Shifted PAUQ 🕷: Distribution shift in text-to-SQL

Oleg Somov[1], **Elena Tutubalina**[1,2,3]

[1]AIRI, [2]Kazan Federal University, [3]HSE University

Correspondence: somov@airi.net

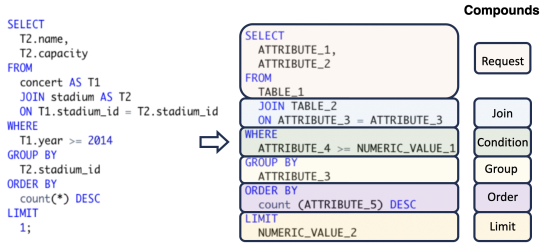
Abstract

Semantic parsing plays a pivotal role in advancing the accessibility of human-computer interaction on a large scale. Spider, a widely recognized dataset for text-to-SQL, contains a wide range of natural language (NL) questions in English and corresponding SQL queries. Original splits of Spider and its adapted to Russian language and improved version, PAUQ, assume independence and identical distribution of training and testing data (i.i.d split). In this work, we propose a target length split and multilingual i.i.d split to measure compositionality and cross-language generalization. We present experimental results of popular text-to-SQL models on original, multilingual, and target length splits. We also construct a context-free grammar for the evaluation of compositionality in text-to-SQL in an out-of-distribution setting. We make the splits publicly available on HuggingFace hub via `https://huggingface.co/datasets/composite/pauq`.

1 Introduction

In this paper, we focus on a subtask of semantic parsing called text-to-SQL, which involves mapping natural language (NL) questions to Structured Query Language (SQL). Sequence-to-sequence (seq-to-seq) models such as RAT-SQL (Wang et al., 2020), BRIDGE (Lin et al., 2020), and RESDSQL (Li et al., 2023) have been widely employed for the text-to-SQL task. However, recent studies have highlighted the limitations of seq-to-seq models in out-of-distribution settings (Shaw et al., 2021; Gu et al., 2021; Chang et al., 2023).

The evaluation process for text-to-SQL models is currently a topic of active research. Yu et al. (2018a) determined the difficulty of requests based on the number of SQL components, resulting in four distinct categories: “Easy”, “Medium”, “Hard”, and “Extra Hard”. Finegan-Dollak et al. (2018) demonstrated that the division of the popular text-to-SQL dataset into training and test sets is



```
SELECT
  T2.name,
  T2.capacity
FROM
  concert AS T1
  JOIN stadium AS T2
  ON T1.stadium_id = T2.stadium_id
WHERE
  T1.year >= 2014
GROUP BY
  T2.stadium_id
ORDER BY
  count(*) DESC
LIMIT
  1;
```

```
SELECT
  ATTRIBUTE_1,
  ATTRIBUTE_2
FROM
  TABLE_1
  JOIN TABLE_2
  ON ATTRIBUTE_3 = ATTRIBUTE_3
WHERE
  ATTRIBUTE_4 >= NUMERIC_VALUE_1
GROUP BY
  ATTRIBUTE_3
ORDER BY
  count (ATTRIBUTE_5) DESC
LIMIT
  NUMERIC_VALUE_2
```

Figure 1: SQL is transformed into an SQL template via masking technique for subsequent compositional evaluation. Token compositions, referred to as compounds, are extracted via context-free grammar and then compared against the predicted query.

inadequate for assessing the model’s generalization abilities. Ribeiro et al. (2020) highlighted that it is important to access performance on functional test sets and out-of-distribution examples, as random train and test splits can overestimate real-world performance and miss important error cases. Chang et al. (2023) proposed a text-to-SQL specific perturbation benchmark . This benchmark encompasses 17 categories for evaluation on text-to-SQL models robustness. Shaw et al. (2021) proposed new train and test splits of non-synthetic datasets for modeling out-of-distribution (OOD) setting in order to measure compositionality in semantic parsing.

