No LLMit Final Group Project Report for CS 7643 Fall 2023

Hersh Godse
hgodse3@gatech.edu

Jean-Luc Marsh
jmarsh37@gatech.edu

Alex Velasco
avelasco8@gatech.edu

Almat Yeraly
ayeraly3@gatech.edu

Abstract

Given the demanding resource requirements of training large language models from scratch, fine-tuning models that were pre-trained for different tasks on a new task can be a useful tool. We survey 3 alternative methods: in-context learning, pattern based fine tuning, and context distillation. We compare the approaches by evaluating their accuracy on both in-domain and out of domain MNLI validation sets, as well as the training (or lack thereof) required for each method. Our results show that the ideal method depends on various factors, such as model size and the data for which predictions are to be made.

1. Introduction/Background/Motivation

Our project conducts a survey and comparison of various methods for fine-tuning pre-trained large language models (LLMs) to better fit to a new target data distribution different from the LLM pre-training. As a result, this produces more relevant or accurate outputs during model inference.

In particular, we study the approaches of In-Context Learning (ICL), Pattern-Based Fine-Tuning (PBFT), and Knowledge Context Distillation. We compare resource requirements for these methods in terms of training time. We measure accuracy on two different validation sets–one in-domain (in line with training distribution), and one out-of-domain. Hereafter, these sets are referred to as the matched and mismatched datasets, respectively. The goal of this analysis is to gain better insight into the trade-offs engineers must make when evaluating, selecting between, and implementing these various methods, and to understand which solution may be optimal for a given problem's constraints.

With ICL, at inference time the prompt is prepended with additional instructions and data. As described in Mosbach et al., (2023) [7], we prepend the data with instructions for the task and a series of examples from the target training set that included an example input as well as corresponding ground truth labels. These examples, along with instructions and the actual task for the model are combined via a natural language sentence, and inputted into the model. These extra examples in the prompt are the ”context”, and by observing the input-label example pairs at the beginning of the prompt sequence during inference time (thus learning ”in” context), the idea is the model will produce a more relevant output for the actual task versus without any context. The major limitation of this approach is memory–a large portion of the context window is used up by the ICL context, which thus limits the total output length the model can generate, or requires training and running a more expensive model to support a larger context window size–which is resource intensive. Furthermore, gradients are generally frozen during inference time, so ICL may be wasteful since the context learned for one input task sample will not be reused for another input task sample. This requires the model to ingest another potentially expensive context.

Normal ”vanilla” fine-tuning refers to taking a pre-trained model, and further few-shot training this model to fit to a particular target data distribution, via a very similar process as the pre-training routine. PBFT differs from vanilla fine-tuning in that it makes predictions using a pre-trained language modeling head rather than a classifier that has been instantiated randomly [7]. This head is pre-trained by defining a pattern among the inputs and using a ”verbalizer” to map vocabulary tokens to labels. For natural language inference tasks, this takes the form of a premise and a hypothesis. The premise can be a question while the hypothesis can be an answer to the question (e.g., ”yes” or ”no”). The answers, in turn, become the tokens. Once this is provided, fine-tuning proceeds as in the vanilla version. Fine-tuning can be limited by the dataset size of the target fine-tuning dataset, and as mentioned in Mosbach et al., (2023) [7] , PBFT has been shown to be more efficient at few-shot fine-tuning for low volume target datasets. Another limiting factor for fine-tuning is distribution shift during the few-shot training procedure–the model may overfit to the target distribution, and reduce performance in other or more general tasks such as those seen in pre-training.

Context distillation aims to help with the overfitting problem described in PBFT above, and involves a teacher model as well as a student model. Both models may have identical architectures, or the student model may have fewer parameters (in the case of model compression tasks). The goal is to capture the knowledge of the teacher model into the student model [9], and the general process is similar to the "knowledge distillation" technique [2]. The teacher model may be trained (for example with vanilla FT or PBFT) to fit to the target data distribution, and then this added context can be distilled down into the student model, for example via a KL Divergence loss. See Section 2.4 (Context Distillation) for more details on the approach.

Each approach comes with its advantages and trade-offs relative to other methods. For example, the very popular ICL training paradigm is perhaps easiest to implement, but has trade-offs in memory and learning efficiency as described above. As we survey each of these training methods, we will be successful if we can come to a conclusion on if there are certain problem constraints or situations in which one approach is clearly more accurate or better performing than another. This has implications for driving and shaping best-known practice in LLM fine tuning, a major area of research today.

