Virtual Task Selection in Meta-Learning for Domain Generalization in Semantic Parsing

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recent work has explored applying meta-learning strategies to improve domain generalization in natural language processing tasks. We propose a simple strategy for improving such model training with meta-learning algorithms through strategic sampling of virtual tasks. We experiment with our task selection strategies on a challenging text-to-SQL semantic parsing dataset and show that it is possible to achieve performance comparable to a strong baseline training strategy while using half as many training steps. Furthermore, comparison with a strong multi-task learning baseline suggests that the benefits of our strategies are complementary to the benefits conferred by meta-learning itself.

1 Introduction

Recent studies have shown that meta-learning can be used to enhance domain generalization across numerous vision and language tasks(Balaji et al., 2018; Li et al., 2019). In particular, optimization-based Model-Agnostic Meta-Learning (MAML) seeks to optimize domain generalization by simulating zero-shot learning during training through virtual source and target tasks (Finn et al., 2017; Li et al., 2018). However, little work has explored the effects of virtual task selection for MAML. In this work, we explore how redefining domains for source/target tasks and utilizing different task selection methods affects the meta-training process with respect to model performance. In particular, we test our methods in the realm of semantic parsing, which is the task of translating natural language utterances into formal meaning representations (e.g., logical forms, database queries, programs). Domain generalization in semantic parsing involves producing new programs from previously unseen natural language utterances stemming from new domains and databases. This is a topic of interest since gathering data to train semantic parsers is difficult and data during inference can be largely different from data available during training (Chang et al., 2020; Huang et al., 2018; Conklin et al., 2021; Chen et al., 2020).

We experiment with different strategies for sampling virtual tasks during meta-learning for Spider, a zero shot text-to-SQL semantic parsing dataset [1] (Yu et al., 2019). The Spider task is a particularly interesting setting to study meta-learning strategies in for a number of reasons. There is value in studying the effects of meta-learning in a challenging semantic parsing task where adaptation to unseen domains is an explicit goal inherent to the task, while simultaneously, semantic parsing tasks are relatively under-explored compared to image classification tasks where meta-learning is most commonly applied. Meanwhile, the task is relatively tractable despite its difficulty; the domains

[1] Our repo is based on Wang et al. (2021)'s implementation in `https://github.com/berlino/tensor2struct-public` and will be available at https://github.com/clarana/tensor2struct-meta-domain

Submitted to the 6th Workshop on Meta-Learning at NeurIPS 2022, New Orleans. Do not distribute.

in the dataset are well-defined (different SQL databases correspond to different domains), with an underlying similarity in desired logical form structure (i.e., acceptable model outputs are well-formed SQL queries which produce correct outputs when executed).

Similar to Wang et al. (2021), we create virtual zero-shot tasks during meta-training by sampling examples from source and target domains pre-defined for Spider. We experiment with alternative domain definitions based on different similarity scores between natural language text and its corresponding programs, and consider different source and target domain selection strategies during training in addition to a random baseline: namely choosing similar domains, different domains, increasingly different domains, or increasingly similar domains. We show that it is possible to achieve full benefits of DG-MAML while drastically reducing training time needed to achieve the results, or outperform baseline DG-MAML using our domain and task selection strategies. Other notable results include the capacity of "bad" virtual task sampling strategies to dramatically reduce task accuracy, and the stabilizing effect of DG-MAML versus a multitask learning baseline.

2 Methods

2.1 Redefining Domains

Text-to-SQL datasets contain pre-defined domains since queries tend to reference a single database (i.e., a database is a domain). In particular, the publicly available Spider dataset contains 166 databases including 146 for the training set and 20 for the validation set. However, samples from the same domain may not be homogeneous when considering text representations (see Figure 5). Since meta-learning assumes that samples in source and target domains are cohesive, we re-define data domains in the following ways:

Similarity of Question representation: the text-SQL pairs in the Spider training set are clustered into 220 new domains using KNN based on cosine similarity between the natural language question representations.
Similarity of SQL representation: the training samples are clustered into 220 new domains based on SQL similarity using the same method.
Similarity of Question-SQL representation similarity: domains are redefined by the degree to which a natural language text is similar to its SQL representation. We calculate the cosine similarity between a text embedding and its corresponding SQL embedding. The new domains are the cosine similarities defined by increments of 0.1. The re-defined domains represent degrees of semantic and syntactic similarity and serve as a fine-grained proxy for training example *difficulty*.

