedings of the VLDB Endowment](#), 8(1):73–84.
- Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation learning using multi-task deep neural networks for semantic classification and information retrieval.
- Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-task deep neural networks for natural language understanding. [arXiv preprint arXiv:1901.11504](#).
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. [arXiv preprint arXiv:1907.11692](#).
- Qin Lyu, Kaushik Chakrabarti, Shobhit Hathi, Souvik Kundu, Jianwen Zhang, and Zheng Chen. 2020. [Hybrid ranking network for text-to-sql](#). Technical Report MSR-TR-2020-7, Microsoft Dynamics 365 AI.
- Ankur P Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqi, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. Totto: A controlled table-to-text generation dataset. [arXiv preprint arXiv:2004.14373](#).
- Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. [arXiv preprint arXiv:1508.00305](#).
- Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. [arXiv preprint arXiv:1802.05365](#).
- 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](#), pages 8372–8388.
- Lappoon R Tang and Raymond Mooney. 2000. Automated construction of database interfaces: Integrating statistical and relational learning for semantic parsing. In [2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora](#), pages 133–141.
- Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In [Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics](#), pages 7567–7578, Online. Association for Computational Linguistics.
- Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface’s transformers: State-of-the-art natural language processing. [ArXiv](#), abs/1910.03771.
- Navid Yaghmazadeh, Yuepeng Wang, Isil Dillig, and Thomas Dillig. 2017. Sqlizer: query synthesis from natural language. [Proceedings of the ACM on Programming Languages](#), 1(OOPSLA):1–26.
- Ziyu Yao, Yiqi Tang, Wen-tau Yih, Huan Sun, and Yu Su. 2020. An imitation game for learning semantic parsers from user interaction. [arXiv preprint arXiv:2005.00689](#).
- Semih Yavuz, Izzeddin Gur, Yu Su, and Xifeng Yan. 2018. What it takes to achieve 100% condition accuracy on wikisql. In [Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing](#), pages 1702–1711.
- Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. [arXiv preprint arXiv:2005.08314](#).
- Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. 2020. Grappa: Grammar-augmented pre-training for table semantic parsing. [arXiv preprint arXiv: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 Radev. 2018. 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. Association for Computational Linguistics.
- John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In [Proceedings of the national conference on artificial intelligence](#), pages 1050–1055.
- Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019. Editing-based sql query generation for cross-domain context-dependent questions. In [Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International](#)

Joint Conference on Natural Language Processing  
(EMNLP-IJCNLP), pages 5341–5352.

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