Meta-Learning and Data Augmentation for Robust Question Answering

Stanford CS224N Default Project (RobustQA)
TA mentor: Allan Zhou, No external collaborators, external mentor, or sharing project.

Zhiyin Lin
Department of Computer Science
Stanford University
`zhiyinl@stanford.edu`

Beining (Cathy) Zhou
Department of Computer Science
Stanford University
`cathyzbn@stanford.edu`

Abstract

Question Answering systems has proved success in answering in-domain queries. However, it suffers from poor generalizations in out-of-distribution contexts. In this paper, we use both data augmentation and meta-learning to address this problem. Starting from a DistilBERT baseline, we provide an ablation study of 4 data augmentation methods and 3 data split variations of the reptile algorithm in improving OOD generalizability. Among all augmentations, random deletion performs the best (F1 = 57.54, EM = 39.15 on test leaderboard). For meta-learning, using our tweak of adding different data in each iteration under the Even Split data split produces the best results, an F1 score 57.99 and an EM score 41.33 on the test leaderboard.

1 Introduction

An user asks a question on a piece of text. A question answering (QA) system reads it and answers. In particular, the model answers by identifying a span of text in the provided passage (i.e. a direct quote). This is a classic yet challenging natural language understanding task and serves as common benchmark for language model evaluation.

The state-of-the-art models such as IE-Net and SA-Net have achieved better scores than human performance [1]. However, similar to many other modern machine learning models, QA models suffer from over-fitting to given dataset and thus generalize poorly to out-of-domain (OOD) problems. This paper focuses on improving OOD performance of QA models and making them robust and trustworthy. To do so, we investigate meta-learning methods and four data augmentation operations.

2 Related Work

Meta-learning roots on the idea of "learn to learn". It trains across a variety of tasks to develop a model that is quick at adapting to a new task given few samples. Metaphorically speaking, it aims to find the best en-garde position for the model to attack a problem. With its flexibility and quickness in adapting, its shows promise in solving domain generalization problems. Since its MAML's initial release in 2017 [1], meta-learning has grabbed great attention from the AI community. Indeed, meta-learning has proven great success in a wide variety of fields. With MAML, Finn et al. outperformed state-of-the-art on statistical regression problems, computer vision (image classification using Omniglot and MiniImagenet), and reinforcement learning. Similar success is observed from Reptile [2], a simpler, less expensive version of MAML. In the natural language context, Dou et al. have experimented with meta-learning on low-resourced tasks (General Language Understanding Evaluation Benchmark) and showed that Reptile outperforms MAML, yet they reached only slightly

[1]https://rajpurkar.github.io/SQuAD-explorer/

better performance than a BERT baseline [3]. To build generalizable NLP classification models, Bansal et al. proposed LEOPARD, a meta-learning method that allows different number of classes across task [4].

Meanwhile, data augmentation has shown impressive results on improving robustness, most notably in computer vision. The augmented data increases the size of the training sample and may help to reduce biases. Here, we explore easy data augmentation proposed by Wei and Zou [5] which has been extensively used in NLP tasks such as contrastive self-supervised NLP model [6] and adversarial attacks [7]. In this paper, we will investigate the best combination of operations and hyperparameters to boost robust performance.

3 Approach

Baseline. We use the pre-trained DistilBERT [8] with AdamW optimizer [9] as baseline. The DistilBERT is a smaller version than the original BERT, contains 66 rather than 340 million parameters.

Data Augmentation. We consider four sentence-wise data augmentation techniques proposed by Wei and Zou in 2019 [5].

- Synonym Replacement (SR): Replace n words (not stop word) with a random synonym.
- Random Insertion (RI): For n times, insert a random synonym of a random word (non-stop word) into a random position.
- Random Swap (RS): Choose two random words in the sentence and swap their positions.
- Random Deletion (RD): Randomly remove each word in the sentence with probability p.

