Improving Out-of-Domain Question Answering with Mixture of Experts

Stanford CS224N Default Project (Robust QA track)

Haofeng Chen, Danni Ma
Department of Computer Science
Stanford University
alexhc@stanford.edu, dannima@stanford.edu

Abstract

Question answering (QA) is an important problem with numerous application in real life. Sometimes, the resource of certain QA task is limited. This work aims to build a robust QA system that can generalize to novel QA tasks with few examples and gradient steps. We propose a Mixture-of-Experts(MoE) style training framework, and use meta-learning methods for domain adaptation. We also explored data augmentation techniques, and successfully improve out-of-domain QA performance of baseline models on F-1 score from 50.81 to 53.84 and exact match (EM) score from 34.82 to 39.27. Our approach achieves a F-1 score of 60.8 and EM score of 42.2 on the out-of-domain QA testing leaderboard.

1 Introduction

Question answering(QA) has achieved significant performance improvements over the last few decades and is widely used in our daily life such as online search and smart assistants. Although the development of QA is rapid, it usually needs to collect and feed in large amount of training datasets of natural human language. Our goal is to build a robust QA system that automatically answer questions posed by humans in a natural language with few examples and gradient steps based on the provided DistilBERT model. In order to improve the performance of DistilBERT model, we explore and implement models and techniques including Mixture-of-Experts (MoE), meta-learning (MAML Algorithm), and data augmentation. Eventually, we successfully integrate the three mentioned approaches with a focus on MoE and boost the F1/EM scores on the out-of-domain test datasets.

2 Related Work

- **Mixture-of-Experts (MoE)**
  The paper "Adaptive mixtures of local experts"[1] authored by R. Jacobs, Michael I. Jordan, S. Nowlan, and Geoffrey E. Hinton in 1991 first introduces Mixture-of-Experts (MoE): a supervised procedure for systems composed of many separate networks learning to handle a subset of the complete set of training cases which can be viewed as either a modular version of a multi-layer supervised network, or as an associative version of competitive learning. Since then, MoE training has been implemented in many NLP research papers[2, 3] and demonstrate great model improvement.
  We reference the original paper in 1991[1] as our main approach to boost the performance of DistilBERT model on the robustQA task: training k models (or "experts") along with a gating function (in this case, MLP) that controls the mixture and update the experts.
- **Data Augmentation in NLP**
  Data augmentation technique, from simple to complex, is vastly utilized in many NLP papers[4, 5, 6, 7, 8] due to limited labeled training datasets available. The paper "An

exploration of data augmentation and sampling techniques for domain-agnostic question answering"[9] shows us applicable techniques for our in-domain data augmentation in order to prevent model from learning brittle correlations that hurt the out-of-domain performance. In this paper, the researchers investigate the various data sampling strategies, query and context paraphrases generated by back-translation: English translated to German and translated back to English, to add quantity to the original in-domain datasets. Their experiments show that data augmentation techniques including back-translation and word substitutions can improve the F1/EM scores on testing on the out-of-domain datasets and the robustness of QA system.

In addition, we reference "Semantically Equivalent Adversarial Rules for Debugging NLP Models"[10] which presents that data augmentation significantly reduces bugs, while maintaining accuracy. The paper introduces an original data augmentation technique called SEAR: semantically equivalent adversarial rule, which can be applied to any type of models and allows human users to accept/reject rules based on whether or not they preserve semantics.

- **Meta-Learning**

  We review "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks"[11] to explore possible applications of meta-learning in question answering task. The paper covers step-by-step implementation of Model-Agnostic Meta-Learning(MAML) algorithm that is compatible with gradient-descent trained model and its application to various learning problems including text classification. The paper shows that meta-learning can solve new learning tasks using only a small number of training samples, which matches our goal of learning to answer questions with few training examples.

  Many other papers[12, 13, 14, 15] in NLP area also uses meta-learning to boost model performance since labeld NLP datasets are very limited. We particularly look at "Investigating Meta-Learning Algorithms for Low-Resource Natural Language Understanding Tasks"[16] and "Learning to few-shot learn across diverse natural language classification tasks"[17] for insights. Both paper covers how to build few-shot models via meta learning with very few resource for training, which takes a small number of samples from a task (called the support set) as input and outputs an adapted model that can make accurate predictions on new examples from the task (the query set). The goal of meta-learning training is to maximise accuracy on the query set after adapting to the support set. Both papers validate that the MAML algorithm can strongly beat the baseline.

