Improving Generalization in Language Model-Based Text-to-SQL Semantic Parsing: Two Simple Semantic Boundary-Based Techniques

Daking Rai[1], Bailin Wang[2], Yilun Zhou[2], Ziyu Yao[1]
[1]George Mason University, [2]MIT
[1]{drai2, ziyuyao}@gmu.edu, [2]{bailinw, yilun}@mit.edu

Abstract

Compositional and domain generalization present significant challenges in semantic parsing, even for state-of-the-art semantic parsers based on pre-trained language models (LMs). In this study, we empirically investigate improving an LM's generalization in semantic parsing with two simple techniques: at the *token* level, we introduce a token preprocessing method to preserve the semantic boundaries of tokens produced by LM tokenizers; at the *sequence* level, we propose to use special tokens to mark the boundaries of components aligned between input and output. Our experimental results on two text-to-SQL semantic parsing datasets show that our token preprocessing, although simple, can substantially improve the LM performance on both types of generalization, and our component boundary marking method is particularly helpful for compositional generalization.[1]

1 Introduction

Pre-trained language models (LMs)[2] such as T5 (Raffel et al., 2020) have now been more and more widely adopted for semantic parsing due to their promising performance and straightforward architectures (Shaw et al., 2021; Scholak et al., 2021; Yin et al., 2021; Qi et al., 2022; Xie et al., 2022; Qiu et al., 2021). However, recent work revealed that these LMs still struggle to generalize on out-of-distribution (OOD) samples (Lake and Baroni, 2018; Keysers et al., 2019; Shaw et al., 2021; Qiu et al., 2022b). For example, if a parser has learned "how many heads are in the department" and "how many people are older than 56", it is expected to generalize to "how many heads of the departments are older than 56". Generalizing to such novel component compositions is known as *compositional generalization*. Additionally, generalizing to new domains (e.g., from "entertainment" to "flight") is referred to as *domain generalization*.

Token Preprocessing (applied to database schema)	
Before:	department_management | department :| id , budget_in_billions , num_employees
After:	department_management | department :| id , budget _ in _ billions , num _ employees
Token Preprocessing (applied to SQL)	
Before:	select avg (flight.price) where flight.origin = 'New York'
After:	select average (flight . price) where flight . origin = 'New York'
Component Boundary Marking (applied to NL input and SQL output)	
Before:	How many heads of the departments are older than 56 ? select count (head.*) where head.age > 56
After:	[sep0] How many heads of the departments [/sep0] [sep1] are older than 56 ? [/sep1] [sep0] select count (head.*) [/sep0] [sep1] where head.age > 56 [/sep1]

Table 1: Our proposed techniques. Top: we preprocess the text such that its T5 tokenization aligns with word semantics. Coloring indicates tokenization; for example, "avg" is converted into three tokens of "a", "v" and "g". Bottom: we add separator tokens to mark the boundaries of aligned semantic components in the input and output.

In this paper, we investigate these two types of generalization of LMs in text-to-SQL semantic parsing, i.e., given a natural language (NL) input and the database schema, producing a SQL query that can be executed against the database for desired output. We conduct experiments using the cross-database Spider benchmark (Yu et al., 2018b) and its derivation Spider-CG (Gan et al., 2022). Compared with existing benchmarks (Keysers et al., 2019; Lake and Baroni, 2018), this task setting is both more realistic (e.g., containing larger language variations) and more challenging (e.g., requiring grounding to the database context).

[1]The source code for our implementation is available at https://github.com/Dakingrai/ood-generalization-semantic-boundary-techniques.

[2]We use "LMs" to refer to a broad set of models that are pre-trained in (masked/autoregressive) language modeling objectives, with encoder-decoder or decoder-only architecture.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics Volume 2: Short Papers, pages 150–160
July 9-14, 2023

Although previous work tackling the two types of generalization all requires non-trivial engineering effort (see Section 2), in this work, we present two simple yet effective techniques, which are extremely easy to implement with LMs (Table 1). Our techniques improve the generalization of LMs by preserving the *semantic boundaries* at the token and the sequence levels. At the token level, our first technique rewrites the inputs to handle naming conventions in database schemas and SQL queries such that a pre-trained LM tokenizer can split them into semantically meaningful tokens. At the sequence level, our second technique introduces special tokens to mark the semantic boundaries (e.g., phrases) aligned between the source NL and the target SQL. These special tokens implicitly help the LM-based parser build more precise input-output correspondences that are crucial for compositional generalization.

