UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Rustam Abdumalikov

Comprehensive research of parameter-efficient fine-tunings in open domain question answering

Master's Thesis (30 ECTS)

Supervisor(s): Yova Kementchedjhieva, Post-doc
Kairit Sirts, Post-doc

Tartu 2023

Comprehensive research of parameter-efficient fine-tunings in open domain question answering

Abstract:

Write an abstract

Keywords:

List of keywords

CERCS:

CERCS code and name: https://www.etis.ee/Portal/Classifiers/Details/d3717f7b-bec8-4cd9-8ea4-c89cd56ca46e

Põhjalik uuring parameetrite tõhusate peenhäälestuste kohta avatud kujuldomeeni küsimusele vastamine

Lühikokkuvõte:

One or two sentences providing a basic introduction to the field, comprehensible to a scientist in any discipline.

Two to three sentences of more detailed background, comprehensible to scientists in related disciplines.

One sentence clearly stating the general problem being addressed by this particular study.

One sentence summarising the main result (with the words “here we show´´ or their equivalent).

Two or three sentences explaining what the main result reveals in direct comparison to what was thought to be the case previously, or how the main result adds to previous knowledge.

One or two sentences to put the results into a more general context.

Two or three sentences to provide a broader perspective, readily comprehensible to a scientist in any discipline, may be included in the first paragraph if the editor considers that the accessibility of the paper is significantly enhanced by their inclusion.

Võtmesõnad:

List of keywords

CERCS:

CERCS kood ja nimetus: `https://www.etis.ee/Portal/Classifiers/Details/d3717f7b-bec8-4cd9-8ea4-c89cd56ca46e`

Acknowledgement

Thank you everyone

Contents

Unsolved issues

1 Introduction

What is it in simple terms (title)?

Why should anyone care?

Question Answering is an essential task in Natural Language Processing (NLP), which encompasses different variations. Some of these variations include answering factual questions learned during pre-training, predicting the answer span within a given context, or generating answers from a given context, which is particularly useful when the answer is distributed across the text. The first variation relies on parametric knowledge, which refers to the knowledge contained within the model weights. In contrast, the latter two variations use contextual knowledge. Contextual knowledge is an external knowledge(like Wikipedia passage) provided to the model during inference along with the question.

Parametric knowledge is fixed at the time of model training, which can become a problem as the answers could become outdated. For instance, if the model was trained on facts up until 2017 and we ask it to answer the question "Who is the President of the US?" today, the model might provide an outdated answer like "Barack Obama" instead of the current President "Joe Biden". This highlights the need to continuously update the model's knowledge to ensure accurate answers. Replacing outdated facts inside model weights with new ones isn't an easy task. Instead model input can be augmented with contextual knowledge, that would have updated information. And fine-tune a model to extract an answer from a given context, rather than from its weights. However, a recent study [LPC+22] identified a potential shortcoming of this approach: the model may over-rely on its memorized knowledge and completely ignore the contextual knowledge, leading to hallucinations.

In order to address the issue of ignoring contextual knowledge, researchers in [NAH+22] implemented a combination of counterfactual data augmentation and fine-tuning. This approach was designed to force the model to consider context more carefully, as relying solely on parametric knowledge could lead to incorrect responses. Similarly, researchers in [WMB+20] encountered a problem where their model was not paying sufficient attention to the questions being asked. To rectify this issue, they utilized a combination of data augmentation and adversarial training techniques, which ultimately led to improved model robustness and generalization.

Both studies were successful in making the model pay attention to the context, with data augmentation proving to be especially important. However, I believe that we can further improve generalization by using parameter-efficient fine-tuning instead of traditional fine-tuning. Traditional fine-tuning often results in the model memorizing the training data due to its high capacity, but this is not the case with parameter-efficient fine-tuning since only a small number of parameters are being fine-tuned, limiting the model's capacity for memorization and forcing it to generalize instead. Furthermore, although

adversarial training has not been used in the generative question answering domain before, it is known for its ability to increase model robustness and generalization in other domains. It is therefore intriguing to see how it will perform in the context of generative question answering. As such, I plan to conduct a comprehensive study of improving generalization in generative question answering by combining data augmentation with parameter-efficient fine-tuning and adversarial training.

