UniSAr: A Unified Structure-Aware Autoregressive Language Model for Text-to-SQL

Anonymous ACL submission

Abstract

Existing text-to-SQL semantic parsers are typically designed for particular settings such as handling queries that span multiple tables, domains or turns which makes them ineffective when applied to different settings. We present UNISAR (***Uni***fied ***S***tructure-***A***ware Auto***r***egressive Language Model), which benefits from directly using an off-the-shelf language model architecture and demonstrates consistently high performance under different settings. Specifically, UNISAR extends existing autoregressive language models to incorporate three non-invasive extensions to make them *structure-aware*: (1) adding *structure mark* to encode database schema, conversation context, and their relationships; (2) *constrained decoding* to decode well structured SQL for a given database schema; and (3) *SQL completion* to complete potential missing JOIN relationships in SQL based on database schema. On seven well-known text-to-SQL datasets covering multi-domain, multi-table and multi-turn, UNISAR demonstrates highly comparable or better performance to the most advanced specifically-designed text-to-SQL models. Importantly, our UNISAR is non-invasive, such that other core model advances in text-to-SQL can also adopt our extensions to further enhance performance.[1]

1 Introduction

Text-to-SQL translates a user's natural language question into a corresponding SQL query (Zhong et al., 2017; Yu et al., 2018; Guo et al., 2019; Yu et al., 2019b). This greatly reduces the entry barrier of data analysis to lay users daunted by the technical nuances of SQL. As text-to-SQL techniques matured, enhancements have been proposed to tackle different settings. These enhancements can be roughly organized into three settings of research (Figure 1): (1) *multi-domain* where a parser must generalize to databases in various domains (Zhong et al., 2017; Sun et al., 2020); (2) *multi-table* where the parser must understand the database structure and generate complex SQL query bridging multiple tables (Yu et al., 2018; Wang et al., 2020b); and (3) *multi-turn* where a parser must understand the dialog history, often requiring co-reference resolution and ellipsis recovery (Yu et al., 2019a,b; Guo et al., 2021).

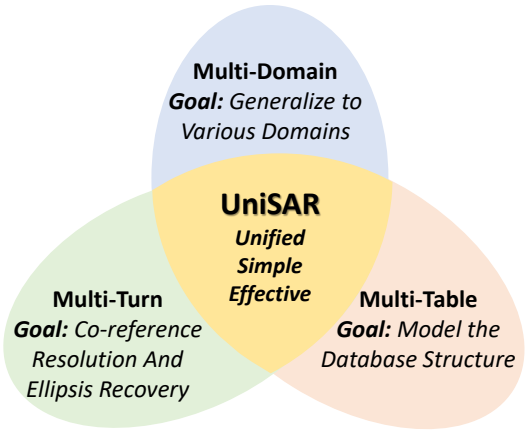

Figure 1: Our view of three main research settings in text-to-SQL.

To address the unique challenges of different settings, researchers have proposed various neural architectures, such as grammar-based decoder for SQL generation (Yin and Neubig, 2018; Guo et al., 2019; Wang et al., 2020a), GNN for database structure modeling (Bogin et al., 2019b), and stacked interaction-layer for context modeling (Cai and Wan, 2020). However, these architectures are prone to 'overfitting' specific datasets, making them non-trivial to adapt to others. For example, the parsers developed for the WikiSQL (Zhong et al., 2017) typically cannot work well on Spider benchmark (Yu et al., 2018).

In this work, we present a ***simple*** yet ***effective*** text-to-SQL parser: UNISAR (***Uni***fied ***S***tructure-***A***ware Auto***r***egressive Language Model). Com-

[1] Codes are public available.http://anonymous.com

pared with the specifically-designed models (invasive), UNISAR is ***simple*** as it does not need any specifically designed DNN modules other than a pre-trained language model (non-invasive). Such a simple architecture gives UNISAR the opportunity to be easily adapted to different datasets, and thus UNISAR enjoys high ***generalizability***. Besides, benefiting from our proposed non-invasive extensions, UNISAR achieves consistently high performance across different settings, which is very ***effective***. Concretely, we propose three non-invasive extensions to make an off-the-shelf autoregressive language model structure-aware. First, we encode structural information (e.g. database schema, conversation context and their relationships) by inserting some special tokens named *structure marks* into the serialized schema and question as inputs. Second, we adopt *constrained decoding* to avoid the decoder generating invalid tokens (e.g., synonyms of schema) during SQL generation. Finally, we propose *SQL completion* to make the SQL complete through inferring potential missing JOIN components based on the database schema.

To prove the effectiveness and generalizability of our UNISAR, we conduct experiments on seven popular text-to-SQL datasets covering multi-domain, multi-table and multi-turn. With a simple and unified architecture, our model achieves comparable or even better performance against task specific models. Importantly, the simple architecture enables different tasks to share the sample training protocol. UNISAR could be easily improved by multi-task training.