The above ways of redefining domains purely rely on the question or SQL representations, which ignores some orthogonal features in the table schemes. We also propose another naive deconvolution method which we deconvolve domain-specific and domain-agnostic representation (see A.1), making use of both the similarity of representation and the pre-defined domain information. Hence we also propose an alternative method below.

Similarity of Domain-Specific Question Representation: The text-SQL pairs in the training set are clustered into 136 new domains using KNN and cosine similarity between the deconvolved domain-specific question representations.

2.2 Virtual Task Selection

We explore task selection strategies to potentially improve meta-learning performance by: 1) Consistently selecting similar domains; 2) Consistently selecting distinct domains; 3) Selecting domains with decreasing similarity; 4) Selecting domains with increasing similarity, and; 5) Random selection.

To quantify similarity between domains, we calculate the cosine similarity matrix for each pair of domains, where the mean embedding for each domain is used. In choosing a target domain for a source domain $d \in D$ during meta-training, we retrieve the vector $\vec{s}$ of similarity between that d and all other domains $D - d$ from the cosine similarity matrix, which we use to obtain a probability

distribution under which we pick the target domains. Formally, in each step $i \in 1, ..., n$ where n is the maximum training steps, we calculate the target domain with the probability distribution, $P(t)$, as:

$$P(t) = \text{softmax}(\vec{s}^{\ r}) \quad \text{where}$$
$$r = \begin{cases} 1 & \text{if selection strategy 1 (constant rate)} \\ -1 & \text{if selection strategy 2 (constant rate)} \\ -\frac{2i}{n} + 1 & \text{if selection strategy 3 (linear rate)} \\ \frac{2i}{n} - 1 & \text{if selection strategy 4 (linear rate)} \end{cases} \tag{1}$$

3 Experiments

We experiment with RAT-SQL (Wang et al., 2019), a model with a relation-aware transformer encoder and a LSTM-based decoder, as our baseline model. We use similar configurations as Wang et al. (2021), including their batch size of $\mathcal{B}_s = \mathcal{B}_t = 12$ and their inner and outer learning rates of $\alpha = 5 \times 10^{-4}$ and 6×10^{-4}, where their outer learning rate schedule includes a 500 step warmup to 6×10^{-4} and a polynomial cooldown to their last step.

We run meta-training using the DG-MAML algorithm using combinations of our three domain definitions and our five virtual task selection methods. In addition, we proposed an alternative, stronger baseline of multi-task learning (Multi-task in Table 1), which allows us to implement task selection on top of regular supervised training. This allows for an additional fairer comparison between meta-learning and regular supervised training when comparing task selection strategies across models. Details of this model can be found in Appendix A. Select results are depicted in Table 1, with additional tables (A.2, 2) containing more complete results. We run a subset of experiments to 20k training steps and report those results in Table 3.

Model	Set Match Acc.	Exec. Acc.
DG-MAML Wang et al. (2021), 20k train steps	*68.9*	*69.3*
DG-FMAML Wang et al. (2021), 20k train steps	*N/A*	*N/A*
Regular	62.3	65.0
Multi-task + orig domain + .	64.5	66.6
DG-FMAML + orig domain + .	66.8	67.3
DG-MAML + orig domain + .	66.8	67.6
DG-FMAML + orig domain + different	67.5	68.8
DG-MAML + orig domain + similar	68.7	**69.2**
DG-MAML + question sim domains + different $\rightarrow$ similar	66.7	68.3
DG-MAML + question sim domains + similar $\rightarrow$ different	68.1	66.7
DG-MAML + domain-specific sim domains + different	**68.8**	68.3
DG-MAML + domain-specific sim domains + similar $\rightarrow$ different	68.7 / 67.3	68.6 / 68.1

Table 1: Evaluation accuracy % on baseline and DG-MAML model with 10k training iterations, except the first section, which contains lines *in italics* which are reported dev accuracies from Wang et al. (2021) (DG-FMAML accuracy is unreported for English-only Spider). The second section contains our own baseline metrics, and the third section contains our most promising models. Each model name consists of three parts: The first component is the learning scheme, the second is the domain definition scheme, and the third is the task selection scheme, where '.' indicates the default selection scheme – otherwise, similarity calculations among domains are based on example embeddings. Note that if task selection scheme is random, no similarity calculation is used.