On five evaluation sets, the experimental results based on T5-base show that, albeit simple, our token-level technique dramatically improves both types of LM generalization, and our sequence-level technique is particularly helpful for compositional generalization. Combining them together leads to further improvements. Our additional experiments further demonstrate the generalizability of our approaches (e.g., to text-to-LISP expression parsing (Semantic Machines et al., 2020)).

2 Related Work

Text-to-SQL Semantic Parsing. This task has received considerate attention since the creation of the WikiSQL (Zhong et al., 2017) and Spider (Yu et al., 2018b) datasets. While a large amount of existing work designed specialized architectures for this task (Yu et al., 2018a; Zhang et al., 2019; Wang et al., 2020; Lin et al., 2020), there has been a trend of directly fine-tuning pre-trained sequence-to-sequence models as semantic parsers (Shaw et al., 2021; Scholak et al., 2021; Xie et al., 2022; Qi et al., 2022). Our work follows the same line and proposed approaches to further improve the LM performance. On the other hand, Guo et al. (2019); Gan et al. (2021); Herzig et al. (2021) showed that simplifying the SQL representation in a way that the new representation can semantically better align with the NL can dramatically improve the parsing performance. In our work, we follow the NatSQL representation (Gan et al., 2021) as it has better alignments with the NL.

Injecting Priors into Semantic Parsers. Our two techniques can be viewed as injecting human prior knowledge into neural models for better generalization, which has been one of the major research efforts on improving domain and compositional generalization. The key consideration to be taken when injecting priors is the trade-off between the form and the generalizability. Strong priors in the form of specialized model architectures (Shaw et al., 2021; Herzig and Berant, 2021; Wang et al., 2021) are either too expensive or not applicable across domains. Weaker priors in terms of specialized training algorithms (Yin et al., 2021; Conklin et al., 2021) are more general, but often weaker in performance compared to other lines of methods. Our work is in the spirit of the third line on the use of data augmentation (Andreas, 2020; Akyürek et al., 2020; Qiu et al., 2022a). However, instead of synthesizing new data from scratch, we "annotate" the data with semantic boundary markers, which is not only much simpler but also brings better performance. The final line of work (Qiu et al., 2022b; Levy et al., 2022) is based on the learning capacities in the context of large LMs, which is out of the scope of this work.

3 Methods

3.1 Token Preprocessing

Before preprocessing	After preprocessing
Snake case in schema items (add space)	
booking_status_code	booking _ status _ code
document_type	document _ type
Dot notation in column references (add space)	
farm.cows	farm . cows
origin.flight	origin . flight
SQL keyword (expand spelling)	
avg	average
desc	descending

Table 2: Three token preprocessing types. Coloring indicates tokenization, same as Table 1.

We present our two techniques for improving the generalization of LM-based semantic parsers. LM pre-training learns high-quality contextualized word representation (Devlin et al., 2019), but to effectively use it on a downstream task, the tokenization needs to "make sense." For example, if the text "pet_age" is tokenized as "pet", "_" and "age", then the semantics of "pet" and "age" acquired during pretraining can be directly used. However, if it is

Dataset	Size	Usage	Generalization Type
Spider$_T$	7,000	Train	None (in-distribution)
Spider$_D$	1,034	Eval	Domain
CG-SUB$_T$	20,686	Eval	None (in-distribution)
CG-SUB$_D$	2,883	Eval	Domain
CG-APP$_T$	18,793	Eval	Composition
CG-APP$_D$	3,237	Eval	Domain & Composition

Table 3: Datasets in our experiments.

tokenized as "pe", "t_a" and "ge", then pre-training is hardly useful because the model does not even recognize the two semantic words.

