N-BEST HYPOTHESES RERANKING FOR TEXT-TO-SQL SYSTEMS

Lu Zeng, Sree Hari Krishnan Parthasarathi, Dilek Hakkani-Tur

Alexa, Amazon, USA.

{luzeng, sparta,hakkanit}@amazon.com

ABSTRACT

Text-to-SQL task maps natural language utterances to structured queries that can be issued to a database. State-of-the-art (SOTA) systems rely on finetuning large, pre-trained language models in conjunction with constrained decoding applying a SQL parser. On the well established Spider dataset, we begin with Oracle studies: specifically, choosing an Oracle hypothesis from a SOTA model's 10-best list, yields a 7.7% absolute improvement in both exact match (EM) and execution (EX) accuracy, showing significant potential improvements with reranking. Identifying coherence and correctness as reranking approaches, we design a model generating a query plan and propose a heuristic schema linking algorithm. Combining both approaches, with T5-Large, we obtain a consistent 1% improvement in EM accuracy, and a 2.5% improvement in EX, establishing a new SOTA for this task. Our comprehensive error studies on DEV data show the underlying difficulty in making progress on this task.

Index Terms— Text-To-SQL, Semantic parsing

1. INTRODUCTION

Large language models (LM) are widely used for natural language generation [1, 2, 3]. Recently, the use of large LMs has extended to semantic parsing tasks such as code generation. For general-purpose code generation, large LMs such as Codex [4] are trained on massive, paired codebases (code, NL). For domain-specific code generation tasks, such as Text-to-SQL that aims to convert natural language instructions to SQL queries, multiple public domain datasets are available (with associated leaderboards), including Spider [5], CoSQL [6] and SParC [7]. However, the amount of training data in these datasets is much smaller than the natural language and code pairs mined from the internet. In such cases, instead of training a new model from scratch, the pretrain/finetune strategy with publicly available LMs has been shown to be more accurate. For example, in UnifiedSKG [8], the T5 model [1] is finetuned for various semantic parsing tasks (including Text-to-SQL), achieving SOTA performance. Besides the amount of the training data, another challenge with SQL generation is that the generated code is underspecified without the corresponding schema [1]. To handle this challenge, implicit or explicit schema linking becomes an important sub-task for SQL code generation [9, 10, 11].

In this paper, we focus on complex, cross-domain, SQL generation using Spider [5] a large-scale, Text-to-SQL dataset consisting of 200 complex databases covering 138 domains. Spider is a well established dataset with nearly 70 entries on the leaderboard, demonstrating the difficulty of the task. The data is split into training, development (DEV), and test sets without overlaps in databases across these sets, as the aim of this task is to learn models that can issue queries from natural language text to previously unseen databases in the training set. Furthermore, SQL queries in this dataset contain nested sub-queries, requiring the model to understand compositional structures. The model performance is based on two metrics, exact-set-match accuracy (EM) and execution accuracy (EX); the former compares individual query components between the predicted and groundtruth SQL queries, while the latter compares their execution output.

Similar to general-purpose code generation, the output of Text-to-SQL models is constrained to follow SQL grammar. Previous solutions have employed encoder/decoder models, with the decoder being constrained to produce well-formed outputs [12, 13]. An approach that is more compatible with large pretrained LMs is to remove the constraints on decoder: [14, 15] prune the finalized hypotheses from beam search to those that are syntactically correct. Meanwhile, PICARD [16], a top entry on the Spider leaderboard [2], in addition to finetuning the pretrained T5 model, also imposes SQL syntax constraints during beam search (via constrained decoding). It achieves an EM of 74.8% and an EX of 79.2% on the Spider DEV set. Natural ways to potentially improve these metrics include increasing the model size or collecting and/or synthesizing additional training data [17].

In this paper, we take a different approach: we first perform an Oracle analysis on n-best lists obtained from PICARD and observe significant improvements (7.7% absolute improvement in EM and EX) even at small beam sizes (such as 10). The gap in performance between 1-best and Oracle motivates reranking approaches. We propose 2 reranking

[1]For Text-to-SQL, in addition to SQL grammar constraints, there are schema constraints.

[2]https://yale-lily.github.io/spider

978-1-6654-7189-3/22/$31.00 ©2023 IEEE

approaches that are motivated by the issues with the current, large pretrained LMs: *coherence* [18] and *correctness*. To improve coherence, we explore n-best reranking using a query plan produced by an independent model. Next, to improve correctness, we propose a heuristic algorithm performing schema linking on n-best lists – imposing constraints that are missing in PICARD. The combined reranking approaches consistently yield improvements across all T5 model sizes, obtaining a 1.0% absolute improvement in EM and a 2.5% absolute improvement on EX. Lastly, to analyze persistent gap with Oracle results, we performed a detailed analysis of errors and metrics. Our studies show that annotation and metrics issues can be significant factors to improving models and evaluation on Spider.