To summarize, our contributions are three-fold:

- To the best of our knowledge, we are the first to propose a unified parsing framework for various text-to-SQL settings including multi-domain, multi-table and multi-turn, without reliance on specific architecture designs.
- We make autoregressive language models structure-aware via simply incorporating three non-invasive extensions: structure mark, constrained decoding and SQL completion.
- We conduct extensive experiments on seven text-to-SQL datasets to demonstrate the better effectiveness of our unified parser UNISAR compared with specifically-designed baselines. UNISAR also shows excellent generalizability in joint training and achieves overall improvements.

2 Related Work

Overall, the recent state-of-the-art models for text-to-SQL use various specific architectures for question/schema encoding and SQL query decoding. Take some popular models for example. To joint encode the question and schema, Xu et al. (2017) proposed column attention strategy to gather information from columns for each question word. EditSQL (Zhang et al., 2019) considered co-attention between question words and database schema nodes. Bogin et al. (2019a) dealed with the graph structure of database schema via GNN. RAT-SQL (Wang et al., 2020a), utilized a complete relational graph attention neural network to handle various pre-defined relations. To ensure the syntactic and semantic correctness of the generated SQL query, existing works (Guo et al., 2019; Wang et al., 2020a) usually adopted grammar-based decoder. In contrast, our UNISAR provides a unified way to encode structural information and decode valid SQL. It is very simple as it does not need specific designed modules (non-invasive) and also incredibly effective like these invasive approaches.

We use the same task formulation as (Dong and Lapata, 2016; Zhong et al., 2017; Lin et al., 2018; Raffel et al., 2020) that treated text-to-SQL as a translation problem. However, they only solve it by applying seq2seq models on sequence of tokens. Thus they can't productively encode the input structure. We extend their work with three non-invasive extensions to effectively encode the input structure. To the best of our knowledge, this is the first time that seq2seq parsers are on par with specially-designed parsers in various text-to-SQL settings.

Recently, Scholak et al. (2021) published PICARD, a different approach to utilizing language models for text-to-SQL, which constrains autoregressive decoders of language models through incremental parsing. This differs from UNISAR in two important ways: (1) they only considered the structure in the decoding phrase but ignored the input structure, which limits the upper bound of performance; (2) they basically relied on T5-3B to achieve good results. To contrast, UNISAR introduces the structure knowledge of input to make language model as effective in modeling as task-specific models. Moreover, UNISAR is built on moderately-sized language model BART-Large (400M). Benefiting from three simple extensions, UNISAR achieves a comparable or better performance to their model.

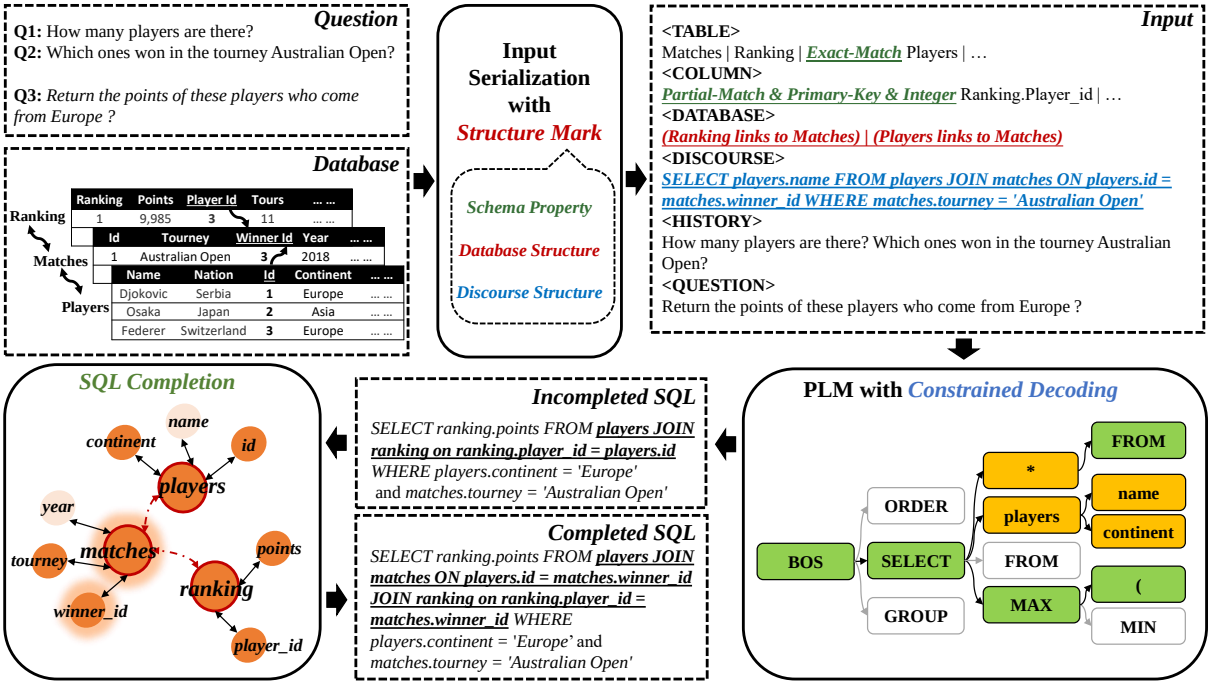

Figure 2: Three non-invasive extensions to make PLMs become structure-aware: *(1)* add ***structure mark*** to encode database schema, conversation context and their relationships; *(2)* ***constrained decoding*** to decode well-structured SQL; *(3)* ***SQL completion*** through inferring the potential missing JOIN components based on the database schema.