Unfortunately, this latter case is very common when tokenizing non-natural language texts, such as database schemas and SQL queries. Thus, we propose a token preprocessing method to induce more natural tokenization by, at a high level, adding white spaces and handling the naming conventions in database schema and SQL queries. We show examples in Table 2 and details in Appendix A.

3.2 Component Boundary Marking

At the sequence level, our second technique further assists LMs in recognizing the semantic boundaries of components aligned between input and output. An example is shown in Table 1. While prior work has attempted the goal via implementing alignment-based attention supervision (Yin et al., 2021), we propose to insert *special tokens* in input and output to inject such bias. Specifically, we use pairs of "`[sep`N`]`" and "`[/sep`N`]`", $N \in \mathbb{Z}$, to mark the boundaries, so as to hint the LM that components within the paired special tokens should be aligned. In practice, we also observed cases where an NL component has to be aligned with a SQL component consisting of multiple non-continuous segments. To handle it, we will apply the same pair of special tokens to each segment of the same component. An example is shown in Table 8 in the Appendix.

Finally, we note that our method assumes the availability of component annotations. Such annotations can be obtained via human labeling (Gan et al., 2021), heuristic rules (Yin et al., 2021), or other advanced machine learning algorithms, but this is beyond the scope of our work.

4 Experiments

4.1 Setup

Datasets. We use two datasets, Spider (Yu et al., 2018b) and Spider-CG (Gan et al., 2022). Spider consists of a training set (Spider$_T$) and a development set (Spider$_D$) with non-overlapping domains but otherwise similar data characteristics (e.g., length). Thus, we train the models on Spider$_T$, and consider Spider$_D$ as the evaluation for domain generalization. Spider-CG is derived from Spider by first dissecting each Spider instance into different components according to its dependency parse and generates data in two ways: substituting a component in one instance with one from another instance and appending one component from one instance to another instance. Depending on whether the instances come from the Spider training or development set, we get four splits: CG-SUB$_T$, CG-SUB$_D$, CG-APP$_T$ and CG-APP$_D$, all of which are only used for evaluation. The instances created under substitution share similar data characteristics while those under appending are much longer, so a good model performance on the latter requires compositional generalization. Table 3 summarizes the dataset information. In addition, we use the NatSQL representation (Gan et al., 2021) throughout the experiment due to its better alignment with the NL input.

Evaluation Metrics. We follow the standard Spider benchmarking and employ two evaluation metrics. **Exact Match (EM)** compares the generated and the ground-truth query by performing exact set matching at the lexical level (Yu et al., 2018b). **Execution Match (EX)** measures whether executing the generated query on the given database can yield the same results as using the ground truth. Notably, for a fair comparison with existing semantic parsers on the Spider leader board, we follow Gan et al. (2022), convert each generated NatSQL query into a SQL query, and report the evaluation results based on the converted SQL query.

Models, Baselines, and Implementation. We evaluate our proposed techniques by applying them to the pre-trained T5 model (Raffel et al., 2020). Our experiments are conducted using T5-base, with the use of database contents following Lin et al. (2020). As our second technique leverages component boundary labels to encourage the compositional generalization of LM, we compare it with a baseline (Yin et al., 2021) which similarly assumes the labels but utilizes them in a more complicated way, i.e., transforming the component alignments into supervision on the cross attention between input and output of the LM. We denote this base-

