Natural Language to Structured Query Generation via Meta-Learning

Po-Sen Huang[⋆], **Chenglong Wang**[†], **Rishabh Singh**[⋆], **Wen-tau Yih**[‡], **Xiaodong He**[*◇]
[⋆]Microsoft Research, [†]University of Washington,
[‡]Allen Institute for Artificial Intelligence, [◇]JD AI Research
pshuang@microsoft.com, clwang@cs.washington.edu,
rishabh.iit@gmail.com, scottyih@allenai.org, xiaodong.he@jd.com

Abstract

In conventional supervised training, a model is trained to fit all the training examples. However, having a monolithic model may not always be the best strategy, as examples could vary widely. In this work, we explore a different learning protocol that treats each example as a unique *pseudo-task*, by reducing the original learning problem to a few-shot meta-learning scenario with the help of a domain-dependent *relevance function*. When evaluated on the WikiSQL dataset, our approach leads to faster convergence and achieves 1.1%–5.4% absolute accuracy gains over the non-meta-learning counterparts.

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 and 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.

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.

[*]Work performed while XH was at Microsoft Research.

Proceedings of NAACL-HLT 2018, pages 732–738
New Orleans, Louisiana, June 1 - 6, 2018.

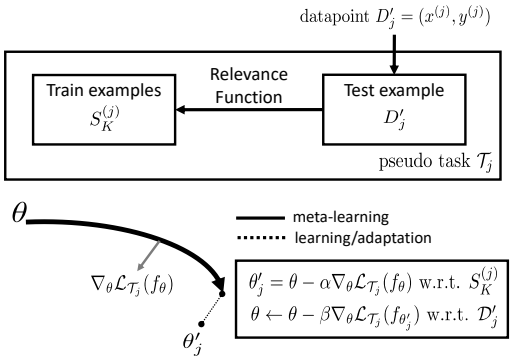

Figure 1: Diagram of the proposed framework. (Upper) we propose using a relevant function to find a support set $S_K^{(j)}$ from all training datapoints given a datapoint D'_j for constructing a pseudo-task $\mathcal{T}_j$ as in the few-shot meta-learning setup. (Bottom) We optimize the model parameters 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. See Section 2 for detail.

2 Background: Meta-Learning

Our work is built on the recently proposed Model-Agnostic Meta-Learning (MAML) framework (Finn et al., 2017), which we describe briefly here. MAML aims to learn the *learners* (for the tasks) and the *meta-learner* in the few-shot meta-learning setup (Vinyals et al., 2016; Andrychowicz et al., 2016; Ravi and Larochelle, 2016). Formally, it considers a model that is represented by a function f_θ with parameters θ. When the model adapts to a new task $\mathcal{T}_i$, the model changes parameters θ to θ'_i, where a task contains K training examples and one or more test examples (K-shot learning). MAML updates the parameters θ'_i by one or a few rounds of gradient descent based on the training examples of task $\mathcal{T}_i$. For example, with one gradient update,

$$\theta'_i = \theta - \alpha\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta),$$

where the step size α is a hyper-parameter; $\mathcal{L}_{\mathcal{T}_i}(f_\theta)$ is a loss function that evaluates the error between the prediction $f_\theta(\mathbf{x}^{(j)})$ and target $\mathbf{y}^{(j)}$, where $\mathbf{x}^{(j)}, \mathbf{y}^{(j)}$ are an input/output pair sampled from the training examples of task $\mathcal{T}_i$. Model parameters θ are trained to optimize the performance of $f_{\theta'_i}$ on the unseen test examples from $\mathcal{T}_i$ across tasks. The meta-objective is:

$$\min_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i}) = \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta - \alpha\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)})$$

The goal of MAML is to optimize the model parameters θ such that the model can learn to adapt new tasks with parameters θ'_i via a few gradient steps on the training examples of new tasks. The model is improved by considering how the test error on unseen test data from $\mathcal{T}_i$ changes with respect to the parameters.

The meta-objective across tasks is optimized using stochastic gradient descent (SGD). The model parameters θ are updated as follows:

$$\theta \leftarrow \theta - \beta\nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i}),$$

where β is the meta step size.

3 Approach

As discussed in Section 1, to reduce traditional supervised learning to a few-shot meta-learning problem, we introduce a *relevance function*, which effectively helps group examples to form *pseudo-tasks*. Because the relevance function is problem-dependent, we first describe the semantic parsing problem below, followed by the design of our relevance function and the complete algorithm.