3 Methodology

Overall, we employ pretrained autoregressive language models as the backbone of UNISAR since they exhibit excellent adaptability and generalizability in many NLP tasks. Furthermore, as shown in Figure 2, we propose three non-invasive extensions to make PLMs become structure-aware: *(1)* we encode structural information (e.g. database schema, conversation context and their linking relationships) by inserting some special tokens named *structure marks* into the serialized schema and question as inputs; *(2)* we adopt *constrained decoding* to decode well-structured SQL via simply filtering invalid tokens (e.g., synonyms of schema) during beam search. *(3)* we propose *SQL completion* to infer the underlying JOIN relationships in the SQL statement based on the predicted incomplete SQL and the database schema.

3.1 Pretrained Language Model

In our experiments, we implement UNISAR on top of BART (Lewis et al., 2020), a widely used pre-trained encoder-decoder model. BART follows a standard sequence-to-sequence Transformer architecture (Vaswani et al., 2017). It is pre-trained via corrupting sentences (e.g., randomly sampling spans and masking each one) and then optimizing a reconstruction loss. We employ the BART-Large for English and the mBART-Large for Chinese.[2]

3.2 Formulate Text-to-SQL as Seq2Seq

The input for UNISAR contains a series of NL questions and their corresponding database. An concrete example of multi-turn is listed in Figure 2. Encoding the NL sentence is relatively straightforward, while encoding the table is non-trivial since it exhibits underlying structures. In practise, we linearize the table into a flatten sequence so that it can be fed directly into the model. By inserting several special tokens to indicate the table boundaries, a linearized table can be represented as $T = \texttt{[TABLE]}, t_1, \cdots, t_N, \texttt{[COLUMN]}, c_1, \cdots, c_N$. [3] Here `[TABLE]` and `[COLUMN]` are special tokens indicating the region of table headers and column names respectively. Notably, we also separate headers or cells in different columns using a vertical bar '|'. Finally, we concatenate the linearized database T with the NL sentence $\mathbf{x}$ and feed them to the encoder. For multi-turn settings, we append the previous history question in reverse chronological order and put them ahead of the current question.

[2] For brevity, we use BART in the following tables.

[3] If the model requires to predict the value, we attach the values of c_i behind c_i and separate each value with symbol '&'.

3.3 Structure Mark

Structure information plays an important role in text-to-SQL. Structure includes database schema, conversation context and their relationships. Recently, in the research line of prompt tuning, Aghajanyan et al. (2021); Chen et al. (2021) make a step by making prompts with additional marks (some special tokens) to encode various structure information. Inspired by their works, we explore the idea of encoding structure information by designing ***structure mark*** and inserting them into the input to make model structure-aware for text-to-SQL.

In fact, we could simply extract structure information from the input and leverage all these structure information to improve the model. Concretely, the structure information can be roughly organized into three types: (1) ***schema property*** to expand the semantic information of schema; (2) ***database structure*** to aggregate the information from schema neighbors; (3) ***discourse structure*** to supply the conversation context in history question. We give a valid example in Figure 2, which shows how to serialize the database schema and insert structure mark into the input.

3.3.1 Schema Property

In a vanilla formulation, the semantic representation of schema only relies on the surface name that easily leads to disambiguation. For example, we could not tell the tiny difference between COUNTRY.CONTINENT and COUNTRY.REGION. However, there exists some obvious and available schema property to enrich the semantic information of schema, which will improve the correctness of alignment between question and database and finally boost the text-to-SQL performance. The information includes (1) the internal schema information from the database schema, such as primary key or column type (INT, STRING or DATE); (2) the name-based linking information between question and schema that serve as the prior of schema linking; (3) the value-based linking information that augment the column representation via leveraging the database content information.

Overall, it is relatively simple to obtain the schema property. Specifically, for the internal schema information, we could derive them from the database definition. For the name-based linking information, we enumerate the n-gram of question and schema then examine if they are aligned. To fine-grained model the alignment, we also add a prefix (exact or partial) before the 'match'. For example, as shown in INPUT of Figure 2, column PLAYER_ID has partially overlapped with the token 'Player', thus we attach PARTIAL_MATCH before the column PLAYER_ID. In order to compute this alignment, we simply derive the schema-linking results using fuzzy string-match following (Sun et al., 2018; Guo et al., 2019; Wang et al., 2020a). For the value-based linking information, we first normalize their data formation (e.g. uniform the representation of date) then match them with the token in question. After obtaining these three types of schema property information, we insert them as prefixes ahead of the schema. Let's take an example for explanation: '***Partial-Match & Primary-Key & Integer Ranking.Player_id***'. In this example, ***Ranking.Player_id*** is the column. The prefix with underline are structure mark that express two schema properties: (1) ***Partial-Match*** indicates that ***Ranking.Player_id*** partially alignment with the question; (2) ***Primary-Key*** and ***Integer*** are all column properties.

