Augmenting Multi-Turn Text-to-SQL Datasets with Self-Play

Qi Liu[1]*, **Zihuiwen Ye**[2]*, **Tao Yu**[1], **Phil Blunsom**[2], **Linfeng Song**[3]
[1]The University of Hong Kong, [2]University of Oxford, [3]Tencent AI Lab, Bellevue, WA, USA
{liuqi, tyu}@cs.hku.hk; {zihuiwen.ye, phil.blunsom}@cs.ox.ac.uk;
lfsong@tencent.com

Abstract

The task of context-dependent text-to-SQL aims to convert multi-turn user utterances to formal SQL queries. This is a challenging task due to both the scarcity of training data from which to learn complex contextual dependencies and to generalize to unseen databases. In this paper we explore augmenting the training datasets using self-play, which leverages contextual information to synthesize new interactions to adapt the model to new databases. We first design a SQL-to-text model conditioned on a sampled goal query, which represents a user's intent, that then converses with a text-to-SQL semantic parser to generate new interactions. We then filter the synthesized interactions and retrain the models with the augmented data. We find that self-play improves the accuracy of a strong baseline on SParC and CoSQL, two widely used cross-domain text-to-SQL datasets. Our analysis shows that self-play simulates various conversational thematic relations, enhances cross-domain generalization and improves beam-search.[1]

1 Introduction

Multi-turn text-to-SQL translation is a powerful semantic parsing paradigm that converts natural language user utterances into executable SQL queries in a conversational environment. Compared to regular text-to-SQL tasks such as Spider (Yu et al., 2018b) and GeoQuery (Zelle and Mooney, 1996), conversational text-to-SQL requires interpreting coreference and omission phenomena that frequently appear in human conversations. To be effective, text-to-SQL models must uncover complex contextual dependencies while grounding user utterances in task specific database schemas.

Numerous architectures and pretraining methods have been proposed for tackling context-dependent text-to-SQL (Suhr et al., 2018; Zhang et al., 2019; Hui et al., 2021; Scholak et al., 2021; Yu et al., 2021; Xie et al., 2022). However, the size of the datasets used has been limited due to the high cost of annotating multi-turn dialogue and SQL pairs, which often requires trained experts. Existing multi-turn text-to-SQL datasets, such as SParC (Yu et al., 2019b) and CoSQL (Yu et al., 2019a), require text-to-SQL parsers to generalize to unseen databases at test time, but doing so is difficult with limited training context.

In this paper we propose the use of self-play to augment multi-turn text-to-SQL datasets in order to achieve more robust generalization. Self-play simulates interactions between multiple artificial agents in order to generate a training signal in addition to supervised data. It has been successfully applied in a wide range of tasks, e.g. board games (Silver et al., 2016, 2018) and multiplayer battle games (Vinyals et al., 2019; Berner et al., 2019). It has also been applied in dialogue simulations, during which a dialogue model converses with a user simulator to generate synthetic dialogues (Schatzmann et al., 2006; Gür et al., 2018; Tseng et al., 2021). In our work, we extend self-play to semantic parsing.

Although self-play has been adopted in task-oriented dialogue, the need to pre-define a domain specific ontology of slot-value pairs (e.g. the slot value "price=expensive" for a restaurant booking) (Henderson et al., 2014; Wen et al., 2016; Budzianowski et al., 2018) prevents self-play from simulating interactions in a new domain. Adding a new domain for task-oriented dialogue is difficult and labor-intensive. On the other hand, text-to-SQL tasks (Yu et al., 2018b, 2019b,a) use a domain-independent formalism, i.e. SQL queries. We demonstrate that self-play is well-suited to simulating interactions in a new domain given a database schema, improving cross-domain generalization.

We use PICARD (Scholak et al., 2021) as the

* Equal Contribution
[1]Our code is available at: https://github.com/leuchine/self_play_picard

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 5608–5620
December 7-11, 2022