  The paper "On First-Order Meta-Learning Algorithms"[18] introduces another first-order gradient-based meta-learning algorithm called Reptile, which works by repeatedly sampling a task, training on it, and moving the initialization towards the trained weights on that task. Like MAML, Reptile learns an initialization for the parameters of a neural network model. When we optimize these parameters at test time, learning becomes faster. The paper shows that combining MAML and Reptile can optimize well for within-task generalization.

- **Few-shot learning in NLP**

  Few-shot modeling[19, 20] is an important problem that can be applied to various areas of natural language processing. With in-context learning, the model, including a general and task-agnostic representation, can adapt to a new problem not being limited to a specific task. We review "Making Pre-trained Language Models Better Few-shot Learners"[21] to explore practical application of few-shot learning in NLP, mainly for fine-tuning purpose. The researchers use smaller language models to improve the computational efficiency of fine-tuning on a small number of annotated examples. They successfully outperform standard fine-tuning procedures in a low resource setting. Specifically, the approach this paper proposes for few-shot language modeling can achieve performance greater than GPT-3, which has an unrealistic number of parameters for common research-purpose devices. This aligns with the setting of our work where only limited computation is available.

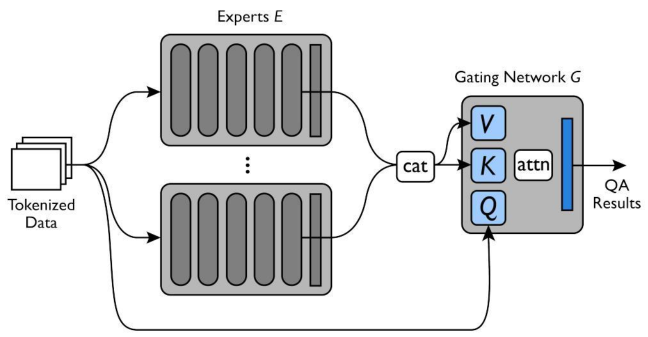


Figure 1: Proposed Mixture-of-Experts Model

3 Approach

3.1 Problem Set-up

For the question-answering (QA) task, we assume a input tuple (c, q) where c is the context and q is the question, and the goal is to find the answer a. We also assume that the answer a is a part of the context c, so instead of forming the problem as a natural language generation problem, we make our model to predict the start and end indices i_{start} and i_{end} of the answer in c.

The goal of this study is to adapt to a set of out-of-domain QA tasks $\mathcal{T}' = \{T_1', ..., T_n'\}$ with datasets from a set of in-domain QA tasks $\mathcal{D}_{in} = \{D_1, ..., D_m\}$ and a few examples from each out-of-domain QA task $\mathcal{D}_{out} = \{D_1', ..., D_n'\}$. Given that all of the QA tasks have the same format, we are able to test each model on different datasets which can help us understand the domain gap between these datasets better.

3.2 Base models

We use the DistilBERT[22] for question answering model implemented by HuggingFace[23] as our baseline models, which is required by the default project handout (robustQA)[24]. The model uses a DistilBERT and a linear classifier to make predictions of the starting and ending indices of an answer in the context. We will refer each of these models as "base models" in the following. We initialize all of the base models using the provided weights of a BERT-based pretrained transformer and finetune on different datasets. Particularly, we set the base model that trains on all three in-domain datasets as our "baseline model" for us to compare with.

3.3 Data augmentation

We explore two data augmentation techniques: synonym replacement and random swap to increase the diversity of the dataset available for training models, without actually collecting new data. We use the `nlpaug` library [25] to generate augmented data. Following [25], we employ the PPDB database [26] for synonym replacement. For each word, we set the probability of it being augmented (either swapped or replaced by synonym mentioned in the following) to be $p = 0.3$.

We employ the augmentation strategy as follows: for a given context-question pair, we use make k augmented example with the augmented context and the same question, and another k with the augmented question and the same context. We use this conservative strategy of augmentation to make training more stable, without having too much deviance to the original data. Unless mentioned, we set k to be 1. Note that randomly swapping two words might possibly change the ground-truth start and end indices of the answer, but practically, we do not observe that this hurts QA performance.