3.3.2 Database Structure

The database structure could improve the representation of the schema via aggregating information from neighboring nodes. As shown in figure 2, the database structure includes (1) the affiliation relations between columns and tables (e.g., ID of MATCHES); (2) the foreign key relations between columns (e.g., WINNERID links to PLAYERID); (3) the tables relations (e.g., MATCHES links to RANKING).

For affiliation relations, we attach the columns with their affiliated table such as MATCHES.ID. For the other two relations, we suppose the tables relations already includes the information of foreign key relations. Thus, we only consider the tables relations here. We adopt the template *'schema1 links to schema2'* and fill in table names. Then concatenate these pairs together and put them into input. In preliminary experiments, we find that affiliation relations and the tables relations could improve the model further. As for affiliation relations, they regulate the generation of column with correct table (e.g. RANKING.YEAR but YEAR comes from MATCHES). As for tables relations, it would directly affect the prediction of tables in FROM clause.

3.3.3 Discourse Structure

The SQL of previous turn provides the mentioned schema and potential intent in current turn (Zhang et al., 2019; Cai and Wan, 2020), which mitigates the burden in entity co-reference resolution and intent ellipsis recovery. Based on this observation, we insert previous SQL in the input to improve the modeling of discourse in terms of entity and intent. We also explore that only inserting the schema from the last turn, and find the model has also been improved to a certain extent.

3.4 Constrained Decoding

Decoding well-structured SQL means that generated SQL not only obeys the SQL grammar but also is faithful to the database schema. We notice that BART is already skilled at learning SQL grammar. However, it sometimes struggles in schema prediction. For example, BART might outputs NATION instead of CITIZENSHIP , which is the synonym of NATION. To alleviate the problem of schema prediction, we first construct the prefix-trie based on the database schema and then filter out the illegal token during the beam search as shown in Figure 2. Note that the difference between UNISAR and grammar-based parser is that we do not need to specify the concrete grammar. As a comparison, UNISAR learns grammar through training, which shows better generalization to various SQL. [4]

3.5 SQL Completion

In our study, we found that the generated SQL statements often miss some JOIN components, since they are often not explicitly mentioned in natural language questions. ***To make the SQL complete***, we need to find back the potential missing JOIN components based on the database schema. Concretely, we first construct a schema graph, where the nodes are tables or columns, and edges are schema relationships. Then we try to find the tables and columns that are located in the shortest path of the existing tables and columns in an incompleted SQL. Take the case in Figure 2 for example. It's an incompleted SQL that does not mention table MATCHES and column WINNER_ID in FROM clause. We infer these two schemas based on their neighbors: PLAYERS and RANKING. MATCHES is located on the path of these two tables. And WINNER_ID is the primary key of MATCHES.

[4] UNISAR learns this complex SQL template efficiently: ***SELECT A (SQL) OP SELECT B (SQL)*** (Wang et al., 2020b). But it's bothersome to design the grammar.

4 Experimental Setup

4.1 Datasets and Evaluation Metrics

We compare UNISAR with previous task-specific parsers using seven popular text-to-SQL benchmark datasets. The dataset statistics are shown in Table 8 (Appendix A). To systemically compare our unified parser with previous task-specific models, we divided the dataset into three group to study the effectiveness of UNISAR: (1) **Multi-Domain**: WikiSQL (Zhong et al., 2017) and TableQA (Sun et al., 2020); (2) **Multi-Table**: Spider (Yu et al., 2018) and DuSQL (Wang et al., 2020b); (3) **Multi-Turn**: CoSQL (Yu et al., 2019a), SParC (Yu et al., 2019b) and Chase (Guo et al., 2021). [5]

For WikiSQL and TableQA, we utilize logic form accuracy (LX) and execution accuracy (EX) as evaluation metrics following Zhong et al. (2017). For Spider and DuSQL, we report exact set match accuracy (EM) following Yu et al. (2018). For SParC, CoSQL and Chase, we report question match accuracy (QM) and interaction match accuracy (IM) following Yu et al. (2019b).

4.2 Baselines

For each setting, we select representative invasive models and non-invasive models as our baselines.

Multi-Domain (1) SQLNet (Xu et al., 2017) is a sketch-based method. (2) SQLova (Hwang et al., 2019) is a sketch-based method which integrates the pre-trained language model; (3) Coarse2Fine (Dong and Lapata, 2018) first generates the SQL template then fills the value. (4) X-SQL (He et al., 2019) enhances the structural schema representation with the contextual embedding. (5) F-SQL (Zhang et al., 2020) improves the representation of schema with table content. (6) HydraNet (Lyu et al., 2020) uses column-wise ranking and decoding. (7) BRIDGE (Lin et al., 2020) further leverages the database content to augment the column representation. (8) SeaD (Xuan et al., 2021) trains a seq2seq model with schema-aware denoising objectives.

