The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21)

Learning Contextual Representations for Semantic Parsing with Generation-Augmented Pre-Training

Peng Shi[1]*, Patrick Ng[2], Zhiguo Wang[2], Henghui Zhu[2], Alexander Hanbo Li[2], Jun Wang[2], Cicero Nogueira dos Santos[2], Bing Xiang[2]

[1] University of Waterloo, [2] AWS AI Labs

peng.shi@uwaterloo.ca, {patricng,zhiguow,henghui,hanboli,juwanga,cicnog,bxiang}@amazon.com

Abstract

Most recently, there has been significant interest in learning contextual representations for various NLP tasks, by leveraging large scale text corpora to train large neural language models with self-supervised learning objectives, such as Masked Language Model (MLM). However, based on a pilot study, we observe three issues of existing general-purpose language models when they are applied to text-to-SQL semantic parsers: fail to detect column mentions in the utterances, fail to infer column mentions from cell values, and fail to compose complex SQL queries. To mitigate these issues, we present a model pre-training framework, **G**eneration-**A**ugmented **P**re-training (GAP), that jointly learns representations of natural language utterances and table schemas by leveraging generation models to generate pre-train data. GAP MODEL[1] is trained on 2M utterance-schema pairs and 30K utterance-schema-SQL triples, whose utterances are produced by generative models. Based on experimental results, neural semantic parsers that leverage GAP MODEL as a representation encoder obtain new state-of-the-art results on both SPIDER and CRITERIA-TO-SQL benchmarks.

Introduction

Recently, deep contextual language models (Devlin et al. 2018; Liu et al. 2019b; Lewis et al. 2019; Dong et al. 2019; Raffel et al. 2019) have shown their effective modeling ability for text, achieving state-of-the-art results in series of NLP tasks. These models capture the syntactic and semantic information of the input text, generating fine-grained contextual embeddings, which can be easily applied to downstream models. Despite the success of large scale pre-trained language models on various tasks, it is less clear how to extend them to semantic parsing tasks such as text-to-SQL (Warren and Pereira 1982; Popescu, Etzioni, and Kautz 2003; Popescu et al. 2004; Li, Yang, and Jagadish 2006), which requires joint reasoning of the natural language utterance and structured database schema information. Recent work (Guo et al. 2019; Wang et al. 2019; Bogin, Gardner, and Berant 2019b,a) shows that with more powerful pre-trained language models, the highly domain-specific semantic parsers can be further improved, even though these language models are trained for pure text encoding.

Pain Point 1: Fail to match and detect the column mentions. **Utterance**: Which professionals live in a city containing the substring 'West'? List his or her role, street, *city* and state. **Prediction**: `SELECT role_code, street, state FROM Professionals WHERE city LIKE '%West%'` **Error**: Missing column `city` in `SELECT` clause.
Pain Point 2: Fail to infer columns based on cell values. **Utterance**: Give the average life expectancy for countries in Africa which are *republics*? **Prediction**: `SELECT Avg(LifeExpectancy) FROM country WHERE Continent = 'Africa'` **Error**: Missing `GovernmentForm = 'Republic'`.
Pain Point 3: Fail to compose complex target SQL. **Utterance**: Which semesters do not have any student enrolled? List the semester name. **Prediction**: `SELECT semester_name FROM Semesters WHERE semester_id NOT IN (SELECT semester_name FROM Student_Enrolment)` **Error**: Should use `semester_id` in nested SQL to align with the column in `WHERE` clause.

Table 1: Error examples collected from the SPIDER development set based on the RAT-SQL + BERT (Wang et al. 2019).

However, based on error analysis on the output of neural language model-based text-to-SQL systems, we observe that these models can be further enhanced if we could mitigate the following three pain points, which are also illustrated in Table 1. (1) *The model is ineffective to match and detect column names in utterances*. The model should learn to detect column names mentioned in utterances by matching utterance tokens with the schema, and use the matched columns in the generated SQL. The error analysis indicates that, in some cases, models miss some columns when synthesizing the target SQL, while the column is mentioned explicitly in the utterance. (2) *The model fails to infer the columns implicitly from cell values*. This problem is trickier than the first one, because the model is expected to infer the column name based on some cell values mentioned in the utterance, instead of just matching the utterance tokens with the schema. This requires the model to have more domain knowledge. For example, as presented in the second

*Work done while at AWS AI Labs.

[1] This refers to the language models that are pre-trained with GAP framework.

section of Table 1, the model should know `republics` is a `GovernmentForm`. (3) *The model should learn to compose complex queries*. Besides the column selection, to generate a correct SQL, the model should learn to attach the selected columns to the correct clauses. This is a non-trivial task, especially when the target SQL is complex, e.g., when the query is nested. As shown in the last section of Table 1, the model should learn to use corresponding column `semester_id` in the nested SQL, instead of using column `semester_name`.

Recent work has demonstrated that jointly pre-training on utterances and table contents (e.g., column names and cell values) can benefit downstream tasks such as table parsing and semantic parsing (Yin et al. 2020; Herzig et al. 2020). These models are pre-trained using the Masked Language Modeling (MLM) task by either masking tokens from the *utterance* input or tokens from the *schema* input. However, this learning objective can only model the alignment between the utterance and schema implicitly. We hypothesize that, in order to cope with the three pain points previously listed, it is necessary to use pre-training objectives that enforce the learning of contextual representations that better capture the alignment between utterances and schema/table contents.