For my study on improving generalization in generative question answering, I will be using the augmented NaturalQuestions dataset provided by [NAH+22]. To explore the effectiveness of different methods, I will be utilizing adversarial training as well as three parameter-efficient fine-tuning methods: LoRA, Adapters, and prompt-tuning. To ensure fairness across all methods, I will give them a fair chance to converge by limiting the maximum number of epochs to 100. Additionally, I will use Early Stopping to prevent overfitting.

Main introduction should end with a thesis statement

1.1 Contribution

What was my contribution?

1.2 Outline

What you are doing in each section (a sentence or two per section)

2 Background

This section provides an overview of the methods used in this thesis, including parameter-efficient fine-tuning techniques (LoRA, Adapters, prompt-tuning), adversarial training, and data-augmentation.

2.1 Fine-Tuning

Do I need this section?

2.2 Parameter-Efficient Fine-Tuning

Pretraining is a process of learning accurate representation of a language via self-supervised learning. To learn a good language representation we need huge amount of data. However, it is a problem since annotation would be tremendously expensive if not infeasible. Thus supervised training is out. Considering our goal we need something in between supervised and unsupervised learning. This something is called self-supervised learning. Which can be implemented buy constructing objective in a way that labels are contained within a dataset. It could be (1) predicting masking word based on surrounding context (2) in similar fashion predicting span corruption, or (3) predicting next word based on preceding words. Considering those objectives a model would be able to learn good language representation by employing a huge amount of data. However, prediction that pretrained model can do isn't practical due to objectives that we were using during pretraining to achieve good language representation. To make it useful for our day to day tasks(like sentiment analysis, question answering, etc.) we could employ finetuning i.e. method of learning a downstream task based on pretrained model representation. However, this approach has a big downside. Downstream task learning implies adapting weights of pretrained model. Thus for each downstream task we need to have a copy of pretrained model. This is wasteful especially consider that today's pretrained models are huge(like GPT-3). The solution to this problem are Parameter-Efficient Fine-Tunings (PEFT). In PEFT weights of pretrained model are frozen, and instead additional small subset of weights is used to steer model to perform downstream task.

2.2.1 LoRA

We can represent finetuned model as follow $W_0 + \Delta W$, where W_0 are pretrained parameters, and ΔW downstream update for a specific task. ΔW has the same cardinality with W_0 i.e. $|W_0| = |\Delta W|$. Which as we mentioned isn't ideal situation we wanna be in.

Lets take a step back and ask ourselves a following question: "What is Neural Network (NN)? NN is a data compressing algorithm. We know that data compression is

only possible in two cases if data has correlations, and/or data contain redundancy. Thus by have either of them or both, NN is able to learn compressed data representation that would lie in smaller subspace. Smaller subspace is called intrinsic dimension (ID).

From previous study we know that majority of NN are over parameterize(that's why such techniques as distillation are possible). This tell us that we can remove those redundant parameters without hurting performance. Thus there exist ID. LoRA is based on the idea that ID exist, which implies that same performance can be achieved by only tuning smaller subset of parameters.

LoRA decomposes ΔW as two low rank matrices multiplications of A and B(if $W_0 \in \mathbb{R}^{d\times k}$, then $A \in \mathbb{R}^{d\times r}$, and $B \in \mathbb{R}^{r\times k}$, thus $\Delta W = BA$, where $r \ll \min(d, k)$). The forward pass would change from $h = W_0x + \Delta Wx$ to $h = W_0x + BAx$. Figure 1 depicts LoRA's architecture.

