Error Detection for Interactive Text-to-SQL Semantic Parsing

Shijie Chen
The Ohio State University
chen.10216@osu.edu

Ziru Chen
The Ohio State University
chen.8336@osu.edu

Huan Sun
The Ohio State University
sun.397@osu.edu

Yu Su
The Ohio State University
su.809@osu.edu

Abstract

Despite remarkable progress in text-to-SQL semantic parsing, the performance of state-of-the-art parsers are still far from perfect. At the same time, modern deep learning based text-to-SQL parsers are often over-confident and thus casting doubt on its trustworthiness when used in an interactive setting. In this paper, we propose to train parser-independent error detectors for text-to-SQL semantic parsers. We test our proposed approach with two parsers, SmBop and Bridge v2, and show our model could outperform parser-dependent uncertainty measures in simulated interactive evaluations. As a result, when used for answer triggering or interaction trigger in interactive semantic parsing systems, our model could effectively improve the usability of the base parser.

1 Introduction

Recent years have witnessed a renewed interest in text-to-SQL semantic parsing [Rubin and Berant, 2021, Lin et al., 2020, Cao et al., 2021, Wang et al., 2020, Gan et al., 2021, Scholak et al., 2021]. Although state-of-the-art semantic parsers have achieved remarkable performance on Spider [Yu et al., 2018], a large-scale cross-domain text-to-SQL benchmark, their performance is still far from satisfactory for real use. To harness the benefit of text-to-SQL parsers in real application scenarios, several interactive semantic parsing frameworks have been proposed. In this work, we explore building parser-independent error detection models to enable easy adaptation of semantic parsers in interactive frameworks. We conduct a case study with SmBop [Rubin and Berant, 2021] and Bridge v2 [Lin et al., 2020], two state-of-the-art text-to-SQL parsers, and examine two key applications of error detectors in interactive semantic parsing: answer triggering and interaction triggering.

When error detectors are used for answer triggering, the base parser refuses to give an answer when an error is detected. The precision of answers is essential to the trustworthiness of interactive systems in real use. For example, NL-EDIT[Elgohary et al., 2020, 2021] corrects parsing errors through user feedback. It would be very frustrating for users, who assume their feedback would be incorporated correctly, to receive an inaccurate answer. In such cases, an accurate error detector could allow the system to acknowledge its failure, leading to improved trustworthiness.

When using error detection models as interaction triggers, the system initiates interactions for error correction only when an error is detected in the initial parse so that unnecessary interactions for correct parses are avoided. MISP [Yao et al., 2019, 2020] initiates interactions by setting a confidence threshold for prediction probability. While this approach is intuitive, it requires the base parser to be well-calibrated when decoding, which does not hold for most modern parsers using deep neural

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

networks. In addition, this design can hardly accommodate some recent parsers, such as SmBop [Rubin and Berant, 2021], whose bottom-up decoding mechanism does not model the distribution over the output space. Several other interactive frameworks [Gur et al., 2018, Li et al., 2020, Zeng et al., 2020] trigger interactions when an incorrect or uncertain span is detected in the input question or predicted SQL query. While these mechanisms have high coverage for parsing errors, they tend to trigger unnecessary interactions for correct base parses. For example, PIIA [Li et al., 2020] triggers interactions on 98% of the questions on Spider dev when its base parser has an accuracy of 49%.

To sum up, a parser-independent interaction trigger that could reliably detect errors in base parser predictions and avoid unnecessary interactions is needed for building effective and efficient interactive text-to-SQL systems. We approach this problem by training parser-independent error detection models on realistic parser errors. To simulate mistakes the parsers might make in real cross-domain environments, we collect errors from weak parsers on out-of-domain data. We show that RoBERTa [Liu et al., 2019] fine-tuned on the collected data could already perform competitively compared to parser-dependent uncertainty metrics. Switching to CodeBERT [Feng et al., 2020], a language model pre-trained on various programming languages, brings significant gain in performance. On top of that, we explore modeling syntactic and semantic features of natural language questions and SQL queries with graph neural networks, which further improves performance. Our proposed models could outperform existing parser-dependent error detection methods without assuming intrusive access to the base parser.