In this work, we present a language model pre-training framework, **G**eneration-**A**ugmented **P**re-training (GAP), that exploits multiple learning objectives (pre-training tasks) and synthetic data generation to jointly learn contextual representations of natural language utterances and table schema. We propose the following three new learning objectives that not only enforce joint learning but also improve the ability of the model to grasp more domain knowledge, which is helpful in cross-domain scenarios: (1) *column prediction task*, which is a pre-training task that consists in giving a label for each column in the input schema to decide whether it is used in the input utterance or not. This task is intent to improve the column detection ability of the model. (2) *column recovery task*, which consists in randomly replacing some of the column names with one of their cell values and asking the model to recover the original column name either based on the cell value itself or based on the contextual information of the utterance when the column is explicitly mentioned in the utterance. This learning objective is meant to enhance the column inferring ability of the model. (3) *SQL generation*, which consists in generating SQL queries given utterances and schema. This task can boost the ability of the model to compose complex queries by leveraging large scale SQL datasets from the Web.

A key challenge to use the proposed pre-training tasks is training data. Although it is easy to obtain large scale datasets of crawled tables and SQL queries, it is difficult to obtain high-quality utterances interrelated with the tables or logically consistent with crawled SQL queries. Recent work used the surrounding text of tables as a proxy of natural language utterances (Yin et al. 2020; Herzig et al. 2020). However, this option is far from optimal because those texts are dissimilar to user utterances in terms of text length, composition and content. The surrounding text of a table is usually a paragraph, while natural language utterances in the downstream task are short sentences. Furthermore, the content of surrounding text of tables can be quite noisy because the text may be irrelevant to the table. In GAP , we overcome the pre-training data challenge through the use of synthetic data. We propose two sequence-to-sequence (seq2seq) generative models, *SQL-to-text* and *table-to-text*, that can produce large scale datasets with enough quality for pre-training. We train our generative models by finetuning BART (Lewis et al. 2019), a state-of-the-art pre-trained language model. Concurrently, Yu et al. (2020b) and Deng et al. (2020) utilized synthetic data generated from synchronized context-free grammar and existing data-to-text datasets (Parikh et al. 2020) for pre-training, respectively, which requires extra crowd and expert annotation efforts.

The outcome of GAP is a pre-trained model that can be plugged into neural semantic parsers to compute contextual representations of utterances and schema. We apply GAP to text-to-SQL semantic parsing datasets, and experimental results show that systems augmented with GAP outperform state-of-the-art semantic parsers on SPIDER and CRITERIA-TO-SQL datasets. In summary, our work presents the following main contributions:

- Based on an error analysis, we spot three main issues in pre-trained LM-based text-to-SQL semantic parsers.
- We propose a new framework for pre-training semantic parsers that exploits multiple pre-training tasks and synthetic data.
- We present three novel learning objectives that alleviate the three main issues spotted with pre-trained LMs for semantic parsing.
- We propose a novel strategy to overcome pre-training data challenges by leveraging SQL-to-Text and Table-to-Text generative models to generate synthetic data for learning joint representations of textual data and table schema.
- To the best of our knowledge, this is the first work to effectively use both crawled SQL and crawled tables to enhance the text-to-SQL semantic parsing task. Our code is public for future work. [2]

Models

We first present the architecture of the semantic parsers, and then introduce the pre-training model in the GAP framework. Lastly, we describe how to obtain the synthetic pre-training data with generative models.

Text-to-SQL Semantic Parser

The Text-to-SQL semantic parser translates natural language utterances to SQL queries. The semantic parsers in our experiments are based on the encoder-decoder architecture. Given an utterance $U = \{x_1, x_2, ..., x_n\}$ and an schema S consisting of tables $T = \{t_1, t_2, ..., t_{|T|}\}$ and columns $C = \{c_1, c_2, ..., c_{|C|}\}$, we leverage the contextual encoder to obtain the representations of utterance tokens and schema. The decoder is required to compute a distribution $P(Y|X, S)$ over SQL programs. Based on different model designs, the decoder learning target Y can be raw SQL tokens (Zhang