During augmentation, We prepossessed the words by converting all to lower-case and removing numbers and punctuations to ensure the consistency of the language structure. We used the synonym dictionary NLTK WordNet [10] for Synonym Replacement. For Random Insertion, stop words are propositions and pronouns including "she", "and", and "at". We adapt the skeleton implementation [2] extensively to QA system. For each question-answer pair, we only implement augmentation on the context.

Table 1: Contexts generated by data augmentation. SR: synonym replacement. RI: random insertion. RS: random swap. RD: random deletion. Example from Relation Extraction dataset. Each operation has $\alpha = 0.3$.

Operation	**Context**
Original	Ray Eberle died of a heart attack in Douglasville, Georgia on August 25, 1979, aged 60.
Synonym Replacement	beam eberle died of a heart flack in douglasville georgia on revered cured.
Random Insertion	ray along pass away eberle died of a heart hoosier state attack in douglasville georgia re on august aged.
Random Swap	august on a died of heart attack in douglasville georgia eberle ray aged .
Random Deletion	ray a heart attack douglasville august.

Meta-learning We implement Reptile proposed by Nichol et al. [2], which is simpler and faster (while yielding comparable results) than the standard meta-learning approach MAML [1]. Motivated by the idea of "learn to learn", Reptile learns a parameter initialization that is quick at finetuning to a new task. This shows its promise in few-shot learning settings since it trains to find a quick learner, which is saved for use later. Starting from an initialization ϕ, it repeatedly samples task τ from the meta-train set (in standard setting, each task is from a different domain), performs k steps of SGD or Adams to obtain a new initialization $\tilde{\phi}$, and updates the initial parameters ϕ in the direction of $\tilde{\phi} - \phi$ just learned on the task.

[2]https://github.com/jasonwei20/eda$_n lp/$

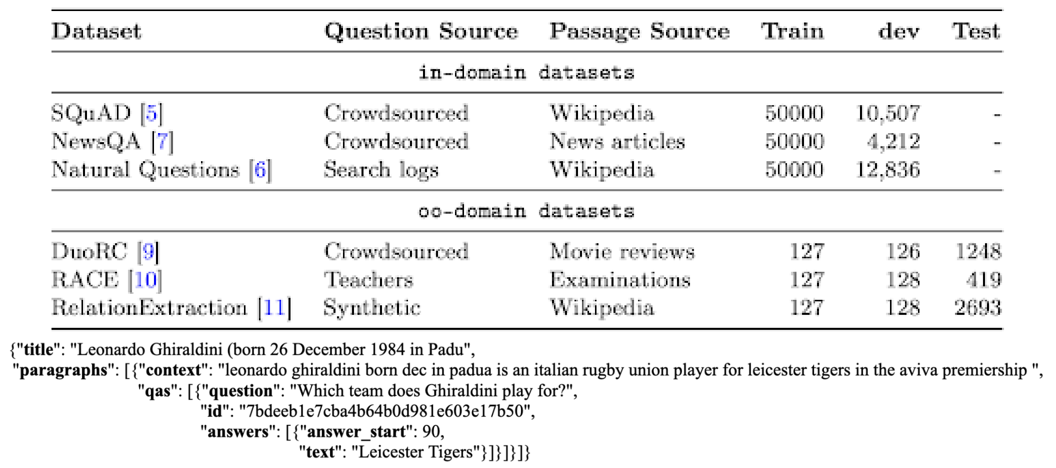
Algorithm 1 The Reptile Algorithm [2]

```
Initialize $\phi$, the vector of initial parameters
for iteration = 1, 2, . . . do
    Sample task $\tau$, corresponding to loss $L_\tau$ on weight vectors $\tilde{\phi}$
    Compute $\tilde{\phi} = U_\tau^k(\phi)$, denoting k steps of SGD or Adams
    Update $\phi \leftarrow \phi + \epsilon(\tilde{\phi})$.
end for
```

4 Experimental Setup

4.1 Data

We use six question-answering datasets, including three large in-domain datasets with a 50,000 sample training size (Natural Questions [11], NewsQA [12], and SQuAD [13]) and three small out-of-domain datasets with a 127 sample training size (Relation Extraction [14], DuoRC [15], and RACE [16]). Datasets are processed to match the format of SQuAD.