4 Discussion

Our methods achieve varying levels of performance on the Spider task. Experiments that boost performance above the baseline at 10k steps in some cases approach the performance of Wang et al. (2021)'s reported results (68.9% and 69.3% accuracy for validation set exact set match and execution accuracy) at 20k steps, using the same batch sizes and other settings (Tables 1, 2). Additionally, we found that intentional task selection did not impact training time (the bottleneck is still by far the time required to compute first and second order gradients during meta-training), so halving the number of training steps also halves the wall clock time. Meanwhile, there are multiple methods that dramatically hurt accuracy compared to baselines, such as domain redefinition based on example-example text cosine similarity with a static sampling strategy (Table A.2 in Appendix), further confirming that task selection strategy does in fact affect model performance. Furthermore, models trained with multi-task learning or DG-FMAML often match or even exceed performance of DG-MAML models. However, meta-learning seems to encourage faster convergence (with respect to the number of steps) and may confer benefits in *stability* throughout individual training runs (Table A.2 in Appendix).

Our domain redefinition experiments are also instructive: When the domains are redefined by question embedding similarity, shifting from selecting similar to dissimilar domains or vice versa throughout training can improve the performance compared to using a consistent selection strategy. Similar-to-dissimilar sampling tends to outperform its inverse when the text-SQL similarity is high and worse when text-SQL similarity is low (Fig. 1 in Appendix). When using the question domain-specific representation as the redefined domains, selecting different samples constantly or shifting from similar to dissimilar domains boosts the performance.

When the domains are redefined by text-SQL similarity, selecting similar source and target domains during training results in higher accuracies when generating hard and extra hard queries compared to selecting dissimilar domains during training (Fig. 7, 8 in Appendix). Additionally, curriculum learning in this setting seems to equalize accuracies over the entire range of queries with different levels of text-SQL similarity, a proxy for example difficulty. Similar-to-dissimilar curriculum learning has a more prominent effect than dissimilar-to-similar curriculum learning (Fig. 3, Fig. 4 in Appendix).

4.1 Future work

Further "tuning" of models trained with redefined domain selection Despite meta-training being a strategy to aim domain generalization, there still may be inherently different distributions of data between training and evaluation or inference time. We can explore meta-training schemes that involve a final tuning of the model using originally defined domains, after meta-training using redefined domains.

Experimentation with alternative curricula In our experiments that involve curriculum learning, the rate in which domain similarity selection increases or decreases is linear with respect to the number of steps. However, it may be beneficial to explore non-linear rates or injecting outliers into the curriculum during training.

Beyond text-SQL Another potential future direction is to evaluate DG-MAML on other zero-shot tasks where domain is not as well-defined as the case where a domain is a SQL table. Our experimentation with redefined domains lays groundwork for domain selection in cases where domains may be defined in different ways – our results suggest that domain selection strategy does matter when meta-learning for domain generalization, and that different strategies may be beneficial depending on how domains are defined. A dynamic task selection strategy could be especially helpful to improve domain generalization in a model-agnostic, task-agnosic way.

4.2 Conclusion

Previous work suggests that a good "curriculum" in meta-learning is important (Zhan et al., 2021; Zhang et al., 2019). Having well-defined inherent domains for a challenging task, as they are in Spider (Yu et al., 2019), can help us ground experimentation and learn from and interpret results. This can in turn help us design human-interpretable curricula for meta-learning, potentially even for tasks beyond those specifically studied.

References

Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. 2018. Metareg: Towards domain generalization using meta-regularization. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Shuaichen Chang, Pengfei Liu, Yun Tang, Jing Huang, Xiaodong He, and Bowen Zhou. 2020. Zero-shot text-to-sql learning with auxiliary task. In *In The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*.

Xilun Chen, Asish Ghoshal, Yashar Mehdad, Luke Zettlemoyer, and Sonal Gupta. 2020. Low-resource domain adaptation for compositional task-oriented semantic parsing.

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.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks.

Po-Sen Huang, Chenglong Wang, Rishabh Singh, Wen-tau Yih, and Xiaodong He. 2018. Natural language to structured query generation via meta-learning.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy Hospedales. 2018. Learning to generalize: Meta-learning for domain generalization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Yiying Li, Yongxin Yang, Wei Zhou, and Timothy M. Hospedales. 2019. Feature-critic networks for heterogeneous domain generalization.