[2] https://github.com/awslabs/gap-text2sql

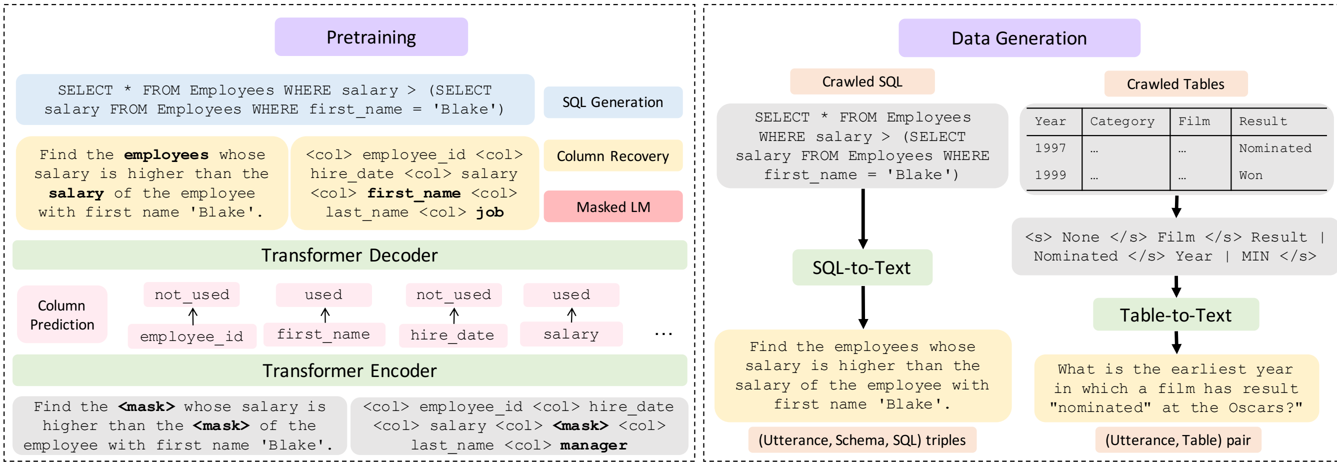

Figure 1: Building blocks of GAP framework. On the left side we illustrate our proposed pre-training tasks. On the right side we depict our proposed data generation strategies.

et al. 2019) or other intermediate representations such as SemQL (Guo et al. 2019) or AST tree (Bogin, Gardner, and Berant 2019b; Yin et al. 2020).

Pre-training Model

The left part of Figure 1 presents an overview of GAP in the pre-training stage. Given an utterance U and schema S, GAP MODEL takes as input the concatenation of U and the column names c in S in the following format $X = \{$<s> U <col> c_1 <col> c_2 ... <col> $c_{|C|}$ </s>$\}$, where c_i denotes the i-th column in schema S. With the 12-layer transformers, each token in the input can be encoded as contextual representations, denoted as $\mathbf{h}$. For different learning objectives, the representations are utilized by different decoders. To jointly learn contextual representations for utterances and schemas and mitigate the three pain points discussed in the intro, we leverage four learning objectives in the pre-training: Besides the Masked Language Model (MLM), we propose learning objectives including Column Prediction (CPred), Column Recovery (CRec), and SQL Generation (GenSQL). Multi-task learning is leveraged for these learning objectives, which helps the model learn to align utterances with schemas and generate correct SQLs.

Column Prediction (CPred): The Column Prediction learning objective encourages the model to capture the alignment signals between the utterance and schema, by predicting whether a column is used in the utterance or not. An illustration is shown in the pink component of Figure 1. Specifically, based on the representations obtained from the transformer encoder, a two-layer MLP is applied on each column representation $\mathbf{g}_{col}$, which is obtained from the output of an average pooling layer that aggregates all sub-tokens of the corresponding column. Afterward, a sigmoid activation function is applied to obtain the probability that the corresponding column is mentioned in the utterance. The GAP MODEL maximizes $P_{\theta_{enc}}(Y_c|X)$ where Y_c is a 0/1 label for a column and X is in its unmasked version.

Column Recovery (CRec): The Column Recovery learning objective strengthens the model's ability to discover the connections between the cell values and the column names, by recovering the column name based on a sampled cell value. For example, as shown in left yellow part of Figure 1, the model recovers the column name `job` from cell value `manager`. Generally, the transformer decoder recovers column names based on two information sources: one is the actual cell value, and the other one is the column name mentions in the utterance. We design the following rules for the value replacement:

- If a column is not mentioned in the utterance, we will replace the column name with its cell value with probability of 0.5. In this case, the column name will be recovered from cell value without other contextual information.
- If a column is mentioned, we will directly replace the column name with its cell value. In this case, the model can leverage the contextual information from the utterance and the cell value to recover the column name.

SQL Generation (GenSQL): This learning objective is directly related to the downstream task. Based on the representation from the transformer encoder, the GAP MODEL decoder maximizes $p_{dec}(y_{sql}|\mathbf{h})$. This learning target encourages the model to learn to compose complex SQL that requires logical reasoning, considering that there are a large number of sophisticated SQLs in crawled data. For example, the GAP MODEL decoder needs to generate the column in appropriate position such as in the `ORDER BY` clause or `WHERE` clause, instead of just predicting the column is used or not. Specifically, the GAP MODEL decoder emits the target SQL token by token with a close vocabulary set, which is composed of the SQL keywords vocabulary and column names. The embeddings of the SQL keywords are randomly initialized and trained during the pre-training phase. The column representations are obtained in the same way as the one used in Column Prediction learning objective, by averaging the column's sub-tokens representations. At each decoding step, the decoder generates a hidden vector and then a dotproduct operation is applied on it and the target vocabulary representations, yielding a probability distribution over the vocabulary set.

Masked Language Model(MLM): We use the standard MLM objective, with a masking rate of 35% sub-tokens in the whole input sequence, including the utterance and schema. Based on the representation from transformer encoder, GAP MODEL employs a transformer decoder to maximize $p_\theta(x|x_m)$ on large scale utterance-schema pairs, where x_m is the masked version of x.

Pre-training Data Generation

As discussed, previous pre-training approaches such as TaBERT (Yin et al. 2020) and TAPAS (Herzig et al. 2020) use the surrounding texts of the tables as a proxy of natural language utterance. However, those texts are noisy and sometimes are not directly related to the table contents. In the downstream task, the input texts are usually utterances/user queries, which are short and highly dependent on the schema and contents of the structured data. In order to minimize the gap between pre-training and downstream tasks, we adopt a state-of-the-art pre-trained sequence-to-sequence model, such as BART, to generate high-quality utterances based on crawled SQLs or structured tables.