3.1 The Semantic Parsing Task

The specific semantic parsing problem we study in this work is to map a natural language question to a SQL query, which can be executed against a given table to find the answer to the original question. In particular, we use the currently largest natural language questions to SQL dataset, WikiSQL (Zhong et al., 2017), to develop our model and to conduct the experiments.

3.2 Relevance Function

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

Algorithm 1 Pseudo-Task MAML (PT-MAML)

Require: Training Datapoints $\mathcal{D} = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$
Require: α, β: 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 θ
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**

izing 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.

3.3 Algorithm

Given a relevance function, the adaptation of the meta-learning using the MAML framework can be summarized in Algorithm 1, called Pseudo-Task MAML (PT-MAML). For each training example $\mathbf{x}^{(j)}$, 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 remaining steps of the algorithm mimics the original MAML design, update task-level models (Step 8) and the meta-level, general model (Step 10) using gradient descent.

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

4 Experiments

In this section, we introduce the WikiSQL dataset and preprocessing steps, the learner model in our meta-learning setup, and the experimental results.

4.1 Dataset

We evaluate our model on the WikiSQL dataset (Zhong et al., 2017). We follow the data preprocessing in (Wang et al., 2017). Specifically, we first preprocess the dataset by running both tables and question-query pairs through Stanford Stanza (Manning et al., 2014) using the script included with the WikiSQL dataset, which normalizes punctuations and cases of the dataset. We further normalize each question based on its corresponding table: for table entries and columns occurring in questions or queries, we normalize their format to be consistent with the table. After preprocessing, we filter the training set by removing pairs whose ground truth solution contains constants not mentioned in the question, as our model requires the constants to be copied from the question. We train and tune our model only on the filtered training and filtered development set, but we report our evaluation on the full development and test sets. We obtain 59,845 (originally 61,297) training pairs, 8,928 (originally 9,145) development pairs and 17,283 test pairs (the test set is not filtered).

4.2 Learner Model

We use the model of Wang et al. (2017) as the *learner* in our meta-learning setup. The model is a grammar-aware Seq2Seq encoder-decoder model with attention (Cho et al., 2014; Bahdanau et al., 2014). The encoder is a bidirectional LSTM, which takes the concatenation of the table header (column names) of the queried table and the question as input to learn a joint representation. The decoder is another LSTM with attention mechanism. There are three output layers corresponding to three decoding types, which restricts the vocabulary it can sample from at each decoding step. The three decoding types are defined as follows:

- τ_V (SQL operator): The output has to be a SQL operator, i.e., a terminal from $V = \{$`Select`, `From`, `Where`, `Id`, `Max`, `Min`, `Count`, `Sum`, `Avg`, `And`, $=$, $>$, $\geq$, $<$, $\leq$, `<END>`, `<GO>`$\}$.
- τ_C (column name): The output has to be a column name, which will be copied from either the table header or the query section of

Model	Dev		Test	
	Acc_{lf}	Acc_{ex}	Acc_{lf}	Acc_{ex}
PointerNet (2017)	44.1%	53.8%	43.3%	53.3%
Seq2SQL (2017)	49.5%	60.8%	48.3%	59.4%
Pointer loss (2017)	46.8%	52.1%	46.1%	51.8%
Meta + Pointer loss	**52.0%**	**57.7%**	**51.4%**	**57.2%**
Max loss (2017)	61.3%	66.9%	60.5%	65.8%
Meta + Max loss	**62.1%**	**67.3%**	**61.6%**	**67.0%**
Sum loss (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".

the input sequence. Note that the column required for the correct SQL output may or may not be mentioned explicitly in the question.

- τ_Q (constant value): The output is a constant that would be copied from the question section of the input sequence.

The grammar of SQL expressions in the the WikiSQL dataset can be described in regular expression as "`Select` f c `From` t `Where` (c op v)*" (f refers to an aggregation function, c refers to a column name, t refers to the table name, op refers an comparator and v refers to a value). The form can be represented by a decoding-type sequence $\tau_V\tau_V\tau_C\tau_V\tau_C\tau_V(\tau_C\tau_V\tau_Q)^*$, which will ensure only decoding-type corrected tokens can be sampled at each decoding step.

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.

4.3 Model Hyperparameters

We use the pre-trained n-gram embeddings by Hashimoto et al. (2017) (100 dimension) and the GloVe word embedding (100 dimension) by Pennington et al. (2014); each token is embedded into a 200 dimensional vector. The encoder is a 3-layer bidirectional LSTM with hidden states of size 100, and the decoder is a 3-layer unidirectional LSTM with hidden states of size 100. The model is trained with question-query pairs with a batch size of 200 for 100 epochs. During training, we clip gradients at 5 and add gradient noise with $\eta = 0.3$, $\gamma = 0.55$ to stabilize training (Neelakantan et al., 2015). We found the meta-learning model is trained stably without back-propagating to second order gradients. We select the support set size K to be 2 based on the development set. Empirically, the performance does not improve when we use a larger K. We set the learning rates $\alpha = 0.001$ and $\beta = 0.1$ based on the development set. The model is implemented in Tensorflow and trained using Adagrad (Duchi et al., 2011).

