

Meta Learning on Topics as Tasks for Robust QA Performance

Stanford CS224N Default Project

Josh Nkoy x Arafat Mohammed

Department of Computer Science

Stanford University

joshnkoy@stanford.edu x arafatm@stanford.edu

Abstract

A key pain point of current neural QA-focused NLP systems is the lack of generalization — often these systems learn parameters that fail to generalize to never-before-seen data domains, unlike how humans can take previous knowledge and build accurate inferences beyond "training" data distributions. Clearly, advances in meta-learning have shown promise in improving model resiliency and adaptability across many AI domains, and thus we hope to modify our given Transformer QA model to improve performance on out-of-domain QA tasks and data. Specifically, we hope to use the Reptile meta-learning algorithm applied to multiple prelearning tasks — which we interpret to be topics from within a single dataset — to create a metalearner on which we test out-of-domain QA, in order to hopefully show that this model would be more robust than baseline (higher EM and F1 scores).

1 Key Information to include

- Main Mentor: Eric Mitchell
- External Collaborators (if you have any): None
- Sharing project: N/A

2 Introduction

Among the wide applications of and innovations in natural language processing (NLP), reading comprehension, specifically question-answering (QA) tasks have benefitted immensely from the refinement of NLP models and techniques as they have evolved from formal language-based methods, to statistical/probabilistic models, and finally to neural models in the past decades. Most recently, advancements in machine QA have been achieved via transition from formerly state-of-the-art neural QA models like BiDAF (Seo, Kembhavi, Farhadi, Hajishirzi, ICLR 2017) that incorporate attention as an accessory to the main compute power of bidirectional LSTMs to newer models like BERT (Devlin et al. 2018) that are based off Transformers that use attention as the main layer-connecting building block.

However, despite these advancements and like many neural models, we see that QA models also have hard times making effective and accurate predictions on datasets that are outside the domain of the distribution of training data. This out-of-domain-QA problem is a significant open problem in natural language understanding for a couple of main interesting issues. First, and most obviously, real-world QA systems are rarely completely in-domain; insofar as a given QA system is not designed to comprehend language from and answer questions about absolutely every subject as per open-domain QA that drives search engines, robustness for a limited QA system to be able to handle domains that may still be related to the training set — its area of expertise as we might call it. Second, and most applicably to how we view neural models as a way to make AI more like human intelligence, humans

themselves can take previous knowledge and generalize it to previously-unseen domains of words and tasks over them, regardless of their area of expertise or how they might have learned what they did previously. The truth of the matter is that because models are essentially memorizing the domain they become a bit paralyzed when thrown into a new one.

An emergingly effective way to better models toward these tendencies is to introduce meta-learning, which involves finding a set of global parameters θ that allow for learning and predicting on specific tasks, in this case viewing tasks as domains of data. As such, we would find task-specific parameters ϕ_i for each task $\mathcal{T}_i$ and feed a DistilBERT model using these parameters over a support set $\mathcal{D}_{\mathcal{T}_i}^S$ of a small number of samples from $\mathcal{T}_i$; then using $\phi_i = f_{\theta}(\mathcal{D}_{\mathcal{T}_i}^S)$, we update the parameters θ in order to get a new model that can make predictions on the query set $\mathcal{D}_{\mathcal{T}_i}^Q$ of new examples from that task. Clearly, advances in meta-learning have shown promise in improving model resiliency and adaptability across many AI domains, and thus we hope to modify our given Transformer QA model to improve performance on out-of-domain QA tasks and data.

3 Related Work

There is a long history of learning general language representations. Meta learning dates back from 1992 when Bengio et al explored the possibilities of learning rule to solve any new tasks, learning how to learn. One of the first variant of meta-learning was multitask -learning by Rich Caruana In 1997. Meta-learning algorithms, being fairly recent, has proven to have lots of potential applications. For example, Gu et al. (2018) have tried to apply first-order MAML in machine translation and Qian and Yu (2019) propose to address the domain adaptation problem in dialogue generation by using MAML. This paper has shown a new variant, Reptile algorithm is potentially more powerful and useful yet simpler. This serves as a stepping stone for further research. One popular application of meta-learning is in Natural Language understanding (NLU) and text translation for low resource languages. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, Zi-Yi et al presented a variant of meta-learning for NLU of low resource languages in their paper, Investigating Meta-Learning Algorithms for Low-Resource Natural Language Understanding Tasks. This study suggested that meta learning has promising impacts in the field of NLU.