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. 2019. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers.

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. 2019. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task.

Runzhe Zhan, Xuebo Liu, Derek F Wong, and Lidia S Chao. 2021. Meta-curriculum learning for domain adaptation in neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14310–14318.

Xuan Zhang, Pamela Shapiro, Gaurav Kumar, Paul McNamee, Marine Carpuat, and Kevin Duh. 2019. Curriculum learning for domain adaptation in neural machine translation. 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 1903–1915, Minneapolis, Minnesota. Association for Computational Linguistics.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We mentioned what we can do for future work
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] We will release a public version of our repository along with camera-ready revisions. Our implementations are built on top of Wang et al. (2021)'s implementations at `https://github.com/berlino/tensor2struct-public`
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] in §3
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] When applicable (Table 3 in Appendix)
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] We did not strictly control for the hardware we used across all experiments, but our wall clock times for training were generally in line with those reported by Wang et al. (2021). Furthermore, our strategies theoretically halve the training time required to reach previously reported baseline performance given any adequate compute resources.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

Optionally include extra information (complete proofs, additional experiments and plots) in the appendix. This section will often be part of the supplemental material.

A.1 Domain-specific and domain-agnostic representation

For datasets with domain categorical information, after obtaining an initial embedding for data $X^{n\times k}$ and domain label $y \in \{1, \dots, C\}^n$, we form the question as follows:

For domain-specific embedding, we want to find a transformation that minimizes the in-domain variance. Without loss of generality, we assume that we want to find a one-dimensional representation. We define the mean of X as $\hat{X}$ such that

$$\hat{x_i} = \frac{1}{\sum_{j=1}^{n} \mathbf{1}_{\{y_j=y_i\}}} \sum_{j=1}^{n} \mathbf{1}_{\{y_j=y_i\}} x_j$$

We then define the objective as follows:

$$\min_p p^T (X - \hat{X})^T (X - \hat{X}) p, \text{where} \|p\|_2 = 1$$

Then for domain-agnostic (syntax-specific) embeddings, we want to find a transformation that maximizes the in-domain variance to account for the variations within the syntax for each domain. We then define the objective as follows:

$$\max_q q^T (X - \hat{X})^T (X - \hat{X}) q, \text{where} \|q\|_2 = 1$$

The dimension of the embedding can be more than $r = 1$.

Then the final transformation for domain-specific embeddings and domain-agnostic embeddings would be

$$\begin{aligned} z^{spe} &= Xp \\ z^{agn} &= Xq \end{aligned}$$

In theory, p and q should be the last column and first column of the eigenvector matrix in the eigendecomposition of $X - \hat{X}$.

A.2 Multi-task learning baseline

Our multi-task learning implementation is based on the DG-MAML implementation and less efficient than it can be, but it nevertheless cuts training time in half (38 hours vs 19 hours). For multi-task learning, we use:

$$\nabla_\theta \mathcal{L}_\mathcal{T}(\theta) = \nabla_\theta \mathcal{L}_{\mathcal{B}_t}(\theta) + \nabla_\theta \mathcal{L}_{\mathcal{B}_s}(\theta) \tag{2}$$

Furthermore, we implement DG-FMAML, the first order approximation of DG-MAML described but not provided by Wang et al. (2021) and run additional experiments using it to explore a computationally less intensive alternative to DG-MAML. In DG-FMAML, we use:

$$\begin{aligned} \nabla_\theta \mathcal{L}_\mathcal{T}(\theta) &= \nabla_\theta \theta' \nabla_{\theta'} \mathcal{L}_{\mathcal{B}_t}(\theta') + \nabla_\theta \mathcal{L}_{\mathcal{B}_s}(\theta) & (3) \\ &\approx \nabla_{\theta'} \mathcal{L}_{\mathcal{B}_t}(\theta') + \nabla_\theta \mathcal{L}_{\mathcal{B}_s}(\theta) & (4) \end{aligned}$$

Because our implementation adds gradients manually from a copy to the original, our DG-FMAML may require more memory at times, but training time is cut roughly in half from the DG-MAML implementation. In Table 1, we highlight our best performing models and compare them with each other as well as with Wang et al. (2021)'s reported results.