Multi-Table (1) RYANSQL (Choi et al., 2020) recursively predicts nested queries with sketch-based slot filling algorithm; (2) IRNet (Guo et al.,

[5] UNISAR still shows a powerful performance in an internal single-domain dataset. Note that for Spider, CoSQL, SparC and DuSQL, we conduct experiments on dev set since the test set is not publicly available.

Model	WikiSQL Dev		WikiSQL Test		TableQA Dev		TableQA Test	
	LX	EX	LX	EX	LX	EX	LX	EX
Invasive Approaches								
SQLNet (Xu et al., 2017)	-	69.8	-	68.0	-	-	61.4	67.2
Coarse2Fine (Dong and Lapata, 2018)	72.5	79.0	71.7	78.5	-	-	72.6	76.7
SQLova (Hwang et al., 2019)	81.6	87.2	80.7	86.2	-	-	81.7	85.8
X-SQL (He et al., 2019)	83.8	89.5	83.3	88.7	-	-	83.3	87.6
F-SQL (Zhang et al., 2020)	-	-	85.6	**91.4**	-	-	90.4	93.2
HydraNet (Lyu et al., 2020)	83.6	89.1	83.8	89.2	-	-	-	-
BRIDGE (Lin et al., 2020)	86.2	**91.7**	85.7	91.1	-	-	-	-
Non-invasive Approaches								
BART-Large (Lewis et al., 2020)	83.7	89.4	82.8	88.8	88.7	91.8	90.7	94.4
SeaD (Xuan et al., 2021)	84.9	90.2	84.7	90.1	-	-	-	-
UNISAR	**86.7**	**91.7**	**85.8**	**91.4**	**89.9**	**92.1**	**91.8**	**95.1**

Table 1: Logical Form Accuracy (LX) and Execution Accuracy (EX) of the multi-domain setting. Note that we report the models without using Execution-Guided Decoding.

Model	Spider	DuSQL
Invasive Approaches		
RYANSQL (Choi et al., 2020)	66.6	-
IRNet (Guo et al., 2019)	63.9	38.4
IRNetExt (Wang et al., 2020b)	-	59.8
RAT-SQL (Wang et al., 2020a)	69.7	-
BRIDGE (Lin et al., 2020)	70.0	-
Non-invasive Approaches		
T5-Base (Shaw et al., 2021)	57.1	-
T5-Large (Scholak et al., 2021)	65.3	-
T5-Large† (Scholak et al., 2021)	69.1	-
BART-Large (Lewis et al., 2020)	64.5	82.2
UNISAR	**70.1**	**84.3**

Table 2: Exact-set-match accuracy (EM) of the multi-table setting. The PLM with † indicates the usage of constrained decoding (Scholak et al., 2021).

Model	SParC QM	SParC IM	CoSQL QM	CoSQL IM	Chase QM	Chase IM
Invasive						
EditSQL	47.2	29.5	39.9	12.3	37.7	17.4
IGSQL	50.7	32.5	44.1	15.8	41.4	20.0
RAT-SQL	60.1	38.6	50.8	20.1	35.1	14.6
Non-invasive						
BART	55.0	36.5	47.1	18.8	37.6	19.5
UNISAR	**60.4**	**40.8**	**51.8**	**21.3**	**42.2**	**22.3**

Table 3: Question Match (QM) and Interaction Match (IM) of multi-turn setting.

2019) utilizes SemQL as an abstraction representation of SQL queries; (3) RAT-SQL (Wang et al., 2020a) utilizes a complete relational graph attention neural network to handle various pre-defined relations; (4) IRNetExt (Wang et al., 2020b) extends IRNet to parse calculation questions and predict values.

Multi-Turn (1) EditSQL (Zhang et al., 2019) adopts an editing-based encoder-decoder model; (2) IGSQL (Cai and Wan, 2020) proposes schema interaction graph encoder to utilize historicalal information of database schema items; (3) RAT-SQL-con (Wang et al., 2020a) is the extension of RAT-SQL for multi-turn settings.

4.3 Implementation Details

We conduct the experiments using the Fairseq (Ott et al., 2019) to do data pre-processing, training and inference. Our training can be done within ***10 hours*** using four V100-16G GPUs or day and a half using one V100-16G GPU.[6] The hyper-parameters could be found in Appendix B. Implementation of constrained decoding is based on Cao et al. (2021a,b).

5 Results and Analysis

5.1 Main Results

The results of multi-domain, multi-table and multi-turn are listed in Table 1, 2 and 3 respectively. We run each UNISAR experiment three times with different random seeds and report the mean. Most results of previous models are reported by cited papers respectively.[7] *As we can see from the results,* UNISAR *achieves the most consistently strong performance under all settings, which demonstrates the effectiveness of our unified architecture.*

[6] As a comparison, training RAT-SQL (Wang et al., 2020a) takes 7 days with one V100-16G GPU.