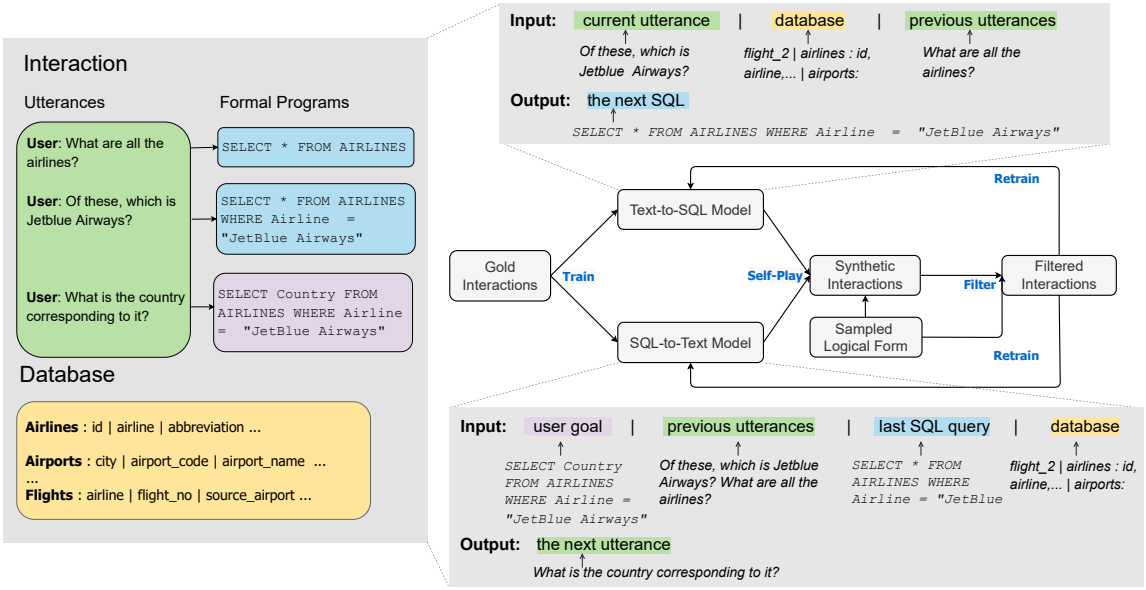

Figure 1: Multi-turn text-to-SQL with self-play. We transform an interaction from SParC on the left to seq2seq formats (top: text-to-SQL, bottom: SQL-to-text). User utterances, SQL queries, databases, and user goals are concatenated with a " | " symbol and shown in green, blue, yellow, and purple respectively. We use self-play to generate synthetic interactions. The synthetic interactions are filtered and used to retrain the text-to-SQL and SQL-to-text models.

base of our text-to-SQL model. When generating a new interaction, we first sample a SQL query with Zhong et al. (2021) as the goal query and condition the SQL-to-text model on this sampled SQL. The text-to-SQL model converses with the SQL-to-text model to simulate a new interaction. We filter out the interactions that are not grounded to the sampled goals and employ self-training (Yarowsky, 1995; Zoph et al., 2020) to retrain the text-to-SQL model and the SQL-to-text model. We conduct extensive experiments on SParC and CoSQL. Our main findings are:

- Self-play helps the text-to-SQL model learn various conversational thematic relations (§5.3) and improves cross-domain generalization (§5.1).
- Self-play improves the performance on the majority of SQL types. Models after self-play perform particularly well on queries of medium difficulty (§5.1).
- Self-play improves beam search. Models after self-play are less sensitive to the beam size and can perform well with even small beam sizes (§5.2).

2 Preliminary

In this section, we formally define the multi-turn text-to-SQL task and introduce the PICARD (Scholak et al., 2021) model, which we use as our baseline. PICARD obtains state-of-the-art results on several text-to-SQL tasks.

2.1 Task Definition

In context-dependent text-to-SQL tasks, we are given interactions between a user and a system. Each interaction spans multiple turns. The user ends the interaction when the query returns the required information from the database. Formally, at each turn t (where $1 \leq t \leq T$), multi-turn text-to-SQL produces a valid and executable SQL query $\mathcal{Q}_t$ given a database $\mathcal{D}$, a current user utterance $\mathcal{U}_t$, and a dialogue context $\mathcal{C}_t$ (which is usually the previous user utterances $\mathcal{U}_{<t}$):

$$p(\mathcal{Q}_t \mid \mathcal{U}_t, \mathcal{C}_t, \mathcal{D}). \quad (1)$$

2.2 Baseline: PICARD

We use PICARD (Scholak et al., 2021) as our baseline conditional model for Equation 1. PICARD serializes the database schema $\mathcal{D}$ into a sequence following Lin et al. (2020). An example of the input and output format is shown in Figure 1. PICARD finetunes T5 (Raffel et al., 2019), a sequence-to-sequence transformer, with input and output sequences. PICARD proposes an incremental parsing method for constrained decoding during beam search. Specifically, it rejects inadmissible

tokens at each beam search step subject to parsing rules that encode lexical and grammatical constraints. Only the beam hypotheses that pass all the constraint checks are kept. PICARD also leverages SQL schema information, such as the column names of each table, to impose checks on the validity of the generated SQL. PICARD greatly reduces the likelihood of decoding invalid SQL queries.

3 Method

Here we introduce how we use self-play for data augmentation. We first design a SQL-to-text model (§3.1). Next, we describe how to use self-play to generate synthetic interactions (§3.2). Finally, we explain how we incorporate the generated data for self-training (§3.3).

3.1 The SQL-to-Text Model

We design a user simulator, which is a SQL-to-text model, to converse with the text-to-SQL model to generate synthetic interactions. Specifically, at each turn t we would like the user simulator to produce a meaningful question that would naturally be asked by a human user. In each interaction, a user has a goal to achieve. We explicitly condition the SQL-to-text model on a user goal, $\mathcal{G}$, to encourage the user simulator to ask questions that are grounded to this goal. Formally, the SQL-to-text model calculates the following conditional at each turn:

$$p(\mathcal{U}_t \mid \mathcal{Q}_{t-1}, \mathcal{C}_t, \mathcal{G}, \mathcal{D}), \quad (2)$$

where the context $\mathcal{C}_t$ contains the previous user utterances $\mathcal{U}_{<t}$. During training, $\mathcal{G}$ is the SQL query of the final turn T, i.e. $\mathcal{Q}_T$. During inference we adopt Zhong et al. (2021) to sample a new goal query as shown in §3.2. We employ the seq2seq approach and parameterize the SQL-to-text model (Eq. 2) with T5. We concatenate the user goal $\mathcal{G}$, the last SQL query $\mathcal{Q}_{t-1}$, the previous user utterances $\mathcal{U}_{<t}$, and the serialized schema $\mathcal{D}$ to predict the next user utterance $\mathcal{U}_t$. For example, one input would be: "user goal | previous utterances | last SQL query | serialized database". Its target label is the correct user utterance for the next turn. We pad the last utterance with a special stop-of-interaction symbol. In SQL-to-text, there could be multiple reasonable questions to ask for the next turn, i.e. a one-to-many relation. A well-trained SQL-to-text model can generate new questions, thereby increasing the diversity of user dialogue flows in the dataset and improving generalization.

Algorithm 2: Self-training.

Input : Gold interactions $\mathcal{I}$, # iteration k for synthetic data generation, threashold w.
Output : A text-to-SQL model and a SQL-to-text model.