We fine tune models on the MNLI target dataset from the GLUE benchmark. We study each method's accuracy on validation data from this target distribution, in two ways: an in-domain "matched" validation set which matches the training distribution, and an out-of-domain "mismatched" validation set that differs from the training distribution.

Each row of the MNLI data consists of a premise, a hypothesis, and a corresponding label that characterizes the relationship between the two. The options are entailment (label 0) if the hypothesis follows logically from the premise, contradiction (label 2) if the hypothesis and premise disagree, and neutral (label 1) if neither is true. Similar to Mosbach et al., (2023) [7], we binarized the data by removing rows with neutral labels and relabeling the contradictions as 1 (thus, 2 total label categories instead of 3).

The models we fine tuned are Meta's Open Pre-trained Transformer (OPT) Language Models with 125m and 350m parameters and a sequence classifier head. We only used these two smallest models due to hardware resourcing limitations. We evaluate both models on matched versus mismatched validation sets, and for each train/evaluation we reproduce the experiment on three different random seeds to de-noise our results.

2. Approach

2.1. In-Context Learning

Our approach to in-context learning, based on Mosbach et al., (2023) [7], was to prepend the data with instructions for the task and a series of examples from the GLUE MNLI training set that included the premise, hypothesis, and corresponding label. Here is an example that uses two samples:

"Instructions: If the hypothesis is entailed by the premise, predict Label 0. If the hypothesis is contradicted by the premise, predict Label 1. Here are some examples: Example 1: Premise: Jon awoke with the warm sun on his face. Hypothesis: It was pitch black. Label: 1. Example 2: Premise: Won't This Hurt Tony Blair's Feelings? Hypothesis: Won't Tony Blair absolutely love this? Label: 1. Now you try - Premise: MLS is major-league sport with minor-league charm. Hypothesis: MLS might have the charm of a minor league but is a major league sport. Label: "

We then ran a forward pass on the prepended matched and mismatched validation sets and compared the results to the actual labels to obtain accuracy. ICL is a method that does not require any parameter updates for the model being used, so there were no additional steps after the forward pass. A Hugging Face text classification notebook [8] was used as the starting point for this. We modified it to use the OPT models described previously and the MNLI data. We also added the prepending method and experimental set up, the latter of which will be described in Section 3.

2.2. Pattern-Based Fine-Tuning

Pattern-based fine-tuning (as detailed in Mosbach et al., 2023 [7]) was implemented for this analysis. This variety of fine-tuning was selected over vanilla fine-tuning as it has previously been shown (through work by Tam et al., 2021 [10] and Logan IV et al., 2022 [5]) to be more efficient for few-shot learning when the number of examples is small, as was the case for the experiments performed in this analysis. Initially, we faced some challenges in interpreting the approach for PBFT implementation based on the literature, but in conjunction with supplementary resources, we landed on an approach that synthesized existing resources and to our knowledge, implemented this method.

As this analysis utilized OPT models, which are decoder-only, the pre-trained modeling head for PBFT was a linear projection. The implementation for PBFT was adapted from a codebase [6] accompanying Mosbach et al., (2023) [7]. This implementation was synthesized with the approaches contained within the aforementioned Hugging Face text classification notebook [8] and a notebook detailing fine-tuning for large models [1]. The existing PBFT implementation was adapted to use a tokenizer from the Hugging Face Transformers package as the verbalizer. Using a custom-made pre-processing function, the example data was iterated through to obtain a premise and hypothesis for each example. These premise-hypothesis pairs were wrapped in a list and appended to a larger list that was passed to the tokenizer as an argument. This tokenizer was in turn passed to the trainer, which was trained and evaluated in a forward pass for each scenario of interest across the permutations of models, datasets, seeds, and number of examples employed for few-shot learning that constituted each run of an experiment. Results were compared to the ground truth to generate accuracy metrics.

As in Mosbach et al., (2023) [7], our analysis applied

Low-Rank Adapatation (LoRA) on top of PBFT. As explained by Hu et al., (2022) [3], LoRA enables more efficient fine-tuning by recycling many of the weights of a pre-trained language model across various tasks.