Contributions. The contributions of this paper are: a) Using Oracle analysis and rerankers on n-best lists, we outperform SOTA performance on a competitive Spider task; b) Analysis of errors on Spider DEV data, showing much work remains to be done on metrics and annotation.

2. RELATED WORK

We review related work in 4 categories: a) reranking approaches; b) coherence in LMs; c) schema linking for text-to-SQL; d) noise in Spider.

Rerankers. N-best list reranking has a long history in speech recognition, machine translation, and semantic parsing communities [19, 20, 21, 22, 23, 24, 25]. These are performed with a much larger 2^{nd}-stage model [26, 27] or using additional knowledge sources not available during the first pass [28, 29]. On Spider, a larger 2^{nd}-stage reranker was successfully used in [30]. Some reranking methods combine outputs from multiple systems [31] or use improved output confidence [32]. Imposing task specific constraints [14, 15] to remove invalid options can be a helpful strategy to improve reranking. In our work, in addition to the output target being structured queries (instead of natural language), the 1^{st}-stage model is large (T5 family), and the baseline model imposes additional knowledge during beam search in the form of SQL parsing. Our proposed methods are designed for this setting.

Coherence with query plan. Coherence issues with LMs [33] are associated with hallucination phenomenon [34, 35]. While these are well-known problems in unstructured text generation with LMs [36, 37, 38, 39], coherence issues with structured text generation is somewhat less unexplored: [40] uses semantic coherence for data augmentation, while [41] uses a reconstruction model for reranking semantic parses, and [14, 15, 16] use parsers to improve coherence with SQL grammar. Our reranker is designed to improve semantic coherence, and it is distinct from [41] in that, our query plan model predicts the structure of a query (from natural language), and orders the n-best list for consistency.

Correctness with schema linking. Historically, schema linking (SL) has been explored for Text-to-SQL. Schema has been encoded in model input for small encoder/decoder models [15, 42, 43] and large pretrained LMs [8, 16]. More work has been done on modeling schema linking [9], where the role of SL in Text-to-SQL is recognized. RAT-SQL [12], a system that had been influential in text-to-SQL, proposed a framework to encode relational structure in the database schema and simultaneously address schema linking. In contrast, we use schema linking as a post-processor and a reranker on the output of large LMs.

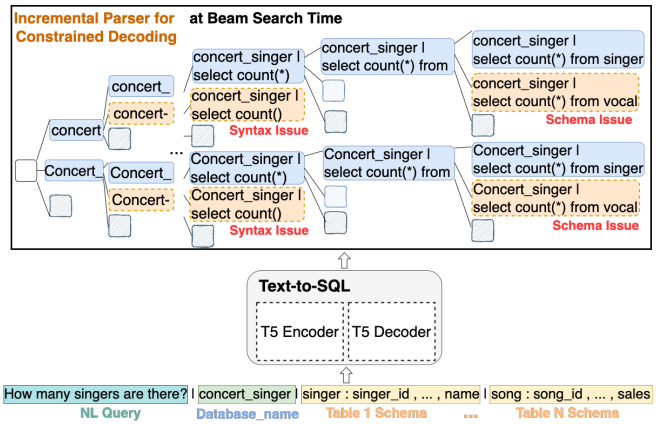

Fig. 1. PICARD explained by an example. The prediction pattern is "⟨Database_name⟩ | ⟨pred_SQL⟩".

Noise in Spider. Spider dataset [5] has been well-explored in Text-to-SQL, and it's corpus noise are documented in related work [9, 12, 44]. One type of corpus noise comes from annotation errors in groundtruth SQL queries and typos in natural language queries [9]. Another source of the noise is in the pairing of only one groundtruth SQL query to each natural language query, which results in a large portion of predicted queries being incorrectly marked as "mispredicted" [12, 44].