Pretrain a text-to-SQL model $p(\mathcal{Q}_t|\mathcal{U}_t, \mathcal{C}_t, \mathcal{D})$ and a SQL-to-text model $p(\mathcal{U}_t|\mathcal{Q}_{t-1}, \mathcal{C}_t, \mathcal{G}, \mathcal{D})$ on $\mathcal{I}$.
$\mathcal{I}' = \emptyset$
for i **in** $(1, ..., k)$ **do**
  Sample a goal query $\mathcal{G}$.
  Generate a synthetic interaction $\mathcal{I}_S$ by self-play between text-to-SQL and SQL-to-text.
  Calculate $score(\mathcal{Q}_T, \mathcal{G})$ on $\mathcal{I}_S$.
  if $score(\mathcal{Q}_T, \mathcal{G}) > w$ **then**
    Add $\mathcal{I}_S$ to $\mathcal{I}'$
Retrain $p(\mathcal{Q}_t|\mathcal{U}_t, \mathcal{C}_t, \mathcal{D})$ and $p(\mathcal{U}_t|\mathcal{Q}_{t-1}, \mathcal{C}_t, \mathcal{G}, \mathcal{D})$ on $\mathcal{I} \cup \mathcal{I}'$.
return the retrained text-to-SQL model and the SQL-to-text model.

3.2 Self-Play

We pretrain both the text-to-SQL and SQL-to-text models on the gold training data by minimizing the negative log likelihood:

$$L = -\sum_{i=1}^{N}\sum_{j=1}^{V} \log p(y_j^i \mid y_1^i, y_2^i, \ldots, y_{j-1}^i), \quad (3)$$

where N is the number of training examples, V is the sequence length, and each y_j^i is a token in the reference sequence. With the models pretrained on the gold dialogues, we can generate synthetic interactions using self-play. First, we need to specify a SQL query as the eventual goal $\mathcal{G}$ of the interaction. We adopt the query sampling method proposed in Zhong et al. (2021) for synthesizing a goal $\mathcal{G}$. Zhong et al. (2021) first builds and samples coarse SQL templates with the SQLs in the training set by replacing the column and value mentions in the queries with typed slots. For example, `SELECT T1.id, T2.name` is converted to the template `SELECT key1, text1`. To adapt the models to an unseen environment, they sample an unseen database and fill in the typed slots with columns and values from the sampled database to form a new SQL query. We follow this approach to synthesize goals in new domains for cross-domain generalization. The complete sampling procedure is given in Appendix A.1. We concatenate the sampled goal $\mathcal{G}$ with an empty context and the serialized schema as shown in Eq. 2 and feed it into the SQL-to-text model to produce the first user utterance. Then, the text-to-SQL model and SQL-to-text model can continue the interaction with Eq. 1 and Eq. 2 until

the end. A synthetic interaction ends whenever the SQL-to-text model decodes the stop-of-interaction symbol.

Filtering Synthetic conversations generated by self-play may diverge from the sampled goals. To filter these low-quality conversations, we compare the generated SQL query $\mathcal{Q}_T$ from the last turn T with the sampled goal $\mathcal{G}$ (see §3.2) using a similarity score $score(\mathcal{Q}_T, \mathcal{G})$. We follow Yu et al. (2018b) and decompose the SQL queries $\mathcal{Q}_T$ and $\mathcal{G}$ into SQL substructures $\mathcal{Q}_{T_s}, \mathcal{G}_s$ (e.g. `select`, `where`, `group_by`, `order_by` statements) and calculate the accuracy on each substructure. We let $score(\mathcal{Q}_T, \mathcal{G})$ be the average of the accuracy over all the substructures. We keep a synthetic conversation if $score(\mathcal{Q}_T, \mathcal{G})$ is larger than a threshold value w. A high score means that the synthetic conversation is grounded to the sampled goal.

3.3 Self-Training

We re-train a new text-to-SQL model and a new SQL-to-text model with both the gold training data and the filtered synthetic interactions. Algorithm 2 shows the overall procedures. The complete self-play and self-training steps are shown in Figure 1. Our method is an instance of self-training as the models are re-trained with their own outputs. To re-train the text-to-SQL and SQL-to-text models, we can either combine the filtered synthetic data with gold interactions, or pretrain on the synthetic interactions before fine-tuning on the gold interactions. We employ the second approach as we observe that the second approach performs slightly better than the first one.

4 Datasets and Main Results

In this section, we evaluate the performance of self-play on cross-domain multi-turn semantic parsing. We first introduce the datasets (§4.1), then detail the evaluation metrics (§4.2), and finally we show the main results (§4.3).

4.1 Datasets

We evaluate our method on two large-scale benchmark datasets, SParC (Yu et al., 2019b) and CoSQL (Yu et al., 2019a). Table 1 summarises the statistics of the two datasets. Following PICARD, we additionally pretrain the text-to-SQL model on a single-turn text-to-SQL dataset Spider (Yu et al., 2018b). All these datasets require generalization to new domains as they contain different databases for training, development, and testing, respectively, to evaluate the cross-domain performance. We discuss SParC and CoSQL in detail.

SParC SParC is a multi-turn text-to-SQL dataset that spans 200 databases in which the tables cover 138 different domains. Each question in an interaction belongs to one of the four thematic relations: refinement, theme-entity, theme-property, and answer refinement (Bertomeu et al., 2006). For example, given a question "Which major has the fewest students?", the next query can be an "refinement" query, "What is the most popular one?", which asks for the same entity as the previous question but with a different constraint.

CoSQL CoSQL is the dialogue version of SParC. In CoSQL, besides a SQL query, the system also generates a natural language response. It is collected with the Wizard-of-Oz setting (Budzianowski et al., 2018). The dataset is used for three tasks including state-tracking, user act prediction, and response generation. We use this dataset for state-tracking, where the goal is to map user utterances into a SQL query at each turn.