Dataset	Question Source	Passage Source	Train	dev	Test
		in-domain datasets			
SQuAD [5]	Crowdsourced	Wikipedia	50000	10,507	-
NewsQA [7]	Crowdsourced	News articles	50000	4,212	-
Natural Questions [6]	Search logs	Wikipedia	50000	12,836	-
		oo-domain datasets			
DuoRC [9]	Crowdsourced	Movie reviews	127	126	1248
RACE [10]	Teachers	Examinations	127	128	419
RelationExtraction [11]	Synthetic	Wikipedia	127	128	2693

```
{"title": "Leonardo Ghiraldini (born 26 December 1984 in Padu",
 "paragraphs": [{"context": "leonardo ghiraldini born dec in padua is an italian rugby union player for leicester tigers in the aviva premiership ",
                 "qas": [{"question": "Which team does Ghiraldini play for?",
                          "id": "7bdeeb1e7cba4b64b0d981e603e17b50",
                          "answers": [{"answer_start": 90,
                                       "text": "Leicester Tigers"}]}]}]}
```

Figure 1: Summary statistics of 3 in-domain datasets and 3 out-of-domain datasets (borrowed from CS224N Project Instruction) and an example datapoint from RelationExtraction Dataset

Datasets are processed to match the formatting of SQuAD. See Figure 4.1 for a snapshot of a processed sample datapoint from RelationExtraction dataset.

We performed our experiments on Azure's Standard NC6s v3 with 6 vcpus and 112 GiB memory.

4.2 Evaluation Method

We evaluate with two metrics: the stricter Exact Match (EM) score returns 1 if the model's prediction matches the ground truth word-by-word, and 0 otherwise, and the more forgiving F1-score gives a harmonic mean of precision and recall.

We evaluated all models on the validation sets of three out-of-domain, given that test sets ground truth are hidden from students. We evaluate with the maximum score across three given human answers to each question to accommodate multiple correct answers with the same meaning.

4.3 Primal Experiments

Baseline. We train the pre-trained DistilBERT on three in-domain dataset and test it on three OOD test sets. Due to the large size of in-domain datasets, the model is trained with 3 epochs and evaluated

Table 2: Data use for experiments. For meta-learning experiments, train set represents meta_train and train_val set represents meta_val. Meta_train and meta_val are disjoint samples from ID_train. DA stands for data augmentation.

	Pretrain	**Train**	**Train_Val**	**Finetune**	**Finetune_Val**	**Test**
Baseline	DistilBERT	ID_train	ID_val	/	/	OOD_test
Finetune	DistilBERT	ID_train	ID_val	OOD_train	OOD_val	OOD_test
DA	DistilBERT	ID_train	ID_val	OOD_train_aug	OOD_val	OOD_test
Meta Baseline	DistilBERT	ID_train	ID_train	/	/	OOD_test
Meta Finetune	DistilBERT	ID_train	ID_train	OOD_train	OOD_val	OOD_test
Meta + DA	DistilBERT	ID_train	ID_train	OOD_train_aug	OOD_val	OOD_test

every 2000 steps. At train time, the maximum length of a prediction is set to 15. The implementation is provided by the starter code.[3] The baseline achieves a 49.88 F1-score and a 34.55 EM score.

Baseline Finetuned on OOD. We continued to finetune the above baseline on the three out-of-domain training sets. This simple finetune boosts the model's performance to a 50.83 F1-score and a 36.91 EM score. This is consistent with the intuition that even after a small portion of the examples from the new domain, the model's performance improved significantly.

