Uncertainty estimation, error detection, and interactivity for robust text-to-SQL systems

Jingxing Fang, Emil Ghitman Gilkes, Mengyuan Li, Robert Roessler

Abstract

Text-to-SQL systems are a subdomain of semantic parsing that involves converting natural language (NL) into SQL queries to enable easy interaction with databases. There are many existing architectures that have produced impressive results for converting NL to SQL queries. However, there is still much to be learned about the errors that occur in test-to-SQL systems and exploring uncertainty in the models. In this project, we intend to evaluate model uncertainty and explore error detection methods for state-of-the-art text-to-SQL systems.

1 Introduction

Text-to-SQL systems are a subdomain of semantic parsing that involves converting natural language (NL) into SQL queries to enable easy interaction with databases. Text-to-SQL systems have garnered attention from both NLP and database communities as it lowers the technical ability required in order to conduct simple analyses of large databases. There are many existing architectures that have produced impressive results, however, there is still much to be learned about the errors and uncertainties in text-to-SQL systems. Since text-to-SQL systems are meant to provide useful information to users, it is important for the user to understand the uncertainty in test-time model predictions.

Another important direction in text-to-SQL research has found that the errors that the parsers make are often minor and can be easily corrected through user feedback. Therefore, there have been efforts to develop text-to-SQL systems that can incorporate simple feedback from users in order to correct the model's predictions. An important component of developing such a system is providing an explanation for why a given model made the prediction so that the user can more easily provide feedback on how to correct the prediction.

2 Related Work

There are three veins of research that are relevant to improving the usability of text-to-SQL systems:

Development of SOTA architectures for semantic parsing and, more specifically, text-to-SQL. Recent research surveys have evaluated the many text-to-SQL models that are out there. Most of the SOTA models are either graph-based models or T5-based models (Deng et al., 2022; He et al., 2019).

Uncertainty estimation and error detection to identify when and why a model fails. Since the end-users of text-to-SQL systems might use the results of the generated queries to make decisions, it's important that the user be able to understand when and where there is uncertainty in the model's predictions. The following references present recent research about uncertainty estimation in text-to-SQL systems: (Qin et al., 2022; Lee et al., 2022). There has also been recent research in error detection frameworks for text classification, such as sentiment analysis. Such frameworks can provide inspiration for error detection frameworks in text-to-SQL systems. The following reference provides an example: (Liu et al., 2021a)

Interactive NL systems that can incorporate user feedback. Evaluations of text-to-SQL models have found that the errors they make are often minor and could be corrected if the user provided simple feedback. Recent research out of Microsoft have developed text-to-SQL systems that provide explanations for why the system generated the query it did, and provided a way for the user to use the explanations to correct the system if needed. (Elgohary et al., 2020, 2021)

3 Datasets

There are numerous relevant datasets available ranging from smaller single domain (e.g. ATIS, SEDE) to extensive cross-domain datasets (e.g. Spider, WikiSQL, SparC, coSQL. Given its scale and complexity as well as its frequent usage as a benchmark dataset, we decided that the Spider (Yu et al., 2018) dataset would be an ideal candidate for our project.

Spider is a large-scale text-to-SQL dataset for developing natural language interface of complex and cross-domain databases. It consists of 200 databases covering 138 different domains, each with multiple tables. There are 10,181 questions, each asks a question on one database, and 5,693 unique complex SQL queries answering the questions.

4 Methodology

To perform uncertainty estimation and error detection of text-to-SQL systems, we start by picking a state-of-the-art T5-based text-to-SQL language model. Text-to-SQL is categorized as a text-to-text generation task, and is composed of an encoding and a decoding process. We use the maximum entropy during the decoding process of the language model as a baseline uncertainty estimation.

4.1 Text-to-SQL Model

The base model we picked for the uncertainty estimation task is Unified Text-To-Text Transfer Transformer (T5) with Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models (PICARD) (Scholak et al., 2021).

4.1.1 T5

The idea of T5 is to pretrain a large neural network on massive amounts of unlabeled text, then the network can be fine-tuned for specific downstream tasks. The T5 model has an encoder-decoder transformer structure (Figure 2), and is pre-trained on 18 different text-to-text tasks which fall into 8 categories: text summarization, question answering, translation, sentiment analysis, natural language inference, co-reference resolution, sentence completion, and word sense disambiguation. The T5 model has reached state-of-the-art performance on the SuperGLUE language benchmark scoring 89.3, which is only slightly lower than the 89.8 human baseline.