[7] For WikiSQL, we re-implement RAT-SQL with BERT-Large. For multi-turn settings, the results of invasice model are reported by Guo et al. (2021).

Model	#Params	Spider	CoSQL	SParC
T5-Base	220M	57.1	-	-
T5-Base†	220M	65.8	-	-
T5-Large	770M	65.3	-	-
T5-Large†	770M	69.1	-	-
T5-3B	3B	69.9	**53.8**	-
BART-Large	400M	64.5	47.1	55.0
UNISAR	400M	**70.1**	51.8	60.4

Table 4: Logical Form Accuracy (LX) of using different sizes of PLM for text-to-SQL tasks. The PLM with † indicates the usage of constrained decoding (Scholak et al., 2021).

Model	Spider	CoSQL	SParC
UNISAR	**70.1**	**51.8**	**60.4**
- SM	66.9(-3.2%)	48.7(-3.1%)	57.4(-3.0%)
- CD	67.5(-2.6%)	50.1(-1.7%)	57.8(-2.6%)
- SC	68.8(-1.3%)	50.8(-1.0%)	58.9(-1.5%)

Table 5: The ablation study results of Logical Form Accuracy on UNISAR. SM stands for structure mark, CD stands for constrained decoding, and SC stands for SQL completion.

Model	Setting	WikiSQL Dev	WikiSQL Test	Spider Dev
RAT-SQL	single	78.0	78.2	69.7
	joint	77.5	77.7	68.8
	-	-0.5	-0.5	-0.9
BART	single	83.7	82.8	64.5
	joint	83.4	83.7	63.9
	-	-0.3	+0.9	-0.6
UNISAR	single	86.7	85.8	70.1
	joint	86.5	86.8	68.9
	-	-0.2	+1.0	-1.2

Table 6: Logical Form Accuracy (LX) of joint training with WikiSQL (single-table) and Spider (multi-table). Note that RAT-SQL is trained with BERT-Large.

Model	Setting	Spider	CoSQL	SParC
RAT-SQL	single	69.7	50.8	60.1
	joint	68.2	50.0	58.2
	-	-1.5	-0.8	-1.9
BART	single	64.5	47.1	55.0
	joint	67.5	50.1	57.8
	-	+3.0	+3.9	+2.8
UNISAR	single	70.1	51.8	60.4
	joint	70.8	52.9	60.9
	-	+0.7	+1.1	+0.5

Table 7: Logical Form Accuracy (LX) of joint training across single-turn (SPider) and multi-turn (CoSQL and SParc) datasets. Note that RAT-SQL and IGSQL are trained with BERT-Large.

Moreover, we compare UNISAR with other existing PLMs in Table 4. It can be observed that UNISAR outperforms most of these PLMs even though they have much more parameters. Note that UNISAR even slightly surpasses T5-3B (8x larger) in Spider. It demonstrates the effectiveness of our non-invasive approaches to make the PLM structure-aware for text-to-SQL.

5.2 Ablation Study

To study the effect of our non-invasive approaches: structure mark, constrained decoding and SQL completion, we conduct ablation study on Spider, CoSQL and SParC. As show in Table 5, the structure mark significantly boosts the performance of BART, implying that it does effectively represent the structure knowledge for autoregressvie language models. At the same time, from the view of decoding, the constrained generation solution also boosts the performance by a large margin, since it alleviates the problem of generating invalid SQL. As a complement to the above two extensions, SQL completion further improves the performance by inferring underlying tables in post-processing.

5.3 Effect of Oracle Structure Mark

To explore the upper-bound of performance boost that structure mark brings, we conduct experiments under oracle setting. Concretely, we derive oracle schema-linking for Spider by human annotation from Lei et al. (2020) and oracle previous SQL for CoSQL, SParC and Chase. Experimental results show that Spider gets a 1.2% performance boost. And for these multi-turn datasets (SParC, CoSQL, Chase), they receive 5.9%, 8.7%, 11.7% boost respectively. It indicates the trends that UNISAR will improve further if we could obtain higher-quality structure mark.

5.4 Generalization Across Different Datasets

The simple architecture of UNISAR enables different tasks to share the sample training protocol. It's easy to perform multi-task train. To examine this, we conduct two types of multi-task: (a) single-table + multi-table and (b) single-turn + multi-turn. For single-table and multi-table, we train WikiSQL and Spider jointly (Table 6). It can be seen that all these three models degrade. But BART and UNISAR generally show robustness to the unbalanced distribution of data amount (80k vs. 10k) and SQL format (simple vs. complex). For single-turn and multi-turn, we train Spider, CoSQL and SParC

Question: *Which template type code is used by most number of documents ?*
BART: SELECT templates.template_type_code FROM templates GROUP BY templates.template_type_code ORDER BY count (*) DESC LIMIT 1
UNISAR: SELECT templates.template_type_code FROM templates **JOIN documents** GROUP BY templates.template_type_code ORDER BY count (*) DESC LIMIT 1