2 Parser-independent Error Detection

Problem Formulation Given a question $X = \{x_1, x_2, \cdots, x_m\}$ and a SQL query $\hat{y} = \{\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_n\}$ predicted by a text-to-SQL parser, the error detector model estimates the probability of $\hat{y}$ being correct $p = (\hat{y} = y^* | X, \hat{y})$.

Data collection Training data is collected from weak base parsers. We first train the base parser using 10%, 30%, and 50% of the Spider training set split by database and collect their beam predictions on respective complement portions of the Spider training set. Then we keep the top-ranked erroneous and correct (if any) SQL predictions according to execution accuracy. The collected samples are further divided into training and development sets by an 80%:20% ratio. In this way, we get training data for the error detection model in a setting that approximates the zero-shot testing environment. From SmBop, we collect 9309 negative samples and 4017 positive samples in the training set as well as 2389 negative samples and 1150 positive samples in the development set. From Bridge v2, we collect 6018 negative samples and 4017 positive samples in the training set as well as 1028 positive samples and 1571 negative samples in the development set.

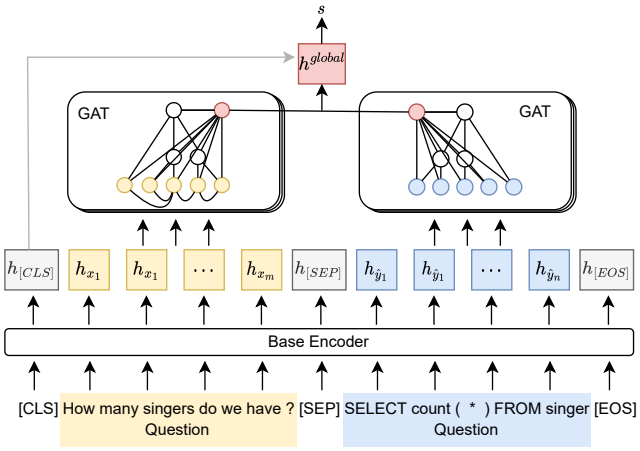

Figure 1: Architecture of our error detection models.

Model Architecture Figure 1 illustrates the architecture of our proposed models. We use pre-trained transformer encoders as our base encoder to obtain contextualized represen-

tations for input question and SQL query. Following CodeBERT's input construction during pre-training, we concatenate questions and SQL queries with special tokens, namely $[CLS], x_1, x_2, \cdots, x_m, [SEP], \hat{y}_1, \hat{y}_2, \cdots, \hat{y}_n, [EOS]$ as input and obtain their representations h_X and $h_{\hat{y}}$. We only use question and SQL as input since we found through preliminary experiments that directly adding database schema information (table and column names) in the input hurts performance.

In light of the compositional nature of questions and SQL queries, we propose to model their syntactic features via graph neural networks. For questions, we obtain their dependency parse trees and constituency parse trees from Stanza [Qi et al., 2020] and merge them together. For SQL queries, we extract their abstract syntax trees via Antlr4[1]. A global node that is connected to all other nodes is added to each input graph to capture global features. We initialize the representations of nodes corresponding to input tokens with CodeBERT's output and randomly initialize representations of other nodes according to their types in the parse tree. These two graphs are encoded by two separate 3-layer graph attention networks [Brody et al., 2022]. Then we concatenate the representation of the global nodes and obtain an aggregated representation:

$$h^{global} = [h_X^{global}; h_{\hat{y}}^{global}]$$

When simply fine-tuning RoBERTa or CodeBERT, the representation for the $[CLS]$ token is used as the aggregated representation, i.e. $h^{global} = h_{[CLS]}$.

