Better Learning with Lesser Data: Meta-Learning with DistilBERT

Stanford CS224N Default Project

Cheuk Hei Yuen
Stanford University
adriany@stanford.edu

Simeng Qu
Stanford University
simonequ@stanford.edu

Linxing Yao
Stanford University
linxingy@stanford.edu

Abstract

While pre-trained transformer models have shown great success in the question answering (QA) tasks, it requires a large amount of task-specific data to finetune. In this paper, we explore three variants of the model-agnostic meta-learning (MAML) algorithm in an attempt to improve the performance on the low resource QA task. To evaluate the effectiveness of this method, we compare its performance against DistilBERT model with and without multi-task learning. We find that the best performing method is MAML with Reptile Meta-Learner, which results in a 2.9% increase in the F1 score. Among the various properties of MAML we have studied, we have carried out a time-cost analysis and expressed our worry that the accuracy gain by MAML in QA task might not justify its increase in runtime.

1 Introduction

Since the advent of transformer, pre-trained transformer models have shown enormous success in improving performance on several downstream tasks, including question answering. However, fine-tuning on a new task still requires large amounts of task-specific labeled data to achieve good performance. This perennial problem sets up the stage of our study, through which we will explore different methods to build Question-Answering (QA) models that are robust against domain shift.

In this project, out main goal is to build a QA model that achieves better performance than the provided DistilBERT model (baseline) for a range of out-of-domain tasks given limited out-of-domain training examples. Our approach is to incorporate the model-agnostic meta-learning (MAML) framework with the DistilBERT model. MAML, first proposed by Finn et al [1], is an optimization-based strategy that aims at locating a good model initialization that can be used to finetune on any task with minimal training iterations and samples. We experiment two variants of the MAML algorithm at the meta-update step, Reptile and First-Order MAML. To evaluate the effectiveness of our method, we first compare the performance of our MAML-DistilBERT algorithm against the baseline for the in-domain training. We then experiment with two different methods during the learning process of the out-of-domain tasks, MAML and multi-task (MT) learning. Our experiments showed that MAML combined with DistiBERT outperforms the baseline DistiBERT in low-resource scenarios of QA tasks.

The rest of this report is organized as follows. Section 2 gives an overview of MAML and related works. Section 3 explains the architecture of the MAML-DistilBERT algorithm we constructed, and Section 4 shows variants of experiment we run for tackling the low-resource out-of-domain tasks, followed by the results measured by F1 and EM scores. In Section 5, we carry out various quantitative analyses to demonstrate why MAML could achieve better performance.

2 Related Work

In 2017, Chelsea et al. [2] proposes a new meta-learning algorithm that is model agnostic, meaning that it can be applied on top of any model that is trained with gradient descent. The training protocol of MAML proposed by Chelsea is a two-part process, starting with a meta-learning process in where we change a meta-model with many tasks forcing that model to learn with a small number of gradient descents. Such meta-model is then finetuned on the downstream tasks of interest. When the MAML model is trained properly, the initialization of the model should be good enough to adapt to any task with minimal training samples. The paper demonstrates MAML's ability of learning with a limited data in visual recognition and reinforcement learning problems.

In the following year, Alex et al. presents Reptile, a variant of MAML, with theoretical support of the algorithm, and compares MAML with multi-task learning, both of which expose the model to all available tasks and allow for transfer learning. The success of MAML is rooted in the uses the k^{th} gradient descents to update the meta-model. This allows the use of the higher-order terms of the loss function in task L_T. Essentially, Alex et al. shows that the update made by taking an average of a collection of tasks after k descent is different from straightly marching through k batches of task with descent in each iteration. The equation below can illustrate the Jensen's inequality (i.e. the LHS represents the SGD process in MAML, while RHS is the normal training process) that is true for k > 1.

$$E[SGD(L_T, \theta, k)] \neq SGD(E[L_T], \theta, k) \tag{1}$$

This MAML algorithm is later adapted into NLP task by Dou et al.[3] with different variants of MetaLearner experimented, including the original MAML, first-order MAML and the Reptile. In his study, promising results are shown with MAML, which set the precedent of the work we carry out in this paper.

3 Approach

3.1 Baselines

Our baseline model is the distilled version of BERT with pretrained parameters first introduced by Sanh et al [2]. This model is trained on three different QA datasets with 50,000 examples each to accumulate general knowledge on QA tasks. Afterwards, the model is finetuned on three out-of-domain datasets with 128 training examples each. Validations of the model performance is carried out using the dev datasets after training and fintuning stages respectively, resulting in two sets of prediction scores, namely baseline-train and baseline-finetune.