In this work, we propose three splits of the existing PAUQ dataset (Bakshandaeva et al., 2022), an improved version of Spider (Yu et al., 2018b), for measuring across-language generalization and compositional generalization. The first split evaluates how models benefit from training in multiple languages of the same task. The split is constructed by joining original splits of Russian and English versions of PAUQ (MPAUQ). The original i.i.d splits of Russian and English will be referred to as **RuPAUQ OS** and **EnPAUQ OS**, respectively. The second two splits, target length, evaluate how the models perform if the text-to-SQL data is split by target length characteristic. The first split is

	RuPAUQ OS		EnPAUQ OS		EnPAUQ TRL		EnPAUQ TSL	
	Train	Test	Train	Test	Train	Test	Train	Test
Split size	8800	1076	8800	1076	7890	1975	7900	1975
Avg. template length	21.04	16.79	21.1	16.8	23.74	8.27	15.28	42.06
Avg. question length	8.95	9.05	12.01	12.31	12.57	9.92	11.68	13.46

Table 1: Statistics for PAUQ original split (OS) for both Russian (RuPAUQ), English (EnPAUQ), and target length splits (TRL and TSL). The length is calculated as the length of sequences in tokens.

when samples in the train set are longer than samples in the test set named **TRL**. The second split is when samples in the test are longer than samples in the train named **TSL**. TRL examines the ability of the model to generalize to simpler SQL without directly learning to construct them. As stated in (Hupkes et al., 2020), TRL evaluates model systematicity, while TSL assesses its productivity.

2 Spider and PAUQ

Spider (Yu et al., 2018b) comprises a substantial collection of 10,181 English questions and 5,693 unique complex SQL queries, spanning 200 databases with multiple tables that encompass 138 distinct domains. Spider was randomly split into train, dev, and test sets. In the case of PAUQ, the Russian version of Spider, all three components - questions, SQL queries, and database content, have been modified and localized. PAUQ improved the original Spider by inserting the missing values, correcting errors, and adding new samples of poorly represented types. As the Spider test set is not publicly available, PAUQ uses the dev set for testing. PAUQ contains 8,800 and 1,076 NL samples for training and testing, respectively. In PAUQ, the total numbers of different elements of databases are: (i) **Databases:** 166 entities (88.0% – train set, 12.0% – dev set); (ii) **Tables:** 876 entities (90.8% – train set, 9.2% – dev set); **Columns:** 4503 entities (90.2% – train set, 9.8% – dev set); **Values:** 531,164 and 533,751 unique values respectively (88.4% – train set, 11.6% – dev set). We adopt three components (NL questions, SQL queries, database) from GitHub via `https://github.com/ai-spiderweb/pauq`, where each query corresponds to Russian and English texts.

3 Proposed Split

We propose a target length-based split to mimic full shift (Hupkes et al., 2020) in the text-to-SQL task for measuring the compositional generalization ability of NLP models. We design split in the following way:

1. Normalize SQL by masking tables, attributes, textual and numeric values with corresponding mask token (see Figure 1) - this way, we get an SQL template by which the target length split will be done. Text-to-SQL solutions are expected to infer attributes, tables, and values from a given question and schema. The ability to handle it correctly is called substitutivity (Hupkes et al., 2020). Since our split is purely for compositional generalization evaluation, we use that template for splitting instead of the original query. During an evaluation, we also use that masking technique and true SQL template to estimate compositional ability of the models.

2. For both splits, we sort SQL templates in ascending or descending order based on template token size. Then, we iterate by the sorted template list in order to fill train and test splits. We require the test size of both splits to be 20% of the original dataset.

3. Since we want to check how the model recombines known tokens to form novel structures, we clear the test set from such queries where there is no full token intersection with train tokens.

We make sure that there is no template intersection in train/test for both TRL and TSL splits. TSL is the most complicated split because the model has to recombine known tokens to form new complex and long queries. TRL split requires the model to also recombine new tokens but to generate queries of shorter length.

For measuring cross-language generalization, we merge English and Russian train sets and evaluate English and Russian test sets separately. Our motivation for the multilingual split is whether training in two languages on the same task can benefit one model. Table 1 shows statistics for OS and target length English and Russian splits.

4 Baselines and Experiments

We focus our dataset on two types for generalization evaluation - compositional generalization and generalization across languages (see Appendix B for the GenBench card (Hupkes et al., 2022)).

We utilized four popular Spider models:

- T5-base (Raffel et al. 2020);
- RESDSQL (Li et al. 2023);
- RAT-SQL (Wang et al. 2020);
- BRIDGE (Lin et al. 2020);

T5-base is a pre-trained encoder-decoder transformer with a language modeling head.