Finally, a 2-layer feed forward neural network with $\tanh$ activation is used to score the aggregated representation v. The score s for each input question-SQL pair is:

$$s = p(\hat{y} = y^* | X, \hat{y}) = \sigma(\text{FFN}(h^{global}))$$

where y^* is the gold SQL query and σ represents the sigmoid function. We train our model by minimizing a binary cross entropy loss:

$$\mathcal{L} = \mathbb{1}_{\hat{y}=y^*} \cdot \log s + (1 - \mathbb{1}_{\hat{y}=y^*}) \cdot \log(1 - s)$$

3 Experiments

3.1 Experiment Setup

Dataset To evaluate error detection methods in a realistic setting, we use grammatically correct predictions made by SmBop and Bridge v2 on Spider Dev as test datasets. SmBop has an accuracy of 75.0%, and we collect 1021 samples in total, including 779 correct SQL parses and 242 incorrect ones. Bridge v2 has an accuracy of 65.5%, and we collect 1028 samples in total, including 705 correct SQL parses and 323 incorrect ones.

Baseline Methods We compare our parser-independent error detectors with parser-dependent uncertainty metrics, including prediction probability and dropout based uncertainty. Since SmBop [Rubin and Berant, 2021] uses bottom-up decoding which separately scores and ranks each candidate prediction, we deduplicate SmBop's beam predictions by keep the maximum score and perform softmax on the deduplicated beam to get a probability distribution over candidate predictions, which can be seen as a reasonable approximation to its confidence. Bridge v2 [Lin et al., 2020] uses autoregressive decoding, and we directly use the log probability of its prediction as its confidence score. Probability-based methods are denoted by superscript p. In terms of dropout-based uncertainty, we follow MISP [Yao et al., 2019] and measure the standard deviation of the scores (SmBop) or log probability (Bridge v2) of the top-ranked prediction in 10 passes. Dropout-based uncertainty is denoted by superscript s.

Evaluation Metrics We report precision, recall, and F1 scores for each method. As we are targeting at error detection, we also report these metrics on negative samples. However, these metrics depend on the threshold used. To evaluate the overall discriminative ability of each method, we also present the area under receiver operating characteristic curve (AUC), which is not affected by the choice of threshold. We apply 5-fold cross validation and report performance using the threshold that maximizes the accuracy of each method. Test samples are partitioned by databases. All results of our models are averages over 5 runs.

[1]Antlr: https://www.antlr.org/, we use a publicly available context-free grammar for SQLite https://github.com/antlr/grammars-v4/tree/master/sql/sqlite

Implementation Our models are trained with a batch size of 32 and are optimized by the AdamW [Loshchilov and Hutter, 2019] optimizer with default parameters. Training lasts 10 epochs with a learning rate of 3e-5 following a linear decay schedule with 10% warm up steps. All models are trained on an NVIDIA RTX A6000 GPU.

3.2 Results

3.2.1 Error Detection

As shown in Table 1, the dropout based uncertainty measure SmBop^s significantly outperforms the approximate confidence measure SmBop^p. However, Table 2 shows the opposite for Bridge v2, which is consistent with the observation of [Yao et al., 2019] that is also based on an autoregressive parser as well. On SmBop, the RoBERTa baseline model already performs competitively compared to parser-dependent metrics. CodeBERT shows a noticeable improvement over RoBERTa despite not pre-trained on SQL. With the added structural features, CodeBERT+GAT further improves in precision and recall on error cases. On Bridge v2, all our models could outperform parser-dependent methods on all metrics. We notice that the improvement brought by using CodeBERT and adding structural features is more evident on SmBop, possibly due to that SmBop is stronger than Bridge v2 and thus its errors are harder to detect.

Table 1: Performance of error detection models on Spider Dev - SmBop.

Model	Precision	Recall	F1	- Precision	- Recall	- F1	Acc	AUC
SmBop^p	80.9	92.4	85.6	42.9	15.5	18.9	76.1	75.5
SmBop^s	**84.3**	90.5	86.4	35.5	**40.8**	**37.1**	77.9	78.6
RoBERTa	82.0	**95.4**	87.7	53.9	19.8	26.1	79.5	74.6
CodeBERT	83.3	**95.4**	88.3	61.6	27.4	33.5	80.8	78.9
CodeBERT+GAT	83.9	95.0	**88.6**	**62.6**	29.7	36.7	**81.5**	**80.7**

Table 2: Performance of error detection models on Spider Dev - Bridge v2.