Some hyperparameter tuning was performed on the PBFT model, with a learning rate of 0.001 and a larger number of epochs (10) delivering improved results. Beyond 10 epochs, the model was observed to overfit. However, in order to enable an equivalent comparison to the other methods explored in our analysis, hyperparameters were kept fixed with 5 epochs, a learning rate of 2e-6, and a weight decay of 0.01.

2.3. Context Distillation

2.3.1 Initial Approach

The initial approach we aimed to take for Context Distillation was to have two exactly identical pre-trained models - a teacher, and a student. The teacher model has its weights frozen, and receives no additional fine-tuning. The student model is free to receive gradients. During few-shot training time, both models receive the exact same inputs (we also experimented with adding context to the teacher model's inputs only), and the loss function is simply a KL Divergence between the teacher versus student model's output distributions over the language vocabulary. The idea is to prevent the student model from overfitting purely to the target training distribution, via a KL Divergence which forces the student model's weights (and thus knowledge) to not differ too far from that of the teacher model (which is not overfit by definition).

Unfortunately, we encountered significant challenges in implementing our logic via the HuggingFace training framework we were using–in particular, we were unable to find a convenient solution to train the student model. Furthermore, as part of computing training loss, we also pass a different input (with context) to a separate model (the teacher model) to then feed to our KL Divergence loss. Additionally, when feeding the teacher and student models the exact same inputs (no context), we observed minimal difference between the two models' outputs, and weak performance on the target distribution's training and validation datasets (implying our models weren't sufficiently learning the target data distribution).

We thus pivoted to the approach described in the below section, with the key change being combining our KL Divergence loss with another standard cross entropy loss to penalize the student model for incorrect output predictions.

2.3.2 Final Approach

Inspired by the work of Snell et al., (2022) [9], in which the authors show that context distillation can internalize abstract task instructions, our final approach was to fine-tune a teacher model with a set of inputs with prepended instructions and a student model with a different set of inputs without instructions. Both the teacher and student models were the same sized models. Out of all methodologies used in our project to fine-tune models, context distillation was the most expensive computationally.

The teacher model was fine-tuned using the vanilla fine-tuning method, while the student model was fine-tuned using the teacher model outputs. We used the same hyperparamaters as the ones used in PBFT. The loss function used to fine-tune the student models was:

$$\mathcal{L}_{total} = (1-\lambda)\mathcal{L}_{student} + \lambda\mathcal{L}_{dist} \tag{1}$$

$$\text{where, } \mathcal{L}_{student} = -\sum_{i} y_i log(s_i) \tag{2}$$

$$\mathcal{L}_{dist} = D_{KL}(t||s). \tag{3}$$

3. Experiments and Results

	Model type			
	125m		350m	
Few-shot type	Matched	Mismatched	Matched	Mismatched
Baseline	47.95%	49.74%	49.67%	47.58%
2-shot				
ICL	51.38%	48.51%	52.03%	50.37%
PBFT	51.64%	51.05%	51.03%	53.05%
Context Distillation	50.49%	50.73%	49.59%	49.26%
4-shot				
ICL	51.26%	48.15%	52.01%	51.16%
PBFT	51.62%	51.05%	51.01%	53.08%
Context Distillation	50.62%	50.48%	48.05%	48.59%
8-shot				
ICL	51.94%	49.21%	51.98%	51.51%
PBFT	51.64%	51.07%	51.06%	53.10%
Context Distillation	51.90%	52.60%	50.00%	50.15%
Overall				
ICL	51.53%	48.62%	52.01%	51.01%
PBFT	51.63%	51.06%	51.03%	53.08%
Context Distillation	51.00%	51.27%	49.21%	49.33%

Table 1: Matched and mismatched accuracies from 125m and 350m models. Cells that are highlighted in green indicate that the results are higher than the baseline.

3.1. Experimental Set Up

We chose the examples used in prepending randomly from the training set. We used three seeds to shuffle the data and then averaged the resulting accuracy for each number of samples (2, 4, and 8). This was done consistently across methods, so the training examples used for prepending in ICL were also the examples used for few-shot training with PBFT and prepending for the teacher model with context distillation. For each model size, we established a performance baseline with the accuracy of the model without any fine-tuning or ICL. Table 1 presents the baseline results and all of our results for each model, few-shot type, and validation sets used. Figure 1 is a visualization of all the results from Table 1.