RESDSQL decouples schema linking and query generation tasks into two stages. The first stage selects the most relevant schema items for the question. During the second stage, the model learns to decode the SQL template of the actual query concatenated with the original actual query. That way, the model can condition generating SQL skeleton before the full original query. RESDSQL uses ROBERTA-large (Zhuang et al., 2021) large for the schema linking and T5-base for query generation. For RuPAUQ and MPAUQ, we replace T5-base model with MT0-base (Muennighoff et al., 2022).

RAT-SQL is an encoder-decoder model that uses a relation-aware transformer within the encoder to model alignments between database schema and content and question tokens. The decoder of the model is tree-structured and generates an abstract syntax tree in the context-free SQL grammar.

BRIDGE, in turn, utilizes database schema and content as input to the model. It has an encoder-decoder architecture with the pointer-generator network using beam-search. The model generates queries in execution-guided order. Both RAT-SQL and BRIDGE use the BERT-base (Devlin et al., 2019) language model as an encoder. The details on hyperparameters are presented in the Appx. A.

5 Overall results

We evaluate 4 models on our splits - T5-base, RESDSQL (with T5-base or MT0-base), RAT-SQL, and BRIDGE. We trained each model on a split train set and evaluated it on a test set with 3 random seeds and averaged predictions. Our evaluation metrics are Exact Matching and Execution Accuracy. Results are presented in Tab. 2, 3, 4.

EnPAUQ	Exact Match		Exec Match	
split	T5	RESDSQL	T5	RESDSQL
OS	0.46	0.69	0.45	0.74
TRL	0.56	0.72	0.55	0.80
TSL	0.10	0.19	0.08	0.30

Table 2: Compositional generalization exact match and exec match metrics for T5-base and RESDSQL. Each model is evaluated on a corresponding test split.

Train	Test	T5-base	RESDSQL	RAT-SQL	BRIDGE
En	En	0.46	0.69	0.66	0.60
M		0.48	0.66	0.66	0.68
Ru	Ru	0.42	0.39	0.51	0.52
M		0.43	0.39	0.57	0.55

Table 3: Exact match metrics for across language generalization. En is EnPAUQ, Ru is RuPAUQ, and M is for MPAUQ.

Train	Test	T5-base	RESDSQL	RAT-SQL	BRIDGE
En	En	0.45	0.74	0.63	0.60
M		0.45	0.68	0.65	0.65
Ru	Ru	0.39	0.43	0.49	0.48
M		0.40	0.42	0.53	0.53

Table 4: Execution match metrics for across language generalization. En is EnPAUQ, Ru is RuPAUQ, and M is MPAUQ.

Target length split is our evaluation towards compositional generalization measurements (Table 5). We evaluate T5-base and RESDSQL on this split. The predictions for both models are stable across different seed runs with standard deviation for less than 1% for both T5-base and RESDSQL. Since RESDSQL is a more advanced model with multiple query generation stages, it tends to over-fit less on TSL and generate novel queries - its execution match is much higher than its exact match. On TRL compared to the OS split, the metrics are higher. It is not surprising because our test set for TRL consists mostly of PAUQ question pairs from **easy** category. Such evaluation shows that the model is able to generalize from complex queries to simple ones without overcrowding the training dataset with simple queries for text-to-SQL. In Tab. 3 and 4, we see that training simultaneously on two languages with multilingual models can give a slight boost to some models. In our experiments, we see an increase in execution match metric for T5-base (Russian +1%), RAT-SQL (English +2%, Russian +4%), and BRIDGE (English +5%, Russian +2%). However, RESDSQL had a drop in English (-6%) when trained in such a setting.

EnPAUQ	Syntax accuracy		Compound accuracy		OOD Compound accuracy	
Split	T5	RESDSQL	T5	RESDSQL	T5	RESDSQL
OS	0.75	0.81	0.81	0.82	0.54	0.48
TRL	0.75	0.91	0.93	0.94	0.65	0.66
TSL	0.73	0.57	0.71	0.68	0.47	0.42

Table 5: Compositional generalization in-depth evaluation of two models.