As shown in the right part of Figure 1, we design two different models, namely SQL-to-Text generation model and Table-to-Text generation model, for handling the two different inputs. Specifically, the SQL-to-text generation model takes the SQL as input and generates the utterance that explains the query intent. The other model, the Table-to-Text generation model, generates utterances based on a set of sampled column names and cell values from tables. In this way, we can generate utterances interrelated with tables without composing queries that might be suspicious.

SQL-to-Text Generation: We crawl 30K SQLs from GitHub. To generate utterances for these SQL queries, we train a SQL-to-Text model on the SPIDER dataset. The input is the original SQL and it is directly tokenized by the BART tokenizer without additional pre-processing. After finetuning BART, the model can generate high-quality utterances logically consistent with the input SQL, achieving a 0.1934 BLEU score on the development set. Then we use the model to generate utterances for crawled SQLs. We further extract columns and tables in each SQL as positive schema candidates, denoted as $\texttt{schema}_{pos}$. We also sample columns and tables from the pool which are extracted from other SQLs as negative candidates, denoted as $\texttt{schema}_{neg}$. The final schema is composed of these two parts. The utterance-schema-SQL triples are then collected for the GenSQL learning objective in the pre-training phase.

Table-to-Text Generation: Generating utterances from tables is different because query intents are not given. Instead of synthesizing noisy SQLs and then translating into natural language utterances, we propose a Table-to-Text generation model that can directly transform a set of column names and cell values into user queries without query intent constrains. Specifically, we sample column names and cell values (both are referred as candidates) from tables. For example, based on the table in the right part of Figure 1, we can sample columns `Year`, `Film` and `Result`, and a cell value `Nominated`. We then linearize the sampled candidates into {`column name | associated cell value list`} and concatenate them into a sequence, separated by `<sep>` token. Furthermore, to control the complexity and diversity of the generated text, we integrate three types of control codes into the model input:

- Aggregator-based control code: Including `COUNT`, `MAX`, `MIN`, `AVG`, and `SUM`. For the first two sampled columns, we randomly sample an aggregator for each with the probability γ_1 (we use γ_1 as 0.5) if the column type matches with the selected aggregator, e.g., aggregator `SUM` should be applied on numerical type column. If the control codes are sampled, they will be appended to the associated cell value list of the corresponding column.
- Structure control code: Including `IN`, `NOT IN`, `INTERSECT`, `UNION`, and `EXCEPT`. For each example, with probability of γ_2 (we use γ_2 as 0.35), we randomly sample one of them with uniform distribution. Otherwise, `NONE` is used. This control code is used as the first item of the input sequence.
- Order-based control code: We add {`LIMIT` : number} as a part of the control code, which is usually used in an `ORDER BY` based query. With this control code, the generated utterances usually contain phrases that constrain the number of query results should be returned, e.g., *Show the name of aircrafts with top three lowest speed.*.

We fine-tune a BART model on SPIDER dataset to create the generator. To align with our designed input, we convert the SQL into the format we expected. We extract all the columns and their associated aggregators and values from the SQL. We also obtain any special control codes that appears in the SQL. After fine-tuning, the model achieves 0.1821 BLEU score on the development set. Afterwards, we apply the finetuned model to the crawled tables and generate high-quality utterances. The utterance-schema pairs are collected for the learning objectives including MLM, CPred, and CRec in pre-training phase.

For the pre-training step, we need to decide whether a column is mentioned in the utterance or not. To create the label for this, we directly regard all the sampled columns to have positive label. This is based on the assumption that the generation model uses all the columns to synthesize the utterance, and does not have the hallucination issue that models generate some columns names or cell values that are not presented in the input.

Experiments

In the pre-training, we train our GAP MODEL with the underlying transformers initialized with BART (Lewis et al. 2019) model. During the fine-tuning phase, we only leverage the encoder component of the GAP MODEL with 12-layer transformers as the encoder for the semantic parsers.

Tasks, Datasets and Baseline Systems

For the downstream tasks, we conduct experiments on two datasets to show the effectiveness of our framework.

Model	Dev	Test
RyanSQL V2 + BERT (Choi et al., 2020)	0.706	0.606
RAT-SQL v2 + BERT (Wang et al. 2019)	0.658	0.619
AuxNet + BART	0.700	0.619
RAT-SQL v3 + BERT (Wang et al. 2019)	0.697	0.656
RAT-SQL + BERT (our replicate)	0.665	-
RAT-SQL + BART Encoder (ours)	0.676	0.651
RAT-SQL + GAP MODEL (ours)	**0.718**	**0.697**

Table 2: Exact set match accuracy on the public development set and hidden test set of SPIDER.

Model	SQL Acc.	Exec. Acc.
SQLNet (Xu et al. 2017)	0.132	0.139
YXJ (Yu et al. 2020c)	0.142	0.158
YXJ + Roberta (ours)	0.294	0.538
YXJ + BART Encoder (ours)	0.307	0.558
YXJ + GAP MODEL (ours)	**0.327**	**0.594**

Table 3: Test results of Criteria-to-SQL. The SQL accuracy and the execution accuracy are reported.

SPIDER: SPIDER dataset (Yu et al. 2018) is a text-to-SQL dataset with 10,181 annotated parallel utterance-database-SQL triples. The exact set match accuracy is the evaluation metrics. The test set is not publicly available. For baseline parser, we use RAT-SQL (Wang et al. 2019) model as our baseline system to report the end-to-end performance. RAT-SQL model is the state-of-the-art parser in the SPIDER dataset, which leverages the 8-layer relation-aware transformer to model the connections among tables and utterance. To show that the GAP MODEL can be plugged into different neural semantic parsers, we further use IRNet (Guo et al. 2019) model for ablation study. IRNet semantic parser is based on SemQL grammar, which is an effective intermediate representation for SQL. IRNet is efficient in terms of training time, which requires 1 day for training, while RAT-SQL model requires approximately 5 days for training. We augment the encoder part of our GAP MODEL to these base parsers, by replacing their original contextual encoders.