4.2 Settings and Evaluation Metrics

Following Yu et al. (2018b), we measure the performance with *question match* (QM) and *interaction match* (IM), both of which are based on the exact set match accuracy. The exact set match is computed by decomposing the predicted SQLs into clauses such as `SELECT`, `WHERE`, `GROUP BY` and calculating the set matching score on each. QM is 1 if the exact set match for a question in an interaction is 1. IM is 1 if the exact set matches for all questions in an interaction are 1. The number of the self-play generated training data for SParC (CoSQL) before filtering is 100,000 (100,000) and 49,623 (48,291) after filtering. Appendix A.2 shows implementation details of our experiments.

4.3 Main Results

We report the main results in Table 2. We observe that the configuration "w/ PICARD w/ self-play" achieves the best results on both datasets (measured by QM and IM). This demonstrates the benefit of self-play. The improvement brought by self-play is more salient on SParC than on CoSQL, while T5-Large w/ PICARD w/ self-play outperforms the vanilla T5-3B reported by Scholak et al. (2021). Therefore, we conclude that self-play is an effective

Dataset	System Response	# Dialogues	#Turns	# Databases	# Domains	# Tables	Avg. Q len	Vocab
SParC	✗	4,298	12,726	200	138	1,020	8.1	3,794
CoSQL	✓	3,007	15,598	200	138	1,020	11.2	9,585

Table 1: Comparison of cross-domain context-dependent text-to-SQL datasets.

Models		SParC				CoSQL			
		Dev		Test		Dev		Test	
		QM	IM	QM	IM	QM	IM	QM	IM
T5-Base	w/ PICARD w/ self-play	**62.4**	**42.1**	-	-	**53.0**	**21.5**	-	-
	w/ PICARD w/o self-play	57.5	38.6	-	-	49.9	20.2	-	-
	w/o PICARD w/ self-play	57.2	37.5	-	-	50.6	20.4	-	-
	w/o PICARD w/o self-play	50.3	31.7	-	-	45.2	18.7	-	-
T5-Large	w/ PICARD w/ self-play	**65.5**	**45.6**	**64.0**	**39.6**	**55.7**	**23.2**	**53.4**	**22.7**
	w/ PICARD w/o self-play	63.0	43.0	60.7	36.9	54.3	21.9	52.1	21.6
	w/o PICARD w/ self-play	64.1	44.1	-	-	53.9	21.2	-	-
	w/o PICARD w/o self-play	57.5	38.1	-	-	51.4	20.6	-	-

Table 2: Main results. Models after self-play outperform the baselines under different configurations.

data augmentation method to improve performance on cross-domain context-dependent text-to-SQL. Appendix A.3 shows the system's performance under different configurations of the generated synthetic data.

5 Analysis

In this section we take SParC and systematically analyze the effect of self-play. First, to gain more insight into how a question's position or the query template affect the models, we examine self-play performance stratified by different turn number and SQL templates in §5.1. Then, we study whether self-play improves decoding during beam search (§5.2). We further conduct a case study of self-play interactions in §5.3.

5.1 Turn and Template Analysis

We first plot the distribution of interaction lengths in Figure 2. Self-play produces shorter interactions with a mean length of 2.53, whereas the mean of the training data is 2.97. Figure 3 shows *Question Match* (QM) accuracy stratified by question turns. The performance after self-play increases on the turn numbers ≤ 3 and decreases on the turn number 4. This is because self-play does not generate enough long interactions as shown in Figure 2.

Next we compare the performance of the models with and without self-play stratified by the difficulty of the SQL template. We first convert SQLs into templates using the method in Zhong et al. (2021). To get a sense of the overlap of the templates in self-play and training, 85% of self-play

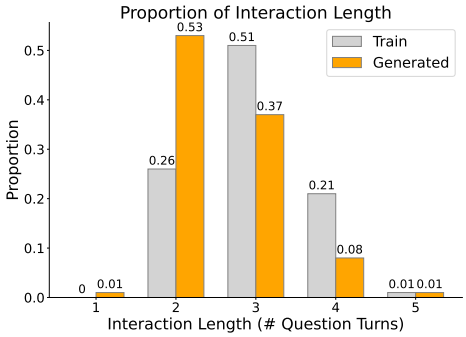

Figure 2: The distribution of interaction lengths of gold interactions and self-play interactions.

templates occur in the training templates. That is to say, 15% of self-play templates are new templates that are unseen during training. As shown in Figure 4, self-play interactions have higher proportions of easy and extra hard templates and lower proportions of medium and hard templates. We compare the performance before and after self-play on the SParC validation set in Table 3. Self-play brings the largest improvement to interactions of medium difficulty, followed by hard and easy ones.

On manually inspecting the performance for templates we observe that the performance on most is improved after self-play. Of the 72 unique templates in the SParC validation set, there are only 12 query templates whose performance decreases. The performance on the templates with the operator "`select counts`" improves significantly (on average an increase of 12 for the 11 templates with "`select counts`"), possibly because the word "count" appears more often in the gen-

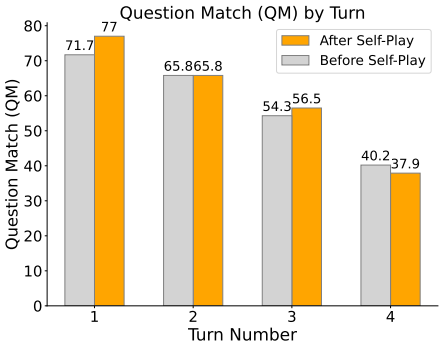

Figure 3: *Question Match* (QM) by turn numbers.