Roadmap Forward Here, we provide an overview of data uses for the following experiments in Table 2.

5 Data Augmentation: an Ablation Study

Unless otherwise specified, we ran all experiments with seed 42, learning rate 3e-5, number of epochs 5, and batch size 16. In training, we evaluated the models every 10 steps on the validation set, and the best model is saved as the final model.

5.1 What Percentage of Text Should an Augmentation Change Per Context?

Following the original paper, we examine each augmentation method independently[5] on the three OOD training sets. We provide an ablation study on the augmentation parameter $\alpha = \{0.05, 0.1, 0.2, 0.3, 0.4\}$, the percent of words changed by each augmentation per context, for each of the four methods while fixing one augmented paragraph per original context. The results are presented in 2.

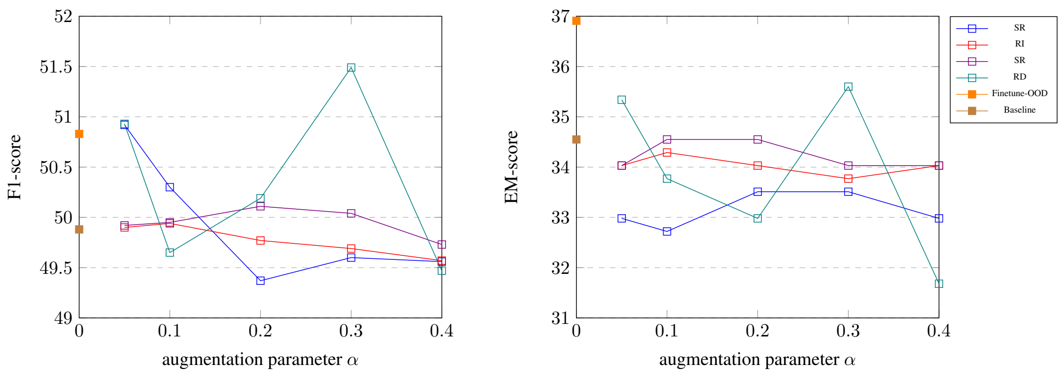


Figure 2: F1-score and EM-score of four data augmentation operations, Finetune-OOD, and Baseline. SR: synonym replacement. RI: random insertion. RS: random swap. RD: random deletion

[3]https://github.com/michiyasunaga/robustqa

Results. All four data augmentation operations show improvements in F1 score from the baseline. For SR, small $\alpha = 0.05$ improved the performance significantly while a larger $\alpha >= 0.2$ hurt the performance. Indeed, replacing an overly high percent of words with synonyms may alter a sentence's meaning. RI shows a smaller performance boost peaking at $\alpha = 0.1$ followed by a flat decline for larger α. Similarly, RS slightly improves upon baseline at $\alpha = 0.2$ and declines afterwards. We hypothesize that over-insertion or -swap is likely to change the entire sentence structure, which is consistent with its declining performance for larger α. Moreover, both RI and RS shows a flatter curve in response to varying α compared to SR and RD, possibly because the nature of random insertion and random swap only adds to (or switch positions of) original sentence's words but does not omit or alter the words original sentence. Retaining original words gives a smaller but more stable impact on F1-score compared to more radical methods. RD achieves highest performance F1-score 51.49 at $\alpha = 0.3$, which is also the best F1-score across all experiments in this ablation study. However, its curve is the most rocky when varying α. This corresponds largely with which words are deleted in a context by randomness. Despite the optimal α differs for each methods, $\alpha = 0.4$ consistently gives below-baseline performance for all data augmentation operations.

EM score's general trend matches that of the F1-score. However, instead of outperforming the baseline at most experiments in F1-score, only RD at $\alpha = 0.05$ and $\alpha = 0.3$ outperforms baseline EM. Moreover, all the data augmentation gives an EM well below the finetune-OOD method. This makes sense because augmentation works by reasonably perturbing the original data, so obtaining an exact match should be harder with augmentation than without, but the more lenient F1-score should show improvement.