3.2 Architecture

Algorithm 1 General MAML

```
Require: $p(\Gamma)$ :  distribution over tasks
Require: $\alpha, \beta$ :  step size hyperparameters
Randomly initialize $\theta$
while not done do
    Sample batch of tasks $\Gamma_i \sim p(\Gamma)$
    for all $\Gamma_i$ do
        Sample K data points  $D = \{x^{(j)}, y^{(j)}\}$ from $\Gamma_i$
        Evaluate  $\nabla_\theta L_{\Gamma_i}(f_\theta)$ using D and $L_{\Gamma_i}$
        Compute adapted parameters with gradient descet: $\theta^{'}_i = \theta - \alpha \nabla_\theta L_{\Gamma_i}(f_\theta)$
        Sample datapoints  $D^{'} = \{x^{(j)}, y^{(j)}\}$ from $\Gamma_i$ from $\Gamma_i$ for the meta-update based on some function
    end for
    Update $\theta \leftarrow MetaLearn(L_{\Gamma_1}(f_{\theta'_1}), L_{\Gamma_2}(f_{\theta'_2}), ...)$ using each $D^{'}_i$ and $L_{\Gamma_i}$
end while
```

The idea of the MAML algorithm is to learn task-agnostic parameters that are suitable for a wide range of datasets (tasks). Using the task-agnostic parameters as a starting point will then enable the task-specific finetune step to converge and learn faster.

To do so, we divide the usual parameter updating process into two steps. Each round, we first learn a set of task-specific thetas θ_t by sampling data from each tasks and doing K gradient descend steps only using the sampled task-specific data.We then update the model-agnostic parameters based on the task-specific parameters, through function $MetaLearner(\theta; \theta_i^{(k)})$. These model-agnostic parameters are then fed back into the task-specific training step as the initial value for each task. The architecture of the MAML model is shown in the diagram in appendix A.

For our study, we implemented two variants of the Meta-Update step, Reptile (2) and First-Order MAML (3), both of which are a first-order approximation of the original MAML. Due to the time and resource limitation, we mainly focused on Reptile when we carried out the subsequent investigation into the different aspects of the MAML algorithm. In order to determine the optimal direction of the next descent for the meta-model, the Reptile meta-update updates the parameters of meta-model θ based on the average of the updated submodel parameters $\theta_i^{(k)}$ after k inner iterations. This method of determining the direction of the next descent has a tendency to move the meta-model to the submodel (task) that is most different than others, as the average is sensitive to outlier.

$$\theta = \theta + \beta \frac{1}{\{|T_i|\}} \sum_{T_i \sim p(\Gamma)} (\theta_i^{(k)} - \theta) \tag{2}$$

$$\theta = \theta - \beta \sum_{\Gamma_i \sim p(\Gamma)} \nabla_{\theta_i}^{(k)} L_i(\theta_i^{(k)}) \tag{3}$$

4 Experiment

4.1 Data

In this paper, We will use three in-domain reading comprehension datasets (Natural Questions [4], NewsQA [5] and SQuAD [6]) for training the baseline QA system. The trained QA system will then be finetuned and evaluated on the limited test examples from three different out-of-domain datasets (RelationExtraction [7], DuoRC [8] , RACE [9]).

4.2 Evaluation Metrics

In this paper, we will use two automatic evaluation metrics, Exact Match (EM) and F1 Score. Both metrics ignore punctuation and articles (a, an, the) [6]. Furthermore, we will take the maximum F1 and EM score across three human-provided gold answers when evaluating on the development-purpose dataset or test dataset. Also, the EM and F1 scores are averaged across the entire evaluation data to get the final reported scores.