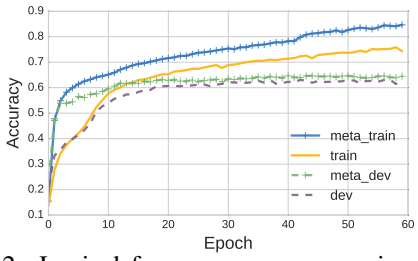

Figure 2: Logical form accuracy comparison, where "meta_train" and "meta_dev" are the train and development set accuracy using the "Meta + Sum loss" model, "train" and "dev" are the train and development set accuracy using the "Sum loss" model (Wang et al., 2017).

4.4 Results

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.

We also investigate the training and development set logical form accuracy over different epochs by "Meta + Sum loss" and "Sum loss" models. The results are shown in Figure 2. One interesting observation is that the "Meta + Sum loss" model converges much faster than the "Sum loss" model especially in the first 10 epochs. We attribute this improvement to the ability to adapt to new tasks even with a small number of training examples.

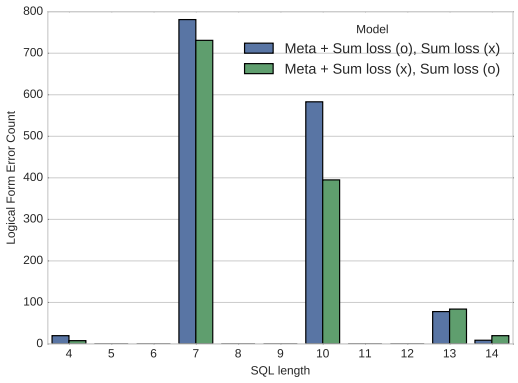

Figure 3: Logical form accuracy comparison. "Meta + Sum loss (o), Sum loss (x)" indicates the generated SQL is incorrect by the "Sum loss" model and is correct by the "Meta + Sum loss" model. Similarly, "Meta + Sum loss (x), Sum loss (o)" indicates the generated SQL is incorrect by the "Meta + Sum loss" model and is correct by the "Sum loss" model.

We compare the logical form error on the test set between the "Sum loss" model (Wang et al., 2017) and the proposed "Meta + Sum loss" model. Among the 17,283 test examples, there are 6,661 and 6,428 errors by the "Sum loss" and the "Meta + Sum loss", respectively. There are 5,190 common errors by both models. We examine the test examples where "Sum loss" is correct while "Meta + Sum loss" is not and vice versa, shown in Figure 3. We observe that the differences are mainly in ground truth SQL length = 7 and 10, where the "Meta + Sum loss" model outperforms "Sum loss" model by a large margin. We show some examples for the two cases in the supplementary material.

5 Related Work

Meta Learning One popular direction of meta-learning (Thrun and Pratt, 1998; Schmidhuber, 1987; Naik and Mammone, 1992) is to train a meta-learner that learns how to update the parameters of the learners model (Bengio et al., 1992; Schmidhuber, 1992). This direction has been applied to learning to optimize deep neural networks (Hochreiter et al., 2001; Andrychowicz et al., 2016; Li and Malik, 2017; Ha et al., 2017). Few-shot learning methods have also adapted meta-learning approaches for image recognition (Koch, 2015; Ravi and Larochelle, 2016; Vinyals et al., 2016) and reinforcement learning (Finn et al., 2017). Given that the few-shot learning setup cannot directly work in standard supervised learning problems, we explore reducing a regular supervised learning problem to the few-shot meta-learning scenario by creating pseudo-tasks with a relevance function.

Semantic Parsing Mapping natural language to logic forms has been actively studied in natural language processing research (Zettlemoyer and Collins, 2005; Giordani and Moschitti, 2010; Artzi and Zettlemoyer, 2011; Berant et al., 2013; Vlachos and Clark, 2014; Yih et al., 2014, 2015; Wang et al., 2015; Golub and He, 2016; Iyer et al., 2017; Krishnamurthy et al., 2017). However, unlike conventional approaches, which fit one model for all training examples, the proposed approach learns to adapt to new tasks. By using the support set based on the relevance function, the proposed model can adapt to a unique model for each example.