3.4 Gating network for mixture-of-experts output aggregation

We train our base models with different datasets and data augmentation techniques. To adapt to novel domains, we employ the method of using a mixture of experts (MoE)[1].

As shown in Figure 1, we train a QA model E_i for each in-domain dataset D_i and another one with all in-domain datasets combined. We call this set of $(m+1)$ QA models "experts", denoted as $\mathcal{E} = \{E_1, ..., E_{m+1}\}$. We have another lightweight model G which we call the gating network, producing QA results by weighting the embeddings produced by each expert model. During training, we fix the parameters of each expert to train G only.

The gating network G, illustrated in Figure 1, takes the concatenated embeddings of each experts before the final layer, $e = \{e_1, ..., e_{m+1}\}$, and the input tokenized data d. The goal is to produce an output embedding by weighting the embeddings of each individual expert. We compute

$$\alpha = softmax(Q(d)^\top K(e))$$

to get the weight of the embeddings from each expert by a query similar with dot-product attention, and Q and K are multilayer perceptron. The final embedding, e_{final}, is obtained by a weighted sum:

$$e_{final} = \sum_i \alpha_i e_i$$

We then add a dropout[27] layer after getting the embedding, and a QA output layer for the final prediction just like the base model mentioned in Section 3.2 to predict the start and end indices of the answer in the given context.

During training, we fix the weights of each of the base models (or the "experts"), and only train the gating network G to generate the QA result by making reference to each of the experts. We refer to training the gating network as MoE training. We also apply meta-learning algorithms for MoE training as discussed next.

3.5 Meta Learning

Meta Learning is a technique to learn to learn a new task. The idea is to form a representation that can be quickly adapted to a new task. MAML [11] is a meta-learning algorithm that tries to improve the query set performance of each in-domain tasks during training. Refer to the paper [11] for further details of the algorithm.

We integrate MAML algorithm into MoE training with the weights of each experts E_i fixed. Algorithm 1 shows the pesudocode of the MAML algorithm we implement for our setup. In brief, at each meta-learning step, we sample a set of tasks (datasets in our context) and adapt to each tasks with certain steps. We update the parameters such that all tasks can have good performance after a few steps of adaptation.

We train the gating network G with the MAML algorithm and finetune on each individual out-of-domain dataset to report the final performance. Details are discussed in the sections below.

4 Experiments

4.1 Data

We utilize three in-domain datasets: the Stanford Question Answering Dataset (SQuAD) [28], Natural Questions [29], and NewsQA [30], all pre-processed in the same format as SQuAD. In addition, we use three out-of-domain datasets: DuoRC[31], RACE[32], and RelationExtraction[33]. All data is provided in the first place and please refer to the default project handout (robustQA)[24] for further detail.

4.2 Evaluation method

We evaluate model performance with Exact Match (EM) score and F1 score, and measure few-shot performance on out-of-domain tasks and comparison to in-domain task scores. This evaluation standard is set by the default project handout (robustQA)[24] as well as the leaderboard.

MAML Algorithm for MoE Training

```
Require: $\mathcal{D}$: a set of datasets
Require: $\alpha$, $\beta$: step size hyperparameters
1: randomly initialize $\theta$
2: while not done do
3:     Sample $n_D$ datasets $D_i \sim \mathcal{D}$
4:     Set meta-loss $\mathcal{L}_{meta} \leftarrow 0$
5:     for all $D_i$ do
6:         $\theta' \leftarrow \theta$
7:         Sample $N$ examples from the training set of $D_i$
8:         for $i$ in 1, ..., $K$:
9:             Evaluate the gradient $\nabla_{\theta'} \mathcal{L}_{D_i}(f_{\theta'})$ with respect to the $N$ examples
10:            Compute adapted parameters $\theta'$ with gradient descent
11:        Sample $K$ examples from the query set of $D_i$
12:        $\mathcal{L}_{meta} = \mathcal{L}_{meta} + \mathcal{L}_{D_i}(f_{\theta'})$
13:    end for
14:    Update $\theta \leftarrow \theta - \beta \nabla_\theta \mathcal{L}_{meta}$
15: end while
```

Table 1: Pseudocode for our MAML implementation

4.3 Experimental details

For training the base networks, we use the AdamW[34] optimizer with a learning rate of 3e-5 for all models for 3 epochs with a batch size of 16. We picked the best model based on in-domain validation performance to obtain a group of experts. Training takes 3 hours for each epoch for the concatenated dataset, and becomes slightly lower for each individual dataset.