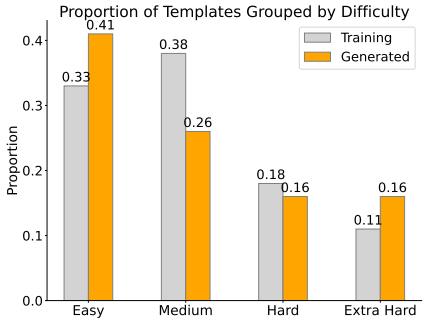

Figure 4: The distribution of template difficulties.

erated ones than training. For example, "`select count (*_col_0)`" is one of the top templates in the generated dataset as shown in Table 4. We find that the accuracy of the template "`select sum (number_col_0)`" increases from 50 to 100 after self-play. Self-play also reduces hallucination to some extent. For example, when asked to display certain record companies, the model would hallucinate the constraint "`having count(*)>2`" before self-play, but the system gives the correct result after self-play. These results confirm the effectiveness of self-play. Appendix A.4 shows more examples of generated templates and the improvement brought by self-play.

Difficulty	Before self-play	After self-play	Improvement
Easy	80.9	82.4	1.5
Medium	62.6	67.7	5.1
Hard	41.2	44.5	3.4
Extra Hard	31.8	31.8	0

Table 3: The performance before and after self-play on SParC validation set grouped by template difficulties.

5.2 Beam Search Analysis

In this section we study whether self-play improves beam search by increasing the recall of the correct SQL. We first define "Recall at beam size k" as the probability that the ground-truth SQL is contained

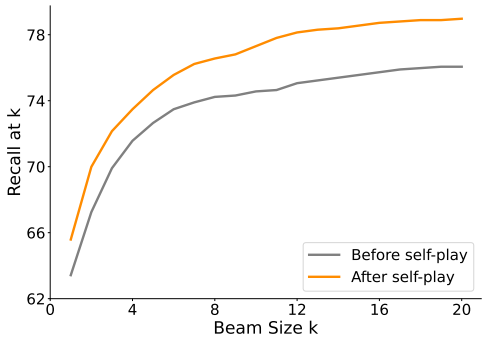

Figure 5: Recall at k plot. Models after self-play have higher recall at different beam sizes.

in the hypotheses of the beam search. This metric measures the recall of the ground-truth SQL when using the beam size k. As shown in Figure 5, we plot "Recall at beam size k" with k from 1 to 20. We observe that the model after self-play has higher recall compared to the model before self-play at different beam sizes. For example, the recall is 76.1 before self-play and 79.0 after self-play when k is 20. This shows that self-play can improve the recall of the ground-truth SQL. The recall at beam size 20 after self-play is 13.5 higher than the corresponding exact match score (79.0 vs. 65.5), demonstrating that the model has a high recall of the ground-truth SQL yet the ground-truth SQL may not have the highest beam score.

As shown in Figure 6, we further plot the exact match score of the T5-Large model with 4 configurations at different beam sizes to understand the effect of beam size on model performance. In general, the exact match score improves with larger beam sizes. We observe that (1) the model with self-play outperforms the model without self-play in all configurations; (2) the models with PICARD are more sensitive to beam sizes because the performance improves significantly with increasing beam sizes;[2] (3) the models with self-play are less sensitive to beam sizes as they can obtain high exact match scores with even small beam sizes; (4) self-play can improve the performance with/without PICARD.

5.3 Case Study of Self-Play Interactions

Table 5 shows successful (5a) and failed interactions (5b, 5c) generated with self-play. In Table 5a, given the sampled goal (the same as the final system query from turn 3), the SQL-to-text model

[2]This conforms with the plot on the Spider dataset in Scholak et al. (2021), where the authors observe pronounced improvements with larger beam sizes.

Top Templates in Train	Proportion	Top Templates in Self-Play	Proportion
`select text_col_0`	7.56%	`select text_col_0`	13.33%
`select text_col_0 where text_col_1 = value`	4.99%	`select *_col_0`	5.13%
`select *_col_0`	3.67%	`select count ( *_col_0 )`	4.57%

Table 4: The top templates and their proportions in the SParC training and generated data.

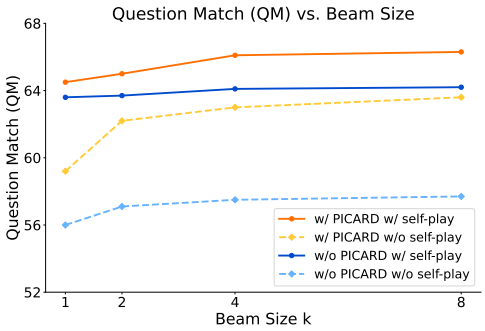

Figure 6: Question Match with different beam sizes.

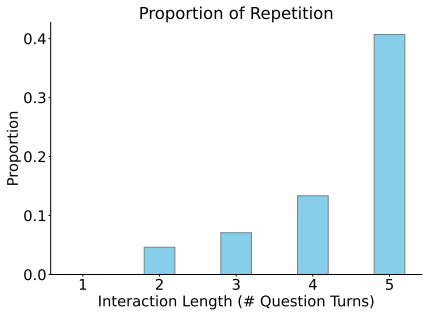

Figure 7: The proportion of repetition in generated interactions grouped by interaction length.

can decompose it into smaller questions over multiple turns. After asking for the locations of gas stations in the first turn, the SQL-to-text model asks for the company names of the gas stations, a theme-entity question, and then proceeds to query the assets of these companies in descending order, an answer refinement question. This demonstrates that different thematic relations are learned by the SQL-to-text model. Meanwhile, the text-to-SQL model produces the correct SQL queries in a context-dependent way.