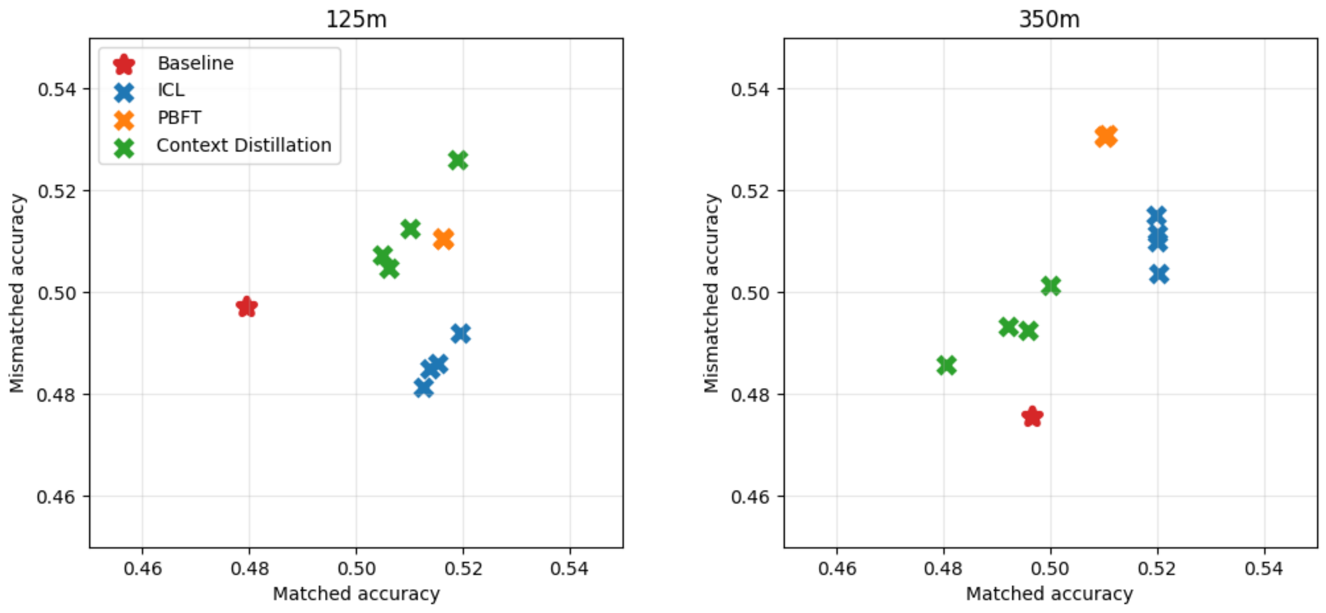

Figure 1: MNLI Matched vs mismatched validation accuracies by model type

3.2. In-Context Learning

Across all sample sizes used for prepending, the 350m model performed better than the 125m model on accuracy for both matched and mismatched validation sets. The difference was slight for the matched validation sets at just 0.48%, but it was more pronounced for the mismatched validation set at 2.39%. Based on this, it seems that ICL may yield greater improvements for larger models.

For both model sizes, performance on the matched validation set was better than the mismatched one. This is not surprising since the validation set comes from the same distribution as the examples from the training set used for prepending. The difference was larger for the 125m model: 2.9% compared to 0.99% for the 350m model. These results indicate that larger models using ICL generalize to out of distribution data better than smaller ones.

The number of examples used in prepending did not have much impact on performance on the matched validation set. The accuracy improvement on the 125m model was only 0.55% when going from 2 examples to 8, while the 350m model actually saw a slight drop in performance (-0.05%) when going from 2 to 8 examples. Performance on mismatched data, however, improved with an increasing number of samples, particularly for the 350m model. It saw a 1.14% improvement when going from 2 to 8 examples for the 350m model. The effect for the 125m model was less pronounced at 0.7%.

3.3. Pattern-Based Fine-Tuning

The pattern-based fine-tuning model was trained and evaluated 36 times in total, corresponding to the two model sizes (125m and 350m), the two datasets (matched and mismatched), the three random seeds, and the 3 values used for the number of samples employed in few-shot learning (2, 4, and 8). Across each of these 36 runs, the training loss, validation loss, and validation accuracy were recorded for each epoch. The highest validation accuracy among the epochs was selected as the accuracy for the run as a whole, and utilized as the metric for comparison among learning methods, with higher accuracy denoting better performance. Accuracies are compared against baseline accuracies for each model size and dataset, which represent a model trained on data lacking context.