3. TEXT-TO-SQL USING PRE-TRAINED LMS

We use PICARD as our baseline system, and use it to produce the n-best hypotheses in this paper. It uses finetuned T5 models, and implements incremental SQL parsing to constrain decoding during beam search. Input to the T5 model includes natural language query, database name, and database schema (`table_name : col1 , ... , coln`). PICARD employs a post-processor during beam search, and to save computing time it only checks the next top-k tokens for validity. Figure 1 shows how predictions are generated when PICARD is enabled, using $k = 2$ and beam size of 3. Shadowed box means that the result is not expanded or explored for the next time stamp; blue (solid) boxes show valid SQL query prefixes, while orange (dashed) boxes show sequences that violate SQL syntax.

The incremental parser not only filters out hypotheses that violate the SQL grammar, but also imposes rudimentary schema checks. Fig. 1 gives two examples of filtering hypotheses, the first one is due to a syntax error (e.g., `count()`) and the other one is due to schema error (e.g.,

table name `vocal` does not exist in the database).

4. RERANKING APPROACHES

In this section, we discuss our two reranking approaches over n-best hypotheses.

4.1. Improving Coherence with Query Plan

Text generation is a challenging topic, and large LMs tend to lose coherence [18, 39]. This is compounded in text-to-SQL with such models: apart from conforming to syntactic structures, models have to perform semantic mapping from natural language to SQL over long spans in complex questions involving compositionality, grouping/ordering/counting. Furthermore, nested queries are likely to appear with `WHERE, EXCEPT, UNION, and INTERSECT` clauses, while grouping/ordering/counting tend to occur with `GROUP BY, HAVING, ORDER BY, LIMIT`.

We build a model that focuses specifically on this aspect: a multi-label classification model that generates a query plan predicting whether a SQL query contains the aforementioned 8 clauses. The output of this model is then used as a coherence check reranker against the n-best hypotheses from the baseline model. We finetune a RoBERTa-Large using the classification head with binary cross entropy (BCE) loss. The model is trained using the training partition of Spider dataset. Input of the model consists of a natural language query and a database schema, while the labels are obtained from groundtruth SQL queries.

4.2. Improving Correctness with Schema Linking

The motivation for this approach comes from the fact that certain aspects of schema linking (values, columns, and tables) from natural language query (on the input side of the model) to a SQL query (on the output side of the model) can be hard for the model to learn to map: this is because these mappings tend to be irregular, and a relatively small training dataset may not cover a large fraction of the irregularities. For example, a primary key field on "student identifcation" could be named in several ways as a column such as "stud_id", "sid", "s_id" etc. Similarly, values that are strings can be arbitrarily represented, such as "True" vs "T" vs "Yes".

We implement an algorithm to perform value linking in `WHERE` clauses. For each predicted SQL query in the n-best list, we follow three steps:

1. 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 of the query;
2. Obtain a list of candidate slot names and values, which are exact or partial occurrences of the column/table names and string values in the question with name-based and value-based linking described in RAT-SQL [12];
3. Perform prefix and abbreviation matches on slot values with categorical types in a table schema definition (such as matching "left" to "L").

5. EXPERIMENTS

This section presents the experimental setup, baselines, Oracle analysis, and the main results.

5.1. Dataset and Metrics

We briefly describe the dataset and metrics used in this paper. More details can be found in [5].

5.1.1. Dataset

Spider is a large-scale, cross-domain paired text-to-SQL dataset: it contains 10,181 questions and 5,693 unique complex SQL queries on 200 databases (covering 138 domains), each with multiple tables. The standard protocol splits this into 8,659 examples on 146 databases for training, and 1034 examples on 20 databases are used for development (DEV); the test set that includes 2,147 examples from 34 databases are held back. The split is done without overlaps of databases across these sets.

Task difficulty levels. Spider categorizes the SQL query complexity into 4 difficulty levels: *easy*, *medium*, *hard*, and *extra hard*. Queries that contain more SQL keywords (`GROUP BY`, `ORDER BY`, `INTERSECT`), nested subqueries, column selections and aggregators are considered harder.

5.1.2. Metrics

On Spider, Text-to-SQL model performance is evaluated based on two metrics: exact-set-match accuracy (EM) and execution accuracy (EX). Note that all our results are on the Spider DEV set.

Exact-set-match accuracy (EM). EM compares each clause[3] 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. EM can lead to false positives and false negatives.

Execution Accuracy (EX). EX compares the execution output of the predicted SQL query and its corresponding groundtruth SQL queries. However, it is important to note that EX can also lead to false positives and false negatives.

5.2. Models

We use PICARD [16] as our baseline model in this paper. We finetune on three T5 model sizes, T5-Base[4], T5-Large[4], and T5-3B on p3dn.24xlarge instances (8 NVIDIA Tesla V100 GPUs) and employ DeepSpeed to save memory[5]. The input to the models contain several segments separated by `|`,

[3]Note that EM implemented by Spider ignores table join conditions.