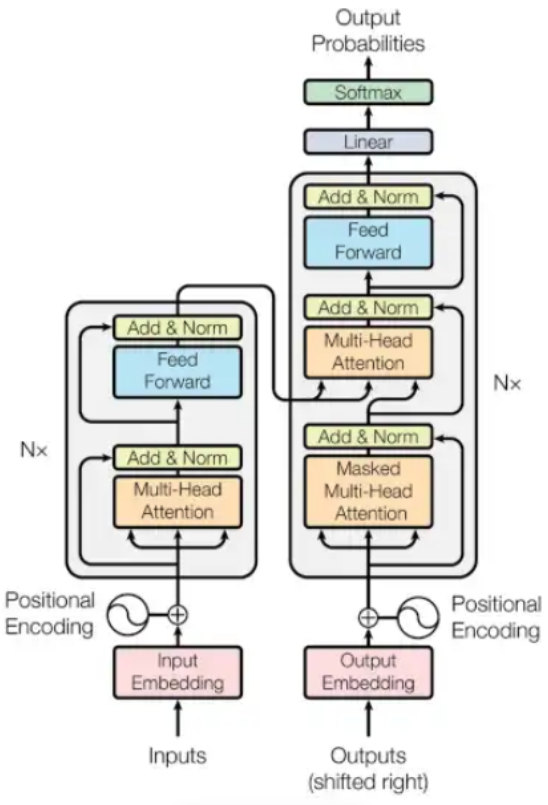

Figure 1: The architecture of the transformers model (Vaswani et al., 2017).

The model we used is based on the pre-trained `t5-base` model and is fine-tuned on the Spider(Yu et al., 2018) dataset. the model architecture is shown in Figure 2, with $N = 12$ encoder layers and the same number of decoder layers. The size of the encoder layers and the pool layers is 768. There are approximately 220 million parameters in total in the `t5-base` pre-trained model.

4.1.2 PICARD

PICARD is an improved text-generation parsing method based on beam search, which constrains auto-regressive decoders of language models through incremental parsing.

The last step of a text-generation model in the inference is to generate tokens from the logits calculated by the model in an auto-regressive manner – the output token at the next step is based on the output token at the current and previous steps. The most naive solution is the greedy search, which outputs the token index with the largest logit at each step and generates the next token based on the single token selected at the current step. A more advanced method would be the beam search. Beam search keeps the N most probable sequences at

each step, generates a new token based on the current N sequences, and keeps the N most probable from the newly generated sequences.

One unique problem of text-to-SQL tasks is that the generated sequence should have the correct SQL grammar. Therefore, PICARD, in addition to the beam search method, performs a grammatical check after generating the new token, and assigns zero-probability to the newly generated sequences that do not satisfy the SQL grammar rules. After the grammatical check, the new N most probable sequences are selected and kept for the next step's token generation. N is chosen to be 4 in this paper. PICARD was proven to significantly improve the performance of this T5-based Text-to-SQL model.

4.2 Uncertainty-Based Error detection

The principal goal of our project is to effectively detect model predictions that are highly uncertain, in hopes of contributing to the development of interactive and usable text-to-SQL systems. In this section, we propose a method to estimate the uncertainty of model predictions and perform error detection based on the uncertainty representations.

4.2.1 Uncertainty Estimation

The T5 model for text-to-text generation is autoregressive, i.e., prediction at the current step depends on the prediction at previous steps. We take a Bayesian viewpoint of the structured autoregressive predictions. Let $\boldsymbol{\theta}$ be the random variable of the model parameters and $\mathcal{D}$ be the training data. The model captures the mapping between a variable-length sequence of inputs $\{x_1, \cdots, x_T\} = \boldsymbol{x} \in \mathcal{X}$ and a variable-length output sequence $\{y_1, \cdots, y_L\} = \boldsymbol{y} \in \mathcal{Y}$, where $x_t \in \{w_1, \cdots, w_V\}$, and $y_l \in \{w_1, \cdots, w_K\}$, the posterior distribution of the predictions is

$$P(y_l \mid \boldsymbol{y}_{<l}, \boldsymbol{x}, \mathcal{D}) = E_{q(\boldsymbol{\theta})}\left[P(y_l \mid \boldsymbol{y}_{<l}, \boldsymbol{x}, \boldsymbol{\theta})\right] \tag{1}$$