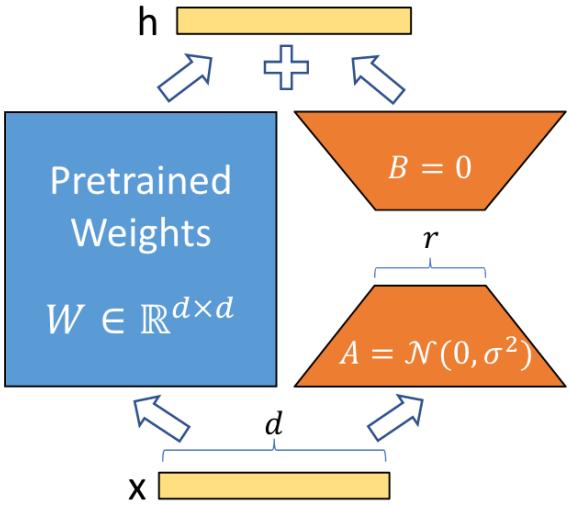

Figure 1. Pretrained parameters W will be frozen, and only A and B will be trained

To initialize A random Gaussian initialization was used and B was initialized with zeros, so $\Delta W = BA$ is zero at the beginning of the training.

2.2.2 Prompt-tuning

To begin exploring prompt-tuning, it's important to understand the concept of association. Association is a powerful tool for organizing memories in a way that facilitates easy retrieval of relevant information. For instance, when you hear the word "banana," you may immediately think of the color yellow or monkeys, because these concepts are associated with bananas based on your past experiences. In psychology, this phenomenon

is known as priming, which refers to the ability of association to influence a person's response to a stimulus, such as an image or a set of words.

In natural language processing (NLP), priming can also be used to steer the output of language models. Specifically, word-level steering is often accomplished using hard or discrete prompt, which involve selecting concrete words to trigger desired associations. However, this method has limitations, such as the need to manually search for relevant words and the lack of scalability. To address these limitations, soft prompts were introduced as an alternative. Unlike hard or discrete prompts, which involve selecting concrete words, soft prompts operate on virtual words(i.e. words that are not part of vocabulary) representation that can be learned.

Soft prompts serve as the conceptual foundation for prompt-tuning, which is a practical implementation of this approach. Prompt-tuning involves concatenating prompt with the input, and feeding it into a transformer model. This can be represented as a dot product of frozen pretrained weights W_0, and concatenation of input and prompt, denoted as $z = concat([\text{prompt}, x])$. The length of the prompt can be adjusted as a hyperparameter. In essence, prompt-tuning provides a way to fine-tune a pre-trained language model by leveraging the power of association through soft prompts.

2.2.3 Adapters

Same as prompt-tuning bottleneck adapter also based on the concept of priming. However, if in case of prompt-tuning priming was directly i.e. specified in an input, in case of bottleneck adaptors priming is hidden in a way, which also called subconscious priming. For example, if someone would ask you to come up with word that start with 'B', and you saw bearded man on the street before you would answer 'Beard', or if you ate bread before you would answer 'Bread'.

2.3 Adversarial training

Adversarial training is a powerful technique used to improve the robustness and generalization capabilities of machine learning models. This approach involves modifying the training process by introducing adversarial examples, which are specifically designed to fool the model into making incorrect predictions.

To implement adversarial training, there are typically two main steps involved. The first step is to generate adversarial examples. The second step is to modify the training objective to incorporate both the original and adversarial data. This is achieved by optimizing two losses, one on the original data and another on the adversarial data, followed by their sum.

$$\mathcal{L}^{\text{Total}} = \mathcal{L}(\Omega) + \lambda \cdot \mathcal{L}\left(\Omega'\right) \tag{1}$$

where Ω is original data, Ω' is adversarial data, and $\lambda > 0$ is a hyperparameter [WMB+20] i.e. 0,25.

Overall, adversarial training is a powerful tool for improving the performance of machine learning models, particularly in scenarios where the model needs to be robust against adversarial attacks or needs to perform well on out-of-distribution data.

