How Far are We from Effective Context Modeling? An Exploratory Study on Semantic Parsing in Context

Qian Liu[1]*, **Bei Chen**[2], **Jiaqi Guo**[3]*, **Jian-Guang Lou**[2], **Bin Zhou**[1] and **Dongmei Zhang**[2]
[1]School of Computer Science and Engineering, Beihang University, China
[2]Microsoft Research, Beijing, China
[3]Xi'an Jiaotong University, Xi'an, China
{qian.liu, zhoubin}@buaa.edu.cn, {beichen, jlou, dongmeiz}@microsoft.com,
jasperguo2013@stu.xjtu.edu.cn

Abstract

Recently semantic parsing in context has received considerable attention, which is challenging since there are complex contextual phenomena. Previous works verified their proposed methods in limited scenarios, which motivates us to conduct an exploratory study on context modeling methods under real-world semantic parsing in context. We present a grammar-based decoding semantic parser and adapt typical context modeling methods on top of it. We evaluate 13 context modeling methods on two large complex cross-domain datasets, and our best model achieves state-of-the-art performances on both datasets with significant improvements. Furthermore, we summarize the most frequent contextual phenomena, with a fine-grained analysis on representative models, which may shed light on potential research directions. Our code is available at https://github.com/microsoft/ContextualSP.

1 Introduction

Semantic parsing, which translates a natural language sentence into its corresponding executable logic form (e.g. Structured Query Language, **SQL**), relieves users from the burden of learning techniques behind the logic form. The majority of previous studies on semantic parsing assume that queries are context-independent and analyze them in isolation. However, in reality, users prefer to interact with systems in a dialogue, where users are allowed to ask context-dependent incomplete questions [Bertomeu *et al.*, 2006]. That arises the task of Semantic Parsing in Context (**SPC**), which is quite challenging as there are complex contextual phenomena. In general, there are two sorts of contextual phenomena in dialogues: *Coreference* and *Ellipsis* [Androutsopoulos *et al.*, 1995]. Figure 1 shows a dialogue from the dataset SPARC [Yu *et al.*, 2019b]. After the question "What is id of the car with the max horsepower?", the user poses an elliptical question "How about with the max MPG?", and a question containing pronouns "Show its make!". Only when completely understanding the context, could a parser successfully parse the incomplete questions into their corresponding SQL queries.

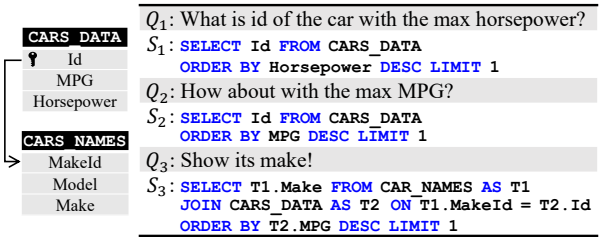


Figure 1: An example dialogue (right) and its database schema (left).

A number of context modeling methods have been suggested in the literature to address SPC [Iyyer *et al.*, 2017; Suhr *et al.*, 2018; Yu *et al.*, 2019b; Zhang *et al.*, 2019; Yu *et al.*, 2019a]. These methods proposed to leverage two categories of context: recent questions and precedent logic form. It is natural to leverage recent questions as context. Taking the example from Figure 1, when parsing Q_3, we also need to take Q_1 and Q_2 as input. We can either simply concatenate the input questions, or use a model to encode them hierarchically [Suhr *et al.*, 2018]. As for the second category, instead of taking a sequence of recent questions as input, it only considers the precedent logic form. For instance, when parsing Q_3, we only need to take S_2 as context. With such a context, the decoder can attend over it, or reuse it via a copy mechanism [Suhr *et al.*, 2018; Zhang *et al.*, 2019]. Intuitively, methods that fall into this category enjoy better generalizability, as they only rely on the last logic form as context, no matter at which turn. Notably, these two categories of context can be used simultaneously.

However, it remains unclear how far we are from effective context modeling. First, there is a lack of thorough comparisons of typical context modeling methods on complex SPC (e.g. cross-domain). Second, none of previous works verified their proposed context modeling methods with the grammar-based decoding technique, which has been proven to be highly effective in semantic parsing [Krishnamurthy *et al.*, 2017; Yin and Neubig, 2018; Guo *et al.*, 2019; Lin *et al.*, 2019]. To obtain better performance, it is worthwhile to study how context modeling methods collaborate with the grammar-based decoding. Last but not least, there is a limited understanding of how context modeling methods perform on various contextual phenomena. An in-depth anal-

*Work done during an internship at Microsoft Research.

ysis can shed light on potential research directions.

In this paper, we try to fulfill the above insufficiency via an exploratory study on real-world semantic parsing in context. Concretely, we present a grammar-based decoding semantic parser and adapt typical context modeling methods on top of it. Through experiments on two large complex cross-domain datasets, SPARC [Yu *et al.*, 2019b] and COSQL [Yu *et al.*, 2019a], we carefully compare and analyze the performance of different context modeling methods. Our best model achieves state-of-the-art (SOTA) performances on both datasets with significant improvements. Furthermore, we summarize and generalize the most frequent contextual phenomena, with a fine-grained analysis of representative models. Through the analysis, we obtain some interesting findings, which may benefit the community on the potential research directions.

