A Survey and Comparison of LLM Fine Tuning Methods

Erik Nordby
Georgia Institute of Technology
Atlanta, GA 30332
enordby@gatech.edu

Wenzhe Ou
Georgia Institute of Technology
Atlanta, GA 30332
wou30@gatech.edu

Chloe Pomeroy
Georgia Institute of Technology
Atlanta, GA 30332
cpomeroy6@gatech.edu

Gaofeng Sha
Georgia Institute of Technology
Atlanta, GA 30332
gsha3@gatech.edu

Abstract

Numerous open source large language models have been trained and released by research institutes and organizations. These models have shown great general natural language processing capabilities, making them competent for article summarizing, education, and being AI chatbots. However, these models may have limited capabilities in domains which the models have not been trained on, such as summarizing healthcare data and research. In order to better understand the effectiveness of different domain adaptation techniques, we investigate the effectiveness of a series of approaches including vanilla fine tuning, prompt-based fine tuning, in-context learning, and context distillation. By testing these techniques across three language models and two datasets, a quantitative comparison has been drawn between these four techniques. Our results are aligned closely with prior work done comparing these techniques under similar circumstances.

1. Introduction and Motivation

Today, domain adaptation in language models is highly dependant on the domain and the capabilities of the model which is being adapted. Current frontier models have shown remarkable performance with in-context learning, few-shot fine-tuning, and prompt-based methods. Notably, current advancements in frontier models have allowed for significant increases in context length which allows in-context learning to achieve remarkable results[9]. Smaller language models have been shown to benefit from different techniques and have even been shown to learn in a different way from large language model with in-context learning [16].

The paper "Few-shot Fine-tuning vs. In-context Learning: A Fair Comparison and Evaluation" (Mosbach et al.) compares few-shot fine-tuning and in-context learning for the purpose of task adaptation. We intend to further compare other techniques in the hope of improving performance and understanding the strengths/weaknesses of various domain adaptation techniques on smaller language models. In this work, we compare the performance of fine-tuning, in-context learning, context distillation, and prompt-based fine-tuning for domain adaption.

Performant task adaptation capabilities are important for allowing for out-of-domain robustness. Some notable examples of this are allowing the model to adapt to a specific code base and to allow for longer conversations which retain prior context. Further, continuing to better understand smaller language models and techniques for their improvement remains relevant. While larger models tend to receive more attention, smaller models can be particularly useful when models need to be fine-tuned while taking time and computational constraints into account. Further, studying techniques and how they improve models may be more tractable on smaller models.

Mosbach et al. evaluated fine-tuning and in-context learning by using datasets which evaluate inference (MNLI [17] and RTE [3]), paraphrase identification (QQP [12] and PAWS-QQP [21]), and others. These datasets evaluate language models on a variety of tasks including entailment which determines whether one sentence entails another. RTE and MNLI are further a part of the GLUE benchmark. In our work, we have chosen to focus on the entailment task which includes the datasets MNLI and RTE.

2. Background

Given the remarkable results pre-trained language models have shown across a wide variety of natural language tasks, there has been significant work done to improve their performance and adapt them to new tasks efficiently. As of

the writing of this report, there are several techniques that have been shown to be effective at this. In this study we utilize fine tuning, in-context learning, prompt based fine-tuning, and context distillation and compare their performance on entailment tasks.

2.1. Fine-Tuning

Fine tuning fine tunes pre-trained LLMs on a small dataset for specialized tasks while preserving its generality. Fine-tuning of LLMs can significantly improve model performance, reduce training costs, and achieve better results. Two major fine-tuning approaches are feature extraction and full fine tuning. Feature extraction only retrains the last layer of the original LLM on task-specific data[4] while the full fine tuning modifies all the hyper-parameters by training the model on the task-specific data.

2.2. Prompt-based Fine tuning

Prompt-based fine tuning combines prompt engineering and fine tuning methods to improve model performance on specific tasks. With this method, the prompts can be manually crafted and optimized so that most informative demonstrations can be prioritized to fine tune the model. When given a small number of training samples, prompt-based fine tuning largely outperforms traditional fine tuning methods[5].