CRITERIA-TO-SQL: CRITERIA-TO-SQL is a dataset to facilitate retrieving eligible patients for a trial from the electronic health record database. The task is to translate the eligibility criteria to executable SQL queries. For example, a criteria statement *any infection requiring parenteral antibiotic therapy or causing fever (i.e., temperature $>$ 100.5f) $\leq$ 7 days prior to registration* is required to be interpreted into SQL `SELECT id FROM records WHERE active_infection = 1 AND (parenteral_antibiotic_therapy = 1 OR causing_fever = 1 OR temperature > 100.5)`. The dataset contains 2003 annotated examples, and the evaluation metrics are the SQL accuracy and execution accuracy. Our baseline system for CRITERIA-TO-SQL dataset is adopted from (Yu et al. 2020c), a slot-filling based model that takes advantages of the prior grammar knowledge to design the sketch. We denote this system as **YXJ** model. The system uses the BERT-base as the contextual encoder.

Spider Results

Table 2 shows the end-to-end results on SPIDER dataset. Based on the codebase[3] provided by Wang et al. (2019), we replicate the RAT-SQL + BERT large model, achieving 0.665 exact set match accuracy on the development set. This matches the RAT-SQL v2 + BERT but still worse than its v3. By replacing the BERT-large with the encoder of BART[4], we obtain accuracy of 0.676 on the development set and 0.651 on test set. The BART Encoder based model achieves comparable results with RAT-SQL v3 + BERT large model on the hidden test set with less encoder layer (BART encoder has 12-layer transformers while BERT large model has 24-layer transformers). With our GAP MODEL, the RAT-SQL can be further augmented, benefiting from enhanced contextual encoding ability. The model achieves accuracy of 0.718 on the development set and 0.697 on the hidden test set. This confirms the effectiveness of the Generation-augmented pre-training. This performance achieves the state-of-the-art performance with less model parameters on SPIDER dataset at the time of writing. Comparing scores of the development set and the test set, we observe BART based models (+BARR Encoder or GAP MODEL) have better generalization ability on the hidden test, considering that the gap between the development set and test set is smaller than the model such as RAT-SQL v3 + BERT.

Criteria-to-SQL Results

Table 3 shows the test results of the CRITERIA-TO-SQL dataset. The YXJ model (Yu et al. 2020c) is built upon BERT-base encoder and sketch-based decoder, achieving the state-of-the-art performance of 0.142 SQL accuracy and 0.158 execution accuracy. We use this system as our baseline. Instead of using the BERT encoder, we augment the model with more powerful pre-trained language models such as RoBERTa and BART. These two pre-trained language models yield significant improvement over the BERT baseline, achieving 0.294 and 0.307 on the SQL accuracy, respectively. After executing the generated SQL queries against the database, these two models obtain 0.538 and 0.558 execution accuracy, respectively. By replacing the BART encoder with GAP MODEL, the parser obtains 2.0% improvement on the SQL accuracy and 3.6% improvement on the execution accuracy, which registers new state-of-the-art performance. This also confirms our assumption that the parsers can benefit from better quality of contextual encoders that jointly reason over utterances and schemas.

Impact of Learning Objectives

We investigate four different learning objectives in this work, namely Masked Language Model (MLM), Column

[3]https://github.com/microsoft/rat-sql

[4]The encoder of BART has 12-layer transformers while BERT-large has 24-layer transformers.

Model	Dev. Acc.	
IRNet + BERT (Ours)	0.620	
IRNet + TaBERT	0.652	
IRNet + RoBERTa (Ours)	0.658	
Learning Objectives	w/o GenSQL	w/ GenSQL
baseline	0.680	0.699
MLM	0.697	0.717
CPred	0.699	0.710
CRec	0.705	0.719
MLM + CPred	0.704	0.716
MLM + CRec	0.711	0.728
MLM + CPred+ CRec	0.715	0.723

Table 4: Ablation study on different learning objectives.

Prediction (CPred), Column Recovery (CRec) and SQL Generation (GenSQL). We conduct the ablation study on SPIDER development set to compare the first three learning objectives under two different conditions: One is with GenSQL learning objective and the other one is without. We use the IRNet based model in the ablation study because it is more efficient in training than RAT-SQL based model, and it can achieve comparable performance. We also want to show that our GAP MODEL is plugin-able and can augment different semantic parsers. Table 4 shows the ablation results.

The first section of the Table 4 shows the results of three baseline systems that are based on IRNet model: IRNet + BERT, IRNet + TaBERT and IRNet + RoBERTa. These results confirm that improving the encoder quality of the semantic parser is a promising direction to pursue.

In the second section of the Table 4, we present detailed ablation study results. Without the GenSQL learning objective, compared with baseline (IRNet + BART Encoder), the three learning objectives (MLM, CPred, CRec) can improve the performance of the parser, with a 1.7%, 1.9% and 2.5% increase, respectively. This indicates that these learning objectives improve the encoding quality of the transformer encoder. Based on the standard unsupervised learning objective MLM, we observe that the CPred and CRec learning objectives are helpful, which lead the model to the accuracy of 0.704 and 0.711, respectively. When we further combine the three learning objectives, the semantic parser's effectiveness is furthered boosted, achieving accuracy of 0.715, a 3.5% increase over its baseline.