2 Methodology

In the task of SPC, we are given a dataset composed of dialogues. Denoting $\langle \mathbf{x}_1, ..., \mathbf{x}_n \rangle$ a sequence of natural language questions in a dialogue, $\langle \mathbf{y}_1, ..., \mathbf{y}_n \rangle$ are their corresponding SQL queries. Each SQL query is conditioned on a multi-table database schema, and the databases used in test do not appear in training. In this section, we first present a base model without considering context. Then we introduce 6 typical context modeling methods and describe how we equip the base model with these methods. Finally, we present how to augment the model with BERT [Devlin *et al.*, 2019].

2.1 Base Model

We employ the popularly used attention-based sequence-to-sequence architecture [Sutskever *et al.*, 2014; Bahdanau *et al.*, 2015] to build our base model. As shown in Figure 2, the base model consists of a question encoder and a grammar-based decoder. For each question, the encoder provides contextual representations, while the decoder generates its corresponding SQL query according to a predefined grammar.

Question Encoder

To capture contextual information within a question, we apply Bidirectional Long Short-Term Memory Neural Network (BiLSTM) as our question encoder [Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997]. Specifically, at turn i, firstly every token $x_{i,k}$ in $\mathbf{x}_i$ is fed into a word embedding layer ϕ^x to get its embedding representation $\phi^x(x_{i,k})$. On top of the embedding representation, the question encoder obtains a contextual representation $\mathbf{h}^E_{i,k} = [\overrightarrow{\mathbf{h}}^E_{i,k}; \overleftarrow{\mathbf{h}}^E_{i,k}]$, where the forward hidden state is computed as following:

$$\overrightarrow{\mathbf{h}}^E_{i,k} = \mathbf{LSTM}^{\overrightarrow{E}}\left(\phi^x(x_{i,k}), \overrightarrow{\mathbf{h}}^E_{i,k-1}\right). \quad (1)$$

Grammar-based Decoder

The decoder is grammar-based with attention on the input question [Krishnamurthy *et al.*, 2017]. Different from producing a SQL query word by word, our decoder outputs a sequence of *grammar rule* (i.e. action). Such a sequence has one-to-one correspondence with the abstract syntax tree of the SQL query. Taking the SQL query in Figure 2 as an example, it is transformed to the action sequence $\langle$ Start→Root, Root→Select Order, Select→Agg, Agg→max Col Tab, Col→Id, Tab→CARS_DATA, Order→desc limit Agg, Agg→none Col Tab, Col→Horsepower, Tab→CARS_DATA $\rangle$ by left-to-right depth-first traversing on the tree. At each decoding step, a *nonterminal* is expanded using one of its corresponding grammar rules. The rules are either schema-specific (e.g. Col→Horsepower), or schema-agnostic (e.g. Start→Root). More specifically, as shown at the top of Figure 2, we make a little modification on Order-related rules upon the grammar proposed by Guo *et al.* [2019], which has been proven to have better performance than vanilla SQL grammar. Denoting $\mathbf{LSTM}^{\overrightarrow{D}}$ the unidirectional LSTM used in the decoder, at each decoding step j of turn i, it takes the embedding of the previous generated grammar rule $\phi^y(y_{i,j-1})$ (indicated as the dash lines in Figure 2), and updates its hidden state as:

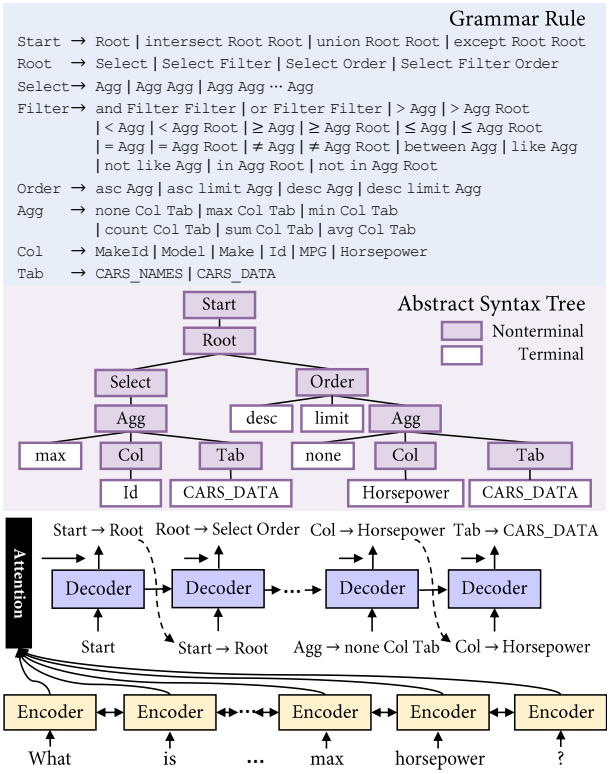


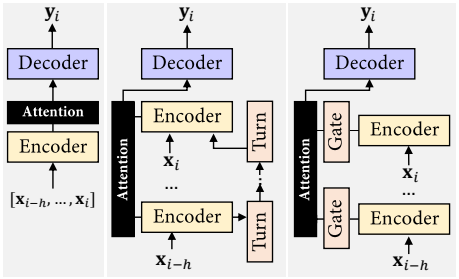
Figure 2: The grammar rule and the abstract syntax tree for the SQL `SELECT Id FROM CARS_DATA ORDER BY Horsepower DESC LIMIT 1`, along with the framework of our base model. Schema-specific grammar rules change with the database schema.