Table 3: F1-score and EM-score of $N_{aug} = \{1, 2, 4, 8, 16\}$ for the four data augmentation operations, each with best performing augmentation parameter α. Optimal performance of each method is bolded for F1-score and italicized for EM score. The best performance across all experiments is marked in red.

N_{aug}	1		2		4		8		16	
Operation	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM
SR-alpha-0.05	50.92	32.98	50.32	***34.55***	**51.33**	34.03	49.98	30.63	50.44	31.94
RI-alpha-0.1	49.94	***34.29***	**50.76**	32.46	49.97	***34.29***	49.52	33.77	50.37	34.03
RS-alpha-0.2	50.11	34.55	**50.58**	***34.82***	49.58	33.25	49.95	34.29	50.37	33.25
RD-alpha-0.3	**51.49**	***35.60***	49.69	31.94	49.43	33.51	50.56	34.82	50.04	34.55

5.2 How Many Augmented Context Per Original Context?

The natural question is: how many augmented data should we use per original data? We provide a ablation study on n_{aug}, the number of generated augmented contexts per original context, for each of the four data augmentation operations, while using the optimal augmentation parameter α shown in Figure 1. Details of the performances are summarized in Table 1. Both EM and F1 scores show fluctuating performances when varying N_{aug}. But data augmentation generally achieves slightly better performance for $N_{aug} = \{1, 2, 4\}$, probably because larger N_{aug} could generate repetitive data and lead to overfit.

6 Meta-learning: Three Main Variations

6.1 Experimental Design

When implementation our version Reptile, we considered both Facebook's Higher, an open-source MAML written in TensorFlow, and Huang's implementation of Reptile in PyTorch https://github.com/gabrielhuang/reptile-pytorch/blob/master/reptile$_sine.ipynb$.

Meta-learning is usually applied to few-shot domain adaption, which is similar to our case. However, we aim to develop a model that performs well on all three tasks after adaption rather than a singular task. Therefore, we investigated three different data splits for the meta-train and meta-validation steps to adapt meta-learning our problem. For both train and validation, we use data from the three ID train sets: Nat Questions, NewsQA, and SQuAD.

For each experiment, we use the following hyperparameters: number of epochs(epochs)= 5, number of iterations (inner_iter_n) =5, meta-learning rate (inner_lr)=0.1, sample size in each task (sample_size)=18, and number of tasks = 21. In Split by Task, task_num is set to 3 because there are only three ID datasets, each is used as one task. The data for meta-train differs for different tasks, but the data for meta-val is the same for each batch to ensure consistency. Sample_size is limited to 18 because of the limited memory of our GPU.

- **Even Split.** In each task, both train and validation an even number ($6 \times 3 = 18$ samples) of data from all three of the ID datasets. This ensures that the ratio of data is analogous to the test data on OOD consisting of 3 datasets of 127 samples each. It also uses all three domains to fully exploit the diverse data provided.
- **Separated.** In each task, train contains $9 \times 2 = 18$ samples of data from two datasets, NewsQA and SQuAD. Val contains 18 samples from Nat Questions. This designs allows the validation set to be of different distribution than the train, so that it serves as a out-of-domain example during meta-train
- **Split by Task.** Each task focuses on data from only one dataset. Train and validation both takes in 18 samples. The model trains by looping over 3 tasks representing 3 datasets.

For each of the 5 inner iterations within one task, the conventional meta-learning algorithms takes in the same set of data for each iteration. However, we experimented both methods of taking in 1) **repeated** data and 2) **different** data. The different data completes training 5 times faster than the repeated method yet loops through the data 5 times less, so we also increased the number of epochs by 5.