With the GenSQL learning objective, the comparison of these three learning objectives is based on a higher baseline with accuracy of 0.699. This indicates that the GenSQL learning objective is valuable. Under this experimental condition, we observe that the MLM learning objective brings consistent improvement over the baseline with 1.8% increase on the accuracy. For the CPred and CRec, the accuracy is boosted by 1.1% and 2.0%, respectively. When we combine the MLM with the CPred, we only observe comparable results with the MLM, without further significant improvement. However, the CRec learning objective brings the MLM a step forward, achieving the 0.728 on the accuracy. The combination of the three learning objectives under w/ GenSQL condition improve 2.4% on accuracy over the baseline. These results show that GenSQLand CRecare two salient learning objectives, leading the model to obtain accuracy more than 0.720, registering a new state-of-the-art performance on public development set on SPIDER.

Learning Objective	Dev. Acc.
baseline	0.680
MLM	0.697
MLM w/o utterance	0.678 (-1.7%)
MLM w/o schema	0.679 (-1.6%)
MLM (surrounding text)	0.679 (-1.6%)
CRev	0.705
CRev w/o utterance	0.688 (-1.7%)
CRev (surrounding text)	0.697 (-0.8%)

Table 5: The ablation study on different inputs for the pre-training based on the IRNet based model.

Analysis of Pre-Training Inputs

Whether to use utterance in pre-training: To prove that the utterance is beneficial in the pre-training, we conduct an ablation study by comparing the pre-trained models which are trained with and without utterance. Our experiments are based on the MLM and CRec learning objectives because the other two (CPred and GenSQL) require the utterance as the input based on their task definitions. Similarly, we use IRNet as our base parser.

The experimental results on SPIDER development set are shown in Table 5. As we can see, if the GAP MODEL is trained with MLM learning objective without utterances as part of the input, the semantic parser performance drops to 0.678 from 0.697, which is lower than the baseline (0.680) by 0.2%. For the CRec learning objective, the accuracy drops from 0.705 to 0.688, a 1.7% decrease, if the GAP is trained without utterance. Even though, CRec learning objective trained without utterances is still helpful, which improves the baseline model by 0.8%. This aligns with our analysis of the CRec learning objective: model can leverage two information sources to recover the column name. If there are no utterances, the model can only use the signals the cell values provide to recover the column name. Furthermore, when the model can access more contextual information, which is provided by the utterance, the model can learn better encoding ability by learning to align the cell values and the column names in the utterances.

Whether to use schema in pre-training: Another input choice is to only keep the utterances in the pre-training. This experimental setting is to justify that the model's improvement is not solely from better utterance representation. This input strategy is only applicable to the MLM learning objective as the schema is a necessary component for other learning objectives. As shown in the MLM w/o schema entry in Table 5, the model performance drops to 0.679, indicating that learning joint utterance and schema representation is necessary for this task.

Whether to use the generated text or the surrounding

Model	Match Acc.
BART Encoder	23.17
GAP MODEL (MLM) Encoder	32.72
GAP MODEL (MLM + CRec) Encoder	36.78
GAP MODEL (MLM + CPred) Encoder	44.51

Table 6: Results of Value-Column Matching Probing Task.

text of the table: The value of the generated text is already justified by the learning objectives such as CPred or GenSQL, because the definitions of these learning objectives require the generated utterances that cannot be obtained from the surrounding text of the table (denoted as surrounding text). Here, we further rationalize our generation-augmented framework on MLM and CRec learning objectives by replacing the generated text with the surrounding text.

The results are presented in the entries of MLM (surrounding text) and CRec (surrounding text) of Table 5. Overall, we can observe that the generation technique is superior to using the surrounding text as a proxy in the MLM and CRec learning objectives, considering the models drop 1.6% and 0.8% on accuracy, respectively. We also find that the CReclearning objective is more robust for pre-training, given that the fine-tuned model performance gets less influence compared with the one with MLM learning objective.

Analysis of Pre-trained Model

As the GAP MODEL provides gains on the text-to-SQL benchmarks, understanding what they learn is important. Following previous work (Liu et al. 2019a; Hewitt and Liang 2019; Hewitt and Manning 2019), we design a probing task, Column-Value Matching (CVM), to examine the extent to which the model can align the cell values in the utterances and the columns, i.e., the probes need to predict which column the cell value belongs to.

Specifically, given the column spans and cell value spans (part of utterances), we can obtain their representations with contextual encoders such as BART or GAP MODEL Encoder, and an average pooling layer. We further compress the representations into another space with linear transformation, denoted as $\{\mathbf{c}_j\}$ and $\mathbf{v}_i$, respectively. The probability of selecting column c_j given cell value v_i is determined by $p(\mathbf{c}_j|\mathbf{v}_i) \propto exp(\mathbf{v}_i\mathbf{c}_j)$. During training, parameters of language model encoders are fixed. Here, we conduct the probing task training on the SPIDER dataset. Note that the unavailability of span annotations of cell values in SPIDER dataset leads to further data pre-processing, which relies on the Levenshtein Distance. The evaluation metric is instance-level accuracy, i.e., the prediction is correct if every cell value used in the utterance is matched with the correct column.