2.4 Generative Question Answering

Do I need this section?

2.5 Counterfactual Data-Augmentation

Do I need this section here?

3 Methods

3.1 Dataset

Describe NaturalQuestion dataset

3.2 Model Architecture, Training and Inference

3.2.1 Training Procedure

Describe details about tokenization. Followed by training steps, and optimization.

3.2.2 Inference Procedure

Describe details about tokenization, and generation.

Rule: If you divide the text into subsections (or subsubsections) then there has to be at least two of them, otherwise do not create any.

3.2.3 Evaluation Metric

4 Results

4.1 Fine-tuning

In Table 1 we can observe that the performance on counterfactual test split significantly increase between 'f' and 'f+cf' datasets. Afterwards, on 'f+cf+a' the performance was

slightly lower than on 'f+cf'. On 'f+a' the performance sligtly decreased than on 'f' for factual and counterfactual test splits.

Methods(datasets)	**f**	**cf**	**empty**	**random**
fine-tuning(f)	0.7385	0.6432	0.0	0.0
fine-tuning(f+cf)	0.7128	0.7678	0.0	0.0
fine-tuning(f+a)	0.7223	0.6249	1.0	0.9846
fine-tuning(f+cf+a)	0.7187	0.7531	1.0	0.981

Table 1. Contain details for Fine-tuning on various datasets.

4.2 Adversarial-Training

In Table 2 we can see that performance on factual test split slightly declines after I additionally employed answerability dataset. Which also explains significantly improved performance on random and empty splits. On counterfactual split the performance remained the same.

Methods(datasets)	**f**	**cf**	**empty**	**random**
Adversarial-Training(f)	-	-	-	-
Adversarial-Training(f+cf)	0.7121	0.7729	0.0	0.0
Adversarial-Training(f+a)	-	-	-	-
Adversarial-Training(f+cf+a)	0.693	0.7722	1.0	0.9685

Table 2. Contain details for adversarial-training on various datasets. **NOTE:** '-' indicate impossibility to train due to nature of adversarial-training which requires counterfactual examples.

4.3 Prompt-Tuning

In Table 3 we can observer that performance didn't change at all.

Methods(datasets)	f	cf	empty	random
Prompt-Tuning(f)	0.6132	0.5538	0.0	0.0
Prompt-Tuning(f+cf)	0.6125	0.5553	0.0	0.0
Prompt-Tuning(f+a)	...	...	...	...
Prompt-Tuning(f+cf+a)	...	...	...	...

Table 3. Contain details for prompt-tuning on various datasets. **NOTE:** '...' indicate that training still need to be done.

4.4 Adapters

In Table 4 the most interesting part is that performance on counterfactual test split practically remained the same. However, in case of fine-tuning it increased. Expectedly the performance declined for 'f+a' for factual and counterfactual test split. And performance significantly increased by 14% on counterfactual test split, when it was trained on 'f+cf+a'.

Methods(datasets)	f	cf	empty	random
adapters(f)	0.737	0.6491	0.0	0.0
adapters(f+cf)	0.7326	0.6425	0.0	0.0
adapters(f+a)	0.7165	0.6066	1.0	0.9722
adapters(f+cf+a)	0.7092	0.7832	1.0	0.9656

Table 4. Contain details for adapters on various datasets.

4.5 LoRA

In Table 5 we can observe that the performance on counterfactual test split significantly increase between 'f' and 'f+cf' datasets. For 'f+a' for factual test split performance remained the same with 'f', but declined for counterfactual test split. And for 'f+cf+a' the performance slightly lower in contrast with 'f+cf'. Same behavior as in fine-tuning case. Trends of fine-tuning and LoRA are alike.