4.3 Experimental Details

Table 1: Training configuration

Hyperparameter	Baseline+MT	MAML+uniform	MAML+PPS	FOMAML
Number of tasks (T)	1	8	8	3
Inner update steps (K)	1	5	5	5
Learning rate (α)	3e-5	3e-5	3e-5	1e-5
Meta learning rate (β)	N/A	1	1	1e-5
Task Distribution	N/A	Uniform	Proportional	Proportional

We tackle the domain-shift problem through two routes: multi-task learning (MT) and model-agnostic meta learning (MAML). We denote MAML with Reptile as MAML and FOMAML for First-Order MAML. For MAML, we have experimented with two different task sampling methods, which results in a total of four configurations. Table 1 shows the tuned hyperparameters for each of the setup. During our tuning process, we first attempt to adopt the hyperparameters used by Huang et al. [10]. However, this yields sub-optimal results. With some investigation, we find that the choice of the meta learning rate is highly dependent of the magnitude of the model parameters, especially when using Reptile. The smaller the model parameter, the greater the meta-learning rate needs to be. Furthermore, our study demonstrates that sampling tasks with uniform distribution is more effective, as it prevents the model from overfitting to tasks with a larger size. A more detailed study of the influence of different hyperparameters on the MAML algorithm (Reptile) will be shown in section 5.

Table 2: F1 score (EM score) on out-of-domain test datasets

Configuration	Aggregate
Baseline+MT	57.08 (40.30)
MAML+uniform	**59.51 (42.02)**
FOMAML	59.38 (41.26)

Table 3: F1 score (EM score) on out-of-domain validation datasets

Configuration	RACE	RelationExtraction	DuoRC	Aggregate
Baseline	36.74 (23.44)	64.12 (38.28)	40.31 (33.33)	47.10 (31.68)
Baseline+MT	36.54 (23.44)	**74.69 (53.12)**	38.88 (29.37)	50.10 (35.34)
MAML+uniform	39.41 (25.78)	73.70 (53.12)	42.02 (34.92)	**51.10 (37.17)**
MAML+PPS	37.61 (23.44)	74.37 (54.69)	37.27 (29.37)	48.72 (35.08)
FOMAML	**40.53 (26.56)**	65.27 (41.40)	**42.13 (31.74)**	49.35 (33.24)

With the tuned parameters located in each configuration, we deploy our models on the limited examples in the out-of-domain datasets and find that the MAML algorithm with uniform task sampling distribution preforms the best. There are several observations can be made here. Firstly, we can do a sanity check by comparing different improvement methods with the baseline model that has only learned general QA knowledge from the in-domain dataset, in order to ensure that the new improvement methods are able to pick up information that is specific to the out-of-domain dataset. It is consistent with our expectation that both multi-task learning and MAML have better performance than the baseline who has never seen the out-of-domain dataset. After we ensure that the new methods are working, the next question we address is that, among all the endeavors to improve on top of the baseline model, which one can learn better in the very limited examples we have in the out-of-domain datasets. In our experiment, we find that the MAML with uniform sampling distribution is the best model that can outperform the multi-task learning approach. Despite both approaches allow for transfer learning, we argues that MAML is able to do better in the learning process because the uniform task sampling distribution allows the meta-model parameter to find a more balanced position that serves well for all the subtasks in the downstream. This distinct advantage becomes obvious when we look at the result from MAML with PPS sampling method, we can see that the result that uses PPS sampling scheme resembles that from the multi-task learning process, where the model seems to spend most of its focus to the task RelationExtract.

5 Analysis

5.1 Loss Function

One of the key features of optimization-based meta-learning is its usage of task-specific inner loops to decide the next descent taken by the meta-model. For each of the tasks sampled in the meta-batch, we execute the inner loop with five gradient descents. These gradient descent steps taken in the inner loop are to encourage the model to learn as much as it can in only five steps. In other words, the MAML algorithm allows us to find a set of parameters that can make the steepest descent in limited steps.

Such properties of MAML are demonstrated in Figure 1, where MAML learns at a very rapid rate in the first epoch. This rapid descent in the first epoch leads to a much lower initial loss in the second epoch. The decline in initial loss continues through the subsequent epochs until the fifth one, at which the model seems to have absorbed all the information from the given examples. When we are finetuning the MAML meta-model in the individual tasks, It is very obviously that our MAML model has a much lower initial loss in comparison to the baseline model. It is a sign that the MAML can find a good initialization for multiple downstream tasks.