Learning algorithm	Domain Definition	Domain Selection Strategy	Set Match Acc.	Exec. Acc.
Regular supervised	N/A	N/A	62.3	65.0
Multi-task learning	SQL table	similar (table name)	64.5 / 64.9	66.6 / 66.7
		similar (text)	66.5	66.3 / 67.2
		different (text)	66.6	67.8
		similar → different (text)	67.7	66.8 / 66.9
		different → similar (text)	68.6 / 69.1	67.4
DG-FMAML	SQL table	similar (table name)	66.8	67.3
		random	67.2	68.3
		similar (text)	65.0	67.3
		different (text)	67.5 / 67.6	68.8
		similar → different (text)	66.6 / 66.7	67.8 / 68.4
		different → similar (text)	66.2	67.3.
DG-MAML	SQL table	similar (table name)	66.8	67.6
		random	67.1	67.1
		similar (text)	**68.7**	**69.2**
		different (text)	67.9	68.9
		similar → different (text)	67.6 / 68.2	67.9 / 68.1
		different → similar (text)	67.6	67.9 / 68.0
	example-example text cosine similarity	random	66.8	66.8
		similar (text)	64.0 / 65.4	64.6 / 66.2
		different (text)	65.1	66.2
		similar → different (text)	68.1	66.7
		different → similar (text)	66.7	68.3
	text-SQL cosine similarity	similar (text)	65.4	65.3
		different (text)	62.8	62.7 / 63.2
		similar → different (text)	65.1 / 64.9	64.9 / 65.2
		different → similar (text)	62.5 / 63.7	64.8 / 66.3

Learning algorithm	Domain Definition	Domain Selection Strategy	Set Match Acc.	Exec. Acc.
DG-MAML	example-example domain specific representation cosine similarity	random	67.3	67.4
		similar (text mean cosim)	66.3 / 66.3	67.0 / 66.5
		different (text mean cosim)	**68.8**	68.3
		similar → different (text mean cosim)	68.7 / 67.3	**68.6** / 68.1
		different → similar (text mean cosim)	66.7 / 65.7	66.8 / 66.7
	sql-sql domain specific representation cosine similarity	random	65.7 / 65.7	66.2 / 66.3
		similar (sql mean cosim)	65.7	66.2
		different (sql mean cosim)	65.7	66.7
		similar → different (sql mean cosim)	62.1	63.2
		different → similar (sql mean cosim)	61.3 / 61.9	62.3 / 62.7

Table 2: This description applies to both Table A.2 and Table 2. Validation set accuracy % on dev set with after 10k training iterations for all models trained. All models listed use a RAT-SQL base with BERT-base contextualized embeddings. Wang et al. (2021) define data domains by SQL table of origin (*SQL table*) and by default select domains weighted by cosine similarity of SQL table name embeddings so that domains with similar names are more likely to be selected for a task (*similar (table name cosim)*). We mark accuracy metrics at 'final checkpoint / best checkpoint' when the final checkpoint does not have the highest accuracy. Checkpoints were saved every 1000 steps starting from the 1000th step.

Model	Set Match Accuracy (10k → 20k steps)	Execution Accuracy (10k → 20k steps)
Regular Wang et al. (2021)	*N/A → 66.8*	*N/A → 66.8*
DG-MAML Wang et al. (2021)	*N/A → 68.9*	*N/A → 69.3*
Regular	N/A → 67.4 ± 0.8	N/A → 67.5 ± 1.1
Multi-task + orig domain + .	64.5 → 69.8 ± 1.6	66.6 → 69.3 ± 1.1
DG-FMAML + orig domain + .	66.8 → *69.9	67.3 → *69.2
DG-MAML + orig domain + .	66.8 → 68.2 ± 1.3	67.6 → 69.4 ± 1.3
DG-MAML + orig domain + similar	68.7 → 69.4 ± 1.0	69.2 → 69.7 ± 1.2
DG-MAML + orig domain + different	67.9 → 67.8 ± 0.4	68.9 → 68.4 ± 1.2

Table 3: Evaluation accuracy % on baseline and DG-MAML models trained for 20k iterations. Accuracies reached in (separate) training runs to 10k steps are included for comparison. Reported numbers for 20k step training runs are arithmetic means ($n = 3$) ± standard deviations, unless *otherwise specified. Comparison of 10k step accuracy with 20k step accuracy can hint at the rate of convergence using different training schemes. Although variability is high relative across random seeds, many of the differences we report, including the improvement in set match accuracy from selecting similar domains throughout training (vs baseline DG-MAML), are statistically significant (p-value < 0.05).