Next, we analyze the failure cases. In Table 5b, the user utterances are not grounded to the sample query in the course of the dialogue, e.g. the question in the final turn does not match the semantics of the goal query. Although the final system query matches the sampled goal, language drift happens in the middle of the conversation. For example, the user utterance mentions templates in the second turn, but the text-to-SQL model ignores this keyword in the SQL query. Another failure case is the repetition of user utterances. Figure 7 shows the proportion of generated interactions in which the user utterances repeat. Repetition happens more frequently with increasing interaction lengths. Table 5c shows an example of a repetitive interaction. Although the SQL-to-text model produces the sampled goal in the first turn, the stop-of-interaction symbol does not appear in the user utterance. As a result, the conversation continues, and the user simply repeats its first question in the third turn.

6 Related Work

Data Augmentation for Semantic Parsing Data augmentation (Feng et al., 2021) is an effective strategy to increase the diversity of training data without manually collecting new data. Data augmentation has been applied in NLP (Jia and Liang, 2016) on various tasks such as paraphrase extraction (Barzilay and McKeown, 2001), machine translation (Sennrich et al., 2016; Liu et al., 2021), and question-answering (Longpre et al., 2019). Data augmentation is also widely-adopted in semantic parsing tasks (Jia and Liang, 2016; Hou et al., 2018; Yu et al., 2020; Zhong et al., 2021; Wang et al., 2021; Yang et al., 2022). Most previous work on data augmentation for text-to-SQL tasks use single-turn datasets such as SPIDER. Yu et al. (2018a) create cross-domain augmentation data with question-SQL patterns extracted from Spider. Guo et al. (2018) use a syntax-and-table-aware semantic parser and a copy-based latent variable model to generate SQLs and questions, respectively. Wu et al. (2021) apply the abstract syntax tree grammar for SQL generation and a hierarchical SQL-to-question generation model to generate questions.

Data augmentation for multi-turn SQL-to-text datasets is under-explored. The task is more challenging compared to single-turn datasets as it requires sequential generation that takes into consideration complex contextual dependencies and user goals. Zhong et al. (2021) combine a forward semantic parser with a backward utterance generator to generate multi-turn interactions. The

Turn	User Utterance	System Query	Execution Result
1	Show the location for all gas stations.	`select location from gas_station`	Herne Hill, Channel Hill Reading North ...
2	For each of them, also show the company names.	`select t3.location, t2.company from station_company as t1 join company as t2 on t1.company_id = t2.company_id join gas_station as t3 on t1.station_id = t3.station_id`	(Hennry Hill, Petrobras) (Maindy Ave, BP)...
3	Order them by assets in descending order.	`select t3.location, t2.company from station_company as t1 join company as t2 on t1.company_id = t2.company_id join gas_station as t3 on t1.station_id = t3.station_id order by t2.assets_billion desc`	(Herne Hill, Royal Dutch Shell) (Herne St, ExxonMobil)...

(a) *A successful interaction.*

Turn	User Utterance	System Query	Execution Result
1	Show the names of all documents.	`select document_name from documents`	Introduction of OS, How to write a CV...
2	For each of them, also show the number of templates.	`select document_name, count(*) from documents group by document_name`	(A history of Arts, 1), (About Korea, 1)...
3	Order the document names in descending order of the number of templates.	`select document_name from documents group by document_name order by count(*) desc`	Welcome to NY, Understanding DB...

(b) *Failure case: language drift.*

Turn	User Utterance	System Query	Execution Result
1	What is the average room count?	`select avg(room_count) from properties`	5.73
2	What is the average selling price?	`select avg(agreed_selling_price) from properties`	48653794
3	What is the average room count?	`select avg(room_count) from properties`	5.73

(c) *Failure case: repetition.*

Table 5: Examples of successful and failed interactions generated by self-play.

algorithm truncates the contextual window to 2 and does not condition on a global user goal during generation. As a result, this method fails to consider long-range contextual dependencies. Differently, our method conditions each turn on its full history context and the sampled user goal. This enables capturing longer contextual dependencies. For further comparison of the model performance between our proposed self-play method and Zhong et al. (2021), please see Appendix A.3.

Self-Play in Task-Oriented Dialogue As learning from real users is costly and time-consuming, self-play with user simulators has been employed in task-oriented dialogue systems (Levin et al., 2000). Two types of user simulators including rule-based (Schatzmann et al., 2006, 2007; Schatzmann and Young, 2009; Shah et al., 2018b,a)) and data-driven (Asri et al., 2016; Gür et al., 2018; Kreyssig et al., 2018; Tseng et al., 2021) are widely-adopted. Rule-based user simulators make use of hand-crafted rules in building dialogue schedules, while data-driven user simulators are trained on gold dialogues. In task-oriented dialogue systems, each domain has its own slot-value pairs, which are domain-dependent. As a result, adapting the system to a new domain usually requires data collection, model redesigning, and retraining. Differently, SQL is domain-agnostic for text-to-SQL tasks. Self-play is well-suited for cross-domain text-to-SQL tasks as it can synthesize user and system interactions to generalize to new domains.

Mitigating the Exposure Bias Exposure bias (Bengio et al., 2015; Ranzato et al., 2015) is the mismatch between training and the generation pro-

cedure that happens when the model is only exposed to ground-truth interactions without being conditioned on its own predicted interactions. Several methods (Ranzato et al., 2015; Shen et al., 2016; Leblond et al., 2017; Welleck et al., 2019) have been proposed to bridge this train and test time discrepancy. Our model demonstrates the benefits of using the high-quality predicted interactions to retrain the original model and is a reasonable way to condition the model on its own prediction and mitigate the exposure bias issue.