For MoE training, we employ the same setup as previously discussed. We fix the weights of each individual expert and train the gating network G only. We observe that the network achieves decent metric numbers after only a few hundred steps, and the best performance is often achieved at around 2000 steps, which takes about two hours to train.

For using meta-learning for training the gating network, we use the `learn2learn` package[35] for the meta-learner that handles cloning and adaptation of the parameters, and write most code on our own for the meta-training loop. For meta-learning parameters, we use a meta-step learning rate β of 3e-5, and a domain-adaptation learning rate α of 1e-4. We choose the number of datasets to sample at each meta-step n_D to be 3, the batch size N to be 16, and the number of adaptation steps K within each meta-step to be 2. We train the gating network with MAML for 3 epochs, but we generally observe the best performance is obtained at around 2000 steps just like vanilla MoE training, but it takes about 8 hours. Note that MAML requires more time to train because each meta-step contains multiple steps of adaptation to multiple tasks. After MAML training, we finetune on each individual out-of-domain training sets for domain adaptation, which generally takes within 200 steps.

5 Results and Quantitative Analysis

In this section, we will discuss our results on the out-of-domain testing and development sets. We have achieved the F1 score of 60.8 and EM score of 42.2 on the RobustQA out-of-domain test set leaderboard with our MAML model discussed later in this section. The performance is among one of the best-performing models.

5.1 Results of base-model training

We first analyze the results of base-model training. Table 2 shows the performance of base models trained with different dataset and augmentation. Note that each base model is discussed in Section 3.2; we train different base models in order to select the candidates for experts used in the later MoE training.

		In-Domain		Out-Domain	
Training set	Augmentation	F1	EM	F1	EM
SQuAD	✗	**76.27**	**61.42**	41.25	24.35
	synonym	75.26	59.94	42.04	25.92
	swap	75.31	60.44	**42.63**	**28.27**
NewsQA	✗	**55.75**	**38.44**	38.61	**24.87**
	synonym	52.07	32.69	39.37	24.35
	swap	55.12	38.08	**39.57**	24.08
NatQuestions	✗	66.89	**50.83**	35.88	**20.16**
	synonym	66.81	49.73	34.89	19.90
	swap	**66.91**	49.52	**37.76**	19.90
Combined	✗	70.21	54.26	**50.81**	**34.82**
	synonym	**70.79**	**54.56**	44.87	28.27
	swap	70.26	53.70	46.65	30.10

Table 2: Performance of base models trained on different sets, with or without data augmentation.

We test the performance of each result model with in-domain and out-domain development set. Note that in-domain here refers to the same dataset as the training set (for example, if we train with SQuAD training set, the in-domain development set is SQuAD dev). For out-domain development set, we use the combination of RACE, RelationExtraction, and DuoRC. The "Combined" training set here is the one trained with all three in-domain training sets (SQuAD, NewsQA, and NatQuestions).

We first see that for in-domain testing, the best performance is achieved without augmentation, but augmentation tends to help with out-domain testing. We hypothesize that augmentation diversifies the in-domain data, making the model less prone to overfitting to the in-domain tasks. This would hurt in-domain testing performance but helps the model to generalize to novel tasks better.

Meanwhile, we also observe that synonym augmentation improves in-domain testing performance for the model trained on all three datasets, but augmentation hurts for out-domain sets. We think that the combined training set has far greater diversity than a single dataset, and the problem of overfitting to the training domain is less significant. We also observe that the model trained with the combined dataset without augmentation performs the best on out-of-domain development set. In the following discussion, we will refer to this model as the baseline we try to improve.

5.2 Effect of using Mixture-of-Experts

Next, we use different base models (or "experts") for mixture-of-expert training with the gating network G. Table 3 shows the performance of different MoE models, with the top row as the baseline model mentioned in Section 5.1.