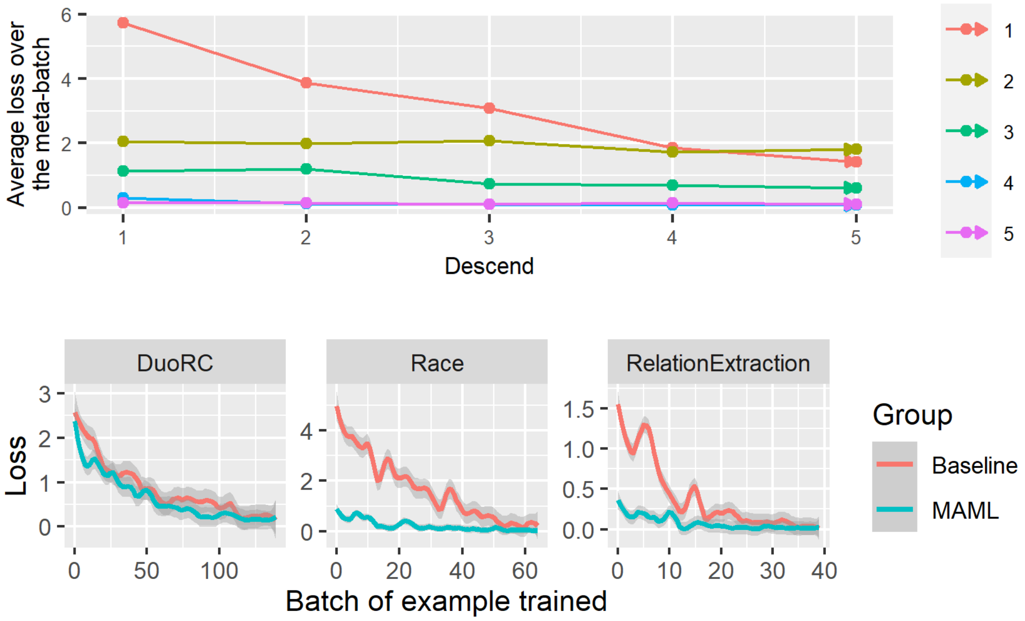

Figure 1: The plot above shows the steepness of the gradient descent over a meta-batch in each epoch (line). Five updates are made for each sampled task in the meta-batch. The plot below shows the evolution of the loss function during the finetuning stage on the out-of-domain dataset.

5.2 Learning rate

Table 4: Parameter Sweep

β/α	3e-3	3e-5	3e-8
1	47.65	**51.10***	49.86
0.5	47.24	48.17	**48.48**
1e-5	47.10	47.10	47.10

For the learning rate, we find that the inner learning rate (α) seems to be universally applicable to all the models that are solving the same problem. Essentially the optimal inner learning rate is transferable between models, should that be multi-task learning or meta-learning.

On the contrary, the meta-learning rate (β) is much more obscure to find. During the process of tuning parameters, we noticed that the meta-learning rate is not transferable between problems even if two problems are in the same field (i.e. NLP). The meta-learning rate provided by Dou et al. does not apply in our model. This non-transferability in meta-learning rate between different problems is not surprising, as it is commonly known that all problem settings require their own set of hyperparameters.

Besides, we also find that the meta-learning rate is non-transferable between different variants of the MAML algorithm. For Reptile MAML algorithm, we find that the optimal meta-learning rate should be around the value of 1. When we try to use the same meta-learning rate for First-order MAML algorithm, the model marches further away from the optimal solution, with the loss progressively getting larger throughout the training process. Despite First-order MAML is not the main focus of our study in this paper, we find that 1e-5 is one of the meta-learning rate that would allows it to work properly. Applying the meta-learning rate that works for the First-Order MAML variants, we find that this is too small for Reptile to acquire any information, hence unable to make update to the meta-model. With this demonstration, we learn that the meta-learning rate suitable for various meta-learning variants sometime lives on different levels of magnitude.

5.3 Number of sampling tasks and inner updates

Table 5: The Effect of Number Of Tasks

Configuration	1	4	8	12
MAML+uniform (K=5)	49.15	50.37	51.10	51.08

Table 6: The Effect of Gradient Descent Steps

Configuration	1	3	5
MAML+uniform (T=8)	47.41	48.99	51.10

Table 6 shows that increasing the number of gradient steps taken in the inner loop for each task can bring forth significant marginal improvement in the performance. This finding is consistent with Alex et al. [11], where the authors studied the Reptile MAML algorithm and demonstrated, mathematically and empirically, that the step of gradient descent can make a noticeable difference in the performance as it allows for the use of higher-order derivatives of the loss function.