2.3. In-Context Learning

In contrast to fine tuning that trains the models, in-context learning doesn't update the parameters of the models. Instead, it adapts the models to a task by changing the prompts that are fed to the models, which is called prompt engineering. Recently, prompt engineering has gained popularity due to its simplicity. Gao[6] has shown that the structure of the template impacts models' performance greatly and Zhao[22] has found when conditioned with an proper textual context, the models improve remarkably.

2.4. Context Distillation

As mentioned in the in-context learning portion and shown in a variety of studies [6] [22] context tokens can greatly influence the performance of a model. Some examples of these context tokens are optimized prompts or a scratch pad which the model may use to perform multi-step responses. Context distillation aims to allow models to internalize these performance increases through a process akin to knowledge distillation [13]. Broadly, this results in a two step process. First, the model is provided a dataset in a context rich environment. This may include providing the model with a scratch pad or creating optimized prompts which are used to format the inputs of the dataset. The outputs from the context rich model are then used to train the original model. In the original paper introducing context distillation [1] this was done by minimizing the KL Divergence between their outputs, though other loss functions may be used [13]. The main distinction between this and knowledge distillation is that while knowledge distillation uses different models for the student and teacher, context distillation uses the same model but the teacher has access to context tokens. This dynamic means that context distillation could be used to recursively improve the models, this paradigm could be used for iterative self-improvement with the student model becoming the teacher at each iteration. [13]

Beyond performance, context distillation is also used by large labs to improve the helpfulness and harmlessness of their models. Initially context distillation was introduced to drive the alignment of models [1] and derivative techniques have seen use in frontier language models to improve how well models follow directions [14] by using the directions as the context.

3. Approach

In order to produce comparable results across the domain adaptation techniques, the same three models were used across the same datasets taken from the GLUE Benchmark [15]. Due to hardware limitations, we chose to test the techniques on the Facebook OPT-125M, Facebook OPT-350M [20], and EleutherAI's Pythia-410M models [2]. These were fine-tuned and evaluated on the RTE dataset as the in-domain dataset. The MNLI dataset was used as the out-of-domain. Both the MNLI and RTE datasets evaluate a model's capability with entailment. The raw MNLI dataset has 3 labels which can be used and to better align with the RTE data, we have dropped data rows with the neutral label (label=1) in MNLI for our purposes.

3.1. Fine tuning

Vanilla fine tuning utilizes the pre-trained models (OPT-125M, OPT-350M and PYTHIA-410M) and optimize their hyper-parameters by training them on RTE dataset respectively. Furthermore, the performance of each model is evaluated by different metrics including loss and accuracy. The detailed steps involve dataset loading from GLUE benchmark, dataset tokenization including padding, dataset subdivision, model loading, trainer parameter initialization, data collator customization, modeling training, and performance evaluation. Unlike [10], in this study we use the Trainer class from the Transformers library. The accuracy of the Fine-tuned models are evaluated both on in-distribution dataset and out-of-distribution dataset.

3.2. Prompt-based finetuning with Null Prompts

Prompt-based finetuning typically involves extensive prompt engineering and tuning which requires human intu-

ition that takes time and is hard to replicate [11]. However, the importance of prompt engineering is much greater when the model is not finetuned [7]. In this work, we use the null prompt method introduced by Logan et. al that completely bypasses the need to write and tune prompts. Instead, we simply tokenize and concatenate the inputs into a 'null' prompt, which has been found to have similar accuracy to engineered prompts, but are simpler and faster to create[7].

We combine this method with BitFit, a lightweight tuning alternative that speeds up finetuning and lowers the computational requirements by only updating the model's bias terms[19]. This method can achieve accuracy with small to medium datasets that is competitive to tuning the entire model, while only updating a very small subset of the model's parameters. With larger datasets, the results are competitive to those from other sparse-finetuning methods.

Similar to the Logan et al., we build our few-shot datasets by taking 2K examples from each label and use 4-fold cross validation. In their work, they chose K=16, but in our work we look at three different values of K: 5, 10, and 16. While we referenced the codebase from Logan et al., we created our own code with similar methodology[7].

3.3. In Context Learning with Finding Fantastically Ordered Prompts

Though prompt engineering has gained its popularity, it is not until the publication of paper[8] that the effect of the order in prompts on pre-trained language models' performances has been studied.