[4]Note that the models are *LM adapted*, initialized from T5 model and trained for additional 100K steps on the LM objective discussed in [1]

[5]https://github.com/microsoft/DeepSpeed

including natural language query, database name, serialized database schema. We take the serialization scheme mentioned in [45] and enable database content by appending database values to the column names [15]. All the T5 models are fine-tuned with teacher forcing and cross-entropy (CE) loss for up to 3000 epochs using a batch size of 2000 and a learning rate of $1e^{-4}$. During inference, an incremental SQL parser is integrated into beam search, with a beam size of 4.

To improve coherence, we finetune a RoBERTa-Large model with a sequence classification head on p3.2xlarge instances (1 NVIDIA Tesla V100 GPU) for query plan prediction. We reuse the input from the baseline model. The output is one-hot encoding label extracted from groundtruth queries based on existence of `WHERE`, `GROUP BY`, `HAVING`, `ORDER BY`, `LIMIT`, `EXCEPT`, `UNION`, and `INTERSECT` clauses. The RoBERTa model is finetuned with binary cross entropy (BCE) loss for up to 100 epochs using a batch size of 5 and a learning rate of $1e^{-5}$.

5.3. Oracle Analysis

We open the beam up and perform Oracle analysis on n-best hypotheses. Specifically, we conducted these experiments over a set of beam sizes (and correspondingly n-best hypotheses), 10, 15, 20, and 25; also a few selected T5 model sizes, T5-Base[4], T5-Large[4], and T5-3B.

Table 1. 1-best accuracies as beam size is changed.

Beam size	T5-Base		T5-Large		T5-3B	
	EM%	EX%	EM%	EX%	EM%	EX%
4	66.6	68.3	74.8	79.2	75.5	79.3
10	67.1	68.4	75.1	79.4	75.6	79.3
15	67.1	68.5	74.7	79.2	75.5	79.2
20	67.3	68.7	74.8	79.2	75.5	79.2
25	67.3	68.7	74.7	79.1	75.6	79.2

Search errors. Table 1 presents the analysis by increasing the beam size (and choosing the 1-best hypothesis): opening up the beam, improves the accuracy by a small amount initially (going from 4 to 10) for all models. However, the performance on EM and EX saturate after that, suggesting that search errors contribute a small proportion of overall errors.

Table 2. Oracle accuracies as beam size is changed.

Beam size	T5-Base		T5-Large		T5-3B		Note
	EM%	EX%	EM%	EX%	EM%	EX%	
10	67.1	68.3	75.1	79.4	75.6	79.3	1-best
10	74.7	75.5	82.8	87.1	85.2	87.3	Oracle
15	75.4	76.6	82.9	87.6	86.2	87.9	
20	76.1	77.6	83.5	88.0	86.3	88.0	
25	76.5	78.3	83.8	88.0	86.8	88.8	

Model errors. In Table 2, the first row is the 1-best obtained from baseline, while the other rows show Oracle accuracies for the three models sizes over different beams. As can be observed, both EM and EX improve significantly: for example, the T5-Large model achieves 7.7%, 7.8%, 8.4% and 8.7% absolute improvements for EM; similarly 7.7%, 8.2%, 8.6% and 8.6% absolute improvements for EX.

Table 3. T5-large: 1-best and Oracle over difficulty levels.

Difficulty	count	1-best		Oracle	
		EM%	EX%	EM%	EX%
Easy	248	89.1	91.9	93.1	96.0
Medium	446	80.9	84.8	88.6	92.6
Hard	174	65.5	71.3	74.7	79.3
Extra	166	48.8	54.8	60.2	67.5
Total	1034	75.1	79.4	82.8	87.1

Difficulty levels. Table 3 compares the 1-best with Oracle using baseline (with T5-Large) over different difficulty levels. It can be observed that the Oracle results are consistently higher over both metrics (EM and EX) at all difficulty levels. The biggest improvement occurs on extra hard queries, with absolute improvements of more than 10% for both metrics. Overall, these results suggest that significant improvements can be obtained using reranking approaches over n-best lists.

5.4. Results

5.4.1. Improving Coherence with Query Plan Modeling

Table 4 lists reranking results using query plan (QP) coherence on the 10-best hypotheses obtained from the baseline using a T5-Large model. It can be seen that using structural consistency with QP can help both EM and EX, obtaining 0.7% and 0.5% absolute improvements respectively. We present an example improved by QP. A query plan that combines `GROUPBY` and `HAVING` clauses is presented below.