Methods(datasets)	f	cf	empty	random
lora(f)	0.7297	0.6527	0.0	0.0
lora(f+cf)	0.7253	0.7897	0.0	0.0
lora(f+a)	0.7289	0.6198	1.0	0.9832
lora(f+cf+a)	0.7216	0.7648	1.0	0.9853

Table 5. Contain details for LoRA on various datasets.

4.6 Trained on factual data

In Table 6 we can see all my methods except prompt-tuning outperformed my and disentQA fine-tuning.

Methods(on 'f')	f	cf	empty	random
adversarial-training	-	-	-	-
prompt-tuning	0.6132	0.5538	0.0	0.0
adapters	0.737	0.6491	0.0	0.0
lora	0.7297	**0.6527**	0.0	0.0
fine-tuning	**0.7385**	0.6432	0.0	0.0
Local: DisentQA fine-tuning	0.7282	0.6308	0.0	0.0
Paper: DisentQA fine-tuning	0.7634	0.6784	0.0	0.0

Table 6. Comparison of methods trained on 'f' dataset

4.7 Trained on factual and counterfactual data

In Table 7 ...

Methods(on 'f+cf')	f	cf	empty	random
adversarial-training	0.7121	0.7729	0.0	0.0
prompt-tuning	0.6125	0.5553	0.0	0.0
adapters	**0.7326**	0.6425	0.0	0.0
lora	0.7253	**0.7897**	0.0	0.0
fine-tuning	0.7128	0.7678	0.0	0.0
Local: DisentQA fine-tuning	0.7194	0.7495	0.0	0.0
Paper: DisentQA fine-tuning	0.7575	0.7604	0.0	0.0

Table 7. Comparison of methods trained on 'f+cf' dataset

4.8 Trained on factual and answerability data

In Table 8 for factual test split LoRA achieved the highest accuracy. Adapters slightly underperformed in contract with fine-tunings.

Methods(on 'f+a')	**f**	**cf**	**empty**	**random**
adversarial-training	-	-	-	-
prompt-tuning	-	-	-	-
adapters	0.7165	0.6066	1.0	0.9722
lora	**0.7289**	0.6198	1.0	0.9832
fine-tuning	0.7223	**0.6249**	1.0	0.9846
Local: DisentQA fine-tuning	0.7253	0.6198	1.0	0.9817

Table 8. Comparison of methods trained on 'f+a' dataset

4.9 Trained on factual, counterfactual and answerability data

In Table 9 for counterfactual test split all method outperformed fine-tuning. Whereas in factual test-set lora and adversarial training slightly underperformed. And prompt-tuning still to be trained.

Methods(on 'f+cf+a')	**f**	**cf**	**empty**	**random**
adversarial-training	0.693	0.7722	1.0	0.9685
prompt-tuning	-	-	-	-
adapters	0.7092	**0.7832**	1.0	0.9656
lora	**0.7216**	0.7648	1.0	0.9853
fine-tuning	0.7187	0.7531	1.0	0.981
Local: DisentQA fine-tuning	0.7011	0.7516	1.0	0.981

Table 9. Comparison of methods trained on 'f+cf+a' dataset

5 Discussion

6 Conclusion and Future Work

what did you do?

What are the results?

future work?

References

[LPC+22] Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. Entity-based knowledge conflicts in question answering, 2022.

[NAH+22] Ella Neeman, Roee Aharoni, Or Honovich, Leshem Choshen, Idan Szpektor, and Omri Abend. Disentqa: Disentangling parametric and contextual knowledge with counterfactual question answering, 2022.

[WMB+20] Johannes Welbl, Pasquale Minervini, Max Bartolo, Pontus Stenetorp, and Sebastian Riedel. Undersensitivity in neural reading comprehension, 2020.

Appendix

I. Glossary

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Rustam Abdumalikov**,
*(*author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Comprehensive research of parameter-efficient fine-tunings in open domain question answering,
   *(*title of thesis)

   supervised by Yova Kementchedjhieva and Kairit Sirts.
   *(*supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Rustam Abdumalikov
09/05/2023