At each step of decoding, we get $P(y_l \mid \boldsymbol{y}_{<l}, \boldsymbol{x}, \mathcal{D})$, which is the probability that each token should be generated at the current step. The total uncertainty in the prediction of y_l is given by the entropy of the predictive posterior

$$\begin{aligned} &\mathcal{H}\left[P(y_l = w_k \mid \boldsymbol{y}_{<l}, \boldsymbol{x}, \mathcal{D})\right] = \\ &-\sum_{k=1}^{K}(P(y_l = w_k \mid \boldsymbol{y}_{<l}, \boldsymbol{x}, \mathcal{D}) \\ &\cdot \log P(y_l = w_k \mid \boldsymbol{y}_{<l}, \boldsymbol{x}, \mathcal{D})) \end{aligned} \tag{2}$$

To measure the uncertainty of a sequence prediction, we tried three different representations: the maximum of the sequence entropy, the l2-norm of the sequence entropy, and the sequence representation itself. Since different predictions have variant lengths, for the sequence representation we pad all predictions to the max sequence length using a value larger than the max length of all predictions.

4.2.2 Error detection

After classifying the predictions into errors and confident predictions, we use several metrics to measure model performance on confident predictions. Based on different uncertainty representation types, we tried different methods to do error detection.

- For scalar representations like max entropy and l2-norm of sequence entropy, we could simply set an "uncertainty threshold". For example, the threshold could be some percentile of max entropy values from the development dataset. This method is inspired by recent research to detect errors in test time inferences (Lee et al., 2022). At test time, if a prediction yields a maximum or l2-norm entropy that is greater than this uncertainty threshold, the prediction is flagged as a likely error.

- Use an unsupervised machine learning method, such as k-means, to get the classification boundary. For scalar representations, we perform k-means directly on those values; for sequence representation, we perform k-means on the padded sequences.

- By using the sequence entropy and the ground truth results of the validation set as training samples and labels, we could build a machine learning classifier like logistic regression to do the error detection and use it on the test set.

4.3 Feature-Based Error Detection

In this section, we present an error detection method based on the context-rich embedding extracted by the transformer model which is motivated by Liu et al., 2021b.

The sequence-to-sequence model in this project works as follows: The input sentence (NL question) is tokenized and put into the encoder to generate contextual-rich embeddings that contain both the information of each token themselves and the information of each token's context. The embeddings

are then put into the decoder, whose architecture is very close to the encoder's, to generate hidden states, i.e., embeddings for the output tokens. The decoder hidden states are mapped to the probability that each token in the vocabulary being the next output token through a dense layer.

The error detection method previously proposed in this report uses the entropy of the predictions as a measurement of the uncertainty of the model inference. In this section, we consider the decoder hidden states as the features of each sample extracted by the transformer model, and we use the features to directly model the errors made by the model. A natural motivation is that some features can be inherently related to ambiguous results. For example, in a sentiment classification task, words such as "kashmir", "midterm", and "netflix" will make the sentences where they occur more likely to be wrongly predicted. In this project, because we are doing a text-to-SQL task, the features are not as explainable and explicit as in sentiment classifications, but the general idea that some features lead to more uncertainty still holds.

At each step in token generation, the decoder hidden states for each sample is 13 tensors of shape 4×768, each corresponding to the hidden state of a decoder layer. The size of the first dimension of each tensor corresponds to $N = 4$ beams used in the PICARD beam search. We propose two models to predict model errors from these hidden states:

- *Method 1* Given an input sequence of tokens, average each of the 13 decoder hidden states over all the tokens to get thirteen 4×768 tensors. Then, take the L2-norm of the second dimension, and concatenate thirteen tensors into 52 features for the error detection task.