There has also been works on using meta-learning for fast adaptation of deep networks from a study by Chelsea Finn, Pieter Abbeel, and Sergey Levine which proposed a variant of meta-learning compatible with any model trained with gradient descent and a variety of learning problems including reinforcement learning and classification.

Another potential application of meta-learning is in Question Answering systems. In 2017, Danqi Chen, Adam Fisch, Jason Weston, Antoine Bordes proposed a open- domain question answering system using Wikipedia as the unique knowledge source.

4 Approach

- Our baseline system is DistilBERT (a smaller, distilled version of the original BERT model [1]) that finetunes on all the training data. This baseline is explained in details in the project document. []
- In this paper, we use Meta Learning to improve the ability of our model (DistilBERT) to generalize: work well on out of domain data. The meta learning problem is basically a supervised learning with additional data $D_{metatraining}$ given by:

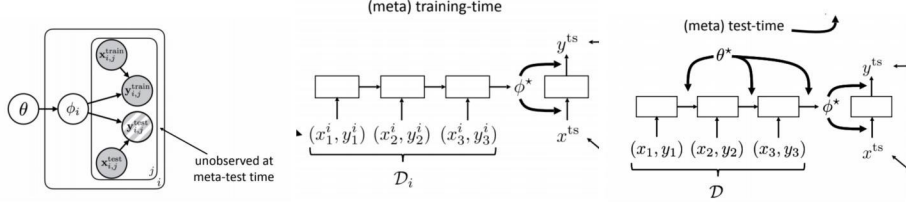
$$\operatorname{argmax}_{\phi} \log p(\phi | D, D_{metatraining})$$

where D is a set of training data points in the form: $(paragraph, question, answer)$ and $D_{metatraining}$ is a set of training dataset similar to D in the form $\{(D_{1train}, D_{1test}), \dots (D_{ktrain}, D_{ktest})\}$, $D_i : (paragraph, question, answer)$ where each D_i is represented as a task T . We can rewrite the above equation in the following form:

$$\begin{aligned} \text{meta-learning: } \theta^* &= \operatorname{argmax}_{\theta} \log p(\theta | D_{metatraining}) \\ \text{adaptation: } \phi^* &= \operatorname{argmax}_{\phi} \log p(\phi | D, \theta^*) \end{aligned}$$

where we first learning some parameters from $D_{metatrain}$ during meta-learning stage and use these learned parameters coupled with the training data to obtain ϕ during the adaptation stage.

- We split each task into support, D_s and query D_q datasets where support is used for training and query is used for evaluation. The method for obtaining each task is elaborated more in the data section of this paper.
- The above process could be demonstrated by the picture below from [].



- Generally, we define DistilBERT model f_θ and use its parameters to initialize another DistilBERT model f_ϕ .
- During meta-training, we uniformly sample a batch of tasks T from the distribution of tasks $P(T)$. For each task, we get the $D_s, D_q, \in D_i \in T$. We then serve the f_ϕ with the D_s as input and then after training we use D_q for evaluation as seen from the second picture. We then use the new parameters of f_ϕ to update the parameters of f_θ .
- The above process can be seen formally from the image below:

$$\theta = \text{MetaUpdate}(\theta; \{\theta_i^{(k)}\}). \quad (2) \quad \theta_i^{(k)} = \theta_i^{(k-1)} - \alpha \nabla_{\theta_i^{(k-1)}} L_i(f_{\theta_i^{(k-1)}}), \quad (1)$$

Algorithm 1 Training procedure.

Pre-train model parameters θ with unlabeled datasets.
while not done **do**
 Sample batch of tasks $\{T_i\} \sim p(T)$
 for all T_i **do**
 Compute $\theta_i^{(k)}$ with Eqn. 1.
 end for
 Update θ with Eqn. 2.
end while
Fine-tune θ on the target task.

5 Experiments

5.1 Data

Given that we aimed to implement a meta-training algorithm with limited compute space and time, we focused on finding appropriate ways to divide one dataset and simulate the process of a model seeing questions from multiple different domains, in order to save computational space while also making learning more structured and yielding more effective parameters for out-of-domain tasks. As such, we used the SQuAD 1.1 database of questions and, from a general standpoint, topicalized the individual questions into groups in such a way that picking tasks would be synonymous to picking topics. The aim of this was to coax the model to learn over the dataset of questions and context by building expertise in each topic, such that out-of-domain datasets can be matched to one or more of these topics in prediction.