$$\overrightarrow{\mathbf{h}}^D_{i,j} = \mathbf{LSTM}^{\overrightarrow{D}}\left([\phi^y(y_{i,j-1}); \mathbf{c}_{i,j-1}], \overrightarrow{\mathbf{h}}^D_{i,j-1}\right), \quad (2)$$

where $\mathbf{c}_{i,j-1}$ is the context vector produced by attending on each encoder hidden state $\mathbf{h}^E_{i,k}$ in the previous step:

$$e_{i,k} = \mathbf{h}^E_{i,k}\,\mathbf{W}^e\,\overrightarrow{\mathbf{h}}^D_{i,j-1},\ a_{i,k} = \frac{\exp(e_{i,k})}{\sum_k \exp(e_{i,k})},$$
$$\mathbf{c}_{i,j-1} = \sum_k \mathbf{h}^E_{i,k} \cdot a_{i,k}, \quad (3)$$

where $\mathbf{W}^e$ is a learned matrix. $\overrightarrow{\mathbf{h}}^D_{i,0}$ is initialized by the final encoder hidden state $\mathbf{h}^E_{i,|\mathbf{x}_i|}$, while $\mathbf{c}_{i,0}$ is a zero-vector.



(a) CONCAT (b) TURN (c) GATE

Figure 3: Different methods to incorporate recent h questions $[\mathbf{x}_{i-h}, ..., \mathbf{x}_{i-1}]$. (a) CONCAT: concatenate recent questions with $\mathbf{x}_i$ as input; (b) TURN: employ a turn-level encoder to capture the inter-dependencies among questions in different turns; (c) GATE: use a gate mechanism to compute the importance of each question.

For each schema-agnostic grammar rule, ϕ^y returns a learned embedding. The embedding of a schema-specific grammar rule is obtained by passing its schema (i.e. table or column) through another unidirectional LSTM, namely schema encoder $\mathbf{LSTM}^{\overrightarrow{S}}$. For example, the embedding of $\text{Col}\to\text{Id}$ is:

$$\phi^y(\text{Col}\to\text{Id}) = \mathbf{LSTM}^{\overrightarrow{S}}\left(\phi^x(\text{“Id”}), \overrightarrow{\mathbf{0}}\right). \tag{4}$$

As for the output $y_{i,j}$, if the expanded nonterminal corresponds to schema-agnostic grammar rules, we can obtain the output probability of action γ as:

$$P(y_{i,j}=\gamma) \propto \exp\left(\tanh\left([\mathbf{h}^{\overrightarrow{D}}_{i,j}; \mathbf{c}_{i,j}]\,\mathbf{W}^o\right)\phi^y(\gamma)\right), \tag{5}$$

where $\mathbf{W}^o$ is a learned matrix. When it comes to schema-specific grammar rules, the main challenge is that the model may encounter schemas never appeared in training due to the cross-domain setting. To deal with it, we do not directly compute the similarity between the decoder hidden state and the schema-specific grammar rule embedding. Instead, we first obtain the unnormalized linking score $l(x_{i,k}, \gamma)$ between the k-th token in $\mathbf{x}_i$ and the schema inside action γ. It is computed by both handcraft features (e.g. word exact match) [Bogin *et al.*, 2019] and learned similarity (i.e. dot product between word embedding and grammar rule embedding). With the input question as bridge, we reuse the attention score $a_{i,k}$ in Equation 3 to measure the probability of outputting a schema-specific action γ as:

$$P(y_{i,j}=\gamma) \propto \exp\left(\sum\nolimits_k a_{i,k}\cdot l(x_{i,k},\gamma)\right). \tag{6}$$

2.2 Recent Questions as Context

To take advantage of the question context, we provide the base model with recent h questions as additional input. As shown in Figure 3, we summarize and generalize three ways to incorporate recent questions as context.

CONCAT. The method concatenates recent questions with the current question in order, making the input of the question encoder be $[\mathbf{x}_{i-h}, \ldots, \mathbf{x}_i]$, while the architecture of the base model remains the same. We do not insert special delimiters between questions, as there are punctuation marks.

TURN. A dialogue can be seen as a sequence of questions which, in turn, are sequences of words. Considering such hierarchy, Suhr *et al.* [2018] employed a turn-level encoder (i.e. an unidirectional LSTM) to encode recent questions hierarchically. At turn i, it takes the previous question vector $\mathbf{h}^E_{i-1} = [\overleftarrow{\mathbf{h}^E_{i-1,1}}, \overrightarrow{\mathbf{h}^E_{i-1,|\mathbf{x}_{i-1}|}}]$ as input, and updates its hidden state to $\mathbf{h}^{\overrightarrow{T}}_i$. Then $\mathbf{h}^{\overrightarrow{T}}_i$ is fed into the question encoder as an implicit context. Accordingly Equation 1 is rewritten:

$$\mathbf{h}^{\overrightarrow{E}}_{i,k} = \mathbf{LSTM}^{\overrightarrow{E}}\left([\phi^x(x_{i,k}); \mathbf{h}^{\overrightarrow{T}}_i], \mathbf{h}^{\overrightarrow{E}}_{i,k-1}\right). \tag{7}$$