On the other hand, the number of sampled tasks has limited impact in the performance. As we can see in Table 5, increase in performance has become saturated as we increase the number of sampled tasks in meta-batch to 8. Looking into the tasks sampled in the meta-batch, we find that the performance of the MAML algorithm is heavily correlated with whether all the tasks are seen and included in the meta-batch. This explains why the performance is worse when the number of sampled is in the lower end, since there is a high probably we will see only subset of all available tasks in the meta-batch. Despite the increase of sampled tasks does not bring as noticeable benefit as the increase of gradient descent steps, it is worth to note the saturated point, so that the practitioner can achieve the best result without spending unnecessary runtime on the additional tasks.

5.4 Task Sampling Distribution

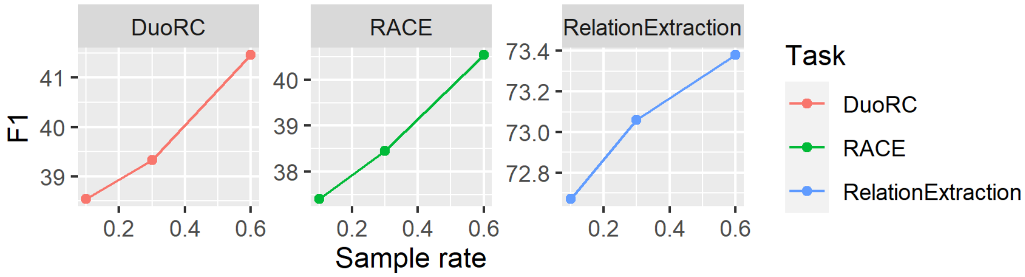

Figure 2: The effect of task sample rate during meta learning.

In this section, we look into the reason why the uniform task sampling scheme works better than the proportional sampling scheme. We design an imbalance sampling experiment, where each task is sampled with probability of $\{0.6, 0.3, 0.1\}$ in rotation. Figure 2 compares the F1 score against the sampling probability for each task. As we can see, the higher the sampling probability of a task, the better the model performs on it. In other words, MAML algorithm tends to learn a meta-model that is leaning toward the tasks that are most frequently sampled. From another perspective, to learn a model that allows better transfer learning, we should adopt a more balanced sampling scheme without favoring tasks with a large sample size.

5.5 Runtime analysis

$$\text{ITC} = \frac{\triangle(\text{percentage increase in performance})}{\triangle(\text{percentage change in runtime})} \tag{4}$$

Table 7: Time efficiency of different approaches

	Baseline		Modified		Aggregate
Configuration	runtime (mins)	performance	runtime (mins)	performance	ITC
Baseline+ML	1133	47.10	1138	50.10	15.92
MAML+uniform	1133	47.10	1155	51.10	8.49

While the MAML algorithm is able to achieve better performance than the baseline and the multi-task learning method, we are interested in gauging the value of applying MAML method in practice. In the artificial intelligence field, there are myriad of new ideas emerged on daily basis. The majority of focus is often placed on the absolute gain in accuracy by the newly proposed method, whereas limited time is spent on whether the newly proposed method is worthy to carry out in practice. The best method should not only be the one that outperform the best score that has been archived in the past, but the one that increases performance with the least increase in resource consumed.

In order to assess the practicality of the proposed improvement methods, we introduce the Improvement-To-Cost (ITC) ratio, define in (4), a simple metric for practitioner to easily screen for the most cost-effective improvement method comparing to some baseline. When comparing two ITCs with respect to the same baseline, the great the value, the lesser sacrifice in runtime is needed for the accuracy gain. Thanks to its simple and general form, this metric can be used in many different settings to help understand the comparative advantage of a model as well as regulate the tendency of indefinitely increasing the size of a model and its input dataset.

As shown in Table 7, the additional runtime consumed by MAML overshadows its performance improvement based on our ITC ratio.

6 Conclusion

In our study, we demonstrate MAML algorithm's ability to locate better initialization that allows faster convergence in each of the downstream tasks compared to multi-task learning. In the process of tuning hyperparameters, we find that the meta-learning rate is not transferable between different MAML variants. More studies and investigations are required in order to understand the potential relationship between the order of magnitude in the meta-learning rate and the magnitude of the model parameters. In terms of sampling tasks for the meta-batch, we find that uniform sampling scheme gives the best result, for it enables the model to have equal exposure to all the available tasks, and thereby promotes transfer learning without overfitting to one or a few tasks with larger sample size. We then demonstrates the important of making multiple updates in the MAML inner loop of each task. This allows the model to make use of the higher-order terms in the loss function, and it is one of the main reasons why the MAML model is able to find good model initialization for all tasks so quickly. Last but not least, we conduct run-time analysis and introduce a new metric, ITC, to measure the "worthiness" of implementing any improvement on top of the baseline. Based on our ITC metric, the improvement brought front by the MAML algorithm is trivial when taking into account the increase in runtime. We largely attribute this inefficiency to the double-loop structure used by the MAML algorithm.