The results are shown in Table 6. We report the accuracy of the BART Encoder model as our probing baseline, which achieves accuracy of 23.17%. With GAP MODEL (MLM) Encoder, the accuracy raises to 32.72%, indicating that the model learns to align the cell values and column names implicitly. By providing stronger supervisions, the MLM+CRec based model and MLM+CPred based models obtain higher accuracy (36.78% and 44.51%), showing that the models capture more alignment signals, contributing to better semantic parser performance.

Related Work

Semantic Parsing: Recently, more interests are concentrated on the SQL-based semantic parsing, and most of the work try to solve the problem with general encoder-decoder architecture. Overall, they enhance the models based on following aspects: (1) Improving the decoding mechanism (Yin and Neubig 2017; Dong and Lapata 2018; Rubin and Berant 2020); (2) Improving the decoding target (Guo et al. 2019); (3) Improving the model encoding ability (Wang et al. 2019; Bogin, Gardner, and Berant 2019a; Yin et al. 2020; Scholak et al. 2020; Ma et al. 2020; Deng et al. 2020; Yu et al. 2020b); (4) Reranking over the generated candidates to improve parses quality (Kelkar et al. 2020; Yin and Neubig 2019). GAP advances the line of (3) by leveraging generation models and three novel learning objectives to enhance the utterance-schema representations.

Question Generation and Table-to-Text Generation: The question generation task is to generate grammatically and semantically correct questions. The generated questions are usually used for enhancing the question answering models (Duan et al. 2017; Guo et al. 2018; Yu et al. 2020a; Zhong et al. 2020). The table-to-text generation task is to generate declarative sentences that describe the information provided by the table (Liu et al. 2017; Gong et al. 2019; Parikh et al. 2020; Radev et al. 2020). Our Table-to-Text model is a combination of these two directions, focusing on generating questions from table, i.e., composing questions based on the sampled columns and cell values, without providing the detailed information about "what to ask".

Pre-training Models: Recent pre-training techniques exploit external knowledge (e.g. entity-level information, commonsense knowledge, knowledge graph) into large-scale pretrained language models (Xiong et al. 2019; Wang et al. 2020; Peters et al. 2019; Rosset et al. 2020). More recently, Yin et al. (2020), Herzig et al. (2020), leverage the semi-structured table data to enhance the representation ability of language models. Concurrently, Yu et al. (2020b) and Deng et al. (2020) leveraged synchronous context-free grammar to generate synthetic data and utilized existing high-quality data-to-text dataset for pre-training, respectively. Different from these work, we explore the direction of utilizing the generators to enhance the joint utterances and structured schema encoding ability of the pre-trained models.

Conclusion

In this work, we spot three pain points in the Text-to-SQL semantic parsing task, and propose a generation-augmented pre-training framework to alleviate them, with four different learning objectives. Experimental results on SPIDER dataset and CRITERIA-TO-SQL dataset show the effectiveness of this framework, which achieves state-of-the-art performance on both datasets.

References

Bogin, B.; Gardner, M.; and Berant, J. 2019a. Global Reasoning over Database Structures for Text-to-SQL Parsing. *arXiv preprint arXiv:1908.11214* .

Bogin, B.; Gardner, M.; and Berant, J. 2019b. Representing schema structure with graph neural networks for text-to-sql parsing. *arXiv preprint arXiv:1905.06241* .

Deng, X.; Awadallah, A. H.; Meek, C.; Polozov, O.; Sun, H.; and Richardson, M. 2020. Structure-Grounded Pretraining for Text-to-SQL. *arXiv preprint arXiv:2010.12773* .

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* .

Dong, L.; and Lapata, M. 2018. Coarse-to-fine decoding for neural semantic parsing. *arXiv preprint arXiv:1805.04793* .

Dong, L.; Yang, N.; Wang, W.; Wei, F.; Liu, X.; Wang, Y.; Gao, J.; Zhou, M.; and Hon, H.-W. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, 13063–13075.

Duan, N.; Tang, D.; Chen, P.; and Zhou, M. 2017. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 866–874.

Gong, H.; Feng, X.; Qin, B.; and Liu, T. 2019. Table-to-text generation with effective hierarchical encoder on three dimensions (row, column and time). *arXiv preprint arXiv:1909.02304* .

Guo, D.; Sun, Y.; Tang, D.; Duan, N.; Yin, J.; Chi, H.; Cao, J.; Chen, P.; and Zhou, M. 2018. Question generation from sql queries improves neural semantic parsing. *arXiv preprint arXiv:1808.06304* .

Guo, J.; Zhan, Z.; Gao, Y.; Xiao, Y.; Lou, J.-G.; Liu, T.; and Zhang, D. 2019. Towards complex text-to-sql in cross-domain database with intermediate representation. *arXiv preprint arXiv:1905.08205* .

Herzig, J.; Nowak, P. K.; Müller, T.; Piccinno, F.; and Eisenschlos, J. M. 2020. TAPAS: Weakly Supervised Table Parsing via Pre-training. *arXiv preprint arXiv:2004.02349* .

Hewitt, J.; and Liang, P. 2019. Designing and interpreting probes with control tasks. *arXiv preprint arXiv:1909.03368* .

Hewitt, J.; and Manning, C. D. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 4129–4138.

Kelkar, A.; Relan, R.; Bhardwaj, V.; Vaichal, S.; and Relan, P. 2020. Bertrand-DR: Improving Text-to-SQL using a Discriminative Re-ranker. *arXiv preprint arXiv:2002.00557* .

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461* .