Similar to CONCAT, Suhr *et al.* [2018] allowed the decoder to attend over all encoder hidden states. To make the decoder distinguish hidden states from different turns, they further proposed a relative distance embedding ϕ^d in attention computing. Taking the above into account, Equation 3 is as:

$$\begin{aligned} e_{i-t,k} &= [\mathbf{h}^E_{i-t,k}; \phi^d(t)]\,\mathbf{W}^e\,\mathbf{h}^{\overrightarrow{D}}_{i,j-1}, \\ a_{i-t,k} &= \frac{\exp\left(e_{i-t,k}\right)}{\sum_t \sum_k \exp(e_{i-t,k})}, \\ \mathbf{c}_{i,j-1} &= \sum\nolimits_t \sum\nolimits_k [\mathbf{h}^E_{i-t,k}; \phi^d(t)] \cdot a_{i-t,k}, \end{aligned} \tag{8}$$

where $t\in[0, \ldots, h]$ represents the relative distance.

GATE. To jointly model the decoder attention in token-level and question-level, inspired by the advances of open-domain dialogue area [Zhang *et al.*, 2018], we propose a gate mechanism to automatically compute the importance of each question. The importance is computed by:

$$\begin{aligned} g_{i-t} &= \mathbf{V}^g \tanh(\mathbf{U}^g \mathbf{h}^E_{i-t} + \mathbf{W}^g \mathbf{h}^E_i), \\ \bar{g}_{i-t} &= \frac{\exp(g_{i-t})}{\sum_t \exp\left(g_{i-t}\right)}, \end{aligned} \tag{9}$$

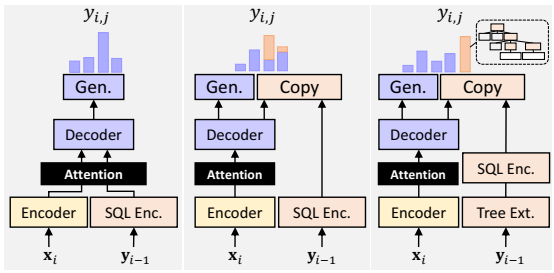
where $\{\mathbf{V}^g, \mathbf{W}^g, \mathbf{U}^g\}$ are learned parameters and $0 \le t \le h$. As done in Equation 8 except for the relative distance embedding, the decoder of GATE also attends over all the encoder hidden states. And the question-level importance $\bar{g}_{i-t}$ is employed as the coefficient of the attention scores at turn $i-t$.

2.3 Precedent SQL as Context

Besides recent questions, as mentioned in Section 1, the precedent SQL can also be context. As shown in Figure 4, the usage of $\mathbf{y}_{i-1}$ requires a SQL encoder, where we employ another BiLSTM to achieve it. The m-th contextual action representation at turn $i-1$, $\mathbf{h}^A_{i-1,m}$, can be obtained by passing the action sequence through the SQL encoder.

SQL ATTN. Attention over $\mathbf{y}_{i-1}$ is a straightforward method to incorporate the SQL context. Given $\mathbf{h}^A_{i-1,m}$, we employ a similar manner as Equation 3 to compute attention score and thus obtain the SQL context vector. This vector is employed as an additional input for decoder in Equation 2.

ACTION COPY. To reuse the precedent generated SQL, Zhang *et al.* [2019] presented a token-level copy mechanism on their non-grammar based parser. Inspired by them, we propose an action-level copy mechanism suited for grammar-based decoding. It enables the decoder to copy actions appearing in $\mathbf{y}_{i-1}$, when the actions are compatible to the current expanded nonterminal. As the copied actions lie in the



(a) SQL ATTN (b) ACTION COPY (c) TREE COPY

Figure 4: Different methods to employ the precedent SQL $\mathbf{y}_{i-1}$. *SQL Enc.* is short for SQL Encoder, *Tree Ext.* for Subtree Extractor, and *Gen.* for Generate. (a) SQL ATTN: attending over $\mathbf{y}_{i-1}$; (b) ACTION COPY: allow to copy actions from $\mathbf{y}_{i-1}$; (c) TREE COPY: allow to copy action subtrees extracted from $\mathbf{y}_{i-1}$.

same semantic space with the generated ones, the output probability for action γ is a mix of generating ($\mathbf{g}$) and copying ($\mathbf{c}$). The generating probability $P(y_{i,j} = \gamma \,|\, \mathbf{g})$ follows Equation 5 and 6, while the copying probability is:

$$P(y_{i,j}{=}\gamma|\mathbf{c}) \propto \textstyle\sum_m \mathbf{1}[\gamma{=}y_{i-1,m}] \cdot \exp(\overrightarrow{\mathbf{h}^D_{i,j}} \mathbf{W}^l \mathbf{h}^A_{i-1,m}), \quad (10)$$

where $\mathbf{W}^l$ is a learned matrix. Denoting $P^{copy}_{i,j}$ the probability of copying at decoding step j of turn i, it can be obtained by $\sigma(\mathbf{W}^c \overrightarrow{\mathbf{h}^D_{i,j}} + \mathbf{b}^c)$, where $\{\mathbf{W}^c, \mathbf{b}^c\}$ are learned parameters and σ is the sigmoid function. The final probability $P(y_{i,j} = \gamma)$ is computed by:

$$P^{copy}_{i,j} \cdot P(y_{i,j} = \gamma \,|\, \mathbf{c}) + (1 - P^{copy}_{i,j}) \cdot P(y_{i,j} = \gamma \,|\, \mathbf{g}). \quad (11)$$