Model	Spider$_D$		CG-SUB$_T$		CG-SUB$_D$		CG-APP$_T$		CG-APP$_D$	
	EM	EX	EM	EX	EM	EX	EM	EX	EM	EX
Semantic Parsers with Specialized Architectures (Gan et al., 2022)										
RATSQL$_{B(S)}$	71.9	-	91.0	-	72.6	-	79.8	-	61.5	-
RATSQL$_{G(S)}$	**74.5**	-	**91.4**	-	**76.7**	-	**82.5**	-	**68.3**	-
Semantic Parsers based on LMs										
T5-base	64.6	67.9	83.8	88.1	69.1	71.1	60.2	70.3	45.0	54.9
T5-base + Tok	<u>71.8</u>	<u>75.6</u>	85.9	89.5	74.1	78.6	65.2	73.8	54.2	65.9
T5-base + Comp	64.4	68.2	86.3	90.2	69.3	73.1	69.8	<u>77.9</u>	53.5	63.4
T5-base + Tok + Comp	69.4	73.2	<u>86.6</u>	<u>90.7</u>	<u>76.6</u>	<u>79.8</u>	<u>71.1</u>	77.8	<u>61.0</u>	<u>69.4</u>
T5-base + Tok + Attn. Sup	69.4	73.7	83.6	87.7	71.7	75.6	62.3	70.8	56.3	66.2

Table 4: Results (%) on different evaluation sets. Top: state-of-the-art model using specialized architecture; numbers are collected from its paper and only EM is reported (code unavailable). Bottom: T5-base models with our proposed or baseline techniques; we report the average performance of each model over three runs. **Tok**: token preprocessing. **Comp**: component boundary marking. **Attn. Sup**: the attention supervision method of Yin et al. (2021).

line as **Attn. Sup**.[3] For both methods, we leverage component annotations from Spider-SS (Gan et al., 2022). These annotations were generated by applying a syntactic parser to decompose the NL question into sub-questions and then manually annotating their corresponding NatSQL components. We also compare with the state-of-the-art models, RATSQL$_{B(S)}$ and RATSQL$_{G(S)}$, from Gan et al. (2022), although their models adopt a specialized architecture (i.e., RATSQL (Wang et al., 2020)) and RATSQL$_{G(S)}$ additionally employed task-specific pre-training (Shi et al., 2021). Both models used the same component annotations from Spider-SS.

Finally, for each of our model variants in Table 4, we repeat the experiment three times, using three random seeds consistently across all models, and report the average results. We include more implementation details in Appendix D.

4.2 Results

Main Results. We present our results in Table 4. First, all models obtain the best performance on the in-distribution evaluation set CG-SUB$_T$ while suffering from more than 10% performance drops on others, confirming the challenges of the domain and compositional generation. As expected, all models have the worst performance on CG-APP$_D$, which requires both types of generalization. Between the two types, it is also observed that compositional generalization (as measured by CG-APP$_T$) is more challenging than domain generalization (as measured by Spider$_D$ and CG-SUB$_D$).

Second, our results show that the token preprocessing method, albeit simple, can improve both domain and compositional generalizations of LMs dramatically. For example, comparing T5-base with T5-base+Tok, the latter is improved by around 5-7% EM and 7% EX for domain generalization (on Spider$_D$ and CG-SUB$_D$), 5% EM and 3.5% EX for compositional generalization (on CG-SUB$_T$), and 9% EM and 11% EX for the challenging case when both types occur (on CG-APP$_D$). Additionally, we also show the effectiveness of token preprocessing with T5-3B on Spider$_D$ in App. B.

Moving on to our proposed component boundary marking method, it shows to be particularly helpful for compositional generalization. Specifically, applying it to T5-base leads to a 9% EM and 7% EX increase on CG-APP$_T$, and an 8% EM and 8% EX increase on CG-APP$_D$. On the in-distribution evaluation set, this technique also gives slight improvement, whereas, for domain generalization, there is no obvious impact from this technique.

Finally, augmenting T5-base with both techniques (i.e., T5-base+Tok+Comp) leads to better performance than applying each technique individually in most evaluation sets, implying that our two techniques are complementary to each other. Specifically, for in-distribution evaluation, using each technique individually or both of them together yield similar results; for domain generalization, there is no additional gain from applying component boundary marking on the top of the token preprocessing; for compositional generaliza-

[3] In our implementation, we apply the supervision to cross-attention distribution averaged across all decoder layers and heads. We also tried cross-attention from only the top decoder layer, but the results are similar.

tion, the two techniques together contribute the best EM across all models and baselines. Overall, combining the two techniques shrinks the performance gap between in-distribution and domain OOD by around 2-4% EM, composition OOD by 7%, and joint OOD by 13%.

