CONVERSATIONAL TEXT-TO-SQL: AN ODYSSEY INTO STATE-OF-THE-ART AND CHALLENGES AHEAD

Sree Hari Krishnan Parthasarathi, Lu Zeng, Dilek Hakkani-Tür
Alexa AI, Amazon

ABSTRACT

Conversational, multi-turn, text-to-SQL (CoSQL) tasks map natural language utterances in a dialogue to SQL queries. State-of-the-art (SOTA) systems use large, pre-trained and finetuned language models, such as the T5-family, in conjunction with constrained decoding. With multi-tasking (MT) over coherent tasks with discrete prompts during training, we improve over specialized text-to-SQL T5-family models. Based on Oracle analyses over n-best hypotheses, we apply a query plan model and a schema linking algorithm as rerankers. Combining MT and reranking, our results using T5-3B show absolute accuracy improvements of 1.0% in exact match and 3.4% in execution match over a SOTA baseline on CoSQL. While these gains consistently manifest at turn level, context dependent turns are considerably harder. We conduct studies to tease apart errors attributable to domain and compositional generalization, with the latter remaining a challenge for multi-turn conversations, especially in generating SQL with unseen parse trees.

Index Terms — Conversational Text-To-SQL

1. INTRODUCTION

Text-to-SQL is an important research topic in semantic parsing [1, 2, 3, 4, 5, 6, 7]. Spider [3] and CoSQL [5] datasets allow for making progress in complex, cross-domain, single and multi-turn text-to-SQL tasks respectively, utilizing a common set of databases, with competitive leaderboard results, demonstrating the difficulty in the tasks. In contrast to Spider, CoSQL was collected as entire dialogues, and hence includes additional challenges for the text-to-SQL task in terms of integrating dialogue context. In addition to the challenges in general-purpose code generation [8, 9], where the output of the system is constrained to follow a grammar, the text-to-SQL problem is underspecified without a schema. Since public text-to-SQL tasks use relatively small datasets, previous solutions employ: a) small encoder/decoder models, with constraints on the decoder [10, 11]; b) large pretrained language models (LMs) without constraints on the decoder [12, 13], pruning finalized hypotheses. PICARD [14], a top entry on the text-to-SQL leaderboard, finetunes a pretrained T5 model and imposes SQL syntax during beam search.

Our focus is on multi-turn, conversational text-to-SQL (CoSQL), and we build a system utilizing PICARD. Motivated by the notion that multi-task training (MT) [15] using coherent tasks and utilizing inductive biases can improve accuracy, we aggregate coherent task data (from CoSQL, Spider, and SParC) along with task-specific discrete prompts during training. Next, persuaded by our previous research [16], we conduct Oracle studies on n-best lists from PICARD CoSQL system, inferring that reranking can be helpful. We adapt the two reranking methods from [16], query plan (QP) and schema linking (SL), and show that both methods can help improve multi-turn text-to-SQL. With accuracy on CoSQL being reported using exact-set-match accuracy (EM) and execution accuracy (EX), with T5-Large we observed: a) MT leads to 2.4% and 1.7% absolute improvement on EM and EX; b) combined reranking approaches yield 1.9% and 2.2% improvements; c) combining MT with reranking, with T5-Large we obtain improvements of 2.1% in EM and 3.7% in EX over a T5-Large PICARD baseline. This improvement is consistent on larger models, using T5-3B yielded about 1.0% in EM and 3.4% in EX over SOTA baseline. We also submitted our system to the CoSQL leaderboard, our system consistently improves over PICARD baseline on the held out test set on question match (1.2% absolute) and interaction match (1.1% absolute). Lastly, we analyze errors in terms of zero-shot domain and compositional generalization. All improvements presented in this paper are absolute gains (i.e., not relative).

Contributions. The contributions of this paper are: a) proposing MT and combining with n-best reranking to improve over SOTA on a competitive multi-turn, conversational text-to-SQL task; b) analysis of gains at turn level, zero-shot domain generalization, and compositional generalization – showing challenges in compositional generalization for multi-turn conversations.