Program Induction / Synthesis Program induction (Reed and De Freitas, 2016; Neelakantan et al., 2015; Graves et al., 2014; Yin et al., 2015; Devlin et al., 2017) aims to infer latent programs given input/output examples, while program synthesis models (Zhong et al., 2017; Parisotto et al., 2017) aim to generate explicit programs and then execute them to get output. The learner model we used in this work follows the line of program synthesis models and trains on pairs of natural language (question) and program (SQL) directly.

6 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.

While the initial finding is encouraging, we believe the potential of this meta-learning framework has not yet been fully realized. In the future, we plan to explore more variations of the meta-learning setup, such as using different relevance functions, including the ones that are jointly learned. We also would like to understand this approach better by testing it on more natural language processing tasks.

Acknowledgments

We thank Chelsea Finn and Eugene Brevdo for helpful discussions in meta-learning, and Adith Swaminathan, Asli Celikyilmaz, and anonymous reviewers for their valuable feedback.

References

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. 2016. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*. pages 3981–3989.

Yoav Artzi and Luke Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. pages 421–432.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Samy Bengio, Yoshua Bengio, Jocelyn Cloutier, and Jan Gecsei. 1992. On the optimization of a synaptic learning rule. In *Preprints Conf. Optimality in Artificial and Biological Neural Networks*. Univ. of Texas, pages 6–8.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. *Recognizing Textual Entailment: Models and Applications*. Morgan and Claypool.

Jacob Devlin, Rudy R. Bunel, Rishabh Singh, Matthew J. Hausknecht, and Pushmeet Kohli. 2017. Neural program meta-induction. In *NIPS*. pages 2077–2085.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12(Jul):2121–2159.

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

Alessandra Giordani and Alessandro Moschitti. 2010. Semantic mapping between natural language questions and SQL queries via syntactic pairing. In *Proceedings of the 14th International Conference on Applications of Natural Language to Information Systems*. pages 207–221.

David Golub and Xiaodong He. 2016. Character-level question answering with attention. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural Turing machines. *arXiv preprint arXiv:1410.5401* .

David Ha, Andrew Dai, and Quoc V Le. 2017. Hypernetworks. In *International Conference on Learning Representations*.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Sepp Hochreiter, A Steven Younger, and Peter R Conwell. 2001. Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*. pages 87–94.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. *arXiv preprint arXiv:1704.08760* .

Gregory Koch. 2015. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 1517–1527.

Ke Li and Jitendra Malik. 2017. Learning to optimize neural nets. In *International Conference on Learning Representations*.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*.

Devang K Naik and RJ Mammone. 1992. Meta-neural networks that learn by learning. In *International Joint Conference on Neural Networks*. IEEE, volume 1, pages 437–442.

Arvind Neelakantan, Luke Vilnis, Quoc V Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. 2015. Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807* .

Emilio Parisotto, Abdel-rahman Mohamed, Rishabh Singh, Lihong Li, Dengyong Zhou, and Pushmeet Kohli. 2017. Neuro-symbolic program synthesis. In *International Conference on Learning Representations (ICLR)*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543.

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

Scott Reed and Nando De Freitas. 2016. Neural programmer-interpreters. In *International Conference on Learning Representations (ICLR)*.

Jurgen Schmidhuber. 1987. *Evolutionary Principles in Self-Referential Learning. On Learning now to Learn: The Meta-Meta-Meta...-Hook*. Diploma thesis, Technische Universitat Munchen, Germany.

Jürgen Schmidhuber. 1992. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation* 4(1):131–139.

Sebastian Thrun and Lorien Pratt, editors. 1998. *Learning to Learn*. Kluwer Academic Publishers.

Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*. pages 3630–3638.

Andreas Vlachos and Stephen Clark. 2014. A new corpus and imitation learning framework for context-dependent semantic parsing. *Transactions of the Association for Computational Linguistics* 2:547–559.

Chenglong Wang, Marc Brockschmidt, and Rishabh Singh. 2017. Pointing out SQL queries from text. Technical Report MSR-TR-2017-45. https://www.microsoft.com/en-us/research/publication/pointing-sql-queries-text/.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Association for Computational Linguistics (ACL)*.

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

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL-IJCNLP*.

Wen-tau Yih, Xiaodong He, and Chris Meek. 2014. Semantic parsing for single-relation question answering. In *ACL*.

Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2015. Neural enquirer: Learning to query tables with natural language. *arXiv preprint arXiv:1512.00965* .

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*. pages 658–666.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating structured queries from natural language using reinforcement learning. *CoRR* abs/1709.00103.