- *Method 2* We think the decoder hidden states from the last layer to be the most relevant features for the error detection task. Given an input sequence of tokens, average the final decoder hidden states over all the tokens to get a 4×768 tensors, then average over the first dimension to get 768 features for the error detection task.

4.4 Evaluation

By comparing the metrics results on all predictions and on only confident predictions, we could see how error detection helps in improving the model's accuracy and stability. We will use exact matching accuracy and execution accuracy (Zhong et al., 2020) on those "confident" predictions and errors separately.

Exact matching accuracy is a kind of semantic accuracy that compares the similarity of the ground truth and the predictions. We use exact set matching accuracy instead of exact string match to reduce false negatives when predictions are semantically equivalent to the gold but differ in logical form.

Execution accuracy compares the execution results of the gold and the prediction. To reduce the false positives when a semantically different prediction happens to have the same execution result as gold on a particular database, we will execute the predictions and gold ones in a number of random databases.

To show more information about our error detection results, we calculate the precision, recall and f-1 score for both exact matching and execution matching.

5 Results and Discussion

5.1 Model results

The model will map natural language input to structured queries in the SQL format. Therefore, the input of this model would typically be a natural language statement or question that a user would like to convert into a SQL query. For example, a user might provide the following input: "**For each stadium, how many concerts are there?**".

The output of the model would be a SQL query that is equivalent to the user's input. In the example above, the model might produce the following output:

```
SELECT T2.name, COUNT(*)
FROM concert AS T1 JOIN stadium
AS T2
ON T1.stadium_id = T2.stadium_id
GROUP BY T1.stadium_id
```

In this example, the output SQL query will give us the exact result we want from the question.

5.2 Uncertainty-Based Error Detection Results

For the first uncertainty-based methods in Section 4.2, we designed an experiment to explore the best percentile threshold we should use as the boundary. Table 1 shows execution and exact precision, recall, f-1 scores and accuracy when we use different percentiles of the maximum entropy to detect the errors. Table 2 shows the results when we use

percentile	precision		recall		f-1		accuracy	
	execution	exact	execution	exact	execution	exact	execution	exact
20	0.6021	0.6245	0.9968	0.9969	0.7508	0.7679	0.6667	0.6667
30	0.6161	0.6392	0.9871	0.9876	0.7587	0.7761	**0.7949**	**0.7949**
40	0.6562	0.6797	0.9436	0.9425	**0.7741**	**0.7899**	0.7518	0.7376
50	0.6807	0.7061	0.8615	0.8618	0.7605	0.7762	0.6532	0.6411
60	0.7153	0.7416	0.7891	0.7888	0.7504	0.7645	0.6246	0.6103
70	0.7580	0.7776	0.6860	0.6786	0.7202	0.7247	0.5869	0.5614
80	0.7549	0.7883	0.4364	0.4394	0.5531	0.5643	0.4815	0.4652
100(all)	1	1	0	0	0	0	0.3994	0.3772

Table 1: The execution and exact precision, recall, f-1 score and accuracy results of the error detection method based on different percentile thresholds of max entropy for exact matching and execution matching.

percentile	precision		recall		f-1		accuracy	
	execution	exact	execution	exact	execution	exact	execution	exact
20	0.6021	0.6245	0.9968	0.9969	0.7508	0.7680	0.6667	0.6667
30	0.6158	0.6389	0.9887	0.9891	0.7590	0.7764	**0.8108**	**0.8108**
40	0.6551	0.6786	0.9453	0.9441	**0.7739**	**0.7896**	0.7536	0.7391
50	0.6789	0.7029	0.8647	0.8633	0.7606	0.7749	0.6543	0.6379
60	0.7168	0.7428	0.7987	0.7981	0.7555	0.7695	0.6345	0.6199
70	0.7587	0.7830	0.7037	0.7003	0.7302	0.7393	0.5983	0.5786
80	0.7890	0.8082	0.5298	0.5233	0.6339	0.6353	0.5267	0.5024
100(all)	1	1	0	0	0	0	0.3994	0.3772

Table 2: The execution and exact precision, recall, f-1 score, and accuracy results of the error detection method based on different percentile thresholds of l2-norm entropy for exact matching and execution matching.

	precision		recall		f-1		accuracy	
	execution	exact	execution	exact	execution	exact	execution	exact
maximum	0.7624	0.7822	0.6200	0.6134	0.6838	0.6876	0.5540	0.5290
l2-norm	0.7823	0.8008	0.6135	0.6056	0.6877	0.6897	0.5610	0.5360

Table 3: The execution and exact precision, recall, f-1 score and accuracy results of the error detection method based on maximum and l2-norm k-means boundary for exact matching and execution matching.