TREE COPY. Besides the action-level copy, we also introduce a tree-level copy mechanism. As illustrated in Figure 4, tree-level copy mechanism enables the decoder to copy action subtrees extracted from $\mathbf{y}_{i-1}$, which shrinks the number of decoding steps by a large margin. Similar idea has been proposed in a non-grammar based decoder [Suhr *et al.*, 2018]. In fact, a subtree is an action sequence starting from specific nonterminals, such as Select. To give an example, $\langle$ Select→Agg, Agg→max Col Tab, Col→Id, Tab→CARS_DATA $\rangle$ makes up a subtree for the tree in Figure 2. For a subtree υ, its representation $\phi^t(\upsilon)$ is the final hidden state when feeding its corresponding action sequence into the SQL encoder. Then we can obtain the output probability of subtree υ as:

$$P(y_{i,j} = \upsilon) \propto \exp\left(\overrightarrow{\mathbf{h}^D_{i,j}} \mathbf{W}^t \phi^t(\upsilon)\right), \quad (12)$$

where $\mathbf{W}^t$ is a learned matrix. The output probabilities of subtrees are normalized together with Equation 5 and 6.

2.4 BERT Enhanced Embedding

We employ BERT [Devlin *et al.*, 2019] to augment our model via enhancing the embedding of questions and schemas. We first concatenate the input question and all the schemas in a deterministic order with `[SEP]` as delimiter [Hwang *et al.*, 2019]. For instance, the input for Q_1 in Figure 1 is "What is id ... max horsepower? [SEP] CARS_DATA ... [SEP] Make". Feeding it into BERT, we obtain the schema-aware question representations and question-aware schema representations. These contextual representations are used to substitute ϕ^x subsequently, while other parts of the model remain the same.

Model	SParC		CoSQL	
	Ques.Match	Int.Match	Ques.Match	Int.Match
SyntaxSQL-con	18.5	4.3	15.1	2.7
CD-Seq2Seq	21.9	8.1	13.8	2.1
EditSQL	33.0	16.4	22.2	5.8
Ours	**41.8**	**20.6**	**33.5**	**9.6**
EditSQL + BERT	47.2	29.5	40.0	11.0
Ours + BERT	**52.6**	**29.9**	**41.0**	**14.0**

Table 1: We report the best performance observed in 5 runs on the development sets of both SPARC and COSQL, since their test sets are not public. We also conduct Wilcoxon signed-rank tests between our method and the baselines, and the bold results show the improvements of our model are significant with $p < 0.005$.

3 Experiment & Analysis

We conduct experiments to study whether the introduced methods are able to effectively model context in the task of SPC (Section 3.2), and further perform a fine-grained analysis on various contextual phenomena (Section 3.3).

3.1 Experimental Setup

Dataset. Two large complex cross-domain datasets are used: SPARC [Yu *et al.*, 2019b] consists of 3034 / 422 dialogues for train / development, and COSQL [Yu *et al.*, 2019a] consists of 2164 / 292 ones. The average turn numbers of SPARC and COSQL are 3.0 and 5.2, respectively.

Evaluation Metrics. We evaluate predicted SQL queries using exact set match accuracy [Yu *et al.*, 2019b]. Based on it, we consider three metrics: *Question Match* (Ques.Match), the match accuracy over all questions, *Interaction Match* (Int.Match), the match accuracy over all dialogues[1], and *Turn i Match*, the match accuracy over questions at turn i.

Implementation Detail. Our implementation is based on PyTorch [Paszke *et al.*, 2017], AllenNLP [Gardner *et al.*, 2018] and the library transformers [Wolf *et al.*, 2019]. We adopt the Adam optimizer and set the learning rate as 1e-3 on all modules except for BERT, for which a learning rate of 1e-5 is used [Kingma and Ba, 2015]. The dimensions of word embedding, action embedding and distance embedding are 100, while the hidden state dimensions of question encoder, grammar-based decoder, turn-level encoder and SQL encoder are 200. We initialize word embedding using GloVe [Pennington *et al.*, 2014] for non-BERT models. For methods that use recent h questions, h is set as 5 on both datasets.

Baselines. We consider three models as our baselines. SyntaxSQL-con and CD-Seq2Seq are two strong baselines introduced in the SPARC dataset paper [Yu *et al.*, 2019b]. SyntaxSQL-con employs a BiLSTM model to encode dialogue history upon the SyntaxSQLNet model (analogous to our TURN) [Yu *et al.*, 2018], while CD-Seq2Seq is adapted from Suhr *et al.* [2018] for cross-domain settings (analogous to our TURN+TREE COPY). EditSQL [Zhang *et al.*, 2019] is a STOA baseline which mainly makes use of SQL attention and token-level copy (analogous to our TURN+SQL ATTN+ACTION COPY).