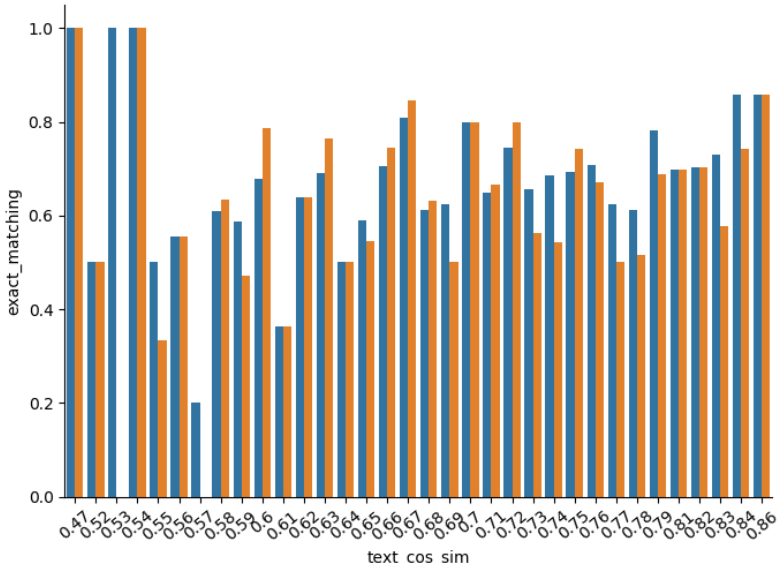

Figure 1: The accuracy among tasks with different text-SQL similarity using curriculum learning (blue) and reverse curriculum learning (orange) with redefined domains based on question embedding similarity

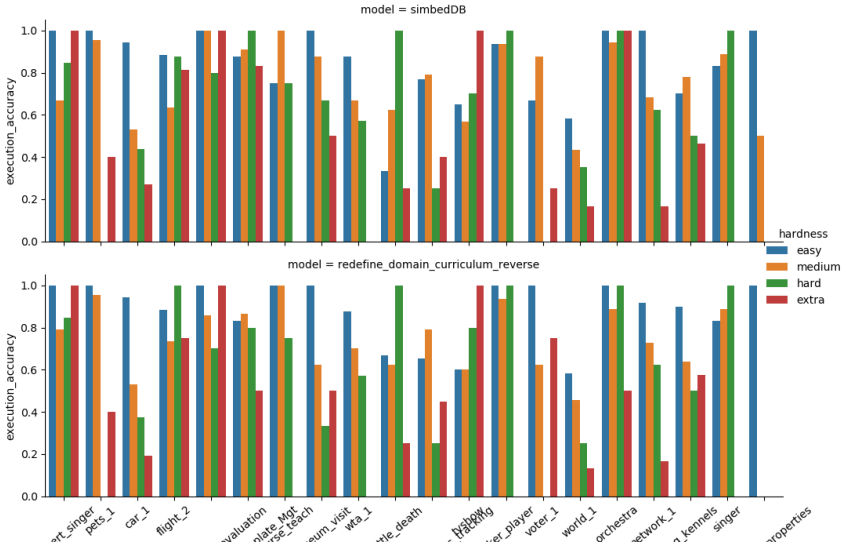

Figure 2: The execution accuracy of predicted SQL with consistently selecting similar domain based on mean question embedding similarity with original domain and reverse curriculum learning with redefined domains based on question embedding similarity