Through extensive experiments with the GPT2 and GPT3 models on a variety of tasks, Lu[8] proposes the following: 1. The order of prompts is crucial for the success of pre-trained language models in few-shot learning. 2. A probing method was created to find the optimal orders of prompt. It consists of a sampling-based probing set and two probing metrics. The probing set is created as such: For each sample in the training set $S_i = (X_i, Y_i)$ where X_i is the sentence and Y_i is the label, a transformation that maps each sample into the natural language space, such that each sample is a text sequence defined by the template, is done. A full permutation of n training samples is then created as $F_m = 1, 2, .., n!$ where F_m concatenates samples into one unique permutation C_m. For example, 4 training samples will give 24 unique ordering permutations $4! = 24$. Additionally, two probing metrics (Local Entropy and Global Entropy) were created to measure the quality of different orders of prompts and to find the optimal orders of prompt for different sizes of language models and for different tasks. Local Entropy(LE) measures if the model is capable of appropriately differentiating between classes. On the other hand, Global Entropy(GE) identifies prompts that avoid the issue of extremely unbalanced predictions. It is computed as $(_(vinV) - p_m^v logp_m^v)$ where p_m^v is calculated as $p_m^v =_i 1_(y(i,m))/|D|$ and $y(i,m)$ is the predicted label for a data point S_i. It proves that GE is more effective in finding the orders of prompts that give the best performance. When using GE to find optimal prompts, the language models have a 13% relative improvement over a wide range of tasks.

3.4. Context Distillation

In order to perform the two step process of context distillation, the models were first prompted with the same context as our In-Context learning experiments. The context rich inputs were then discarded and the output logits were used as targets for the original dataset's features. A KL Divergence loss was used to train based on the softmax of the logit targets and the log-softmax of the outputs from the student model. Finally, the models were evaluated on the same evaluation metrics as the other techniques.

During the initial implementation of this technique, we tried to implement an approach which used the actual outputs of the model to fine-tune the model instead of the logits. In our case, that meant using the output labels "entailment" or "not entailment" and using the original loss function of the model. We believed initially that this could have served as an effective way to generate new data for the model in a similar way to pseudo-labeling. While further refinement or better models may improve this approach, we found that this approach significantly amplified the biases of the original model and led to overfitting. In one case, our context rich model output 97.5 percent entailment. This led our student model to only output entailment with immense confidence.

This issue of bias internalization similarly appeared with the KL Divergence approach, though to a much lesser extent. As discussed earlier, improvements from context can be quite sensitive to the context used. This sensitivity was transferred to the context distillation models. In order to mitigate this, we adjusted the learning rate and temperature of the model. Further, we re-ran our experiments with different seeds to ensure the accuracy across various contexts could be measured.

The code for the context distillation section drew from Hugging Face's official Transformer library [18]. More specifically it drew from their glue dataset script which can be found here. This was adapted with in-context learning logic and ulility functions from the original paper's [10] repository found here. Finally, in order to address the targets becoming an array of logits, the forward functions for the transformer models were copied, altered slightly to resolve errors, and patched back into the respective models.

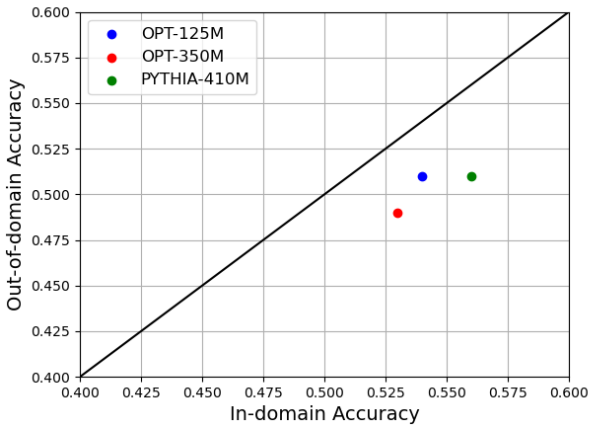

Figure 1. Model Performance Comparison in Vanilla Fine-tuning

4. Experiments and Results

4.1. Fine tuning

After training the three models on RTE train dataset, the performance of individual model is tested on RTE test dataset. Each model is only trained for 3 epochs and each training time is around 10 minutes. The in-domain accuracy for Facebook/OPT-125M, Facebook/OPT-350M and EleutherAI/PYTHIA-410M are 0.54, 0.53 and 0.56, respectively. The fine-tuned models are further tested on MNLI dataset for classification. The model accuracy values on MNLI dataset are 0.51, 0.49 and 0.51, respectively. We further manipulate the parameters such as number of epochs, batch size, and dataset size, but the variance of the accuracy result is very small, less than 0.03.