[1] Int.Match is much more challenging as it requires each predicted SQL in a dialogue to be correct.

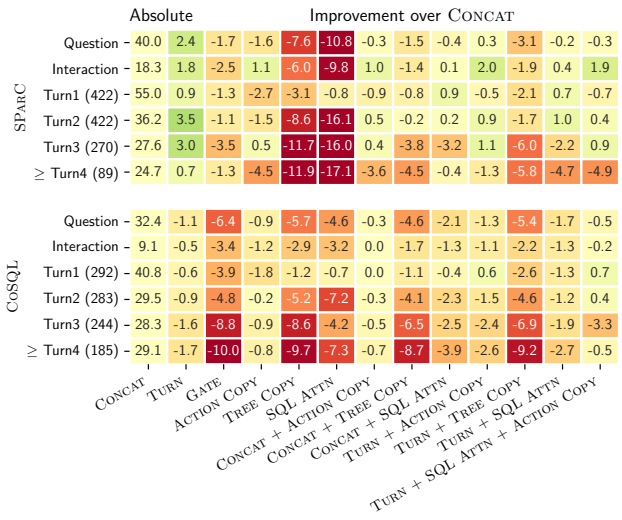


Figure 5: Question Match, Interaction Match and Turn i Match on SPARC and COSQL development sets. The numbers are averaged over 5 runs. The first column represents *absolute* values. The rest are *improvements* of different context modeling methods over CONCAT.

3.2 Model Comparison

Taking CONCAT as a representative, we compare the performance of our model with other models, as shown in Table 1. As illustrated, our model outperforms baselines by a large margin with or without BERT, achieving new SOTA performances on both datasets. Compared with the previous SOTA without BERT on SPARC, our model improves Ques.Match and Int.Match by 8.8 and 4.2 points, respectively.

To conduct a thorough comparison, we evaluate 13 different context modeling methods upon the same parser, including 6 methods introduced in Section 2 and 7 selective combinations of them (e.g., CONCAT+ACTION COPY). The experimental results are presented in Figure 5. Taken as a whole, it is very surprising to observe that none of these methods can be consistently superior to the others. The experimental results on BERT-based models show the same trend. Diving deep into the methods only using recent questions as context, we observe that CONCAT and TURN perform competitively, outperforming GATE by a large margin. With respect to the methods only using precedent SQL as context, ACTION COPY significantly surpasses TREE COPY and SQL ATTN in all metrics. In addition, we observe that there is little difference in the performance of ACTION COPY and CONCAT, which implies that using precedent SQL as context gives almost the same effect with using recent questions. In terms of the combinations of different context modeling methods, they do not significantly improve the performance as we expected.

As mentioned in Section 1, intuitively, methods which only use the precedent SQL enjoy better generalizability. To validate it, we further conduct an *out-of-distribution* experiment to assess the generalizability of different context modeling methods. Concretely, we select three representative methods and train them on questions at turn 1 and 2, whereas test them at turn 3, 4 and beyond. As shown in Figure 6, ACTION

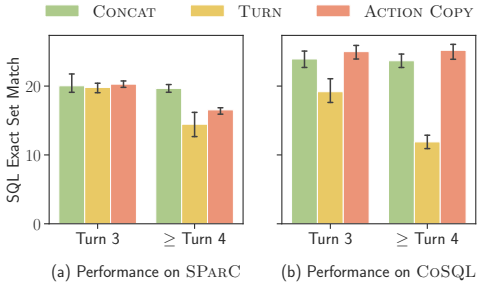


Figure 6: Out-of-distribution experimental results (Turn i Match) of three models on SPARC and COSQL development sets.

COPY has a consistently comparable or better performance, validating the intuition. Meanwhile, CONCAT appears to be strikingly competitive, demonstrating it also has a good generalizability. Compared with them, TURN is more vulnerable to out-of-distribution questions. In conclusion, existing context modeling methods in the task of SPC are not as effective as expected, since they do not show a significant advantage over the simple concatenation method.

3.3 Fine-grained Analysis

By a careful investigation on contextual phenomena, we summarize them in multiple hierarchies. Roughly, there are three kinds of contextual phenomena in questions: *semantically complete*, *coreference* and *ellipsis*. Semantically complete means a question can reflect all the meaning of its corresponding SQL. Coreference means a question contains pronouns, while ellipsis means the question cannot reflect all of its SQL, even if resolving its pronouns. In the fine-grained level, coreference can be divided into 5 types according to its pronoun [Androutsopoulos *et al.*, 1995]. Ellipsis can be characterized by its intention: *continuation* and *substitution*[2]. Continuation is to augment extra semantics (e.g. Filter), and substitution refers to the situation where current question is intended to substitute particular semantics in the precedent question. Substitution can be further branched into 4 types: *explicit* vs. *implicit* and *schema* vs. *operator*. Explicit means the current question provides contextual clues (i.e. partial context overlaps with the precedent question) to help locate the substitution target, while implicit does not. In most cases, the target is schema or operator. In order to study the effect of context modeling methods on various phenomena, as shown in Table 2, we take the development set of SPARC as an example to perform our analysis. The analysis begins by presenting Ques.Match of three representative models on the above fine-grained types in Figure 7. As shown, though different methods have different strengths, they all perform poorly on certain types, which will be elaborated below.