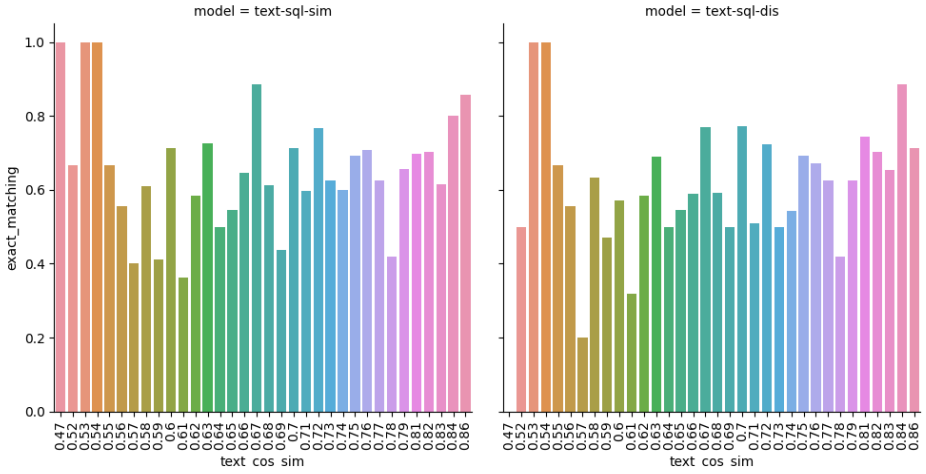

Figure 3: Accuracy among tasks with different cosine similarities between text and SQL. Source and target domains during training are redefined by cosine similarities between text and SQL. Left: similar domain are selected as source and target domains during training. Right: dissimlar domain are selected as source and target domains during training

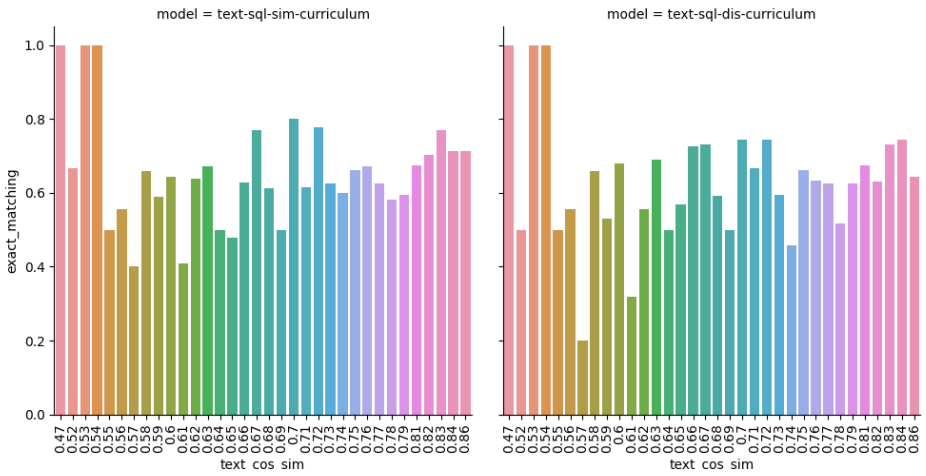

Figure 4: Accuracy among tasks with different cosine similarities between text and SQL. Source and target domains during training are redefined by cosine similarities between text and SQL. Left: curriculum learning is used (decreasing source and target domain similarity over time during training). Right: reverse curriculum learning is used (increasing source and target domain similarity over time during training)

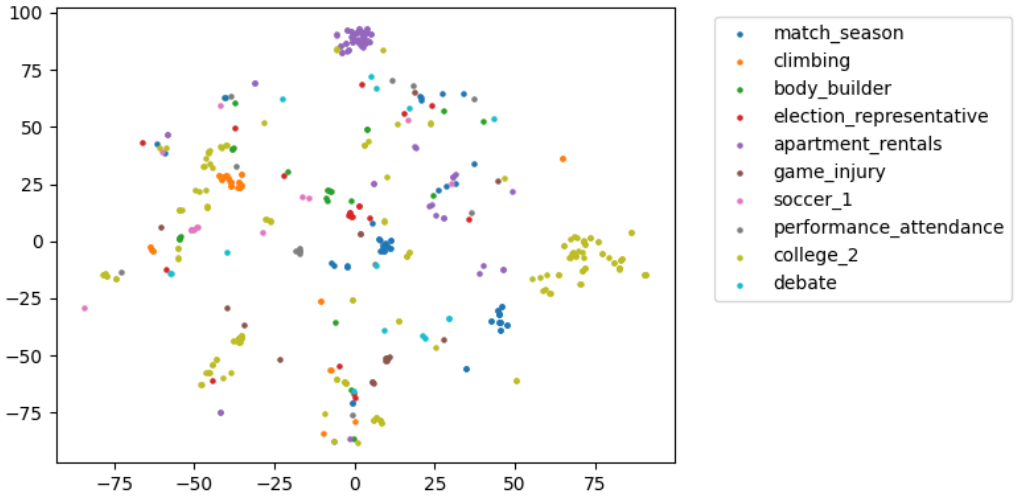

Figure 5: tSNE plot for sentence representation of questions from 10 example domains