Compared with Special Architectures. Despite its simplicity, our T5-base+Tok+Comp model achieves comparable or better performance than the two RATSQL variants on CG-SUB$_D$. It also performs comparably to RATSQL$_{B(S)}$ on CG-APP$_D$.

Compared with Attn. Sup. Surprisingly, the attention supervision has only led to around 2% EM and 1.5% EX gains on CG-APP$_D$, while no further advantage is observed on other evaluation sets. In our conjecture, this is due to the misalignment between the objective of Attn. Sup (Yin et al., 2021) and the attention mechanism of pre-trained LMs. Specifically, Attn. Sup encourages the attention distribution of different heads to be consistent with the component alignment supervision. However, prior work (Voita et al., 2019) suggests that different attention heads of even the same layer may have different functions and roles. Thus, when coarsely defining the objective function, it may not allow for the most effective supervision. Furthermore, similar to our finding, Yin et al. (2021) did not observe performance gain when they applied Attn. Sup to T5-base on CFQ (Keysers et al., 2020).

Qualitative Analysis on Tokenization. To qualitatively understand how our token preprocessing helps the generalization, we randomly sampled 50 examples from the Spider$_D$ to analyze how frequently the T5 tokenizer divides tokens into less meaningful subtokens. Consequently, we found 243 tokenization issues in total, and 140 of them can be resolved by our token preprocessing. The remaining cases are like splitting "id" into "i" and "d" as shown in Table 1, which is beyond our scope.

Error Analysis on Component Boundary Marking. We manually examined 50 error predictions from T5-base+Tok+Comp and contrasted them with the errors of T5-base+Tok. Intriguingly, we observed much more frequent schema items or value hallucinations from the former. For example, it may generate queries accessing non-existing columns in a table, or misspells the literal values in the queries. We conjecture that this is because our component boundaries are only applied to the NL input, not the database schema (note that literal values are grounded and attached to schema items in their input representations; see Appendix D for details). This reveals a new challenge of LM generalization in text-to-SQL semantic parsing, i.e., how to properly handle the database schema when injecting prior knowledge into LMs for compositional generalization.

Model	Exact Match
COARSE2FINE + SS (Span-level Sup.)	47.4
T5-base	63.9
T5-base + Tok	65.1
T5-base + Tok + Comp	**67.7**

Table 5: Results (%) on SMCalFlow-Compositional Skills dataset (16-shot setting). Top: Result from Yin et al. (2021). Bottom: T5-base models with our proposed or baseline techniques; we report the average performance of each model over three runs.

Generalizing to Other Semantic Parsing Tasks. While our main focus in this work is on text-to-SQL parsing, we also investigate whether our approaches can generalize beyond this specific task. To this end, we implemented both of our techniques to SMCalFlow-CS (Yin et al., 2021), a compositional generalization dataset for text-to-LISP expression parsing (Semantic Machines et al., 2020). For "+Comp", We utilize the span-level alignments heuristically derived by Yin et al. (2021) as component annotations.[4] Our results in Table 5 show that: (1) Our token preprocessing can be universally helpful for LMs to model schema items, predicates, etc., leading to 1.2% performance gain over T5-base; (2) Our component boundary marking method is highly effective for compositional generalization, which offers 2.6% additional gain.

5 Conclusion

In this paper, we present two simple yet effective techniques to improve the domain and compositional generalization of LMs in text-to-SQL semantic parsing. Our techniques aid LMs in preserving the semantic boundaries of tokens and components in their input and output. We also demonstrate their potential to be generalized to other semantic parsing tasks.

[4]Yin et al.'s approach requires knowing the ground-truth LISP expression when deriving the component boundaries for the input question. In our experiment, we assume the availability of these question boundaries at test time and focus on showcasing the potential of "Comp", while automating this question decomposition is left as future work.