Question: Find the average life expectancy and total population for each continent where the average life expectancy is shorter than 72?
Pred SQL: SELECT avg(lifeexpectancy), sum(population)
FROM country
WHERE lifeexpectancy < 72
GROUP BY continent
With QP: SELECT continent, avg(lifeexpectancy), sum(population)
FROM country
GROUP BY continent
HAVING avg(lifeexpectancy) < 72

Table 4. Reranking with query plan (QP) coherence checks on 10-best hypotheses from baseline using T5-Large.

Method	T5-Large	
	EM%	EX%
Baseline	75.1	79.4
QP	75.8	79.9

To understand the QP model better, we present its performance on each of the clauses in Table 5 in terms of F-1 score, precision and recall. While the model performs well on `WHERE`, `GROUP BY`, `HAVING`, `ORDER BY`, `LIMIT`, `INTERSECT`, it does not perform as well on

EXCEPT and UNION. We conjecture low training data counts for these clauses as a potential reason. The macro F-1 score for this model is 0.89.

Table 5. QP model performance per clause.

Clause	F1	R	P
where	0.97	0.97	0.98
groupBy	0.96	0.96	0.96
having	0.97	0.95	0.99
orderBy	0.96	0.97	0.95
limit	0.95	0.96	0.94
except	0.73	0.76	0.71
union	0.62	0.44	0.99
intersect	0.96	0.97	0.95

5.4.2. *Improving Correctness with Schema Linking*

Table 6 shows the reranking performance with schema linking over 10-best hypotheses obtained from a baseline model with T5-Large. With this approach, we see an absolute improvement in EX of 1.9%.

Table 6. Reranking with schema linking (SL) on 10-best hypotheses obtained from baseline using T5-Large.

Method	T5-Large	
	EM%	EX%
Baseline	75.1	79.4
SL	75.4	81.3

Below, we present 2 examples of schema linking errors (one of column name; and another of cell value) which are fixed by the proposed method.

Example 1

Question: What is the abbreviation of Airline "JetBlue Airways"?
Pred SQL: SELECT abbreviation FROM airlines
WHERE **abbreviation** = "JetBlue Airways"
Fixed SQL: SELECT abbreviation FROM airlines
WHERE **airline** = "JetBlue Airways"

Example 2

Question: What is the total population of Gelderland district?
Pred SQL: SELECT sum(population) FROM city
WHERE district = "**Geeland**"
Fixed SQL: SELECT sum(population) FROM city
WHERE district = "**Gelderland**"

In Ex 1, the question asks for the abbreviation of Airline "JetBlue Airways", where "JetBlue Airways" is a slot value for a slot name "airline". The model gets misled by an available column name that matches the query word "abbreviation". In Ex 2, we need a SQL query for the total population of "Gelderland" district. The expected slot value is "Gelderland", the predicted SQL obtained a cell value of "Geeland".

Syntax Checks with a Full Parser. Since incremental parsing can still allow syntactically incorrect full hypotheses, we investigated the errors from the baseline model. Around 2% of the hypotheses are indeed syntactically incorrect. For example, intermediate SQL queries from Fig. 1 such as `select count(*) from`, which are invalid queries but valid prefixes, are allowed by the incremental parser as outputs. We run a full SQL parser, filtering invalid hypotheses, and selecting the hypothesis with the highest remaining score.

Analysis. Results from our experiments showed that the full parser does not improve performance. We investigated the errors for the 23 examples with syntactically incorrect 1-best predictions: for 16 of them all other hypotheses are incomplete SQL queries, while the other 7 do not have any hypotheses that improves either EM or EX. These are errors that cannot be improved even with an Oracle selection, meaning that these are residual search errors.

5.4.3. *Combined Results*

We examine if the gains from SL and QP are additive, and if they carryover to different model sizes. We then combine SL and QP in order, presenting the results in Table 7.

Table 7. Performance of rerankers (QP, SL, SL + QP) on 10-best lists obtained from different T5 model sizes.

Method	T5-Base		T5-Large		T5-3B	
	EM%	EX%	EM%	EX%	EM%	EX%
Baseline	67.1	68.3	75.1	79.4	75.6	79.3
QP	67.8	68.9	75.8	79.9	75.9	79.6
SL	67.2	70.0	75.4	81.3	76.1	80.4
SL + QP	67.9	70.4	76.1	81.9	76.4	80.6

From the table, it can be seen that the performance of the different rerankers are consistent across model sizes: the SL approach yields a consistent gain of between 1.1% to 1.9% in EX. Similarly, QP yields a consistent gain as well. Overall the gains combining the approaches are additive on both EM and EX across model sizes. Based on the one-sided t-tests, improvements on T5-base and T5-large are significant at $\alpha = 0.1$, while the gain on T5-3B is significant at $\alpha = 0.15$.