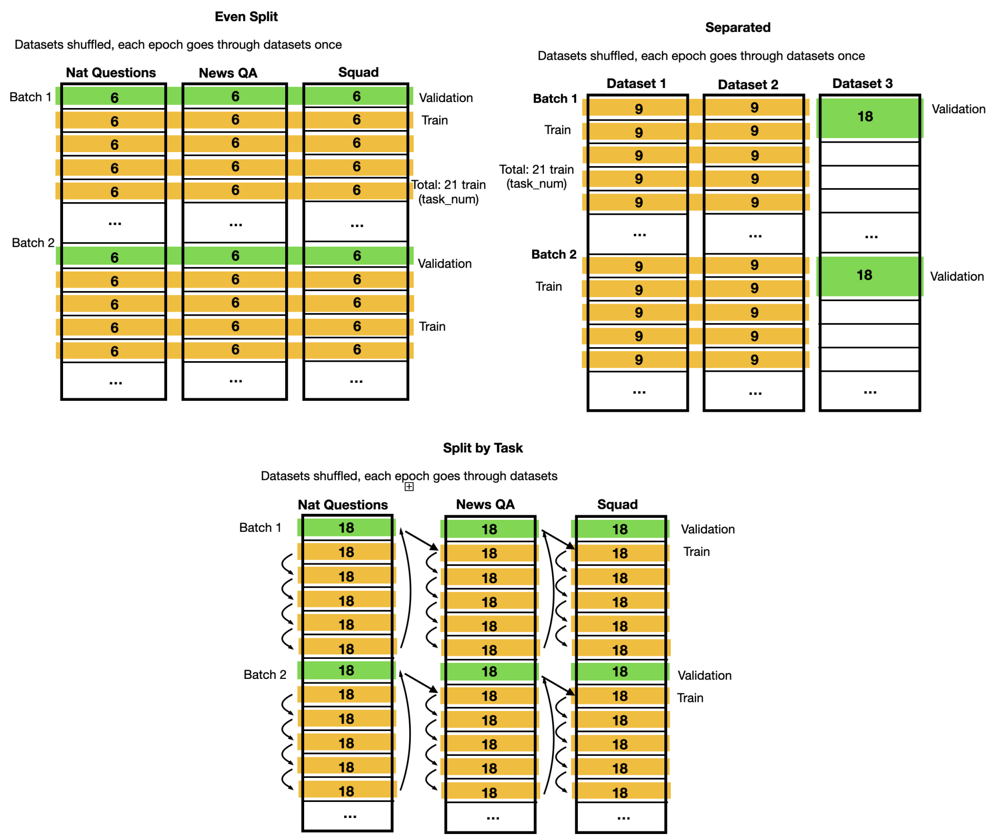

Even Split
Datasets shuffled, each epoch goes through datasets once
Nat Questions
News QA
Squad
Batch 1
Batch 2
Validation
Train
Total: 21 train (task_num)
Separated
Datasets shuffled, each epoch goes through datasets once
Dataset 1
Dataset 2
Dataset 3
Split by Task
Datasets shuffled, each epoch goes through datasets

Figure 3: Design of three data splits

6.2 Results and Analysis

6.2.1 Data Splits

We designed the three data splits each of a different intent. The even split matches the data types of the OOD and provides the most diversity for each task; the separation examines a validation set that serves at OOD examples; the split by task allows the model to jump to different distributions for each task.

Table 4: F1-score and EM-score of meta-learning experiments. The optimal performance of each meta setup (i.e. Baseline, Finetune OOD)is bolded for F1-score and italicized for EM score. Overall best performance is marked in red.