2. CONVERSATIONAL TEXT-TO-SQL

2.1. Related Work

Conversational text-to-SQL: A comprehensive survey of text-to-SQL is provided in [17]. Historically, much more research has been undertaken in single-turn text-to-SQL [1, 3, 18]. SParC and CoSQL are recent multi-turn text-to-SQL datasets, with CoSQL being more realistic dialogues. Most previous works on multi-turn text-to-SQL...
attempt to encode context in utterances and/or previous SQL queries
into the model [19, 20, 21, 22, 23]. On the other hand, PICARD [14]
does not treat utterance/SQL context in a special fashion, and relies
on large pretrained LMs (PLM) with constrained decoding, obtain-
ing SOTA results. Our work brings context modeling into [14] in two
ways: a) using contextual information in the two n-best reranking
methods [16, 22]; b) better context learning via data augmentation
and prompting to reduce cross-task interactions in MT [24, 25].

Compositional generalization is an active area of research in se-
manic parsing [26], with the focus primarily on single-turn utter-
ances [27]. On the other hand, multi-turn text-to-SQL benchmarks
are set up to evaluate cross-domain generalization [4, 5]. In this
work, we attempt to bridge both aspects: compositionality analysis
in multi-turn cross-domain text-to-SQL.

2.2. Proposed Approach
The overview of the proposed system is shown in Fig 1. PICARD is
part (b) in the figure, while multi-task prompts and n-best list rerank-
ing methods are shown in parts (a) and (c).

A) Proposed Multi-task Prompt (MT) Approach: We ensemble
data from similar semantic parsing tasks (CoSQL, Spider, and
SParC), and inject an inductive bias to the model by using discrete,
task-specific, natural language prompts in input. We simply use the
 task names as the prompts, and the CoSQL example shown in Fig 1
has a prompt of “cosql :”.

B) Proposed Reranking with MT Query Plan (QP) Model: Start-
ing from [16], we build a model that focuses specifically on im-
proving long span coherence: a multi-label classification model that
generates a query plan predicting whether the predicted SQL query
should contain any of the 8 clauses (WHERE, EXCEPT, UNION,
and INTERSECT, GROUP BY, HAVING, ORDER BY, LIMIT). We
adapt the approach so that the QP model is trained in an MT fashion.

C) Proposed Reranking with Schema Linking (SL) on conversa-
tional setting: We adapt the heuristic algorithm in [16] for the
multi-turn setting. For each predicted SQL query in n-best list, we
follow three steps: a) extract slot names and their respective values
from the conditions in the WHERE clause; then check if the slot
value exists in any of the referenced tables in the FROM clause. b)
output a list of candidate slot names/values from the current and pre-
vious turns in the interaction, which are exact/partial occurrences of
the column/table names and string values in the question with name-
base and value-based linking described in RAFT-SQL [10]; c) For
value linking, we next consider prefix/abbreviation matches on slot
values with categorical types.

3. EXPERIMENTAL SETUP

3.1. Datasets and Metrics
We briefly describe the datasets and metrics used in this paper [3].

A) Text-to-SQL Datasets: All 3 datasets (CoSQL, Spider, and
SParC) are based on 200 databases (covering 138 domains), each
with multiple tables. The standard protocol splits them into 140
databases for training, 20 databases for development (DEV), and 40
databases are held back for evaluation. Databases have no overlaps
across the splits. CoSQL is a conversational dataset, containing
3k dialogues, with 30k+ turns and 10k+ annotated SQL queries
(collected in Wizard-of-Oz fashion). Dialogues are split into 2,164
for training, and 292 for DEV, and 551 for evaluation. Spider is
a single-turn dataset, containing 10,181 questions with 5,693 SQL
queries. The examples are split into 7,000 for training, 1,034 for
DEV, and 2,147 for evaluation. SParC is a sequential text-to-SQL
dataset containing 4,298 coherent question sequences; these are split
into train (3034), DEV (422), and evaluation (842). We mainly
present results on CoSQL (with some discussion on Spider to ana-
lyze generalization properties).

B) Text-to-SQL Metrics: As mentioned previously, performance
is evaluated using EM and EX on the CoSQL and Spider (DEV),
with DEV being used for eval, and no hyperparameter tuning being
done on them. EM compares each clause between a prediction
and its corresponding groundtruth SQL query. The predicted SQL
query is correct only if all of the components match. This metric
does not take values into account. EX compares the execution out-
put of the predicted SQL query and its corresponding groundtruth
SQL queries. Note that both EM and EX can lead to false positives
and false negatives.