Similar to [10], a plot showing the comparison between in-domain accuracy and out-of-domain accuracy is shown in Figure 1. The performance of these three fine tuned models are comparable and their performance is slightly worse than their counterpart for in domain accuracy. According to accuracy values, our fine tuned models have similar performance as these in [10].

4.2. Prompt-based fine-tuning with Null Prompts

	K=5	K=10	K=16
OPT 125m	0.47	0.55	**0.57**
OPT 350m	0.51	**0.57**	0.52
Pythia 410m	**0.52**	0.48	0.48

Table 1. Results using null prompts with various K values

In their work, Logan et. al used RoBERTa (Large) and ALBERT (XXLarge-V2), which both have a similar size in terms of number of parameters compared to the models we've used. They found that the null prompt method resulted in comparable results to the other prompt-based fine-tuning methods, with all methods performing worse on RTE than most other benchmarks.

Our results are slightly worse than Logan et. al, with the accuracy of the OPT models increasing 6 and 12% respectively compared to the baseline models. The Pythia model actually drops in accuracy after being fine-tuned for all values of K, and it would be an interesting area for further work to investigate why this model architecture does not work well with this technique.

Conceptually, it makes sense that our models would perform worse than the original work, because the models the authors used have architectures that are more suited to the RTE benchmark. The OPT models focus primarily on generation tasks, compared to the RoBERTa and ALBERT models which are focused towards classification and question answering.

This is an important finding, as Logan et. al mentioned that future work using the null prompt method would include testing this with very large and left-to-right LMs. We have shown here that their results hold for left-to-right models of similar size to those they used.

4.3. In Context Learning with Finding Fantastically Ordered Prompts

Two sizes of OPT model (125m and 350m) and one size of the PYTHIA model(410m), and 4-shot setting are used in the experiment. As mentioned above, a template is attached for each sample to transform it into a text sequence. The template is defined as (Premise: Sentence 1; Hypothesis: Sentence 2; Prediction: Label). An Example of the training sets is as below:

Premise: *20th Century spokesman Rick Dinon said that between the time it applied for the rate increase and changed its mind, the estimated quake losses had ballooned by several hundred million dollars.* **Hypothesis**: *Rick Dinon is the senior vice president of 20th Century Insurance Co.* **Prediction**: *True*

A full permutation of 4 training samples is also created. An example is as follows:

Premise: *20th Century spokesman Rick...***Hypothesis**: *Rick Dinon is...***Prediction**: *True.***Premise**: *Harrington, of Fitchburg...* **Premise**: *In the last few days of the war ...* **Premise**: *The Bakun Hydroelectric Project (BHEP) comprises...***Hypothesis**: *A hydroelectric project ...***Prediction**: *False.*

Due to the limited computation budget, only one random seed is used and models are only evaluated in domain.

The experiment is conducted to see if 1. difference in the order of prompts affects performance of the pre-trained model. 2. GE can be used to identify optimal prompts.

Order of prompts affects performance of the models

Figure 2. Model Performance Across Orders of Prompts

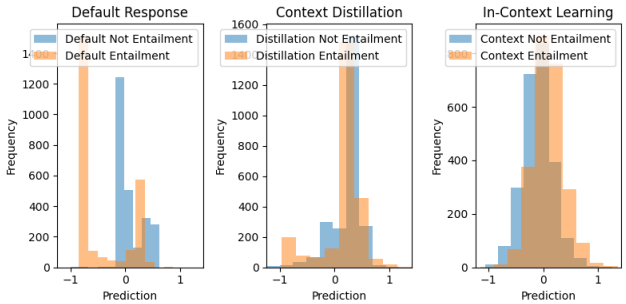

Figure 3. Output Logit Distributions

In Figure 2, we visualize the accuracy of opt-125m, opt-350m and PYTHIA-410m over 24 orders of prompts. We can see that the order of prompt does influence the performance of the model. The accuracy rate ranges from 0.495 to 0.552 for OPT-125M, from 0.484 to 0.552 for OPT-350M, from 0.451 to 0.513 for PYTHIA-410m. While using the best prompt, the OPT-125M has a 5.4% and PYTHIA-410M has a 6.6% improvement over the average performance while OPT-350M has a 7.5% lift.