Model	Precision	Recall	F1	- Precision	- Recall	- F1	Acc	AUC
Bridge v2^p	78.5	86.5	81.8	55.1	41.6	45.3	73.2	77.6
Bridge v2^s	76.2	88.7	81.4	53.5	30.5	36.9	71.7	76.7
RoBERTa	80.2	**90.5**	**84.7**	**65.3**	43.0	50.2	77.6	80.6
CodeBERT	83.9	81.5	82.4	55.0	**57.9**	55.7	76.3	80.2
CodeBERT+GAT	**83.5**	85.3	84.1	60.2	54.9	**56.2**	**77.9**	**82.9**

3.2.2 Cross-parser Generalization

We evaluate the cross-parser generalization ability of our error detectors by training the model on data collected from one parser and test it on the other following the same 5-fold cross-validation setting. Table 3 summarizes cross-parser transfer performance. Overall our models generalize well on unseen parsers. Adding structural features noticeably improves generalization performance.

Notably, CodeBERT^p and CodeBERT+GAT^p trained with data collected from SmBop could outperform Bridge v2^p and Bridge v2^s in a zero-shot setting. This does not hold on the opposite direction. We hypothesize that errors made by stronger parsers might be more diverse and thus are of higher quality for training error detection models.

3.2.3 Application in Interactive Semantic Parsing Systems

We simulate the performance of our error detectors in interactive semantic parsing systems when used as answer trigger and interaction trigger at different thresholds. The base parser is SmBop.

Answer triggering Figure 2(a) demonstrates the change of precision when varying the decision threshold. A high p (or low s) reduces the number of question answered for a higher precision.

Table 3: Cross-parser Generalization Performance.

Model	Precision	Recall	F1	- Precision	- Recall	- F1	Acc	AUC
Bridge v2 → SmBop								
RoBERTa	**82.2**	94.6	**87.3**	54.9	**23.6**	**29.1**	**79.1**	76.9
CodeBERT	81.2	**95.1**	86.8	**55.0**	15.1	16.9	78.1	76.5
CodeBERT+GAT	82.0	93.9	86.9	46.5	19.8	22.8	78.5	**77.9**
SmBop → Bridge v2								
RoBERTa	77.0	**92.4**	83.7	62.4	30.0	38.9	75.1	75.4
CodeBERT	79.7	89.7	84.2	65.3	43.3	51.1	76.9	79.6
CodeBERT+GAT	**81.1**	89.2	**84.8**	**65.8**	**48.0**	**54.7**	**78.1**	**81.5**

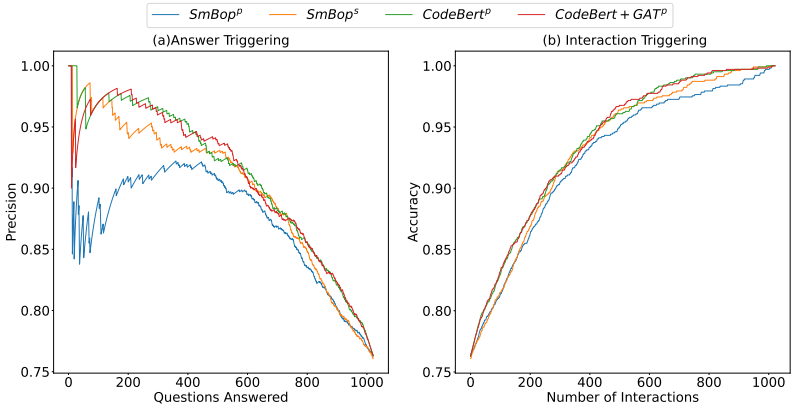

Figure 2: Performance in simulated interactive semantic parsing using SmBop.