A Architecture

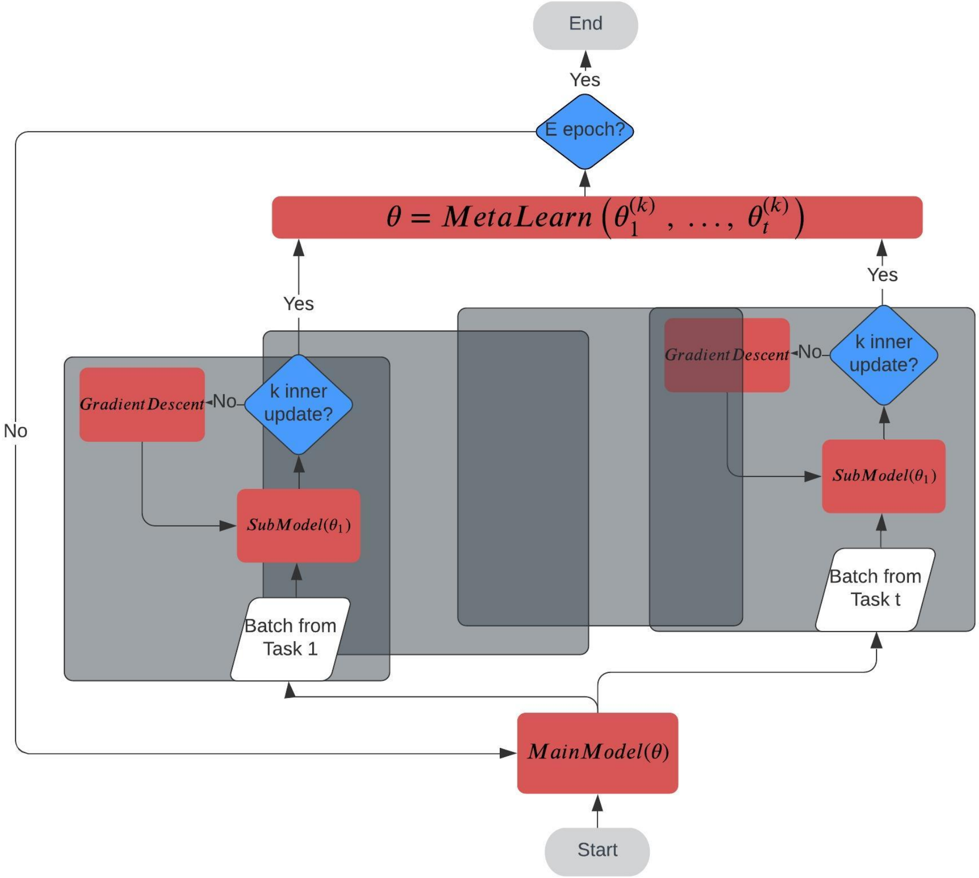

Figure 3: MAML architecture

References

[1] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400, 2017.

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

[3] Zi-Yi Dou, Keyi Yu, and Antonios Anastasopoulos. Investigating meta-learning algorithms for low-resource natural language understanding tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1192–1197, Hong Kong, China, November 2019. Association for Computational Linguistics.

[4] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfeld, Michael CollinsAnkur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: a benchmark for question answering research. *ACL*, 2019.

[5] Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni andPhilip Bachman, and Kaheer Suleman. Newsqa: A machine comprehension dataset. *ACL*, 2017.

[6] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, abs/1606.05250, 2016.

[7] Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. *arXiv preprint*, 2017.

[8] Amrita Saha, Rahul Aralikatte, Mitesh M. Khapra, and Karthik Sankaranarayanan. Duorc: Towards complex language understanding with paraphrased reading comprehension. *ACL*, 2018.

[9] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. Race: Large-scale reading comprehension dataset from examinations. *EMNLP*, 2017.

[10] Po-Sen Huang, Chenglong Wang, Rishabh Singh, Wen-tau Yih, and Xiaodong He. Natural language to structured query generation via meta-learning. *CoRR*, abs/1803.02400, 2018.

[11] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. 2018.