Question: *How many contestants did not get voted ?*
BART: SELECT COUNT (*) FROM **competitor** WHERE **competitor**.contestant_number NOT IN (SELECT votes.contestant_number FROM votes)
UNISAR: SELECT COUNT (*) FROM **contestants** WHERE **contestants**.contestant_number NOT IN (SELECT votes.contestant_number FROM votes)

Question: *What are the different models for the cards produced after 1980 ?*
BART: SELECT DISTINCT model_list.model FROM cars_data JOIN model_list WHERE cars_data.year > 1980
UNISAR: SELECT DISTINCT model_list.model FROM model_list **JOIN car_names** JOIN cars_data on car_names.makeid = cars_data.id WHERE cars_data.year > 1980

Question: *What is the version number of template id 3 ? | What is the document name of template id 25 ?*
BART: SELECT documents.document_name FROM documents WHERE documents.template_id = 25
UNISAR: SELECT documents.document_name FROM documents **JOIN templates** WHERE templates.template_id = 25

Question: *what are airlines that have flights arriving at airport 'AHD' ?*
BART: SELECT airlines.airline FROM airlines JOIN flights WHERE flights.destairport = "AHD"
UNISAR: SELECT airlines.airline FROM airlines JOIN flights **JOIN airports** on where **airports.airportname = "AHD"**

Figure 3: Case study: the first four cases are positive samples while the last one is negative. `FROM` conditions are omitted here for brevity.

jointly (Table 7). It indicates that both BART and UNISAR show a positive trend by jointly training, whereas RAT-SQL not. The divergence could be attributed to the task-specific modules (e.g. grammar-based decoder) which make the model easily overfit to specific dataset.

5.5 Case Studies

In Figure 3, we compare the SQL generated by our model UNISAR with those created by the vanilla BART. We notice that UNISAR performs better than the BART, especially on examples that involve the `JOIN` operation of multiple tables. For example, in the first case where BART fails to identify the existence of table DOCUMENTS. For comparison, UNISAR successfully predicts the connection of two tables since the structure mark presents the database structure. In the second case, vanilla BART predicts a token (e.g., COMPETITOR) that does not exist in the database schema. This will cause an ill-formed SQL but UNISAR ensures the faithful generation with constrained decoding. Moreover, in the third case, we can find that SQL completion infers the missing table CAR_NAMES based on the matched table MODEL_LIST and CARS_DATA. In the fourth case from multi-turn setting, UNISAR still outperforms the BART in contextual modeling with effectively encoding the information of dialogue history. However, in the last case, our UNISAR is awkward to predict unnecessary table AIRPORTS. This error perhaps can be attributed to inappropriate structure mark due to inaccurate schema-linking. A high precision structure mark will alleviate this problem.

6 Conclusion

In this paper, we propose UNISAR, a simple-yet-effective model to solve text-to-SQL across various settings in a unified framework. Concretely, we simply extend BART, an off-the-shelf autoagressive language model, with three non-invasive extensions to make it structure-aware. Experimental results demonstrate the effectiveness of our UNISAR on seven well-known text-to-SQL datasets. UNISAR achieves highly comparable or better performance to the most advanced specifically-designed text-to-SQL models. UNISAR also shows excellent generalizability in joint training and achieves overall improvements.

References

Armen Aghajanyan, Dmytro Okhonko, Mike Lewis, Mandar Joshi, Hu Xu, Gargi Ghosh, and Luke Zettlemoyer. 2021. Htlm: Hyper-text pre-training and prompting of language models. *ArXiv*, abs/2107.06955.

Ben Bogin, Jonathan Berant, and Matt Gardner. 2019a. Representing schema structure with graph neural networks for text-to-SQL parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4560–4565.

Ben Bogin, Matt Gardner, and Jonathan Berant. 2019b. 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 3659–3664, Hong Kong, China. Association for Computational Linguistics.

Yitao Cai and Xiaojun Wan. 2020. IGSQL: Database schema interaction graph based neural model for context-dependent text-to-SQL generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6903–6912, Online. Association for Computational Linguistics.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021a. Autoregressive entity retrieval. In *International Conference on Learning Representations*.

Nicola De Cao, Ledell Wu, Kashyap Popat, Mikel Artetxe, Naman Goyal, Mikhail Plekhanov, Luke Zettlemoyer, Nicola Cancedda, Sebastian Riedel, and Fabio Petroni. 2021b. Multilingual autoregressive entity linking. In *arXiv pre-print 2103.12528*.

Xiang Chen, Ningyu Zhang, Xin Xie, Shumin Deng, Yunzhi Yao, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. 2021. Knowprompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction. *ArXiv*, abs/2104.07650.

DongHyun Choi, Myeongcheol Shin, EungGyun Kim, and Dong Ryeol Shin. 2020. RYANSQL: recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases. *CoRR*, abs/2004.03125.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43. 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 (Volume 1: Long Papers)*, pages 731–742, Melbourne, Australia. Association for Computational Linguistics.