5.4.4. *Analysis of gains across difficulty levels*

Table 8 shows that the gains from the proposed approach (combining SL and QP) consistently carry over all the predefined difficulty levels. Furthermore, the gains are bigger on harder examples, considering both EM and EX.

6. DISCUSSION ON ERRORS

Given that proposed reranker still has a gap with Oracle, we analyze errors with metrics computation and annotation.

6.1. Metrics Errors

PK/FK Swaps. EM implementation by Spider is done by replacing foreign keys with primary keys. This is problematic

Table 8. T5-large model: Performance of rerankers (combining SL and QP) on 10-best lists

Diff	count	Baseline		SL + QP	
		EM%	EX%	EM%	EX%
Easy	248	89.1	91.9	89.5	94.8
Med	446	80.9	84.8	81.6	87.9
Hard	174	65.5	71.3	68.4	73.6
Extra	166	48.8	54.8	49.4	55.4
Total	1034	75.1	79.4	76.1	81.9

and may lead to false positives. The following example contains a prediction that is marked as exact match incorrectly.

Question: How many flights depart from 'APG'?

Gold SQL: SELECT count(*) FROM flights
WHERE **sourceairport** = 'APG'

Pred SQL: SELECT count(*) FROM flights
WHERE **destairport** = 'APG'

*Note that **airportcode** from **airports** table is PK for **sourceairport** and **destairport** from **flights** table

We fixed the EM implementation and obtained the revised EM^* in Table 9. Generally there is a 3-4% absolute gap between the original and revised EM implementation.

Table 9. Revised EM implementation.

Method	T5-large LM		T5-3b	
	EM%	EM*%	EM%	EM*%
PICARD	75.1	72.1	75.6	71.9
SL + QP	76.1	72.9	76.4	72.5

Query rewrite errors. Since SQL queries can sometimes be rewritten in multiple ways, EM can yield false negatives. We conducted error analysis over hypotheses selected after our combined reranking system (SL + QP) using finetuned T5-large model with PICARD. We found that 7.9% of predictions are correct using EX, but incorrect with EM; also, 4.8% of hypotheses are correct SQL but different to the corresponding groundtruth SQL. In such examples with SQL rewrites, EX serves as a complementary metric to EM.

Content-only match errors. As described in [5], it is possible to obtain a predicted SQL that returns the same result as the groundtruth SQL but semantically different. We found that 4.4% of the hypotheses on DEV data generated from T5-Large model using PICARD are false positives of this type.

Redundant column errors. We found that some predicted or groundtruth SQL queries contain redundant columns. The columns repeat information or do not need to be included, however retaining such column won't affect the semantics of the query, but can result in EX errors.

Question: What is first, middle, and last name, along with id and number of enrollments, for student who enrolled the most in any program?

Gold SQL: SELECT t1.student_id, t1.first_name, t1.middle_name, t1.last_name, count(*), **t1.student_id**
FROM ...

Pred SQL: SELECT t1.student_id, t1.first_name, t1.middle_name, t1.last_name, count(*)
FROM ...

6.2. Annotation Errors

We found 5 unique groundtruth SQL queries with annotation errors in DEV set. In Spider, some natural language queries are presented multiple times; consequently, the 5 unique groundtruth SQL queries are assigned to 7 DEV examples. We show an example with annotation error whose natural language query has been written in two different ways below. The annotation error is at least 0.7%.

Question: Find all airlines that have **at least 10** flights.
Which airlines have **at least 10** flights?

Gold SQL: SELECT t1.airline FROM airlines as t1
JOIN flights as t2 ON t1.uid = t2.airline
GROUP BY t1.airline HAVING **count(*) > 10**

Fixed SQL: SELECT t1.airline FROM airlines as t1
JOIN flights as t2 ON t1.uid = t2.airline
GROUP BY t1.airline HAVING **count(*) >= 10**

6.3. Summary and Outlook on Errors

As we have seen, 0.7% of the DEV examples have incorrect groundtruth SQL queries. We also identified issues with EM implementation in Spider – foreign keys are automatically replaced with the corresponding primary keys, resulting in 3-4% of the predictions on DEV examples being incorrectly marked as exact matches (false positives). Also, each example in Spider is annotated with only one groundtruth SQL – leading to about 4.8% of the predictions on DEV examples being false negatives. Using EX, results in 4.4% of the predictions on DEV being false positive, while another 0.8% of the predictions are false negative. Our studies show that metrics and annotation need further research attention on Spider.