Limitations

Future work can further apply our approaches to other semantic parsing tasks. For example, for parsing texts to lambda-calculus expressions for knowledge base question answering (Dong and Lapata, 2016), one can similarly preprocess the schema items (e.g., "`department_time`" into "`department _ time`") and typed values (e.g., "`dallas:ci`" into "`dallas : ci`") for more meaningful subword tokenization results. In addition, our experiments are based on T5. To further verify the effectiveness of our techniques, one can apply them to other pre-trained language models such as BART (Lewis et al., 2020) and GPT-2 (Radford et al., 2019) as well.

Acknowledgments

We would like to thank all anonymous reviewers for their constructive comments. We also thank Yujian Gan and Xinyun Chen for their help in using the NatSQL and the Spider-SS datasets, as well as Pengcheng Yin for using the code base of Attn. Sup. This project was supported by resources provided by the Office of Research Computing at George Mason University (`https://orc.gmu.edu`) and funded in part by grants from the National Science Foundation (Awards Number 1625039 and 2018631).

References

Ekin Akyürek, Afra Feyza Akyürek, and Jacob Andreas. 2020. Learning to recombine and resample data for compositional generalization. *arXiv preprint arXiv:2010.03706*.

Jacob Andreas. 2020. Good-enough compositional data augmentation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7556–7566, Online. Association for Computational Linguistics.

Henry Conklin, Bailin Wang, Kenny Smith, and Ivan Titov. 2021. Meta-learning to compositionally generalize. 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 3322–3335, Online. Association for Computational Linguistics.

DeepSpeed. 2023. https://github.com/microsoft/deepspeed.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany. Association for Computational Linguistics.

Yujian Gan, Xinyun Chen, Qiuping Huang, and Matthew Purver. 2022. Measuring and improving compositional generalization in text-to-SQL via component alignment. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 831–843, Seattle, United States. Association for Computational Linguistics.

Yujian Gan, Xinyun Chen, Jinxia Xie, Matthew Purver, John R. Woodward, John Drake, and Qiaofu Zhang. 2021. 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. Association for Computational Linguistics.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-SQL in cross-domain database with intermediate representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4524–4535, Florence, Italy. Association for Computational Linguistics.

Jonathan Herzig and Jonathan Berant. 2021. Span-based semantic parsing for compositional generalization. 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 908–921, Online. Association for Computational Linguistics.

Jonathan Herzig, Peter Shaw, Ming-Wei Chang, Kelvin Guu, Panupong Pasupat, and Yuan Zhang. 2021. Unlocking compositional generalization in pre-trained models using intermediate representations. *arXiv preprint arXiv:2104.07478*.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations*.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, et al. 2019. Measuring compositional generalization: A comprehensive method on realistic data. *arXiv preprint arXiv:1912.09713*.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2873–2882. PMLR.

Itay Levy, Ben Bogin, and Jonathan Berant. 2022. Diverse demonstrations improve in-context compositional generalization. *arXiv preprint arXiv:2212.06800*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020. Bridging textual and tabular data for cross-domain text-to-SQL semantic parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4870–4888, Online. Association for Computational Linguistics.

Jiexing Qi, Jingyao Tang, Ziwei He, Xiangpeng Wan, Chenghu Zhou, Xinbing Wang, Quanshi Zhang, and Zhouhan Lin. 2022. Rasat: Integrating relational structures into pretrained seq2seq model for text-to-sql. *arXiv preprint arXiv:2205.06983*.

Linlu Qiu, Peter Shaw, Panupong Pasupat, Pawel Nowak, Tal Linzen, Fei Sha, and Kristina Toutanova. 2022a. Improving compositional generalization with latent structure and data augmentation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4341–4362, Seattle, United States. Association for Computational Linguistics.

Linlu Qiu, Peter Shaw, Panupong Pasupat, Paweł Krzysztof Nowak, Tal Linzen, Fei Sha, and Kristina Toutanova. 2021. Improving compositional generalization with latent structure and data augmentation. *arXiv preprint arXiv:2112.07610*.