	Accuracy		Precision		Recall	
	Execution	Exact	Execution	Exact	Execution	Exact
Method 1	0.5080	0.4870	0.4389	0.4123	0.9283	0.9266
Method 2	0.5500	0.5310	0.4580	0.4282	0.8235	0.8179

Table 4: The accuracy, precision, and recall for predicting execution and exact match with Method 1 and Method 2.

l2-norm of the sequence entropy instead of max entropy. We could see from both of the tables that when we set the percentile to be 40%, the f-1 score would be the highest for both uncertainty representations (max entropy and l2-norm). And when doing entropy thresholding, the execution accuracy and exact match accuracy will both increase compared to the results without error detection. The execution accuracy and exact match accuracy could reach 81.08% when we set 30% percentile of l2-norm entropy as the boundary to classify confident predictions and errors.

We also tried to use the k-means boundary on max entropy and l2-norm entropy to do error detection. We run k-means on the validation set, and the boundary we got for max entropy is 0.5600, and the boundary for l2-norm representation is 0.6649. Table 3 shows the execution and exact precision, recall, and f-1 scores when we use k-means boundary based on maximum entropy and l2-norm entropy. By comparing Table 3 with Table 1 and Table 2, we could find that the boundary we got by k-means is similar to setting a percentile between 70% to 80%.

For the last uncertainty-based method in Section 4.2, we use a logistic regression model trained on validation sequence entropy and labels to predict on test sequence entropy. This model could achieve 73.2% error detection accuracy on the validation dataset but could only get 45.16% accuracy on the test dataset. Since this test accuracy is less than 50%, this method could not give us a stable and accurate error detection result. Therefore, for uncertainty-based error detection, it's better to use scalar uncertainty representations (maximum entropy and l2-norm of sequence entropy) instead of using sequence representation with padding.

5.3 Feature-Based Error Detection Results

We use the features calculated with Methods 1 and 2 in Section 4.3 to train a logistic regression model, which predicts whether the SQL queries predicted for each sample is an correct execution (or exact match) (labeled as 1), or not (labeled as 0), and the result is shown in Table 4. We can consider predicting label 1 as equivalent to being confident about the prediction, so the accuracy here is equivalent to the execution accuracy or exact match accuracy of confident predictions. Compared with the result in the previous section, it can be seen that filtering out uncertain predictions with this feature-based error detection methods can significantly improve the accuracy of predictions, but feature-based error detection does not significantly outperform entropy-based error detection. This implies that the decoder hidden states do not provide more information or extra correlations with the errors that the model is prone to making than the final output logits.

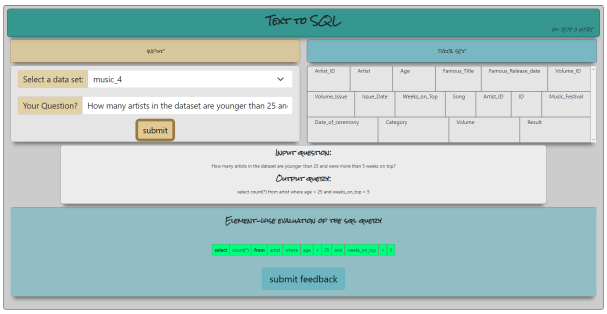

Figure 2: web-app with four components. 1) input fields (top left), 2) dataset explorer (top right), 3) prediction (center), 4) element-wise user feedback

6 Web App

In addition to training and optimizing our model, we've also created a web app that allows users to generate and explore queries in a user-friendly, fast, playful, and visually pleasing way. Furthermore, the web app also allows us to store user feedback, i.e. users can flag erroneous elements of the query. This feedback is sent back to the server and stored in a database. We intend to use this information in the future, beyond the scope of the course, to train a semi-supervised model to increase performance. As for the architecture of the web app, we're using a python server (flask) - which allows us to efficiently run our model and store user feedback. You can find the code for the web app on GitHub (https://github.com/rob3rtroessler/textToSQL). Currently, you will need to run it locally on your machine, as the model is too big to use any of the conventional cloud application platforms for free.