7. CONCLUSIONS

We began with Oracle studies, showing large gains can be obtained with n-best hypotheses reranking for a SOTA Text-to-SQL system. Motivated by correctness and coherence issues in text generation using large PLMs, we proposed two reranking approaches: a) schema linking; b) query plan modeling. On a competitive Spider dataset, applying our proposed reranking methods to 10-best hypotheses obtained from a SOTA system (PICARD), we achieve improvements of 1% in EM and 2.5% in EX. We analyzed errors in metrics computation and annotation – showing these can be significant factors to improving models and evaluation on Spider.

8. REFERENCES

[1] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.

[2] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *arXiv preprint arXiv:1910.13461*, 2019.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al., "Language models are few-shot learners," *Proc. of NeurIPS*, vol. 33, pp. 1877–1901, 2020.

[4] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al., "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[5] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al., "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task," *arXiv preprint arXiv:1809.08887*, 2018.

[6] Tao Yu, Rui Zhang, He Yang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, et al., "Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases," *arXiv preprint arXiv:1909.05378*, 2019.

[7] Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Heyang Er, Irene Li, Bo Pang, Tao Chen, et al., "Sparc: cross-domain semantic parsing in context," *arXiv preprint arXiv:1906.02285*, 2019.

[8] Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I Wang, et al., "Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models," *arXiv preprint arXiv:2201.05966*, 2022.

[9] Wenqiang Lei, Weixin Wang, Zhixin Ma, Tian Gan, Wei Lu, Min-Yen Kan, and Tat-Seng Chua, "Re-examining the role of schema linking in text-to-sql," in *Proc. of EMNLP*, 2020, pp. 6943–6954.

[10] 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," *arXiv preprint arXiv:1905.08205*, 2019.

[11] Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev, "Typesql: Knowledge-based type-aware neural text-to-sql generation," *arXiv preprint arXiv:1804.09769*, 2018.

[12] Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson, "Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers," *arXiv preprint arXiv:1911.04942*, 2019.

[13] Pengcheng Yin and Graham Neubig, "A syntactic neural model for general-purpose code generation," *arXiv preprint arXiv:1704.01696*, 2017.

[14] Alane Laughlin Suhr, Kenton Lee, Ming-Wei Chang, and Pete Shaw, "Exploring unexplored generalization challenges for cross-database semantic parsing," 2020.

[15] Xi Victoria Lin, Richard Socher, and Caiming Xiong, "Bridging textual and tabular data for cross-domain text-to-sql semantic parsing," *arXiv preprint arXiv:2012.12627*, 2020.

[16] Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau, "Picard: Parsing incrementally for constrained auto-regressive decoding from language models," *arXiv preprint arXiv:2109.05093*, 2021.

[17] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei, "Scaling laws for neural language models," *arXiv preprint arXiv:2001.08361*, 2020.

[18] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al., "Webgpt: Browser-assisted question-answering with human feedback," *arXiv preprint arXiv:2112.09332*, 2021.

[19] Richard Schwartz, Long Nguyen, and John Makhoul, "Multiple-pass search strategies," in *Automatic Speech and Speaker Recognition*, pp. 429–456. Springer, 1996.

[20] Long Nguyen and Richard M Schwartz, "Efficient 2-pass n-best decoder.," in *EuroSpeech*. Citeseer, 1997.

[21] Richard Zens, Oliver Bender, Saša Hasan, Shahram Khadivi, Evgeny Matusov, Jia Xu, Yuqi Zhang, and Hermann Ney, "The rwth phrase-based statistical machine translation system," in *Proc. of IWSLT*, 2005.

[22] Ngoc-Quang Luong, Laurent Besacier, and Benjamin Lecouteux, "Word confidence estimation for smt n-best list re-ranking," in *Proc. of the Workshop on HaCaT during EACL*, 2014.

[23] Michael Collins and Terry Koo, "Discriminative reranking for natural language parsing," *Computational Linguistics*, vol. 31, no. 1, pp. 25–70, 2005.

[24] Ruifang Ge and Raymond Mooney, "Discriminative reranking for semantic parsing," in *Proc. of COLING/ACL*, 2006, pp. 263–270.

