Natural Language to Structured Query Generation via Meta-Learning

Po-Sen Huang [1] Chenglong Wang [2] Rishabh Singh [3] * Wen-tau Yih [4] Xiaodong He [5] *

1. Introduction

Conventional supervised training is a pervasive paradigm for NLP problems. In this setting, a model is trained to fit all the training examples and their corresponding targets. However, while sharing the same surface form of the prediction task, examples of the same problem may vary widely. For instance, *recognizing textual entailment* is a binary classification problem on whether the hypothesis follows a given textual statement, but the challenge datasets consist of a huge variety of inference categories and genres (Dagan et al., 2013; Williams et al., 2017). Similarly, for a semantic parsing problem that maps natural language questions to SQL statements, the number of conditions in a SQL query or the length of a question can vary substantially (Zhong et al., 2017).

The inherently high variety of the examples suggests an alternative training protocol: instead of learning a monolithic, one-size-fits-all model, it could be more effective to learn multiple models, where each one is designed for a specific "task" that covers a group of *similar* examples. However, this strategy is faced with at least two difficulties. As the number of tasks increases, each task will have much fewer training examples for learning a robust model. In addition, the notion of "task", namely the group of examples, is typically not available in the dataset.

In this work, we explore this alternative learning setting and address the two difficulties by adapting the *meta-learning* framework. Motivated by the few-shot learning scenario (Andrychowicz et al., 2016; Ravi & Larochelle, 2016; Vinyals et al., 2016), meta-learning aims to learn a general model that can quickly adapt to a new task given very few examples without retraining the model from scratch (Finn et al., 2017). We extend this framework by effectively creating *pseudo-tasks* with the help of a *relevance function*. During training, each example is viewed as the test example of an individual "task", where its top-K relevant instances are used as training examples for this specific task. A general model is trained for all tasks in aggregation. Similarly during testing, instead of applying the general model directly, the top-K relevant instances (in the training set) to the given test example are first selected to update the general model, which then makes the final prediction. The overview of the proposed framework is shown in Figure 1.

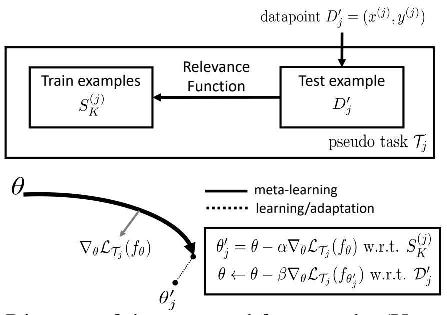
datapoint $D'_j = (x^{(j)}, y^{(j)})$
Train examples $S_K^{(j)}$
Relevance Function
Test example D'_j
pseudo task $\mathcal{T}_j$
θ
meta-learning
learning/adaptation
$\nabla_\theta \mathcal{L}_{\mathcal{T}_j}(f_\theta)$
$\theta'_j = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_j}(f_\theta)$ w.r.t. $S_K^{(j)}$
$\theta \leftarrow \theta - \beta \nabla_\theta \mathcal{L}_{\mathcal{T}_j}(f_{\theta'_j})$ w.r.t. $\mathcal{D}'_j$
θ'_j

Figure 1. Diagram of the proposed framework. (Upper) we propose using a relevant function to find a support set $S_K^{(j)}$ from all training datapoints given a datapoint D'_j for constructing a pseudo-task $\mathcal{T}_j$ as in the few-shot meta-learning setup. (Bottom) We optimize the model parameters θ such that the model can learn to adapt a new task with parameters θ'_j via a few gradient steps on the training examples of the new task. The model is updated by considering the test error on the test example of the new task.

When empirically evaluated on a recently proposed, large semantic parsing dataset, WikiSQL (Zhong et al., 2017), our approach leads to faster convergence and achieves 1.1%–5.4% absolute accuracy gain over the non-meta-learning counterparts, establishing a new state-of-the-art result. More importantly, we demonstrate how to design a relevance function to successfully reduce a regular supervised learning problem to a meta-learning problem. To the best of our knowledge, this is the first successful attempt in adapting meta-learning to a semantic task.