3.2. Models
A) Baseline Model: PICARD [14] is our baseline, and it is trained
on Spider and CoSQL; we focus on two T5 model sizes, T5-Large
and T5-3B. Input to the model includes current natural language
turn, database name, and serialized database schema (table, name
: col1, ..., colN) with database content, and previous
turns from the dialogue in reverse chronological order. During
inference, constrained decoding (CD) is integrated into beam search,
with a beam size of 10.

B) Multi-tasking Prompting (MT) T5 Model: We introduced
two changes: a) augment CoSQL and Spider with SParC and
weight them equally during training; b) to reduce variance in es-
estimated parameters, we inject inductive biases with task specific
discrete prompts, extending the input. We finetune the model on
p3dn.24xlarge instances (8 NVIDIA Tesla V100 GPUs) with teacher
forcing and cross-entropy loss for 3000 epochs using a batch size of
2000 and a learning rate of 1e-4.

C) MT Query Plan (QP) Model: We finetune RoBERTa-Large
models with a sequence classification head on p3.2xlarge instances
(1 NVIDIA Tesla V100 GPU). We reused the input from MT T5
model (without database content), and output 1-hot encoded to pre-
dict labels extracted from groundtruth queries based on the existence
of the 8 clauses. Models are finetuned with binary cross entropy loss
for 100 epochs using a batch size of 5 and a learning rate of 1e-5.

4. RESULTS

4.1. Multi-Tasking Approach
While we follow a restrictive MT strategy (3 coherent tasks, Spi-
der, CoSQL, and SParC), UnifiedSKG [25] follows a generalist MT
strategy of training on 21 structured knowledge grounding (SKG)
tasks (including the 3 tasks above). Table 1 shows results on CoSQL
without CD against baseline. Note that the baseline (52.5%) pe-
forms better than UnifiedSKG (51.6%). Furthermore, the proposed
MT approach performs significantly better than both models.

| Table 1: T5-large: MT performance on CoSQL without CD. |
|-----------------|-----------------|
|                | MT Methods      | EM%  |
| Baseline        | 52.5            |
| UnifiedSKG MT-P | 51.6            |
| Proposed MT     | 54.6            |

4.2. Oracle Analysis and Reranking Approaches
We enable CD and perform Oracle analysis on 10-best hypotheses.
The study is done on the baseline and proposed MT model at two se-
lected T5 model sizes: T5-Large and T5-3B. In Table 2, for each row
block, the first row is the 1-best, while the other row shows Oracle
accuracies. As can be observed, both EM and EX improve significantly: for example, the T5-Large proposed MT model gains 12.6% and 10.8% absolute for EM and EX respectively. Similar gains can be seen for the baseline at the same model size, as well as for the T5-3B models, following trends observed in our paper [16]. Note that with T5-large models, we observed that with CD the first and third rows in Table 1 yielded an EM of 54.4% and 56.8% respectively.

Table 3 lists reranking results on 10-best hyptheses obtained from baseline using a T5-Large model. SL contributes 0.4% and 2.0% absolute improvement on EM and EX, while QP yields smaller gains, with the gains from SL and QP being additive.

Table 2: Oracle accuracies with beam size of 10.

<table>
<thead>
<tr>
<th>Model</th>
<th>T5-Large EM%</th>
<th>T5-Large EX%</th>
<th>T5-3B EM%</th>
<th>T5-3B EX%</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>54.4</td>
<td>63.7</td>
<td>57.1</td>
<td>66.6</td>
<td>1-best Oracle</td>
</tr>
<tr>
<td>Proposed MT</td>
<td>56.8</td>
<td>65.4</td>
<td>58.3</td>
<td>68.7</td>
<td>1-best Oracle</td>
</tr>
</tbody>
</table>

4.3. Combined Results with MT and Reranking

We present results combining MT and reranking approaches (SL and QP) in Table 4. Compared to the baseline (T5-Large PICARD CoSQL), the proposed system achieves significant improvement: we observe 2.1% and 3.7% absolute improvement on EM and EX. The table also shows the effect of each component: MT affects the model performance the most on EM and SL leads the most improvement on EX, while QP contributes the least. Note that the overall gains also carry over to T5-3B, with the EM of 58.0% and EX of 70.0% representing improvements over the SOTA baseline model. We also submitted our system to CoSQL leaderboard, on the held out test set, PICARD baseline obtained a question match of 54.6%, while our system achieved 55.8%. We also improved the interaction match from 23.7% to 24.8% with our proposed approach.