Data Split	Inner Iteration	Meta Baseline		Meta Finetune OOD	
		F1 Score	EM Score	F1 Score	EM Score
Even Split	Repeated	44.04	29.06	45.93	30.89
	Different	38.79	21.73	43.53	29.06
	Different (Epoch = 15)	**47.13**	*30.63*	**48.04**	*34.82*
Separation	Repeated	43.25	26.18	45.58	31.68
	Different	35.71	19.11	43.33	29.58
	Different (Epoch = 15)	42.06	25.65	45.75	31.41
Split by Task	Repeated	16.71	35.80	7.07	23.56

Even Split. Even split showed the best performance among the three types of data splits. Although meta-learning is usually implemented without mixing different tasks, the even split matches the structure of the objective OOD tasks given at test time. Furthermore, since it exploits all three datasets compared to the separation method, it provides more diversity to training, which could be the factor that caused the edge over other methods.

Separation. The intuition of allocating a separate distribution for the validation set to simulate OOD in meta-validation did not work as anticipated, and it performed worse than the even split in almost all aspects. This may be due to the reduction from 3 different types of distributions to 2, or it may be due to the fact that meta-learning focuses on picking up information specific to the validation distribution. If given more ID datasets, we could have performed a more rigorous analysis. Also, this method showed comparatively high scores in EM and a more significant increase after finetune.

Split by Task. This method generated the least favorable performance. This may be due to the nature of the task given: at test time, OOD data is mixed.

6.2.2 Inner Iteration Data

Repeated feeding This method is the standard implementation for meta-learning, as it performs a k=5 step gradient update within each task, as provided in the implementation by Higher. However, it is slow — it roughly takes 12 hours to train with sample_size = 18 on our Azure Standard NC6s v3 with 6 vcpus and 112 GiB memory.

Different Feeding We tweaked the original method by feeding different data into each step of the inner iteration. This enlarges the data taken by one task by 5 folds. Because the data was exploited 5 times less, we increased the number of epoches to 15, which is 5 times the size of the original. Surprisingly, this method improved the F1 score by 2-3 for the experiments we performed. This proved that feeding diverse data, even in the inner iteration steps, helps to boost performance.

Overall, model may have some success in finding good initialization, yet still did not beat the baseline. Our highest score was roughly the same as the baseline. Perhaps, meta-learning is limited in tasks with different OOD tasks combined at test time. However, we did gain many insights on the data splits and

6.3 Combining Data Augmentation and Meta-Learning

Overall, data augmentation did not prove to be helpful when combined with meta-learning. As shown in 5, most of the F1 and EM scores were lower than those when finetuned on the OOD dataset without augmentation. Only synonym replacement resulted in small performance increase, yet it could be due to random chance. In addition, when examining the loss during finetuning, the loss decreased in many cases as more epochs of finetuning were performed. This provides an illustration that the augmentation and augmentation parameters may be very model-dependent, and they may not generalize well to other tasks.

Table 5: F1-score (a) and EM-score (b) of meta-learning experiments with data augmentation finetune. The best hyperparameter combination (alpha, N_{aug}) is used for each data augmentation operations: SR (synonym replacement), RI (Random Insertion), RS (Random Swap), RD (Random Deletion). The optimal performance of each data augmentation operation is bolded for F1-score and italicized for EM score. Overall best performance is marked in red.

Data Split	Inner Iteration	SR	RI	RS	RD
Even Split	Repeated	45.57	44.84	44.00	44.83
	Different	42.69	41.08	41.28	40.22
	Different (Epoch = 15)	**48.11**	*46.99*	**47.49**	*46.86*
Separation	Repeated	43.31	44.09	43.98	43.50
	Different	37.93	37.81	37.24	36.82
	Different (Epoch = 15)	43.59	43.35	43.37	42.64

(a) F1 score

Data Split	Inner Iteration	SR	RI	RS	RD
Even Split	Repeated	29.06	29.06	28.80	26.96
	Different	24.61	23.30	23.04	22.51
	Different (Epoch = 15)	**31.41**	*29.84*	**31.41**	*30.37*
Separation	Repeated	26.18	25.92	26.44	26.18
	Different	20.94	20.94	19.11	19.11
	Different (Epoch = 15)	26.96	24.87	23.82	26.18

(b) EM score

7 Future Directions

Data augmentation methods has shown improvement, yet some augmentation techniques, especially random deletion, is prone to rocky behaviors, while others are more stable. We hope to investigate the reason behind these behaviors. Are there certain factors, such as parts of speech or the position of a word in sentence, that make certain augmentation methods perform better? For instance, does deleting a verb and a proposition have the same affect on the performance?