2. Approach

In this section, we first describe the design of our relevance function and then the complete algorithm.

2.1. Relevance Function

The intuition behind the design of a relevance function is that examples of the same type should have higher scores.

*Work performed at Microsoft Research. [1]Microsoft Research [2]University of Washington [3]Google Brain [4]Allen Institute for Artificial Intelligence [5]JD AI Research. Correspondence to: Po-Sen Huang <pshuang@microsoft.com>.

Published at the ICML workshop Neural Abstract Machines & Program Induction v2 (NAMPI) — Extended Abstract, Stockholm, Sweden, 2018. Copyright 2018 by the author(s).

For the questions to SQL problem, we design a simple relevance function that depends on (1) the predicted type of the corresponding SQL query and (2) the question length.

There are five SQL types in the WikiSQL dataset: {`Count`, `Min`, `Max`, `Sum`, `Avg`, `Select`}. We train a SQL type classifier f_{sql} using SVMs with bag-of-words features of the input question, which achieves 93.5% training accuracy and 88.0% test accuracy in SQL type prediction. Another soft indication on whether two questions can be viewed as belonging to the same "task" is their lengths, as they correlate to the lengths of the mapped SQL queries. The length of a question is the number of tokens in it after normalizing entity mentions to single tokens.[1] Our relevance function only considers examples of the same predicted SQL types. If examples $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ have the same SQL type, then their relevance score is $1 - |q_{len}(\mathbf{x}^{(i)}) - q_{len}(\mathbf{x}^{(j)})|$, where q_{len} calculates the question length. Notice that the relevance function does not need to be highly accurate as there is no formal definition on which examples should be grouped in the same pseudo-task. A heuristic-based function that encodes some domain knowledge typically works well based on our preliminary study. In principle, the relevance function can also be jointly learned with the meta-learning model, which we leave for future work.

2.2. Algorithm

Algorithm 1 summarizes Pseudo-Task MAML (PT-MAML), which adapts the MAML meta-learning framework to use a relevance function. For each training input $\mathbf{x}^{(j)}$ and target $\mathbf{y}^{(j)}$, we create a pseudo-task $\mathcal{T}_j$ using the top-K relevant examples as the support set $\mathcal{S}_K^{(j)}$ (Step 1). The step size α and the meta step size β are hyper-parameters. $\mathcal{L}_{\mathcal{T}_j}(f_\theta)$ is a loss function that evaluates the error between the prediction $f_\theta(\mathbf{x}^{(i)})$ and target $\mathbf{y}^{(i)}$, where $\mathbf{x}^{(i)}, \mathbf{y}^{(i)}$ are an input/output pair from task $\mathcal{T}_j$. The remaining steps of the algorithm mimic the original MAML design (Finn et al., 2017), which updates task-level models (Step 8) and the meta-level general model (Step 10) using gradient descent.

Algorithm 1 Pseudo-Task MAML (PT-MAML)

```
Require: Training Datapoints $\mathcal{D} = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$
Require: $\alpha, \beta$: step size hyperparameters
Require: $K$: support set size hyperparameter
 1: Construct a task $\mathcal{T}_j$ with training examples using a support
    set $\mathcal{S}_K^{(j)}$ and a test example $\mathcal{D}'_j = (\mathbf{x}^{(j)}, \mathbf{y}^{(j)})$.
 2: Denote $p(\mathcal{T})$ as distribution over tasks
 3: Randomly initialize $\theta$
 4: while not done do
 5:   Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
 6:   for all $\mathcal{T}_i$ do
 7:     Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{S}_K^{(j)}$
 8:     Compute adapted parameters with gradient descent:
        $\theta'_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
 9:   end for
10:   Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i})$ using each $\mathcal{D}'_i$
      from $\mathcal{T}_i$ and $\mathcal{L}_{\mathcal{T}_i}$ for the meta-update
11: end while
```