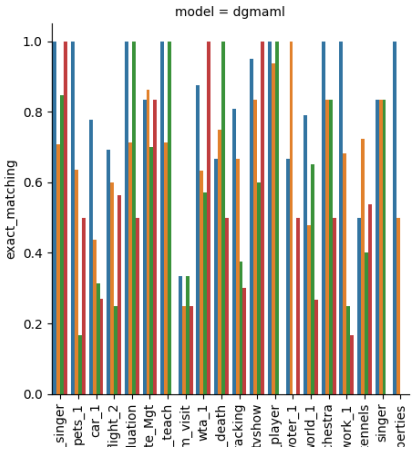

Figure 6: Accuracy among domains defined by hardness of SQL query using DGMAML

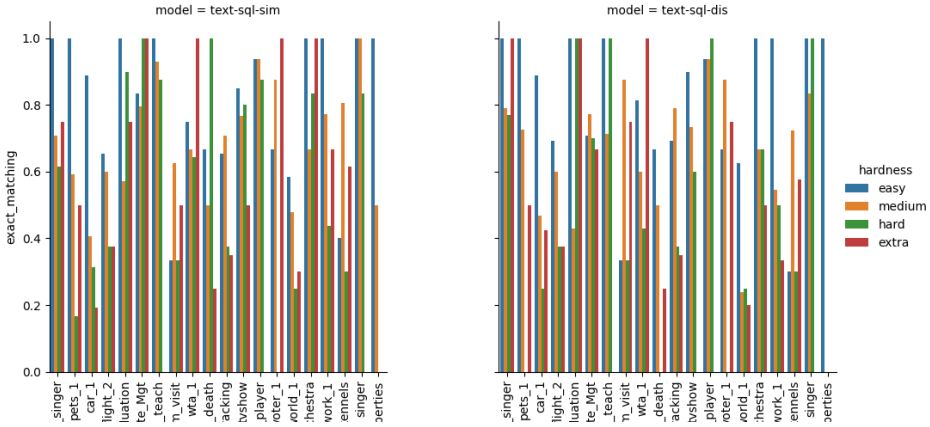

Figure 7: Accuracy among domains defined by hardness of SQL query. Source and target domains during training are redefined by cosine similarities between text and SQL. Left: similar domain are selected as source and target domains during training. Right: dissimlar domain are selected as source and target domains during training

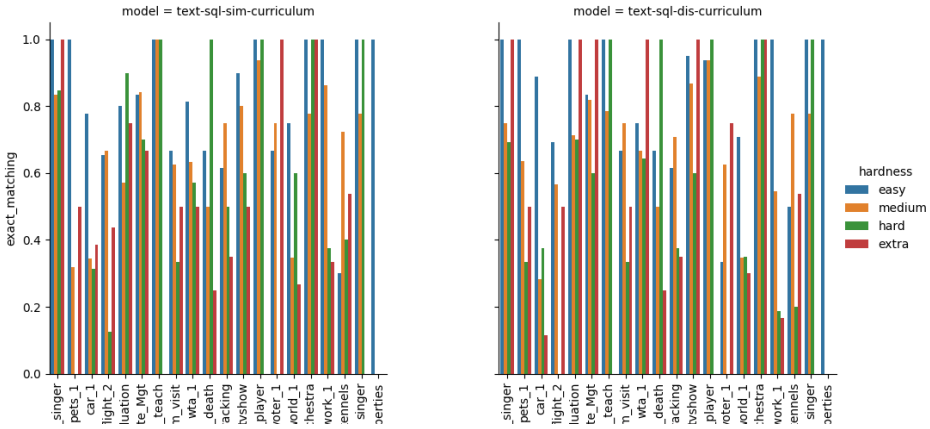

Figure 8: Accuracy among domains defined by hardness of SQL query. Source and target domains during training are redefined by cosine similarities between text and SQL. Left: curriculum learning is used (decreasing source and target domain similarity over time during training). Right: reverse curriculum learning is used (increasing source and target domain similarity over time during training)