Coreference. Diving deep into the coreference (left of Figure 7), we observe that all methods struggle with two fine-grained types: *definite noun phrases* and *one anaphora*. Through our study, we find the scope of antecedent is a key factor. An antecedent is one or more entities referred by a

[2]The fine-grained types of ellipsis are proposed by us because there is no consensus yet.

Contextual Phenomena	Fine-grained Types		Count	Example: Precedent Question	Example: Current Question
Semantically Complete	Context Independent		149	Show the nationality of each person.	Group people by their nationality.
Coreference	Bridging Anaphora		31	Show the version number for all templates.	What is the smallest value?
	Definite Noun Phrases		67	Which country has a head of state named Beatrix?	What languages are spoken in that country?
	One Anaphora		59	Order the pets by age.	How much does each one weigh?
	Demonstrative Pronoun		195	Which students have pets?	Of those, whose last name is smith?
	Possessive Determiner		88	How many highschoolers are liked by someone else?	What are their names?
Ellipsis	Continuation		131	What are all the flight numbers?	Which land in Aberdeen?
	Substitution	Explicit	61	What is id of the car with the max horsepower?	How about with the max MPG?
		Implicit	60	Find the names of museums opened before 2010.	How about after?
		Schema	80	How many losers participated in the Australian Open?	Winners?
		Operator	41	Who was the last student to register?	Who was the first to register?

Table 2: Different fine-grained types, their count and representative examples from the SPARC development set. one means *one* is a pronoun. Winners means *Winners* is a phrase intended to substitute *losers*.

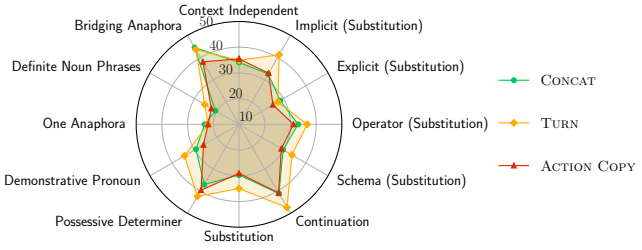


Figure 7: The average Ques.Match (across 5 runs) of different context modeling methods on fine-grained types.

pronoun. Its scope is either *whole*, where the antecedent is the precedent answer, or *partial*, where the antecedent is part of the precedent question. The above-mentioned fine-grained types are more challenging as their partial proportion are nearly 40%, while for *demonstrative pronoun* it is only 22%. It is reasonable as partial requires complex inference on context. Considering the 4^{th} example in Table 2, "one" refers to "pets" instead of "age" because the accompanying verb is "weigh". From this observation, we draw the conclusion that current context modeling methods do not succeed on pronouns which require complex inference on context.

Ellipsis. As for ellipsis (right of Figure 7), we obtain some findings by comparisons in three aspects. The first finding is that all models have a better performance on continuation than substitution. This is expected since there are redundant semantics in substitution, while not in continuation. Considering the 8^{th} example in Table 2, "horsepower" is redundant and it may raise noise in SQL prediction. The second finding comes from the unexpected drop from *implicit(substitution)* to *explicit(substitution)*. Intuitively, explicit should surpass implicit on substitution as it provides more contextual clues. The finding demonstrates that contextual clues are obviously not well utilized by the context modeling methods. Third, compared with *schema(substitution)*, *operator(substitution)* achieves a comparable or better performance consistently. We believe it is caused by the cross-domain setting, which makes schema related substitution more difficult.

4 Related Work

The most related work is the line of semantic parsing in context. In the topic of SQL, Zettlemoyer and Collins [2009] proposed a context-independent CCG parser and then applied it to do context-dependent substitution, Iyyer *et al.* [2017] applied a search-based method for sequential questions, and Suhr *et al.* [2018] provided the first sequence-to-sequence solution in the area. More recently, Zhang *et al.* [2019] presented a edit-based method to reuse the precedent generated SQL, while Liu *et al.* [2019] introduced an auxiliary task. With respect to other logic forms, Long *et al.* [2016] focused on understanding execution commands in context, Guo *et al.* [2018] on question answering over knowledge base in a conversation, and [Iyer *et al.*, 2018] on code generation in environment context. Our work is different from theirs as we perform an exploratory study, not fulfilled by previous works.

There are also several related works that provided studies on context. Hwang *et al.* [2019] explored the contextual representations in context-independent semantic parsing, and Sankar *et al.* [2019] studied how conversational agents use conversation history to generate response. Different from them, our task focuses on context modeling for semantic parsing. Under the same task, Androutsopoulos *et al.* [1995] summarized contextual phenomena in a coarse-grained level, while Bertomeu *et al.* [2006] performed a Wizard-of-Oz experiment to study the most frequent phenomena. What makes our work different from them is that we not only summarize contextual phenomena by fine-grained types, but also perform an analysis of context modeling methods.

5 Conclusion & Future Work

This work conducts an exploratory study on semantic parsing in context, to realize how far we are from effective context modeling. Through a thorough comparison, we find that existing context modeling methods are not as effective as expected. A simple concatenation method can be much competitive. Furthermore, by performing a fine-grained analysis, we summarize two potential directions as our future work: incorporating common sense for better pronouns inference, and modeling contextual clues in a more explicit manner. We believe our work can facilitate the community to debug models in a fine-grained level and make more progress.