GE is not so robust as the paper mentioned

	Average performance	Global Entropy performance
OPT-125M	0.523	0.522
OPT-350M	0.514	0.514
PYTHIA-410M	0.481	0.476

Table 2. Model Performance When Using Global Entropy

The original paper observed a 13 percent relative improvement across tasks when using GE to find the top 4 prompts. However, such improvements are not seen with the experiment. Table 2 shows the performance when using top 4 prompts and the average performance. We can see there is no improvement when using the top 4 prompts found by GE. Codes for experiment are partially adapted from the Github of the original paper found here.

4.4. Context Distillation

The three models were trained and tested on the RTE dataset and further tested on MNLI. This was performed using a learning rate of 1e-5, 1e-6, and 1e-7. Temperatures of .5, 1, and 5 were used for the KL Divergence. Finally, this was performed with three different seeds to ensure the same contexts were being used across the models and hyper-parameters. Ultimately, the context being used showed the largest impact on the final accuracy. While the learning rate and temperature did seem to have an impact, without suitable context the model was unable to significantly improve over the baseline. Table 3 summarizes the difference across the varied parameters

Variable	Accuracy Standard Dev	Max Average Accuracy For Variations
Learning Rate	1.1e-3	0.495
Temperature	4.9e-4	0.495
Context	3e-3	0.51

Table 3. Model Performance Across Hyper-parameters

Figure 3 shows the distribution of the logits across the default model, our context distilled model, and an in-context learning model. The in-context learning and context distilled models were provided with the same context. As can be seen from the figure, the context distilled model's outputs have aligned with the in-context learning model.

The values for context distillation in Table 4 show the average across the seeds for the best combination of hyper-parameters for each model

Overall, the performance seen from this method does not significantly improve from the baseline. Much of this could be related to challenges seen with in-context learning combined with the challenges of optimally fine-tuning a model with KL-Divergence.

4.5. Comparisons

Overall Prompt-based fine tuning has the best performance

Table 4 shows the accuracy rates for OPT-125M, OPT-350M, and PYTHIA-410M across the techniques tested. We can see that for both in domain and out of domain, prompt-based fine tuning has the highest accuracy rates for the three models except for PYTHIA-410M when evaluating in domain.

We see context distillation provides the smallest improvement in the performance of the model among the approaches tested. When compared to regular fine-tuning, the quality of the training data is of significantly lesser quality. In regular fine-tuning, the labels offer the ground truth answer while in context distillation the labels are not guaranteed to be correct. For both context distillation and in-context learning, the model is supplied with the context. Context distillation adds the additional step of discarding

	In domain			Out of domain		
	OPT-125M	OPT-350M	PYTHIA-410M	OPT-125M	OPT-350M	PYTHIA-410M
Fine Tuning	0.54	0.53	0.56	0.51	0.49	0.51
Prompt-based Fine Tuning	0.57	0.57	0.55	0.53	0.55	0.51
Fantastically Ordered Prompt	0.55	0.55	0.48	Not evaluated due to computation budget limit		
Context Distillation	0.49	0.49	0.50	0.49	0.51	0.49

Table 4. Accuracy Rates Across Fine-Tuning, Prompt-based Fine-Tuning with Null Prompts, In-context Learning with Fantastically Ordered Prompt and Context Distillation

the context and using the output to fine-tune the model. We expect that this will result in some information from the initial context being lost and unused by the context distillation model. Further, the combination of the dynamics from fine-tuning and in-context learning makes this approach more fragile than the others.

It's interesting to note as well that the PYTHIA 410M model performed worst with all methods, even though it's a larger model than both of the OPT models we chose. This model was created with interpretability in mind rather than downstream performance. So, though it sees competitive performance with similar sized models in many cases, this could be an explanation for why it does poorly here[2].

It can also be seen that the difference between the models is only subtle. All of the models have an accuracy rates of around 0.5 for both in domain and out of domain. One possible reason for such small difference could be that the task itself is difficult, but it may also be the case that the models we use only are not sophisticated enough to generalize.