We hypothesize that the models in section 5.1 all overfit to their corresponding training set to some extent, so we choose different base models as our experts in this section. We first choose three models trained without augmentation, one from each dataset, to construct the set of three experts. We the train the gating network with these three networks and refer to it as MoE_3 in the table. We made another experiment by adding the baseline model into the experts and call the MoE model MoE_4. The last experiment is replacing each individual model with the one trained with the swapping augmentation, but keeping the baseline model unchanged. We call this model $\text{MoE}_{4,swap}$.

We first observe that none of the MoE models perform better than the baseline on in-domain development set. It is reasonable because the baseline network has a bias towards in-domain tasks; adding weaker experts to the baseline might not help with in-domain testing. However, we do observe that MoE_4 improves out-of-domain development set performance. We believe that with more experts, the model is less prone to overfitting to achieving a better in-domain performance. This technique can be the source of the model's ability to generalize to unseen domains. $\text{MoE}_{4,swap}$ further improves the performance of MoE_4 by using base models that are more robust to out-of-domain tasks. As shown in Table 2, base models trained on augmented single tasks tend to perform better outside the training domain.

However, we do observe that MoE$_3$ performs much worse than the baseline. We hypothesize that each of the three models might be overfitting to their own training tasks because of the lack of diversity and enough training data, which makes it hard for the gating network to form good representation for the downstream task. Adding the baseline network helps alleviate this form of overfitting to individual datasets.

	In-Domain		Out-Domain	
Model	F1	EM	F1	EM
Base$_{Combined}$	**70.21**	**54.26**	50.81	34.82
MoE$_3$	68.73	53.02	42.78	28.01
MoE$_4$	70.19	54.23	51.92	35.34
MoE$_{4,swap}$	68.42	51.81	**52.00**	**36.13**

Table 3: Performance of MoE training with different base models, with or without data augmentation.

5.3 Meta Learning for training the Gating Network

We then explore using meta-learning for MoE training. As discussed in Section 3.5, for meta-learning training we need a set of datasets with the training set used in adaptation and the query set used for meta-adaptation. For each of the in-domain datasets, we use its training set as the training set in Algorithm 1 and the development set as the query set. When we use out-of-domain datasets, we set their training sets as query set instead because we need the development set for the purpose of evaluation.

We explore two different meta-learning setups: one with the three in-domain tasks ($\mathcal{D}_{in}$ = {SQuAD, NewsQA, NatQuestions}), and another with the three in-domain tasks and three out-domain tasks ($\mathcal{D}_{out}$ = {RACE, RelationExtraction, DuoRC}). We denote them as MAML$_{\mathcal{D}_{in}}$ and MAML$_{\mathcal{D}_{in}+\mathcal{D}_{out}}$ in table 4.

In Table 4, we observe that using MAML with finetuning on individual out-of-domain training sets improves the performance on all metrics to a great extent. This proves our hypothesis that MAML can learn to adapt to unseen tasks really well by learning to learn during the meta-training phase. We also observe that when we bring out-of-domain tasks into meta-training, even if the training set is small and we use the training set also as the query set, the performance on out-of-domain development set is improved greater. This technique might help the model handling the three out-of-domain datasets better during domain adaptation.

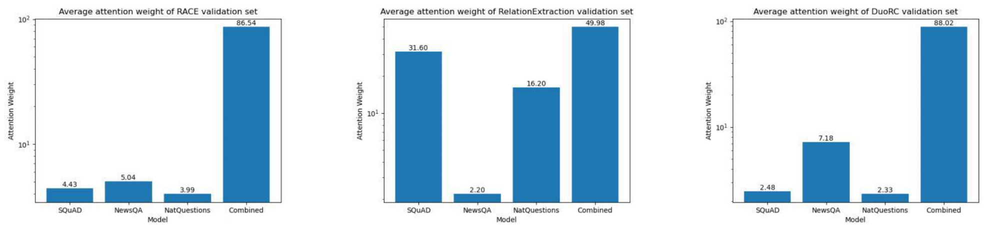
However, when we submitted the best-performing MAML model trained on all 6 tasks to the testing leaderboard, we obtained an F1 score of 58.59 and EM of 39.66, which is lower than we expected. We hypothesize that because the training and development sets of the out-of-domain datasets are very small, there might be a gap between these sets and the larger test set. Then we used the MAML model trained on in-domain set only, MAML$_{\mathcal{D}_{in}}$, and lowered the number of domain adaptation steps to around 100. This model achieves the testing performance of F1 of 60.80 and 42.25, which is among the best-performing models. Therefore, the second model that uses the tiny out-of-domain training set might be "overfitting to the development set" although not using the data from the development set. The first MAML model, trained on the three large in-domain dataset, might have a greater capability in generalization.