Jiaqi Guo, Ziliang Si, Yu Wang, Qian Liu, Ming Fan, Jian-Guang Lou, Zijiang Yang, and Ting Liu. 2021. Chase: A large-scale and pragmatic Chinese dataset for cross-database context-dependent text-to-SQL. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2316–2331, Online. Association for Computational Linguistics.

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. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535, Florence, Italy. Association for Computational Linguistics.

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

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

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Wenqiang Lei, Weixin Wang, Zhixin Ma, Tian Gan, Wei Lu, Min-Yen Kan, and Tat-Seng Chua. 2020. Re-examining the role of schema linking in text-to-SQL. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6943–6954, Online. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. 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 *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4870–4888, Online. Association for Computational Linguistics.

Xi Victoria Lin, Chenglong Wang, Luke Zettlemoyer, and Michael D. Ernst. 2018. NL2Bash: A corpus and semantic parser for natural language interface to the linux operating system. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

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

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. Picard - parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. 2021. Compositional generalization and natural language variation: Can a semantic parsing approach handle both? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 922–938, Online. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15:1929–1958.

Ningyuan Sun, Xuefeng Yang, and Yunfeng Liu. 2020. Tableqa: a large-scale chinese text-to-sql dataset for table-aware sql generation.

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 (Volume 1: Long Papers)*, pages 361–372, Melbourne, Australia. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020a. 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*, Online. Association for Computational Linguistics.

Lijie Wang, Ao Zhang, Kun Wu, Ke Sun, Zhenghua Li, Hua Wu, Min Zhang, and Haifeng Wang. 2020b. DuSQL: A large-scale and pragmatic Chinese text-to-SQL dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6923–6935, Online. Association for Computational Linguistics.

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

Kuan Xuan, Yongbo Wang, Yongliang Wang, Zujie Wen, and Yang Dong. 2021. Sead: End-to-end text-to-sql generation with schema-aware denoising. *CoRR*, abs/2105.07911.

Pengcheng Yin and Graham Neubig. 2018. TRANX: A transition-based neural abstract syntax parser for semantic parsing and code generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 7–12, Brussels, Belgium. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter Lasecki, and Dragomir Radev. 2019a. CoSQL: A conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. 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 1962–1979, Hong Kong, China. Association for Computational Linguistics.

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*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, Emily Ji, Shreya Dixit, David Proctor, Sungrok Shim, Jonathan Kraft, Vincent Zhang, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019b. SParC: Cross-domain semantic parsing in context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4511–4523, Florence, Italy. Association for Computational Linguistics.

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 5338–5349, Hong Kong, China. Association for Computational Linguistics.

Xiaoyu Zhang, Fengjing Yin, Guojie Ma, Bin Ge, and Weidong Xiao. 2020. F-sql: Fuse table schema and table content for single-table text2sql generation. *IEEE Access*, 8:136409–136420.

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

A Datasets Statistics

Table 8 lists the dataset we used. We divided the dataset into three group to study the effectiveness of UNISAR: (1) **Multi-Domain**: WikiSQL (Zhong et al., 2017) and TableQA (Sun et al., 2020); (2) **Multi-Table**: Spider (Yu et al., 2018) and DuSQL (Wang et al., 2020b); (3) **Multi-Turn**: CoSQL (Yu et al., 2019a), SParC (Yu et al., 2019b) and Chase (Guo et al., 2021).

B Hyper-parameters

We adopt the BART-Large and set the task of Fairseq as TRANSLATION_FROM_PRETRAINED_BART. The learning rate is $1e$-5. The max tokens is 1400 for V100-16G GPU. We adopt the polynomial_decay with 5,000 warmup-updates. The dropout (Srivastava et al., 2014) rate is 0.1. Optimizer is Adam (Kingma and Ba, 2015) with the default parameters. The max-update is set to 10,000. Empirically, the model obtained best performance about 7000 steps (about $10 \sim 15$ epochs) in Spider, CoSQL and SParC. The Fairseq (Ott et al., 2019) dynamically tunes the batch size to realize higher GPU utilization.

Dataset	# Lan.	# Ques.	# SQL Len.	# Table	# DB	# MD	# MTa	# MTu
WikiSQL (Zhong et al., 2017)	English	80,654	10.6	26,521	26,521	✓	✗	✗
TableQA (Sun et al., 2020)	Chinese	49,974	9.1	5,291	5,291	✓	✗	✗
Spider (Yu et al., 2018)	English	10,181	21.7	1,020	200	✓	✓	✗
DuSQL (Wang et al., 2020b)	Chinese	23,797	20.2	820	200	✓	✓	✗
CoSQL (Yu et al., 2019a)	English	15,598	18.4	1,020	200	✓	✓	✓
SParC (Yu et al., 2019b)	English	12,726	17.8	1,020	200	✓	✓	✓
Chase (Guo et al., 2021)	Chinese	17,940	20.9	1,020	200	✓	✓	✓

Table 8: Comparisons of seven text-to-SQL datasets. MD (multi-domain), MTa (multi-table) and MTu (multi-turn) denote the research focuses of these datsets and # SQL Len. denotes the complexity of predicted SQL.