5. Conclusion

Using the same models and datasets, a fair comparison has been drawn between different supervised fine tuning approaches. The experiment shows that the models with prompt-based fine tuning possess the best performance while models with context distillation show the lowest accuracy. However, the performance difference between the best and the worst fine tuning approaches is only 0.08. Furthermore, model performance with vanilla fine tuning varies least from OPT-125M to PYTHIA-410M. For the most part, our results are consistent with the original paper [10]. One exception to this is that the original paper demonstrated that larger language models will show better performance after fine tuning. However, in this study the PYTHIA-410 does not show significant advantage over OPT-125M after fine tuning.

One limitation of this study is that only smaller LLMs are chosen for supervised fine-tuning and we did not investigate our techniques on larger LLMs like Facebook OPT-13B due to limited computational resources and tight project timeline. We also focused solely on entailment tasks due to these limitations, but future work could extend this to other GLUE or SuperGLUE benchmark tasks. Further work could also be done towards further refining the techniques used. For example, the recursive self improvement which may be offered by context distillation could be explored. Finally, applying these techniques across a wider range of model sizes could also allow their benefits to be better understood.

6. Work Division

All four members contributed equally to this project, with each member selecting one of the four methods (context distillation, fine-tuning, prompt-based fine-tuning, and prompt engineering) to train models and run experiments on for comparison. Erik worked on context distillation, Gaofeng worked on fine-tuning, Chloe worked on prompt based fine-tuning using null prompts and Wenzhe worked on in-context learning using prompt reordering. We all contributed equally to the report. The division of work is in the Table 5.

Student Name	Contributed Aspects	Details
Erik Nordby	Context Distillation	Created and evaluated models for Context distillation
Wenzhe Ou	Finding Fantastically Ordered Prompts	Created and evaluated models for in-context learning with prompt ordering.
Chloe Pomeroy	Prompt-Based Finetuning	Created and evaluated models for prompt-based finetuning with null prompts and bitfit.
Gaofeng Sha	Finetuning	Created and evaluated models for vanilla finetuning.

Table 5. Contributions of team members.

References

[1] Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Jackson Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, and Jared Kaplan. A general language assistant as a laboratory for alignment, 2021. 2

[2] Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR, 2023. 2, 6

[3] Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine learning challenges workshop*, pages 177–190. Springer, 2005. 1

[4] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235, 2023. 2

[5] Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. *CoRR*, abs/2012.15723, 2020. 2

[6] Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*, 2020. 2

[7] Robert L. Logan IV, Ivana Balazevic, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. Cutting down on prompts and parameters: Simple few-shot learning with language models. *CoRR*, abs/2106.13353, 2021. 3

[8] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786*, 2021. 3

[9] Xuezhe Ma, Xiaomeng Yang, Wenhan Xiong, Beidi Chen, Lili Yu, Hao Zhang, Jonathan May, Luke Zettlemoyer, Omer Levy, and Chunting Zhou. Megalodon: Efficient llm pre-training and inference with unlimited context length, 2024. 1

[10] Marius Mosbach, Tiago Pimentel, Shauli Ravfogel, Dietrich Klakow, and Yanai Elazar. Few-shot fine-tuning vs. in-context learning: A fair comparison and evaluation. *arXiv preprint arXiv:2305.16938*, 2023. 2, 3, 4, 6

[11] Ethan Perez, Douwe Kiela, and Kyunghyun Cho. True few-shot learning with language models. *CoRR*, abs/2105.11447, 2021. 3

[12] Lakshay Sharma, Laura Graesser, Nikita Nangia, and Utku Evci. Natural language understanding with the quora question pairs dataset, 2019. 1

[13] Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context, 2022. 2

[14] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. 2

[15] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018. 2

[16] Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, et al. Larger language models do in-context learning differently. *arXiv preprint arXiv:2303.03846*, 2023. 1

[17] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. 1

[18] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Perric Cistac, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-Art Natural Language Processing. pages 38–45. Association for Computational Linguistics, Oct. 2020. 3

[19] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *CoRR*, abs/2106.10199, 2021. 3

[20] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models, 2022. 2

[21] Yuan Zhang, Jason Baldridge, and Luheng He. PAWS: Paraphrase adversaries from word scrambling. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1298–1308, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. 1

[22] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *International conference on machine learning*, pages 12697–12706. PMLR, 2021. 2