Linlu Qiu, Peter Shaw, Panupong Pasupat, Tianze Shi, Jonathan Herzig, Emily Pitler, Fei Sha, and Kristina Toutanova. 2022b. Evaluating the impact of model scale for compositional generalization in semantic parsing. *arXiv preprint arXiv:2205.12253*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

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, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Semantic Machines, Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, Hao Fang, Alan Guo, David Hall, Kristin Hayes, Kellie Hill, Diana Ho, Wendy Iwaszuk, Smriti Jha, Dan Klein, Jayant Krishnamurthy, Theo Lanman, Percy Liang, Christopher H. Lin, Ilya Lintsbakh, Andy McGovern, Aleksandr Nisnevich, Adam Pauls, Dmitrij Petters, Brent Read, Dan Roth, Subhro Roy, Jesse Rusak, Beth Short, Div Slomin, Ben Snyder, Stephon Striplin, Yu Su, Zachary Tellman, Sam Thomson, Andrei Vorobev, Izabela Witoszko, Jason Wolfe, Abby Wray, Yuchen Zhang, and Alexander Zotov. 2020. Task-oriented dialogue as dataflow synthesis. *Transactions of the Association for Computational Linguistics*, 8:556–571.

Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. 2021. Compositional generalization and natural language variation: Can a semantic parsing approach handle both? 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 922–938, Online. Association for Computational Linguistics.

Peng Shi, Patrick Ng, Zhiguo Wang, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Cicero Nogueira dos Santos, and Bing Xiang. 2021. Learning contextual representations for semantic parsing with generation-augmented pre-training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13806–13814.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Bailin Wang, Mirella Lapata, and Ivan Titov. 2021. Structured reordering for modeling latent alignments in sequence transduction. *Advances in Neural Information Processing Systems*, 34:13378–13391.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. 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. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

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. 2022. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. *arXiv preprint arXiv:2201.05966*.

Pengcheng Yin, Hao Fang, Graham Neubig, Adam Pauls, Emmanouil Antonios Platanios, Yu Su, Sam Thomson, and Jacob Andreas. 2021. Compositional generalization for neural semantic parsing via span-level supervised attention. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2810–2823, Online. Association for Computational Linguistics.

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

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018b. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Rui Zhang, Tao Yu, Heyang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019. Editing-based SQL query generation for cross-domain context-dependent questions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5338–5349, Hong Kong, China. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

A Token Preprocessing Details

We propose a simple token preprocessing method. Instead of directly feeding the input to the subword tokenizer, we introduce three preprocessing steps: (1) For schema items in input and output, reversing the snake case to the normal, e.g., "`pet_age`" to "`pet _ age`"; (2) For any call of "`Table.Column`", splitting the tokens around the access operator "`.`" (i.e., "`Table . Column`"); and (3) Replacing any reserved words that cannot be properly handled in NatSQL, e.g., "`avg`" to "`average`". In practice, we also handle formalism-specific special tokens, e.g., adding the "less than" operator "<" to the vocabulary of T5 tokenizer. While we showcase our token preprocessing under text-to-SQL parsing, the intuition can be generalized to other formalisms (e.g., regex, λ-expression) easily.

In addition, we also check the issue of tokenization in other popular LM tokenizers and found that the tokenization issue is not specific to T5. Examples of bad tokenization from BERT (Devlin et al., 2019) and GPT2 (Radford et al., 2019) tokenizers and after our token preprocessing are listed in Table 6.

GPT2 Tokenizer	
Before:	`student_enrolment_courses`
After:	`student _ enrolment _ courses`
Before:	`transcripts.transcript_date`
After:	`transcripts . transcript _ date`
Before:	`avg`
After:	`average`
BERT Tokenizer	
Before:	`singer.NetWorthMillions`
After:	`singer . Net Worth Millions`
Before:	`avg`
After:	`average`
Before:	`asc`
After:	`ascending`

Table 6: Tokenization of snake case, camel case, and token notation in BERT and GPT2 tokenizer. Coloring indicates tokenization, same as Table 1.