Table 4: CoSQL: Baseline, combined results using MT and reranking, and ablations removing one component at a time.

<table>
<thead>
<tr>
<th>Method</th>
<th>T5-Large EM%</th>
<th>T5-Large EX%</th>
<th>T5-3B EM%</th>
<th>T5-3B EX%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>54.4</td>
<td>63.7</td>
<td>57.1</td>
<td>66.6</td>
</tr>
<tr>
<td>Proposed MT</td>
<td>56.5</td>
<td>67.4</td>
<td>58.0</td>
<td>70.0</td>
</tr>
<tr>
<td>− MT</td>
<td>55.3</td>
<td>65.9</td>
<td>57.8</td>
<td>69.1</td>
</tr>
<tr>
<td>− SL</td>
<td>56.6</td>
<td>65.6</td>
<td>58.2</td>
<td>69.2</td>
</tr>
<tr>
<td>− QP</td>
<td>56.7</td>
<td>67.2</td>
<td>58.1</td>
<td>69.5</td>
</tr>
</tbody>
</table>

Task difficulty levels: We now analyze the gains over difficulty levels. Table 5 shows that the gains from the proposed system (T5-Large model combining MT and reranking) consistently carry over all the pre-defined difficulty levels in CoSQL task.

Effect of additional data on MT: The proposed MT approach has two parts: a) utilizing additional SParC data compared to baseline; b) attaching task specific prompts to model inputs. For T5-Large, the effect of adding extra data is presented in Table 6 with CD (1.2% and 0.4% absolute improvement on EM and EX). Adding task specific prompts give another 1.2% and 1.3% improvement on the metrics.

4.4. Turn Level Analysis

Table 7 shows the distribution of difficulty levels across turns. It confirms that later turns have more complex SQL queries. Since the sample counts are extremely small after Turn 5, we merged all subsequent turns into that bin, so that each row in Table 7 has at least 120 examples. These bins are then used in Fig 2 to present the performance of the baseline and proposed systems at each turn. The green line shows sample counts of the bins, and numbers could be read from the y-axis on the right side. From the bar chart, we can clearly see the proposed system consistently performs better than the baseline (never worse than the baseline).

Table 5: T5-large: Performance across difficulty levels on CoSQL.

<table>
<thead>
<tr>
<th>Diff</th>
<th>Total</th>
<th>Baseline EM%</th>
<th>Baseline EX%</th>
<th>Proposed EM%</th>
<th>Proposed EX%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Easy</td>
<td>417</td>
<td>74.1</td>
<td>77.9</td>
<td>75.3</td>
<td>82.3</td>
</tr>
<tr>
<td>Med</td>
<td>320</td>
<td>51.2</td>
<td>60.0</td>
<td>53.8</td>
<td>63.8</td>
</tr>
<tr>
<td>Hard</td>
<td>162</td>
<td>34.0</td>
<td>59.3</td>
<td>37.7</td>
<td>61.1</td>
</tr>
<tr>
<td>Extra</td>
<td>107</td>
<td>17.8</td>
<td>26.2</td>
<td>19.6</td>
<td>29.9</td>
</tr>
<tr>
<td>Total</td>
<td>1006</td>
<td>54.4</td>
<td>63.7</td>
<td>56.5</td>
<td>67.4</td>
</tr>
</tbody>
</table>

Table 6: Additional data effect.

<table>
<thead>
<tr>
<th>Data</th>
<th>EM%</th>
<th>EX%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>54.4</td>
<td>63.7</td>
</tr>
<tr>
<td>W/ extra data</td>
<td>55.6</td>
<td>64.1</td>
</tr>
<tr>
<td>+ MT prompt</td>
<td>56.8</td>
<td>65.4</td>
</tr>
</tbody>
</table>

Fig. 2: T5-large models: Turn level analysis. The y-axes on the left hand side show EM and EX for (a) and (b) respectively, while the right hand side shows sample counts per bin.

Dialogue Context Dependency: Some turns are highly dependent on the context provided by previous turns in the dialogue, while others are independent of the context. To understand performance of the proposed system in both cases, we manually annotated each turn in CoSQL DEV set as to whether the context is needed. We then divide the DEV examples into two groups: a) 712 context-independent examples; b) 294 context-dependent examples. Table 8 shows that for T5-Large the proposed system achieves 2.1% and 3.2% absolute improvement, while QP contributes the least. Note that the overall gains also carry over to T5-3B, with the EM of 58.0% and EX of 70.0% representing improvements over the SOTA baseline model.