	RACE		RelExtr		DuoRC		Total	
Model	F1	EM	F1	EM	F1	EM	F1	EM
MAML$_{\mathcal{D}_{in}}$	37.53	23.44	73.14	53.12	**46.40**	**35.71**	52.39	37.43
MAML$_{\mathcal{D}_{in}+\mathcal{D}_{out}}$	**38.67**	**26.56**	**76.65**	**56.25**	46.09	34.92	**53.84**	**39.27**

Table 4: Performance of MoE models trained with the MAML algorithm.

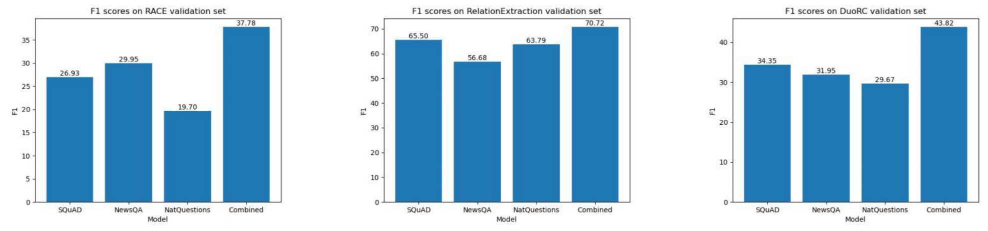
6 Qualitative Analysis

Having observed that the mixture-of-expert training framework improves out-of-domain QA testing performance, a question rises: what have the MoE models learned? In this section, we visualize the attention weights in the gating network to see how it constructs the final embedding from the knowledge of different experts.



a-1 a-2 a-3

(a) Validation F-1 scores



b-1 b-2 b-3

(b) Attention weights

Figure 2: Comparing between the validation F1 scores of each base-model ("expert") and the learned attention weights from each gating network.

We use the model MAML$_{\mathcal{D}_{in}}$ to construct Figure 2. Figure 2(a) shows the validation F1 scores of each individual experts on each out-of-domain tasks. 2(b) shows the weights the gating network puts on each model embedding. We see a correspondence between the attention weights and model performance, especially for RACE and RelationExtraction. In addition, we observe that the output of RACE and DuoRC depends more on the baseline "Combined" model (which performs the best), this can be due to the F1 score of each expert trained on individual datasets to be too low. For RelationExtraction, however, the gating network learns to depend less on the baseline model and use the other models to reduce overfitting.

7 Conclusion

In conclusion, we find that Mixture-of-Experts (MoE)[1] works well on improving the performance of DistilBERT[22] model on the robust QA task along with data augmentation[25] including synonym replacement and random swap, and meta-learning including MAML algorithm[11] for domain adaptations. As a result, we successfully increase F-1 score from 50.81 to 53.84 and exact match (EM) score from 34.82 to 39.27 in out-of-domain validation performance. Eventually, we manage to achieve a F-1 score of 60.8 and EM score of 42.2 on the out-of-domain QA testing leaderboard. The primary limitation of our work is that due to limited time, we have not experimented more types of meta-learning algorithms, parameters and data augmentation techniques. Also, due to limitation of GPU capability, we are unable to perform meta-learning with more adaptation steps. Future directions to improve our work could be experimenting more meta-learning algorithms and parameters, and utilizing the attributes of data in the datasets to form more tasks and scale the number of in-domain and out-of-domain datasets.

8 External Collaborator

We have consulted Jerry Zhenbang Tan (ztan035@stanford.edu) regarding meta-learning in our proposal and training result analysis in our milestone report. However, Jerry has not contributed to the final report.

References

[1] S. Nowlan R. Jacobs, Michael I. Jordan and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation: 3:79–87*, 1991.

[2] Jiahuan Pei, Pengjie Ren, and Maarten de Rijke. A modular task-oriented dialogue system using a neural mixture-of-experts. *CoRR*, abs/1907.05346, 2019.

[3] Phong Le, Marc Dymetman, and Jean-Michel Renders. Lstm-based mixture-of-experts for knowledge-aware dialogues. *CoRR*, abs/1605.01652, 2016.