Conversely, a lower p (or higher s) encourages the system to answer more questions at the cost of making more mistakes. In this scenario, $SmBop^s$ does significantly better than $SmBop^p$. Our error detectors could outperform both baseline methods and enable the system to answer more questions at higher precision.

Interaction triggering We simulate the potential gain of more accurate interaction triggers by assuming oracle error correction interactions, where any detected error would be fixed. Ideally, we would want to get higher accuracy with fewer interactions. As shown in Figure 2(b), $SmBop^s$ again outperforms $SmBop^p$ by a large margin. Our parser-independent models consistently improves upon them, leading to a more efficient interactive semantic parsing system.

4 Conclusion

In this work, we explore building parser-independent error detection models for adapting text-to-SQL parsers to interactive frameworks. Through a case study with state-of-the-art parsers, we demonstrate the effectiveness of our approach when used in interactive semantic parsing systems. Compared to parser-dependent uncertainty metrics, our parser-independent method shows superior performance while doesn't assume any intrusive access to the base parser. Fur future work, we would try to improve performance through different modeling techniques and work on more fine-grained error detection.

Acknowledgements

The authors would like to thank the anonymous reviewers and colleagues from the OSU NLP group for their constructive comments. This research was supported in part by NSF OAC 2112606, NSF IIS 1815674, NSF CAREER 1942980, and Cisco.

References

Shaked Brody, Uri Alon, and Eran Yahav. How Attentive are Graph Attention Networks? In *International Conference on Learning Representations*, February 2022.

Ruisheng Cao, Lu Chen, Zhi Chen, Yanbin Zhao, Su Zhu, and Kai Yu. LGESQL: Line Graph Enhanced Text-to-SQL Model with Mixed Local and Non-Local Relations. 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 2541–2555, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.198.

Ahmed Elgohary, Saghar Hosseini, and Ahmed Hassan Awadallah. Speak to your Parser: Interactive Text-to-SQL with Natural Language Feedback. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2065–2077, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.187.

Ahmed Elgohary, Christopher Meek, Matthew Richardson, Adam Fourney, Gonzalo Ramos, and Ahmed Hassan Awadallah. NL-EDIT: Correcting semantic parse errors through natural language interaction (NAACL21). *arXiv:2103.14540 [cs]*, March 2021.

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. CodeBERT: A Pre-Trained Model for Programming and Natural Languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1536–1547, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.139.

Yujian Gan, Xinyun Chen, Jinxia Xie, Matthew Purver, John R. Woodward, John Drake, and Qiaofu Zhang. Natural SQL: Making SQL Easier to Infer from Natural Language Specifications. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2030–2042, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.174.

Izzeddin Gur, Semih Yavuz, Yu Su, and Xifeng Yan. DialSQL: Dialogue Based Structured Query Generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1339–1349, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1124.

Yuntao Li, Bei Chen, Qian Liu, Yan Gao, Jian-Guang Lou, Yan Zhang, and Dongmei Zhang. "What Do You Mean by That?" A Parser-Independent Interactive Approach for Enhancing Text-to-SQL. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6913–6922, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.561.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 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, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.438.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv:1907.11692 [cs]*, July 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-demos.14.

Ohad Rubin and Jonathan Berant. SmBoP: Semi-autoregressive Bottom-up Semantic Parsing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 311–324, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.29.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding from Language Models. *arXiv:2109.05093 [cs]*, September 2021.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.677.

Ziyu Yao, Yu Su, Huan Sun, and Wen-tau Yih. Model-based Interactive Semantic Parsing: A Unified Framework and A Text-to-SQL Case Study. 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 5447–5458, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1547.

Ziyu Yao, Yiqi Tang, Wen-tau Yih, Huan Sun, and Yu Su. An Imitation Game for Learning Semantic Parsers from User Interaction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6883–6902, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.559.

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. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1425.

Jichuan Zeng, Xi Victoria Lin, Steven C.H. Hoi, Richard Socher, Caiming Xiong, Michael Lyu, and Irwin King. Photon: A Robust Cross-Domain Text-to-SQL System. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 204–214, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-demos.24.