B T5-3B Experiment

To assess the effectiveness of our token preprocessing technique with larger LMs, we apply it to T5-3B and evaluate the model on Spider$_D$. The results

Model	Spider$_D$ EM	Spider$_D$ EX
T5-3B (w deepspeed)	73.2	77.4
T5-3B (w/o deepspeed)	76.0	79.8
T5-3B + Tok (w deepspeed)	74.4	78.7
T5-3B + Tok (w/o deepspeed)	**77.4**	**80.9**

Table 7: Results (%) on Spider$_D$ when T5-3B(+Tok) was trained with or without using deepspeed.

are shown in Table 7. Our results show that T5-3B+Tok has a performance gain of 1.1%, indicating that it is helpful for larger LMs as well. Additionally, we also provide results with and without using DeepSpeed (2023), a deep learning optimization library that is used to train large models more efficiently. Surprisingly, although DeepSpeed (2023) helped us improve training speed, we found a performance drop of around 2.1-2.2% EX while using it. However, our token preprocessing consistently leads to around 1.0% absolute performance gain.

C Component Boundary Marking Details

In Table 8, we present one more example of component boundary marking. In this example, the NL component *"What is the most populace city"* is aligned with two non-continuous SQL segments, "`select city.Name, city.Population`" and "`order by city.Population desc limit 1`". To handle such cases, we apply the same pair of special tokens "`[sep0]`" "`[/sep0]`" twice, one for each segment.

Component Boundary Marking Example	
Before:	What is the most populace city that speaks English? `Select city.Name, city.Population where countrylanguage.Language = "English" order by city.Population desc limit 1`
After:	`[sep0]` What is the most populace city `[/sep0]` `[sep1]` that speaks English? `[/sep1]` `[sep0] select city.Name , city.Population [/sep0] [sep1] where countrylanguage.Language = "English" [/sep1] [sep0] order by city.Population desc limit 1 [/sep0]`

Table 8: An example of component boundary marking when an NL component aligns with non-continuous segments in the SQL side. In this case, we apply the special tokens for each segment.

D Implementation Details

Our experiments are conducted based on the pre-trained T5 model. The input to T5 follows the same format and order as Scholak et al. (2021) (except our additional token preprocessing, if applied), i.e., "`Question | Database 1 | Table 1: Column 1, Column 2,...| Table 2: Column 1, Column 2...`". We also use the database contents as parts of the input, following Lin et al. (2020). For example, if the NL question mentions a literal value (e.g., "New York"), the appearance of whom can be found in the contents of a certain "`Column 1`" via fuzzy string matching, then when we represent the database schema, we will include it via "`Database 1 | Table 1: Column 1 (New York), Column 2, ...`".

We fine-tune the T5-base LM that consists of 220 million parameters on NVIDIA A100 GPU for 10-12 hours. It was trained with a learning rate of 10^{-4} and batch size 16 for T5-base for a maximum of 20K training steps. The model is evaluated on Spider$_D$ for every 1K training steps, and the best checkpoint is selected based on the model EM on Spider$_D$. In inference time, we perform simple greedy decoding.

We use the PyTorch-Transformers library (Wolf et al., 2020), which is a library for state-of-the-art pre-trained models for NLP, to fine-tune our models. Specifically, our code for fine-tuning T5-base is adapted from PICARD's implementation (Scholak et al., 2021). Furthermore, we also use DeepSpeed (2023) to fine-tune all of our T5-base models.

Datasets. We used Spider (Yu et al., 2018b), NatSQL (Gan et al., 2021), Spider-CG (Gan et al., 2022), and SMCalFlow-CS (Yin et al., 2021) in our work. They are under the license of CC BY-SA 4.0. Our use of these datasets is consistent with their intended use, i.e., for scientific research. All datasets are in English. They contain annotated NL and SQL or NatSQL or LISP expression pairs from the open domain.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Limitations

☒ A2. Did you discuss any potential risks of your work?
We don't see the potential of how our two techniques can be misused.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

4

☑ B1. Did you cite the creators of artifacts you used?
2, 4

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
B

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
B

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Sensitive contents are unlikely to be contained in the datasets we used. For example, for Spider-CG, it is annotated by domain experts.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
B

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
No response.

C ☑ Did you run computational experiments?

4.1

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
B

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
4.1

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
4.1

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
4.1

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.