Li, Y.; Yang, H.; and Jagadish, H. 2006. Constructing a generic natural language interface for an XML database. In *EDBT*. Springer.

Liu, N. F.; Gardner, M.; Belinkov, Y.; Peters, M. E.; and Smith, N. A. 2019a. Linguistic knowledge and transferability of contextual representations. *arXiv preprint arXiv:1903.08855* .

Liu, T.; Wang, K.; Sha, L.; Chang, B.; and Sui, Z. 2017. Table-to-text generation by structure-aware seq2seq learning. *arXiv preprint arXiv:1711.09724* .

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019b. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* .

Ma, J.; Yan, Z.; Pang, S.; Zhang, Y.; and Shen, J. 2020. Mention Extraction and Linking for SQL Query Generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 6936–6942.

Parikh, A. P.; Wang, X.; Gehrmann, S.; Faruqui, M.; Dhingra, B.; Yang, D.; and Das, D. 2020. ToTTo: A Controlled Table-To-Text Generation Dataset. *arXiv preprint arXiv:2004.14373* .

Peters, M. E.; Neumann, M.; Logan IV, R. L.; Schwartz, R.; Joshi, V.; Singh, S.; and Smith, N. A. 2019. Knowledge enhanced contextual word representations. *arXiv preprint arXiv:1909.04164* .

Popescu, A.-M.; Armanasu, A.; Etzioni, O.; Ko, D.; and Yates, A. 2004. Modern natural language interfaces to databases: Composing statistical parsing with semantic tractability. In *Proceedings of the 20th international conference on Computational Linguistics*.

Popescu, A.-M.; Etzioni, O.; and Kautz, H. 2003. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces*. ACM.

Radev, D.; Zhang, R.; Rau, A.; Sivaprasad, A.; Hsieh, C.; Rajani, N. F.; Tang, X.; Vyas, A.; Verma, N.; Krishna, P.; et al. 2020. DART: Open-Domain Structured Data Record to Text Generation. *arXiv preprint arXiv:2007.02871* .

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683* .

Rosset, C.; Xiong, C.; Phan, M.; Song, X.; Bennett, P.; and Tiwary, S. 2020. Knowledge-Aware Language Model Pretraining. *arXiv preprint arXiv:2007.00655* .

Rubin, O.; and Berant, J. 2020. SmBoP: Semi-autoregressive Bottom-up Semantic Parsing. *arXiv preprint arXiv:2010.12412* .

Scholak, T.; Li, R.; Bahdanau, D.; de Vries, H.; and Pal, C. 2020. DuoRAT: Towards Simpler Text-to-SQL Models. *arXiv preprint arXiv:2010.11119* .

Wang, B.; Shin, R.; Liu, X.; Polozov, O.; and Richardson, M. 2019. RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. *arXiv preprint arXiv:1911.04942* .

Wang, R.; Tang, D.; Duan, N.; Wei, Z.; Huang, X.; Cao, C.; Jiang, D.; Zhou, M.; et al. 2020. K-adapter: Infusing knowledge into pre-trained models with adapters. *arXiv preprint arXiv:2002.01808* .

Warren, D. H.; and Pereira, F. C. 1982. An efficient easily adaptable system for interpreting natural language queries. *Computational Linguistics* 8(3-4): 110–122.

Xiong, W.; Du, J.; Wang, W. Y.; and Stoyanov, V. 2019. Pretrained Encyclopedia: Weakly Supervised Knowledge-Pretrained Language Model. *arXiv preprint arXiv:1912.09637* .

Yin, P.; and Neubig, G. 2017. A syntactic neural model for general-purpose code generation. *arXiv preprint arXiv:1704.01696* .

Yin, P.; and Neubig, G. 2019. Reranking for neural semantic parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 4553–4559.

Yin, P.; Neubig, G.; Yih, W.-t.; and Riedel, S. 2020. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. *arXiv preprint arXiv:2005.08314* .

Yu, J.; Quan, X.; Su, Q.; and Yin, J. 2020a. Generating Multi-hop Reasoning Questions to Improve Machine Reading Comprehension. In *Proceedings of The Web Conference 2020*, 281–291.

Yu, T.; Wu, C.-S.; Lin, X. V.; Wang, B.; Tan, Y. C.; Yang, X.; Radev, D.; Socher, R.; and Xiong, C. 2020b. GraPPa: Grammar-Augmented Pre-Training for Table Semantic Parsing. *arXiv preprint arXiv:2009.13845* .

Yu, T.; Zhang, R.; Yang, K.; Yasunaga, M.; Wang, D.; Li, Z.; Ma, J.; Li, I.; Yao, Q.; Roman, S.; et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887* .

Yu, X.; Chen, T.; Yu, Z.; Li, H.; Yang, Y.; Jiang, X.; and Jiang, A. 2020c. Dataset and Enhanced Model for Eligibility Criteria-to-SQL Semantic Parsing. In *Proceedings of The 12th Language Resources and Evaluation Conference*, 5829–5837.

Zhang, R.; Yu, T.; Er, H. Y.; Shim, S.; Xue, E.; Lin, X. V.; Shi, T.; Xiong, C.; Socher, R.; and Radev, D. 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 9th International Joint Conference on Natural Language Processing*.

Zhong, V.; Lewis, M.; Wang, S. I.; and Zettlemoyer, L. 2020. Grounded adaptation for zero-shot executable semantic parsing. *arXiv preprint arXiv:2009.07396* .