Interestingly, the runs with the best performance as measured by accuracy were those utilizing the 350m model on the mismatched dataset, which averaged approximately 53% accuracy. The single run with the highest accuracy was the run utilzing the 350m model on the mismatched datatset with the third seed and 4 few-shot examples. This result represented a more than 5% accuracy improvement over the baseline accuracy of 47.6%. Remarkably, all pattern-based fine-tuning runs performed better than the baseline.

Notably, while overall pattern-based fine-tuning accuracy on the mismatched dataset significantly outperformed overall accuracy on the matched dataset for the 350m model by approximately 2%, this trend was reversed for the 125m model. For the 125m model, pattern-based fine-tuning had a higher overall accuracy on the matched dataset, outperforming the mismatched dataset by a narrower margin of 0.5%.

The comparable performance of pattern-based fine-tuning on the matched and mismatched data sets under the 125m model, as well as the stronger performance of pattern-based fine tuning on the mismatched data under the 350m model lend credence to the findings of Webson and Pavlick (2022) [11]. Namely, that models that employ pattern-based fine-tuning can perform well despite patterns that may be deceptive.

Training times showed pronounced differences between model sizes, as is to be expected, with the 350m model requiring approximately 20 minutes to train per run using a Google Cloud T4GPU versus 6-7 minutes for the 125m model. Inference times were nearly instantaneous and showed little variance.

3.4. Context Distillation

Using our approach to distill context, we observed a consistent improvement of 2-3% compared to the baseline models, as can be seen in Table 1 or Figure 1.

The fine-tuned 125m models showed an improvement for both matched and mismatched validation sets across the board. The 2-shot fine-tuning matched set accuracy increased to 50.49% and mismatched set accuracy to 50.73%. Subsequent 4-shot fine-tuning led to an increase of 50.62% and 50.48% for matched and mismatched sets. The 8-shot fine-tuning had the highest accuracy rates with matched set of 51.90% and mismatched set of 52.60%. Combining all results, the accuracies averaged 51.00% and 52.17% for matched and mismatched sets.

As for the fine-tuned 350 models, interestingly, none but 8-shot fine-tuning models had an improvement over the matched baseline. The 8-shot fine-tuning saw an increase of 50.00%, while the other fine-tuning models were comparable to the baseline. Nevertheless, the mismatched set accuracies increased for all models. 2-shot, 4-shot, and 8-shot fine-tuning models increased to 49.26%, 48.59%, 50.15% respectively. The combination of all results showed an improvement in mismatched set accuracy of 49.33%, but no improvement in the matched set accuracy.

3.5. Comparison

Across each of the methods used (ICL, PBFT, and context distillation), the methods generally displayed a 2-3% improvement relative to the baseline results in terms of accuracy. The exceptions to this rule were ICL on the mismatched dataset for the 125m model and context distillation on the matched dataset for the 350m model under the 2 and 4-shot fine-tuning models.

Variations in performance by number of examples used in few-shot learning was generally rather muted, especially for PBFT. However, context distillation bucked this trend, showing a more significant divergence between 8-shot and 2 and 4-shot runs.

Performance according to model size was also a mixed bag, with ICL showing better results across the board for the 350m model, while context distillation showed lesser performance on the same model, failing to improve over the baseline on the matched dataset for two of the three few-shot example values. PBFT achieved highs in terms of accuracy on the mismatched dataset for the 350m model, but the results on the matched dataset were comparable to those of the mismatched datatset under the 125m model.

4. Conclusion and Follow-Ups

With larger models, ICL seems to be the best approach for in-domain data (matched validation set), while PBFT does the best on out of domain data (mismatched validation set). For the smaller models, PBFT was the best on matched data, while context distillation did the best on mismatched data. This aligns with the findings of Devlin et al., (2019) [4], that FT works well across model sizes, whereas ICL struggles with smaller models. Accuracy, however, is not the only consideration. In terms of training time, our version of context distillation requires fine-tuning of both the teacher and student model. PBFT only requires fine-tuning one model. ICL does not require any training.

In terms of project success, we were able to get some improvements on the baseline we used. We were however unsuccessful in implementing the original context distillation experiments we aimed to study (due to framework limitations of HuggingFace APIs), and pivoted to an alternate approach.

This project's experimental survey was conducted on a single type of model architecture (OPT), on a single target dataset (MNLI), on a single task (binary intent classification). Each of these is a dimension we could vary, to better understand how our conclusions presented above generalize to the rest of the learning space.