[4] Yan Xu, Ran Jia, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. Improved relation classification by deep recurrent neural networks with data augmentation. *CoRR*, abs/1601.03651, 2016.

[5] Marzieh Fadaee, Arianna Bisazza, and Christof Monz. Data augmentation for low-resource neural machine translation. *CoRR*, abs/1705.00440, 2017.

[6] S. Yu, J. Yang, D. Liu, R. Li, Y. Zhang, and S. Zhao. Hierarchical data augmentation and the application in text classification. *IEEE Access*, 7:185476–185485, 2019.

[7] Jasdeep Singh, Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. XLDA: cross-lingual data augmentation for natural language inference and question answering. *CoRR*, abs/1905.11471, 2019.

[8] Toan Tran, Trung Pham, Gustavo Carneiro, Lyle J. Palmer, and Ian D. Reid. A bayesian data augmentation approach for learning deep models. *CoRR*, abs/1710.10564, 2017.

[9] Zhucheng Tu Chris DuBois Shayne Longpre, Yi Lu. An exploration of data augmentation and sampling techniques for domain-agnostic question answering. *arXiv:1912.02145*, 2019.

[10] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 856–865, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[11] Sergey Levine Chelsea Finn, Pieter Abbeel. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv:1703.03400*, 2017.

[12] Wenpeng Yin. Meta-learning for few-shot natural language processing: A survey, 2020.

[13] 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.

[14] Fei Mi, Minlie Huang, Jiyong Zhang, and Boi Faltings. Meta-learning for low-resource natural language generation in task-oriented dialogue systems. *CoRR*, abs/1905.05644, 2019.

[15] Trapit Bansal, Rishikesh Jha, Tsendsuren Munkhdalai, and Andrew McCallum. Self-supervised meta-learning for few-shot natural language classification tasks, 2020.

[16] Zi-Yi Dou, Keyi Yu, and Antonios Anastasopoulos. Investigating meta-learning algorithms for low-resource natural language understanding tasks. *CoRR*, abs/1908.10423, 2019.

[17] Trapit Bansal, Rishikesh Jha, and Andrew McCallum. Learning to few-shot learn across diverse natural language classification tasks. *CoRR*, abs/1911.03863, 2019.

[18] John Schulman Alex Nichol, Joshua Achiam. On first-order meta-learning algorithms. *arXiv:1803.02999*, 2018.

[19] Wenpeng Yin, Nazneen Fatema Rajani, Dragomir Radev, Richard Socher, and Caiming Xiong. Universal natural language processing with limited annotations: Try few-shot textual entailment as a start, 2020.

[20] Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. Few-shot natural language generation for task-oriented dialog, 2020.

[21] Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*, 2020.

[22] Julien Chaumond Victor Sanh, Lysandre Debut and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108,*, 2019.

[23] Distilbert. `https://huggingface.co/transformers/model_doc/distilbert.html`. Accessed: 2021-03-05.

[24] Cs 224n default final project: Building a qa system (robust qa track). `http://web.stanford.edu/class/cs224n/project/default-final-project-handout-robustqa-track.pdf`. Accessed: 2021-03-05.

[25] Makcedward. makcedward/nlpaug.

[26] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. Ppdb: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764, 2013.

[27] Dropout. `https://www.tensorflow.org/api_docs/python/tf/keras/layers/Dropout`. Accessed: 2021-03-05.

[28] Konstantin Lopyrev Pranav Rajpurkar, Jian Zhang and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR, abs/1606.05250*, 2016.

[29] Olivia Redfield Michael Collins Ankur 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 Tom Kwiatkowski, Jennimaria Palomaki and Slav Petrov. Natural questions: a benchmark for question answering research. *Association for Computational Linguistics (ACL)*, 2019.

[30] Xingdi Yuan Justin Harris Alessandro Sordoni Philip Bachman Adam Trischler, Tong Wang and Kaheer Suleman. Newsqa: A machine comprehension dataset. *ACL 2017, page 191*, 2017.

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

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

[33] Eunsol Choi Omer Levy, Minjoon Seo and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. *arXiv preprint arXiv:1706.04115*, 2017.

[34] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv:1711.05101*, 2017.

[35] learn2learn. `https://github.com/learnables/learn2learn`. Accessed: 2021-03-05.