We wish to investigate more high-level ideas related to meta-learning, both its performance on the problem at hand and on other data. We could compare the performance compare performance when using each of 3 ID dataset as the meta-validation dataset in the separated data split. Would meta-validation dataset containing sources from Wikipedia be more effective when tested on OOD which also contains data drawn from Wikipedia? This may provide us insights into the differences in each dataset and the degree that meta learning could accommodate those differences.

For meta-learning in general, we hope to investigate what meta-learning could achieve on different dataset setups. With more diverse in-domain tasks such as more types of in-domain datasets, would meta-learning yield better results? Would meta-learning perform better than the baseline methods in a fewer-shot learning environment? We are excited to see the future of this method.

8 Conclusion

This paper provided an ablation study on data augmentation, and experimented with data splits and inner loop data variations for meta learning to improve the robustness of Question Answering System on out-of-domain tasks. Overall, data augmentation, particularly random deletion, showed improvement in performance. Meta-learning methods also reach comparable performance to the baselines. Moreover, it provides far more interesting insights than simple results. An even data split approach performs the best, possibly because it provides diverse data and matches the structure of the test data. Moreover, our tweak of using different data in inner iterations of meta-train shows better result than the conventional method, as it allows the model to see more data in each task.

Previously, meta-learning has shown great success in fields such as statistical regression problems, computer vision, and reinforcement learning. Nevertheless, we have not seen a comparable success when it comes to natural language processing. In this paper, we implement meta-learning algorithms in various ways directly on top of a DistilBERT, but the results have not been the most satisfactory.

So we ask: why is natural language processing so special? Here, we have two reasons. First, natural language is of a special form that encodes complicated human logic and well-crafted structure into a linear array of words. Compared to other data like images, language does not translate naturally to numbers; we need to artificially force an embedding that translate texts into numerical representations. These extra operations are successful, but they also add more complexity to the model. "How to learn NLP problem" is much more complicated than for example "how to learn about regression" for meta-learning. So tackling NLP with meta-learning at a level that directly on top of a million-parameter model might not be the best choice. Going forward, we wonder if meta-learning at more granular level could help with this issue, such as incorporating meta-learning designs right at the embedding, attention layer, or adding some specific designs on meta-learning algorithms from a language perspective.

References

[1] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

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

[3] Zi-Yi Dou, Keyi Yu, and Antonios Anastasopoulos. Investigating meta-learning algorithms for low-resource natural language understanding tasks. *arXiv preprint arXiv:1908.10423*, 2019.

[4] Trapit Bansal, Rishikesh Jha, and Andrew McCallum. Learning to few-shot learn across diverse natural language classification tasks. *arXiv preprint arXiv:1911.03863*, 2019.

[5] Jason Wei and Kai Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*, 2019.

[6] Hongchao Fang, Sicheng Wang, Meng Zhou, Jiayuan Ding, and Pengtao Xie. Cert: Contrastive self-supervised learning for language understanding. *arXiv preprint arXiv:2005.12766*, 2020.

[7] John X Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. *arXiv preprint arXiv:2005.05909*, 2020.

[8] 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.

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

[10] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[11] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

[12] Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*, 2016.

[13] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

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

[15] Amrita Saha, Rahul Aralikatte, Mitesh M Khapra, and Karthik Sankaranarayanan. Duorc: Towards complex language understanding with paraphrased reading comprehension. *arXiv preprint arXiv:1804.07927*, 2018.

[16] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*, 2017.