Model	Dev		Test	
	Acc_{lf}	Acc_{ex}	Acc_{lf}	Acc_{ex}
PointerNet (Zhong et al., 2017)	44.1%	53.8%	43.3%	53.3%
Seq2SQL (Zhong et al., 2017)	49.5%	60.8%	48.3%	59.4%
Pointer loss (Wang et al., 2017)	46.8%	52.1%	46.1%	51.8%
Meta + Pointer loss	**52.0%**	**57.7%**	**51.4%**	**57.2%**
Max loss (Wang et al., 2017)	61.3%	66.9%	60.5%	65.8%
Meta + Max loss	**62.1%**	**67.3%**	**61.6%**	**67.0%**
Sum loss (Wang et al., 2017)	62.0%	67.1%	61.5%	66.8%
Meta + Sum loss	**63.1%**	**68.3%**	**62.8%**	**68.0%**

Table 1. Experimental Results on the WikiSQL dataset, where Acc_{lf} represents the logical form accuracy and Acc_{ex} represents the SQL execution accuracy. "Pointer loss", "Max loss", and "Sum loss" are the non-meta-learning counterpart from Wang et al. (2017). "Meta + X" denotes the meta-learning model with learner "X".

3. Experiments

Wang et al. (2017) propose three cross-entropy based loss functions: "Pointer loss", which is the cross-entropy between target index and the chosen index, "Max loss", which computes the probability of copying a token v in the input as the maximum probability of pointers that point to token v, and "Sum loss", which computes the probability of copying a token v in the input as the sum of probabilities of pointers that point to token v. See Wang et al. (2017) for more detail.

Table 1 shows the experimental results of our model on the WikiSQL dataset. We select the model based on the best logical form accuracy on the development set, and compare our results to augmented pointer network and the Seq2SQL model (with RL) in (Zhong et al., 2017). Both logical form accuracy (denoted by Acc_{lf}) that compares the exact SQL syntax match, and the SQL execution results (denoted by Acc_{ex}) are reported. We compare our approach with its non-meta-learning counterpart using "Pointer loss", "Max loss", and "Sum loss" losses from (Wang et al., 2017). Our model achieves 1.1%–5.3% and 1.2%–5.4% gains on the test set logical form and execution accuracy, respectively.

4. Conclusion

In this paper, we propose a new learning protocol that reduces a regular supervised learning problem to the few-shot meta-learning scenario. This is done by effectively creating *pseudo-tasks* with the help of a *relevance function*. When evaluated on the newly released, large semantic parsing dataset, WikiSQL, our approach leads to faster convergence and enjoys 1.1%–5.4% absolute accuracy gains over the non-meta-learning counterparts, achieving a new state-of-the-art result.

[1]Phrases in questions that can match some table cells are treated as entities.

References

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., and de Freitas, N. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pp. 3981–3989, 2016.

Dagan, I., Roth, D., Sammons, M., and Zanzotto, F. M. *Recognizing Textual Entailment: Models and Applications*. Morgan and Claypool, 2013. ISBN 978-1-59829-834-5.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

Ravi, S. and Larochelle, H. Optimization as a model for few-shot learning. In *International Conference on Learning Representations (ICLR)*, 2016.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pp. 3630–3638, 2016.

Wang, C., Brockschmidt, M., and Singh, R. Pointing out SQL queries from text. Technical Report MSR-TR-2017-45, November 2017. URL https://www.microsoft.com/en-us/research/publication/pointing-sql-queries-text/.

Williams, A., Nangia, N., and Bowman, S. R. A broad-coverage challenge corpus for sentence understanding through inference. *CoRR*, abs/1704.05426, 2017.

Zhong, V., Xiong, C., and Socher, R. Seq2SQL: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103, 2017.