We first needed a method to efficiently evaluate the topical content of each of the $\langle c, q, a \rangle$ data points consisting of context paragraphs c , questions q , and answers a ; we achieved such a topicalization as follows. We first noticed, by construction of QA dataset like the ones we were given, any topical analysis of the data could be encapsulated in a distribution over the context paragraphs. As such, we took the original JSON format of the SQuAD data and read it into a list $\mathcal{D}$ of all unique $\langle c, q, a \rangle$ tuples. From these tuples we also extracted a list C of all unique context paragraphs, over which we generated a probability distribution $p(\mathcal{T}_i | C)$ that returned some probability that a context paragraph

would fit into the top semantic features of the topic $\mathcal{T}_i \in \mathcal{T}$. With this distribution we partition each of the $|\mathcal{D}|$ data points by finding over each context paragraph c_k the most appropriate topic, or

$$\mathcal{T}_{i|c_k}^* = \arg \max_{\mathcal{T}_i} p(\mathcal{T}_i | C = c_k)$$

for any $i \in |\mathcal{T}|$; therefore with these labels over each context paragraph we obtain a (task/topic) distribution of data over each topic as

$$\mathcal{D}_{\mathcal{T}} = \bigcup_{i \in |\mathcal{T}|} \mathcal{D}_{\mathcal{T}_i},$$

in which $\mathcal{D}_{\mathcal{T}_i} \cap \mathcal{D}_{\mathcal{T}_j} = \emptyset$ for all $i \neq j$ — each $\mathcal{D}_{\mathcal{T}_i}$ is a disjoint subset of all of the data — and, furthermore, in which each $\langle c_k, q, a \rangle \in \mathcal{D}_{\mathcal{T}_{i|c_k}^*}$ — each data point corresponding to the context paragraphs in topic i are in the distribution under that topic.

Viewing our context paragraphs c temporarily as documents, we saw that obtaining this probability distribution $p(\mathcal{T}_i | C)$ would most efficiently be handled via a quick information retrieval (IR) method. Since we are relying on generating probabilities of documents being in a specific topic, we looked for a probabilistic IR algorithm; **latent Dirichlet allocation** (LDA) [source here] ended up being perfect for the task at hand. Methodologically, by using LDA, we created a probability distribution of latent topics over the top f features (words) and then via a generative probabilistic model obtained an accurate representation of each document as a mixture of topic probabilities. Computationally then, we vectorized the list of documents with respect to the counts of the top $f = 500$ words in each, and then fed that vectorized representation into a prepackaged LDA model that returned, as needed, a $|C| \times |\mathcal{T}|$ matrix of probabilities $p(\mathcal{T}_i | C = c_k)$ for each document c_k — in our case we chose $|\mathcal{T}| = 100$ topics to create. In essence, finding the topic that is the best for each document computationally involved, quite simply, taking the $\arg \max$ of each row in the resulting matrix and assigning that resulting column number with highest probability as the best topic for that document. In short, by using IR we were able to build our desired distribution of "tasks" within one dataset, this process being extremely easy to extrapolate over multiple datasets as well.

Once the topicalization process was finished, we then sought to divide the data-over-topics distribution randomly into support data $\mathcal{D}_{\mathcal{T}_i}^S$ and query data $\mathcal{D}_{\mathcal{T}_i}^Q$. As such, we iterated over and took a random permutation of all of the $\langle c, q, a \rangle$ data points within $\mathcal{D}_{\mathcal{T}_i}$, assigning 70% of those to $\mathcal{D}_{\mathcal{T}_i}^S$ and the remaining 30% to $\mathcal{D}_{\mathcal{T}_i}^Q$. As such, we randomly partition each topic's data into disjoint support and query sets such that, with this support/query split,

$$\mathcal{D}_{\mathcal{T}} = \bigcup_{i \in |\mathcal{T}|} (\mathcal{D}_{\mathcal{T}_i}^S \cup \mathcal{D}_{\mathcal{T}_i}^Q),$$

where $\mathcal{D}_{\mathcal{T}_i} = \mathcal{D}_{\mathcal{T}_i}^S \cup \mathcal{D}_{\mathcal{T}_i}^Q$ for all i by definition, and of course $\mathcal{D}_{\mathcal{T}_i}^S \cap \mathcal{D}_{\mathcal{T}_i}^Q = \emptyset$ for all i , reflecting disjointness.