6 In-depth compound analysis

We wanted to explore the ability of compositional generalization for proposed target length splits. We have developed context-free grammar (CFG) in order to parse queries. In order to analyze, we utilize the concept of compound. For example, we have dataset tokens `SELECT`, `COUNT`, `SUM`, `Goals`, `Teams` (`Goals` and `Teams` are table attributes). Composition of these tokens such `SELECT COUNT Goals` is called *compounds*. CFG covers compounds for `REQUEST`, `JOIN`, `CONDITION`, `GROUP`, `ORDER`, `LIMIT`. For evaluation, we apply the same masking technique used for split generation for generated SQL. Then we parse it with CFG to extract compounds (see Figure 1). For in-depth compound analysis, we propose and evaluate three accuracy metrics derived from our compounds:

- **Syntax accuracy** - this accuracy metric estimates whether all expected true SQL compounds are present in predicted SQL. For example, if the true SQL has `REQUEST` compound and `CONDITION` compounds `SELECT count(ATTRIBUTE_1) FROM TABLE_1 WHERE ATTR_2 = TEXT_VAL_1 OR ATTR_3 = TEXT_VAL_2`; and predicted SQL has only `REQUEST` compound in `SELECT count(ATTRIBUTE_1) FROM TABLE_1` - the syntax accuracy for such query will be 0.5;

- **Compound accuracy** - this accuracy metric estimates the proportion of predicted compounds to expected ones in the query. For example, if true query has 1 `REQUEST` compound and 2 `CONDITION` compounds `SELECT count(ATTRIBUTE_1) FROM TABLE_1 WHERE ATTRIBUTE_2 = TEXT_VALUE_1 OR ATTRIBUTE_3 = TEXT_VALUE_2`; while the predicted query has 1 `REQUEST` compound and 1 `CONDITION` compounds in `SELECT count(ATTRIBUTE_1) FROM TABLE_1 WHERE ATTRIBUTE_2 = TEXT_VALUE_1` the compound accuracy will be 0.33.

- **OOD Compound accuracy** - compound accuracy, which is only calculated on compounds that were not seen during training.

We calculate the proposed metrics independently for each sample and then average over all test set predictions. In Tab. 5, we see that a more advanced RESDSQL overall model increases only syntax accuracy metric compared to the T5-base on OS and TRL splits, while other two metrics are relatively close to each other for all splits. OOD Compound Accuracy is significantly lower then Compound Accuracy (e.g., 0.66 vs 0.94 for RESDSQL on the TRL split, respectively). The TRL split compound metrics show that both models are able to generate unseen compounds in approximately 1.5 times better than in the TSL split.

7 Discussion and Conclusion

In this work, we have explored two types of generalization - compositional generalization (TRL and TSL splits) and across language generalization (MPAUQ). We have evaluated 4 text-to-SQL models on these splits. For an in-depth analysis on compositional generalization, we have developed CFG and proposed two metrics for compound evaluation. Our results show that TSL split is the most challenging split for the model. TRL split shows that the models are able to generalize to unseen short SQL. Multilingual split shows that some models can benefit from learning on the translated task to gain performance on individual language.

In future work, we plan to perform more experiments to make more compositional generalization splits (e.g., template or target maximum compound divergence splits (Shaw et al., 2021)). We also plan to explore different neural architectures and training strategies to enhance the model's ability to handle complex queries and measure compositional generalization and generalization across languages. We hope to prepare the PAUQ leaderboard and encourage further research in this area.

Acknowledgments

This research was supported in part through computational resources of HPC facilities at HSE University. The work of E.T. has been supported by the Russian Science Foundation grant # 23-11-00358. We would also like to thank the anonymous reviewers for their comments on this paper.

Limitations and Ethics

We note that large language models such as Codex, a 175B GPT model further fine-tuned on code, are out of the scope of this work.

PAUQ and Spider's limitations First of all, the data is still 'artificial', which means that it was created by a limited number of people specifically for training and evaluating text-to-SQL models; thus, it lacks the diversity and complexity of natural data formed by questions that people formulate in order to get the desired information from the database. For instance, the real-world data contain NL queries that require common sense knowledge that cannot be extracted directly from the database, ambiguous questions allowing various ways of interpretation that are quite frequent, and queries with window functions that make the process easier and more convenient, – all of these are not included in the Spider dataset, as well as in PAUQ.

Train and test tokens Our research explores full shift by splitting the dataset based on target template length. Specifically, we examine the scenario where test template tokens are present in the training set and do not explore the more challenging case of modeling unseen tokens.

References

Daria Bakshandaeva, Oleg Somov, Ekaterina Dmitrieva, Vera Davydova, and Elena Tutubalina. 2022. PAUQ: Text-to-SQL in Russian. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2355–2376, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Shuaichen Chang, Jun Wang, Mingwen Dong, Lin Pan, Henghui Zhu, Alexander Hanbo Li, Wuwei Lan, Sheng Zhang, Jiarong Jiang, Joseph Lilien, Steve Ash, William Yang Wang, Zhiguo Wang, Vittorio Castelli, Patrick Ng, and Bing Xiang. 2023. Dr.spider: A diagnostic evaluation benchmark towards text-to-sql robustness.