7 Conclusion

We explore using self-play as a data augmentation method for generating synthetic dialogues in the cross-domain conversational semantic parsing task to address the challenge of data scarcity and cross-domain generalization. Self-play learns various thematic relations in dialogues, improves beam search, and encourages the model's generalization to different domains. Experiments on a T5 text-to-SQL semantic parser demonstrate the benefit of our proposed method. In the future, we will study using rewards in a RL setting to guide self-play to produce better synthetic dialogues.

Limitations

Although the filtered dialogues after self-play are mostly grounded to the sampled user goal, some synthetic dialogues are unnatural as illustrated in section §5.3 Therefore, a more controlled generation of self-play that penalizes producing repetitive questions, and encourages dialogues that last longer turns would be desirable. The experiments require large GPU resources and restrict us to run self-play for one round. Running self-play with the retrained models iteratively for multiple rounds may possibly improve the results more. Dataset-wise, in the real world, as humans do not ask questions in a controlled setting as in SParC and CoSQL, the data distribution may be more noisy and complicated. Self-play is not able to generate synthetic dialogues that diverge from the training data to simulate real-world scenarios.

References

Layla El Asri, Jing He, and Kaheer Suleman. 2016. A sequence-to-sequence model for user simulation in spoken dialogue systems.

Regina Barzilay and Kathleen R. McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 50–57, Toulouse, France. Association for Computational Linguistics.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. 2019. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*.

Núria Bertomeu, Hans Uszkoreit, Anette Frank, Hans-Ulrich Krieger, and Brigitte Jörg. 2006. Contextual phenomena and thematic relations in database QA dialogues: results from a Wizard-of-Oz experiment. In *Proceedings of the Interactive Question Answering Workshop at HLT-NAACL 2006*, pages 1–8, New York, NY, USA. Association for Computational Linguistics.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. Multiwoz – a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling.

Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for nlp.

Daya Guo, Yibo Sun, Duyu Tang, Nan Duan, Jian Yin, Hong Chi, James Cao, Peng Chen, and Ming Zhou. 2018. Question generation from sql queries improves neural semantic parsing.

Izzeddin Gür, Dilek Hakkani-Tür, Gokhan Tür, and Pararth Shah. 2018. User modeling for task oriented dialogues. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 900–906. IEEE.

Matthew Henderson, Blaise Thomson, and Jason D. Williams. 2014. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272, Philadelphia, PA, U.S.A. Association for Computational Linguistics.

Yutai Hou, Yijia Liu, Wanxiang Che, and Ting Liu. 2018. Sequence-to-sequence data augmentation for dialogue language understanding.

Binyuan Hui, Ruiying Geng, Qiyu Ren, Binhua Li, Yongbin Li, Jian Sun, Fei Huang, Luo Si, Pengfei Zhu, and Xiaodan Zhu. 2021. Dynamic hybrid relation network for cross-domain context-dependent semantic parsing.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization.

Florian Kreyssig, Inigo Casanueva, Pawel Budzianowski, and Milica Gasic. 2018. Neural user simulation for corpus-based policy optimisation for spoken dialogue systems.

Rémi Leblond, Jean-Baptiste Alayrac, Anton Osokin, and Simon Lacoste-Julien. 2017. Searnn: Training rnns with global-local losses.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. 2000. A stochastic model of human-machine interaction for learning dialog strategies. *Speech and Audio Processing, IEEE Transactions on*, 8:11 – 23.

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.

Qi Liu, Matt Kusner, and Phil Blunsom. 2021. Counterfactual data augmentation for neural machine translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 187–197.

Shayne Longpre, Yi Lu, Zhucheng Tu, and Chris DuBois. 2019. An exploration of data augmentation and sampling techniques for domain-agnostic question answering. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 220–227, Hong Kong, China. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007. Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 149–152.

Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. 2006. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *Knowledge Engineering Review*, 21(2):97–126.

Jost Schatzmann and Steve Young. 2009. The hidden agenda user simulation model. *IEEE Transactions on Audio, Speech, and Language Processing*, 17(4):733–747.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. Picard: Parsing incrementally for constrained auto-regressive decoding from language models.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Pararth Shah, Dilek Hakkani-Tür, Bing Liu, and Gokhan Tür. 2018a. Bootstrapping a neural conversational agent with dialogue self-play, crowdsourcing and on-line reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 41–51, New Orleans - Louisiana. Association for Computational Linguistics.

Pararth Shah, Dilek Hakkani-Tür, Gokhan Tür, Abhinav Rastogi, Ankur Bapna, Neha Nayak, and Larry Heck. 2018b. Building a conversational agent overnight with dialogue self-play.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1683–1692, Berlin, Germany. Association for Computational Linguistics.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144.

Alane Suhr, Srinivasan Iyer, and Yoav Artzi. 2018. Learning to map context-dependent sentences to executable formal queries. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2238–2249, New Orleans, Louisiana. Association for Computational Linguistics.

Bo-Hsiang Tseng, Yinpei Dai, Florian Kreyssig, and Bill Byrne. 2021. Transferable dialogue systems and user simulators. 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 152–166, Online. Association for Computational Linguistics.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. 2019. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354.

Bailin Wang, Wenpeng Yin, Xi Victoria Lin, and Caiming Xiong. 2021. Learning to synthesize data for semantic parsing.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2019. Neural text generation with unlikelihood training.

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2016. A network-based end-to-end trainable task-oriented dialogue system.