[25] Binyuan Hui, Ruiying Geng, Qiyu Ren, Binhua Li, Yongbin Li, Jian Sun, Fei Huang, Luo Si, Pengfei Zhu, and Xiaodan Zhu, "Dynamic hybrid relation exploration network for cross-domain context-dependent semantic parsing," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, vol. 35, pp. 13116–13124.

[26] Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin, "Document ranking with a pretrained sequence-to-sequence model," *arXiv preprint arXiv:2003.06713*, 2020.

[27] Zhiguo Wang, Patrick Ng, Ramesh Nallapati, and Bing Xiang, "Retrieval, re-ranking and multi-task learning for knowledge-base question answering," in *Proc. of the European Chapter of ACL: Main Volume*, 2021, pp. 347–357.

[28] Manny Rayner, David Carter, Vassilios Digalakis, and Patti Price, "Combining knowledge sources to reorder n-best speech hypothesis lists," *arXiv preprint cmp-lg/9407010*, 1994.

[29] Rebecca Jonson, "Dialogue context-based re-ranking of asr hypotheses," in *SLT*. IEEE, 2006, pp. 174–177.

[30] Amol Kelkar, Rohan Relan, Vaishali Bhardwaj, Saurabh Vaichal, Chandra Khatri, and Peter Relan, "Bertrand-dr: Improving text-to-sql using a discriminative re-ranker," *arXiv preprint arXiv:2002.00557*, 2020.

[31] Antti-Veikko Rosti, Necip Fazil Ayan, Bing Xiang, Spyros Matsoukas, Richard Schwartz, and Bonnie Dorr, "Combining outputs from multiple machine translation systems," in *HLT 2007: The Conference of NAACL; Proc. of the Main Conference*, 2007, pp. 228–235.

[32] Nguyen Bach, Fei Huang, and Yaser Al-Onaizan, "Goodness: A method for measuring machine translation confidence," in *Proc. of the 49th Annual Meeting of ACL: HLT*, 2011, pp. 211–219.

[33] Robert Porzel and Iryna Gurevych, "Contextual coherence in natural language processing," in *International and Interdisciplinary Conference on Modeling and Using Context*. Springer, 2003, pp. 272–285.

[34] Vikas Raunak, Arul Menezes, and Marcin Junczys-Dowmunt, "The curious case of hallucinations in neural machine translation," *arXiv preprint arXiv:2104.06683*, 2021.

[35] Katherine Lee, Orhan Firat, Ashish Agarwal, Clara Fannjiang, and David Sussillo, "Hallucinations in neural machine translation," 2018.

[36] Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald, "On faithfulness and factuality in abstractive summarization," *arXiv preprint arXiv:2005.00661*, 2020.

[37] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi, "The curious case of neural text degeneration," *arXiv preprint arXiv:1904.09751*, 2019.

[38] Clara Meister and Ryan Cotterell, "Language model evaluation beyond perplexity," *arXiv preprint arXiv:2106.00085*, 2021.

[39] Nikolay Malkin, Zhen Wang, and Nebojsa Jojic, "Coherence boosting: When your pretrained language model is not paying enough attention," in *Proc. of the 60th Annual Meeting of ACL (Volume 1: Long Papers)*, 2022, pp. 8214–8236.

[40] Ning Li, Bethany Keller, Mark Butler, and Daniel Cer, "Seqgensql–a robust sequence generation model for structured query language," *arXiv preprint arXiv:2011.03836*, 2020.

[41] Pengcheng Yin and Graham Neubig, "Reranking for neural semantic parsing," in *Proc. of the 57th Annual Meeting of ACL*, 2019.

[42] Ben Bogin, Matt Gardner, and Jonathan Berant, "Representing schema structure with graph neural networks for text-to-sql parsing," *arXiv preprint arXiv:1905.06241*, 2019.

[43] Zhi Chen, Lu Chen, Yanbin Zhao, Ruisheng Cao, Zihan Xu, Su Zhu, and Kai Yu, "Shadowgnn: Graph projection neural network for text-to-sql parser," *arXiv preprint arXiv:2104.04689*, 2021.

[44] DongHyun Choi, Myeong Cheol Shin, EungGyun Kim, and Dong Ryeol Shin, "Ryansql: Recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases," *Computational Linguistics*, vol. 47, no. 2, pp. 309–332, 2021.

[45] Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova, "Compositional generalization and natural language variation: Can a semantic parsing approach handle both?," *arXiv preprint arXiv:2010.12725*, 2020.