7 Conclusion

Our project has demonstrated that uncertainty-based error detection can significantly improve the accuracy of text-to-SQL methods. Although our results for feature-based error detection were not as good, there is still promise for this approach. Recent research in text-to-SQL proposes a data uncertainty constraint to reduce the sensitivity of the learned representations and improve the robustness of the parser (Bowen Qin, 2022). From the model

uncertainty perspective, there is often structural information (dependence) among the weights of neural networks. To improve the generalizability and stability of neural text-to-SQL parsers, a model uncertainty constraint could refine the query representations. It is possible that the features learned using this training approach might provide better results for feature-based error detection.

Ultimately, error detection is an extremely important aspect of building practical text-to-SQL systems, and our project provides a strong foundation for future research into developing more precise detection systems.

8 Ethics Statement

The Spider dataset is a publicly available project that was annotated by 11 Yale students. The terms and conditions have been followed. The intended use of Text-to-SQL systems would be to lower the barrier of entry for roles that involve simple analytics as people with limited technical expertise could extract information from relational databases. Text-to-SQL systems could also be leveraged in educational tools to help professionals upskill.

As text-to-SQL systems become more advanced, there is a possibility that they will reduce growth in entry level analytics jobs. However, we do not believe that text-to-SQL systems will result in widespread job loss. Instead, text-to-SQL systems are intended to be a tool to democratize access to simple analytics and increase the speed and efficiency of professionals who regularly work with SQL.

A potential risk of text-to-SQL systems is when the system makes an incorrect prediction. This is the motivation of the research presented in our report. If an error is undetected, it's possible that the user draws incorrect conculsions from the output, which could lead to wrongly informed decision making. One cannot over rely on these systems and due diligence should always be performed. However, we hope that errror detection methods, such as those presented in our research, can help mitigate the risk of prediction errors.

References

Binyuan Hui Bowen Li Pengxiang Wei Binhua Li Fei Huang Luo Si Min Yang Yongbin Li Bowen Qin, Lihan Wang. 2022. Sun: Exploring intrinsic uncertainties in text-to-sql parsers. In *COLING*.

Naihao Deng, Yulong Chen, and Yue Zhang. 2022. Recent advances in text-to-sql: A survey of what we have and what we expect.

Ahmed Elgohary, Saghar Hosseini, and Ahmed H. Awadallah. 2020. Speak to your parser: Interactive text-to-sql with natural language feedback. In *Annual Conference of the Association for Computational Linguistics (ACL 2020)*.

Ahmed Elgohary, Christopher Meek, Matthew Richardson, Adam Fourney, Gonzalo Ramos, and Ahmed Hassan Awadallah. 2021. Nl-edit: Correcting semantic parse errors through natural language interaction.

Yingying He, Di Bai, and Wantong Jiang. 2019. Text-to-sql translation with various neural networks.

Gyubok Lee, Hyeonji Hwang, Seongsu Bae, Yeonsu Kwon, Woncheol Shin, Seongjun Yang, Minjoon Seo, Jong-Yeup Kim, and Edward Choi. 2022. EHRSQL: A practical text-to-SQL benchmark for electronic health records. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Zhe Liu, Yufan Guo, and Jalal Mahmud. 2021a. When and why does a model fail? a human-in-the-loop error detection framework for sentiment analysis.

Zhe Liu, Yufan Guo, and Jalal Mahmud. 2021b. When and why does a model fail? a human-in-the-loop error detection framework for sentiment analysis. *arXiv preprint arXiv:2106.00954*.

Bowen Qin, Lihan Wang, Binyuan Hui, Bowen Li, Xiangpeng Wei, Binhua Li, Fei Huang, Luo Si, Min Yang, and Yongbin Li. 2022. Sun: Exploring intrinsic uncertainties in text-to-sql parsers.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

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

Ruiqi Zhong, Tao Yu, and Dan Klein. 2020. Semantic evaluation for text-to-sql with distilled test suites. *arXiv preprint arXiv:2010.02840*.