Table 7: Distribution of difficulty level at turns.

<table>
<thead>
<tr>
<th>Turn #</th>
<th>Total counts</th>
<th>Difficulty level Distribution (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Easy</td>
<td>Medium</td>
</tr>
<tr>
<td>1</td>
<td>261</td>
<td>49.8</td>
</tr>
<tr>
<td>2</td>
<td>228</td>
<td>39.0</td>
</tr>
<tr>
<td>3</td>
<td>217</td>
<td>38.2</td>
</tr>
<tr>
<td>4</td>
<td>135</td>
<td>37.8</td>
</tr>
<tr>
<td>5+</td>
<td>165</td>
<td>38.8</td>
</tr>
</tbody>
</table>
5. DISCUSSION

5.1. Performance on Spider

Table 9 shows the proposed approach also improves over the baseline on the Spider task as well. For example, compared to baseline Spider task-specific model (with T5-Large), we observed 0.6% and 1.7% absolute improvement on EM and EX respectively. The T5-Large proposed system even outperforms the SOTA baseline Spider model (a T5-3B model). The table also presents ablations removing one component at a time, showing similar trends as previously in Section 4.3.

Table 9: Cross task generalization: Performance on Spider.

<table>
<thead>
<tr>
<th>Method</th>
<th>EM%</th>
<th>EX%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline (T5-Large)</td>
<td>75.1</td>
<td>79.4</td>
</tr>
<tr>
<td>Baseline (T5-3B)</td>
<td>75.6</td>
<td>79.3</td>
</tr>
<tr>
<td>Proposed (T5-Large)</td>
<td>75.7</td>
<td>81.1</td>
</tr>
<tr>
<td>– MT</td>
<td>73.4</td>
<td>79.6</td>
</tr>
<tr>
<td>– SL</td>
<td>75.2</td>
<td>79.1</td>
</tr>
<tr>
<td>– MT-QP</td>
<td>75.0</td>
<td>80.9</td>
</tr>
</tbody>
</table>

5.2. Zero-Shot Generalization (ZSG)

CoSQL and Spider tasks are designed to be cross-domain (without database overlaps over splits), system performances are then reported in zero-shot. However, we wanted to tease apart generalization performance separately due to compositionality and zero-shot domain. To attribute zero-shot “only” generalization performance, we ignore DEV examples with parse trees unseen in MT training data (meaning that all the remaining parse trees were observed in the training data). However, this contains examples with unseen DB schema: we then report ZSG for the proposed and baseline systems in Table 10. The performance reported for all systems are now better (because they are slightly easier examples, having removed unseen parse trees). The system generalizes well on both CoSQL and Spider task, with at least 2% improvement in the metrics at each case. Consistent with previous results, QP contributes the least in this study. A challenge in performing inference on unseen DB schema is the primary/foreign key relationships among tables, such as the example below (note that the turns are concatenated with separator “|”).

Example: Can you list all the continents? | Okay, now can you list number of countries belonging to each continent, including continent id as well?

Gold SQL: SELECT T1.continent, count (*) FROM continents AS T1 JOIN countries AS T2 ON T1.contid = T2.countryid group by T1.contid

5.3. Compositional Generalization (CG)

To analyze CG, we mix the train and DEV sets of the 3 datasets and re-do the split (meaning that the new train and DEV overlap in DB schema, and therefore are not zero-shot). We re-train the baseline and the proposed approach, and then present the systems with held-out examples having unseen parse trees. Table 11 shows the results: the performance of all systems are poor (even more so on CoSQL, than on Spider), showing that CG is a challenge, especially in conversational multi-turn tasks. The proposed system obtains a 1% absolute improvement on EM and EX on both CoSQL and Spider. Based on the ablution studies, QP hurts performance. Below is an example of a SQL with a novel parse tree that the model has to construct now and gets wrong.

Example: How many customers have loans that total more than 5000?

Gold SQL: SELECT count (*) FROM (SELECT * FROM customer AS T1 JOIN loan AS T2 ON T1.cust_id = T2.cust_id GROUP BY T1.cust_name HAVING sum (T2.amount) > 5000)
7. REFERENCES