Our final step of data manipulation was to randomly sample B batches of our split topic/data groups $\mathcal{D}_{\mathcal{T}_i}$, such that over each stage of our meta-training process we would be batching different groups of topics to learn. Relatively simply then, we again took a random permutation of all of the topics $\mathcal{T}$ and then split these task indices into groups of tasks $\mathcal{T}^b$ for all $b \leq B$ such that

$$\mathcal{D}_{\mathcal{T}} = \bigcup_{b \leq B} \mathcal{D}_{\mathcal{T}^b} = \bigcup_{b \leq B} \left(\bigcup_{k \leq \frac{B}{|\mathcal{T}|}} \mathcal{D}_{\mathcal{T}_k^b} \right) = \bigcup_{b \leq B} \left(\bigcup_{k \leq \frac{B}{|\mathcal{T}|}} (\mathcal{D}_{\mathcal{T}_k^b}^S \cup \mathcal{D}_{\mathcal{T}_k^b}^Q) \right),$$

where k represents the number of topics in each of these batches.

These were the forms of support and query datasets that we used in the meta-learning process described above, with train data and evaluation being split in SQuAD before this whole process. For out-of-domain evaluations for submission, we used the provided newsqa dataset.

5.2 Evaluation method

For evaluation metrics, we stuck to the default QA metrics of F1 and EM scoring over both in-domain and out-of-domain datasets.

5.3 Experimental details

Due to the computational constraints, we were unable to perform an extensive experiment for our project. However, we decided to set a learning rate of $\beta = 0.1$ for the meta-model, while settling on a learning rate of $\alpha = 0.02$ for our inner learning model whose parameters are used to update the outer meta-model. We then set $k = 3$ as our number of total meta-training iterations, with 10 iterations over each task in each sample batch of the overall task distribution. This gave relatively fast compute times with easily obtainable scoring and loss parameters, though far from optimal for all values.

5.4 Results and Analysis

Unfortunately, due to computational constraints and bugs being constantly worked out, we were not able to achieve F1 and EM scores anywhere near baseline. More specifically, because of our small parameters for epoch size and iteration over each task in each batch — respectively, 3 meta-train iterations (below the best-performing marks of 5 and 10 meta-train iterations) and only 10 iterations over each task — our F1 and EM scores rarely rose above 0.1 and 0.001 respectively during each meta-train iteration per batch. Our file of answer predictions realized as strings reflected this primitive behavior: over each of the predictions, there were rarely any specific answers, showing that the model was almost randomly taking snippets from the context paragraph as the answer, showing good topic focus but no specificity at all. In order to slightly fix this, as well as the relatively stalled average losses, we increased the learning rate α of the inner training model from 0.02 to 0.1 to more aggressively ensure that the model wouldn't get stuck at local minima, which helped increase average F1 scores per iteration from 0.03 to 0.07.

In addition, there were some very interesting, and even encouraging, patterns that could be exploited for future work and improvement over the whole model. A baseline test of the meta-trained model on the SQuAD validation set, after meta-training on the SQuAD training split as provided in the default final project package, gave an F1 score of 7.135 and an EM score of 0.785, which seems to be a drastic improvement over the local learning scores over each task; this might show that (1) the meta-model was still able to adapt well to new data it hadn't seen, provided there was no repeat data in the validation set, and that (2) with more thorough iteration over these tasks, we might increase the ability for the meta-model to see patterns in new data and thus answer accurately based on the variety of tasks/topics it has already seen. We hope to see what would happen given we iterate over each task 100 times instead of 10 with these new learning rates, which would bolster learning the patterns in each task and thus each parameter update of the meta-model.

6 Conclusion

Overall, our project lies at the confluence of three potential innovations: meta-learning and the robustness resulting from it, information retrieval as a generator of task distributions for such meta-learning, and stretching single datasets as far as possible for training, which saves a lot of compute space. We were able to successfully combine these three goals and even produce a meta-model that readily evaluated data from new domains, albeit at poor efficacy due to lack of near-adequate training of that meta-model. In all, we hope to continue our work in order to realize the benefits we are confident exist in such a path, with 10-fold increases in training time parameters that likely would bring better performance than baseline.

References

- [1] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108, 2019.
- [2] Shangwen Lv et al. Pre-training text representations as meta learning. arXiv preprint arXiv:2004.05568, 2020.
- [3] Open-domain Question Answering DrQA (Chen, et al. ACL 2017) <https://arxiv.org/abs/1704.00051>
- [4] Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. arXiv.

- [5] 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.
- [6] Jiatao Gu, Yong Wang, Yun Chen, Victor OK Li, and Kyunghyun Cho. 2018. Meta-learning for low- resource neural machine translation. In EMNLP.
- [7] Kun Qian and Zhou Yu. 2019. Domain adaptive dialog generation via meta learning. In ACL.