Kun Wu, Lijie Wang, Zhenghua Li, Ao Zhang, Xinyan Xiao, Hua Wu, Min Zhang, and Haifeng Wang. 2021. Data augmentation with hierarchical sql-to-question generation for cross-domain text-to-sql parsing.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I. Wang, Victor Zhong, Bailin Wang, Chengzu Li, Connor Boyle, Ansong Ni, Ziyu Yao, Dragomir Radev, Caiming Xiong, Lingpeng Kong, Rui Zhang, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2022. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models.

Kevin Yang, Olivia Deng, Charles Chen, Richard Shin, Subhro Roy, and Benjamin Van Durme. 2022. Addressing resource and privacy constraints in semantic parsing through data augmentation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3685–3695, Dublin, Ireland. Association for Computational Linguistics.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, pages 189–196.

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

Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. 2018a. SyntaxSQLNet: Syntax tree networks for complex and cross-domain text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1653–1663, Brussels, Belgium. Association for Computational Linguistics.

Tao Yu, Rui Zhang, He Yang 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 S Lasecki, and Dragomir Radev. 2019a. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases.

Tao Yu, Rui Zhang, Alex Polozov, Christopher Meek, and Ahmed Hassan Awadallah. 2021. {SC}ore: Pre-training for context representation in conversational semantic parsing. In *International Conference on Learning Representations*.

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. 2018b. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task.

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.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI, Vol. 2*.

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.

Victor Zhong, Mike Lewis, Sida I. Wang, and Luke Zettlemoyer. 2021. Grounded adaptation for zero-shot executable semantic parsing.

Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin D. Cubuk, and Quoc V. Le. 2020. Rethinking pre-training and self-training.

A Appendix

A.1 Query Sampling Procedure in GAZP

The query sampling procedure in GAZP (Zhong et al., 2021) is shown in Algorithm 3.

Algorithm 3: Query Sampling Procedure in GAZP.

Input : Databases $\mathcal{D}$, training data X_{tr}.
Output : Sampled query z'.
Preprocess queries in training data X_{tr}.
Replace columns and values in X_{tr} with typed slots to form coarse templates Z.
Sample a database, $d \sim$ UNIFORM($\mathcal{D}$)
$Z' = \emptyset$
for $z \in Z$ **do**
  if *z can be filled with d* **then**
    Z'.*ADD*(z)
  end
end
Build empirical distribution $P_{Z'}$ by counting their occurrences in X_{tr}.
Sample a template $z' \sim P_{Z'}$.
Randomly assign columns and values s, v in d to populate the corresponding typed slots in z'.
return z' .

A.2 Implementation Details

We experiment with T5-Base (~220m parameters) and T5-Large (~770m parameters). Adam (Kingma and Ba, 2014) is used for optimization with a learning rate of 1e-4 and 5e-6 for text-to-SQL and SQL-to-text, respectively. The beam size is set to 4 for text-to-SQL and 20 for SQL-to-text. We used 8 GPUs for all our experiments. PICARD is set to the highest mode "Parse with Guard", and the max number of tokens to check for PICARD is 2. Inputs longer than 512 tokens are truncated. The maximum number of self-play turns is set to 5. The threshold w for filtering interactions is set to 0.5. We generate 100,000 synthetic interactions before filtering.

A.3 Analysis on the Generated Data

To study the effect of the size of generated data on the final accuracy, we show the QM scores after self-play on SParC validation set with T5-base trained on different number of synthetic data before filtering. As shown in Table 6, we do not observe significant improvements after using 100,000 synthetic data before filtering, thus we choose the size to be 100,000 in the experiments.

# of synthetic data	**50,000**	**100,000**	**150,000**
Question Match (QM)	60.2	62.4	62.5

Table 6: The QM score after self-play on SParC validation set with T5-base trained on different number of synthetic data before filtering.

In our experiments, filtering is applied to discard low-quality synthetic interactions that diverge from the user's goals. We find that training on low-quality interactions gives negative effects for the final performance. We study the effect of changing the filter threshold value w, as shown in Table 7. The final threshold w for filtering interactions is set to 0.5 as a larger threshold aggressively filters most synthetic dialogues that are of hard/extra-hard difficulties.

w	**0**	**0.3**	**0.5**	**0.7**
Question Match (QM)	56.2	60.5	62.4	61.8

Table 7: The QM score on SParC validation set with T5-base trained with different filter value w.

We further study if conditioning on a user goal $\mathcal{G}$ when generating interactions is necessary. When we ablate the user goal, the QM score on SParC drops from 62.4 to 59.8. We argue that it is important to condition on the user goal to obtain grounded interactions.

We also reimplement the method used in Zhong et al. (2021) by ablating both the user goal and the context. The QM score on SParC drops from 62.4 to 58.3. We argue that it is important to condition on the user goal and the full context to obtain grounded interactions.

A.4 Template Examples

Top Templates Unseen in Train	**Proportion**
`select text_col_0 group_by key_col_0 order_by count (*_col_0) desc limit_value`	0.19%
`select number_col_0 group_by number_col_0 order_by count (*) desc limit_value`	0.12%
`select text_col_0 , count (*_col_0) group_by text_col_0 order_by count (*) desc`	0.07%

Table 8: Examples of generated templates unseen in SParC train.

Template and Example	Improvement
select sum (number_col_0) e.g. SELECT sum(number_products) FROM shop	50
select text_col_0 , count (*_col_0) group_by text_col_0 e.g. SELECT Nationality , COUNT(*) FROM people GROUP BY Nationality	42.9
select text_col_0 where key_col_0 = value e.g. SELECT AirportName FROM AIRPORTS WHERE AirportCode = "AKO"	28.6
select text_col_0 where key_col_0 not in (select key_col_1) e.g. SELECT Name FROM people WHERE People_ID NOT IN (SELECT People_ID FROM poker_player)	-22.2

Table 9: Examples of templates on which self-play improves (or reduces) performance.