We could also explore how the trends in our findings generalize as we increase the number of examples used in few-shot learning, for each of our model learning strategies. For example, we could explore the potential divergence in performance between PBFT and vanilla fine-tuning as the number of examples used in few-shot learning is varied. It would be interesting to determine the threshold at which the number of examples causes PBFT to lose its performative edge over vanilla fine-tuning for this model, similar to the explorations of Tam et al., (2021) [10].

Additionally, in their paper, Mosbach et al. (2023) [7] present that models with higher number of parameters had significantly better results than the models with lowest number of parameters. Our results were comparable to what was presented in [7] for 125m and 350m models. Hence, we hypothesize that fine-tuning larger models would also show an improvement over the baseline and be comparable to the results presented in [7], and could explore this with more resourcing.

Finally, we could also explore multi-learning fine-tuning tasks, where the same model needs to fine-tune to multiple datasets or multiple learning tasks. How does adding this increased learning requirement affect model performance on each task or data distribution, including as other hyperparameters are adjusted? In the teacher-student training relationship we explored in our context distillation experiments, we only explored how the student model performs on a target distribution/learning task, but did not explore how it performs on the general tasks and distributions as the teacher model. In other words, we learned how well the student model can fit a new distribution, but did not investigate how much the student model forgets the original learned distribution. This is also a compelling area of subsequent investigation.

References

[1] Finetune-opt-bnb-peft.ipynb. https://colab.research.google.com/drive/1jCkpikz0J2o20FBQmYmAGdiKmJGOMo-o?usp=sharing#scrollTo=hsD1VKqeA62Z. 2

[2] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. 2

[3] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. 3

[4] Kenton Lee Jacob Devlin, Ming-Wei Chang and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *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)*, 2019. 5

[5] Robert 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. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2824–2835, Dublin, Ireland, May 2022. Association for Computational Linguistics. 2

[6] mmarius. ft.py, 2023. https://github.com/uds-lsv/llmft/blob/main/ft.py. 2

[7] Marius Mosbach, Tiago Pimentel, Shauli Ravfogel, Dietrich Klakow, and Yanai Elazar. Few-shot fine-tuning vs. in-context learning: A fair comparison and evaluation, 2023. 1, 2, 5, 6

[8] Rocketknight1. text-classification.ipynb, 2023. https://github.com/huggingface/notebooks/blob/main/examples/text_classification.ipynb. 2

[9] Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context, 2022. 2, 3

[10] Derek Tam, Rakesh R. Menon, Mohit Bansal, Shashank Srivastava, and Colin Raffel. Improving and simplifying pattern exploiting training. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4980–4991, Online and Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics. 2, 5

[11] Albert Webson and Ellie Pavlick. Do prompt-based models really understand the meaning of their prompts?, 2022. 5

Student Name	Contributed Aspects	Details
All	Project Report Writing	Writing up the Final Project Report.
Hersh Godse	Context Distillation Initial Implementation	Formulated and wrote the original knowledge distillation approach studied for Context Distillation experiments
	Context Distillation Fine Tuning	Fine-tuned the teacher and student 350m models.
Jean-Luc Marsh	Pattern-Based Fine-Tuning	Pattern-based fine-tuning implementation.
	Fine-Tuning Testing Flow	Trained pattern-based fine-tuning model (through running of experiments), and examined results. Analyzed effect of varying epochs, learning rate, and weight decay.
	Environment Exploration	Explored and evaluated project environments for feasibility of running experiments and conducting analysis (PACE-ICE, Google Colab, Google Cloud).
Alex Velasco	ICL	ICL implementation and analysis of results.
	Prepending	Created function to apply prepending to datasets.
	Experiment Setup	Came up with framework for comparing results across methods and wrote code used to create different training sets and contexts across different seeds and sample numbers.
	Research on Context Distillation	Contributed to research and discussions around details of context distillation.
Almat Yeraly	Research Context Distillation	Researched ways to implement context distillation to our project
	Context Distillation Custom Trainer	Implemented the custom trainer based on the huggingface trainer for context distillation.
	Context Distillation Fine-tuning	Fine-tuned the teacher and student 125m models.
	Visualizations	Put together the table (Table 1) and the scatter plot with final results (Figure 1)

Table 2: Contributions of team members.