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.

Catherine Finegan-Dollak, Jonathan K Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-sql evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360.

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond i.i.d.: Three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*. ACM.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. 2020. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795.

Dieuwke Hupkes, Mario Giulianelli, Verna Dankers, Mikel Artetxe, Yanai Elazar, Tiago Pimentel, Christos Christodoulopoulos, Karim Lasri, Naomi Saphra, Arabella Sinclair, Dennis Ulmer, Florian Schottmann, Khuyagbaatar Batsuren, Kaiser Sun, Koustuv Sinha, Leila Khalatbari, Maria Ryskina, Rita Frieske, Ryan Cotterell, and Zhijing Jin. 2022. State-of-the-art generalisation research in NLP: a taxonomy and review. *CoRR*.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023. Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13067–13075.

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.

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, et al. 2022. Crosslingual generalization through multitask finetuning. *arXiv preprint arXiv:2211.01786*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 4902–4912, Online. Association for Computational Linguistics.

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.

Kaiser Sun, Adina Williams, and Dieuwke Hupkes. 2023. A replication study of compositional generalization works on semantic parsing. In *ML Reproducibility Challenge 2022*.

Bailin Wang, Mirella Lapata, and Ivan Titov. 2021. Meta-learning for domain generalization in semantic parsing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 366–379, Online. Association for Computational Linguistics.

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.

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. 2018a. 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.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 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.

Liu Zhuang, Lin Wayne, Shi Ya, and Zhao Jun. 2021. A robustly optimized BERT pre-training approach with post-training. In *Proceedings of the 20th Chinese National Conference on Computational Linguistics*, pages 1218–1227, Huhhot, China. Chinese Information Processing Society of China.

A Experimental Setup

For English OS, TRL, TSL splits we train T5-base from HF checkpoint provided at `https://huggingface.co/t5-base`. We trained model for 10k iterations with a batch size of 256 and learning rate 10^{-4} as in Sun et al. 2023. As optimizer we have used Adafactor.

For training RESDSQL, we used the original implementation of RESDSQL provided at `https://github.com/RUCKBReasoning/RESDSQL` .

For training RuPAUQ OS, MPAUQ for the first stage of schema linking in RESDSQL we used `https://huggingface.co/DeepPavlov/xlm-roberta-large-en-ru-mnli` and for query generation in both T5-base and RESDSQL we used `https://huggingface.co/bigscience/mt0-base`

We used Tensor2Struct package (Wang et al., 2021) to train RAT-SQL. The hyperparameters are taken from the original implementation of RAT-SQL provided at `https://github.com/berlino/tensor2struct-public`. In monolingual setup, RAT-SQL models are trained for a maximum of 25k iterations, then the best checkpoint on the corresponding dev set in terms of exact match was picked (in all cases it is a checkpoint obtained after training in the range of 20k to 25k iterations). In multilingual setup, when training data is double-sized, the maximum number of iterations is increased to 40k.

As for BRIDGE, in all cases it was trained for a maximum of 20k iterations. Then the best checkpoint according to exact-match top-1 metric on the corresponding dev set was selected. During training, we used default hyperparameters from the original implementation of BRIDGE provided at `https://github.com/salesforce/TabularSemanticParsing`.

To develop the context-free grammar, we use a Yargy library provided at `https://github.com/natasha/yargy`.

For training RuPAUQ OS, MPAUQ RAT-SQL and BRIDGE encoders we have used `https://huggingface.co/bert-base-multilingual-cased`

All models were trained on one Tesla V100 32 GB.

For evaluation we have used original Spider evaluation script `https://github.com/taoyds/test-suite-sql-eval` .

B GenBench Card

Motivation			
Practical	*Cognitive*	*Intrinsic*	*Fairness*
□			

Generalisation type					
Compositional	*Struct-l*	*Cross Task*	*Cross Language*	*Cross Domain*	*Robustness*
□			□		

Shift type			
Covariate	*Label*	*Full*	*Assumed*
		□	

Shift source			
Naturally occuring	*Partitioned natural*	*Generated shift*	*Fully generated*
	□		

Shift locus			
Train–test	*Finetune train–test*	*Pretrain–train*	*Pretrain–test*
	□		