References

[Androutsopoulos *et al.*, 1995] Ion Androutsopoulos, Graeme D Ritchie, and Peter Thanisch. Natural Language Interfaces to Databases–An Introduction. *Natural language engineering*, 1995.

[Bahdanau *et al.*, 2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[Bertomeu *et al.*, 2006] Núria Bertomeu, Hans Uszkoreit, Anette Frank, Hans-Ulrich Krieger, and Brigitte Jörg. Contextual phenomena and thematic relations in database QA dialogues: results from a Wizard-of-Oz experiment. In *NAACL*, 2006.

[Bogin *et al.*, 2019] Ben Bogin, Jonathan Berant, and Matt Gardner. Representing schema structure with graph neural networks for text-to-SQL parsing. In *ACL*, 2019.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[Gardner *et al.*, 2018] Matt Gardner, Joel Grus, Mark Neumann, and et al. AllenNLP: A deep semantic natural language processing platform. In *ACL*, 2018.

[Guo *et al.*, 2018] Daya Guo, Duyu Tang, Nan Duan, Ming Zhou, and Jian Yin. Dialog-to-action: conversational question answering over a large-scale knowledge base. In *NIPS*, 2018.

[Guo *et al.*, 2019] Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. Towards complex text-to-SQL in cross-domain database with intermediate representation. In *ACL*, 2019.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997.

[Hwang *et al.*, 2019] Wonseok Hwang, Jinyeung Yim, Seunghyun Park, and Minjoon Seo. A comprehensive exploration on WikiSQL with table-aware word contextualization. *CoRR*, 2019.

[Iyer *et al.*, 2018] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. Mapping language to code in programmatic context. In *EMNLP*, 2018.

[Iyyer *et al.*, 2017] Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. Search-based neural structured learning for sequential question answering. In *ACL*, 2017.

[Kingma and Ba, 2015] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[Krishnamurthy *et al.*, 2017] Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. Neural semantic parsing with type constraints for semi-structured tables. In *EMNLP*, 2017.

[Lin *et al.*, 2019] Kevin Lin, Ben Bogin, Mark Neumann, Jonathan Berant, and Matt Gardner. Grammar-based neural text-to-SQL generation. *CoRR*, 2019.

[Liu *et al.*, 2019] Qian Liu, Bei Chen, Haoyan Liu, Jian-Guang Lou, Lei Fang, Bin Zhou, and Dongmei Zhang. A split-and-recombine approach for follow-up query analysis. In *EMNLP-IJCNLP*, 2019.

[Long *et al.*, 2016] Reginald Long, Panupong Pasupat, and Percy Liang. Simpler context-dependent logical forms via model projections. In *ACL*, 2016.

[Paszke *et al.*, 2017] Adam Paszke, Sam Gross, Soumith Chintala, and et al. Automatic differentiation in PyTorch. In *NIPS*, 2017.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global vectors for word representation. In *EMNLP*, 2014.

[Sankar *et al.*, 2019] Chinnadhurai Sankar, Sandeep Subramanian, Chris Pal, Sarath Chandar, and Yoshua Bengio. Do neural dialog systems use the conversation history effectively? An empirical study. 2019.

[Schuster and Paliwal, 1997] Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing*, 1997.

[Suhr *et al.*, 2018] Alane Suhr, Srinivasan Iyer, and Yoav Artzi. Learning to map context-dependent sentences to executable formal queries. In *NAACL*, 2018.

[Sutskever *et al.*, 2014] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

[Wolf *et al.*, 2019] Thomas Wolf, Lysandre Debut, Victor Sanh, and et al. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, 2019.

[Yin and Neubig, 2018] Pengcheng Yin and Graham Neubig. TRANX: A transition-based neural abstract syntax parser for semantic parsing and code generation. In *EMNLP*, 2018.

[Yu *et al.*, 2018] Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. SyntaxSQLNet: Syntax tree networks for complex and cross-domain text-to-SQL task. In *EMNLP*, 2018.

[Yu *et al.*, 2019a] Tao Yu, Rui Zhang, Heyang Er, and et al. CoSQL: A conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. In *EMNLP-IJCNLP*, 2019.

[Yu *et al.*, 2019b] Tao Yu, Rui Zhang, Michihiro Yasunaga, and et al. SParC: Cross-domain semantic parsing in context. In *ACL*, 2019.

[Zettlemoyer and Collins, 2009] Luke S. Zettlemoyer and Michael Collins. Learning context-dependent mappings from sentences to logical form. In *ACL-IJCNLP*, 2009.

[Zhang *et al.*, 2018] Weinan Zhang, Yiming Cui, Yifa Wang, Qingfu Zhu, Lingzhi Li, Lianqiang Zhou, and Ting Liu. Context-sensitive generation of open-domain conversational responses. In *COLING*, 2018.

[Zhang *et al.*, 2019] Rui Zhang, Tao Yu, Heyang Er, and et al. Editing-based SQL query generation for cross-domain context-dependent questions. In *EMNLP-IJCNLP*, 2019.