SuLaLoM: Structured Classification of Tabular Data with Large Language Models

Stanford CS224N Custom Project

Su Kara
Department of Computer Science
Stanford University
sukara@stanford.edu

Abstract

This project seeks to improve the classification accuracy of Large Language Models on tabular data. We explore in-context learning (ICL) with various input data formats, including Text, HTML, JSON, and more, to identify the most effective format for LLM prompts in predicting the `Class` column of tabular data. We evaluate these formats using Llama-3-70B and Gemma-7B models accessed through the Together AI API. We also fine-tuned the pre-trained Gemma-7B model from HuggingFace with serialized tabular data rows on Google Colab. We also fine-tune the pre-trained Gemma-7B model from HuggingFace with serialized tabular data on Google Colab, testing datasets with 16, 128, and 512 rows. Our experiments aim to enhance the accuracy and robustness of tabular data classification by leveraging both ICL and fine-tuning methods with different input formats, ultimately providing insights into optimizing LLMs for tabular data tasks.

1 Key Information

- TA mentor: Kaylee Burns
- External collaborators: No
- External mentor: No
- Sharing project: No

2 Introduction

Large Language Models (LLMs) have recently demonstrated remarkable capabilities beyond natural language processing (NLP) tasks that they were initially known for. Despite their impressive growth in so many areas from text to images and videos in so many domains from healthcare to software development, they still struggle with tabular data (Fang et al, 2024).

Traditionally, gradient-boosted trees like XGBoost and methods like logistic regression have commonly been used for predictions on tabular data (Borisov et al, 2022). Researchers have also investigated ways to integrate tabular data with deep neural networks (Hollmann et al., 2023).

Today, there is increasing research in using LLMs on tabular data. There are two main fields that researchers focus on: 1) table understanding and 2) table prediction. Table understanding is the ability of a model to accurately interpret, extract, and reason about data presented in tabular format. A recent study evaluates table understanding of LLMs by developing a benchmark that includes tasks like cell lookup and row retrieval, exploring different input designs and prompting methods (Sui et al., 2023).

In contrast, table prediction uses table serialization (the process of converting tabular data into a sequential text format suitable for input to LLMs) and prompting for an LLM to predict a value in a table. While table understanding is crucial for tasks that require interpreting and reasoning about

data, in this paper, we wanted to focus on table prediction to improve the performance of LLMs on tabular data by using in-context learning (ICL) with different prompting techniques such as zero-shot, few-shot, and rule-based as well as fine-tuning an open source model. Our hypothesis is that since LLMs are mostly being trained on web data, they will perform better with prompts in the HTML Table format.

3 Related Work

With our sharp focus on improving the performance of LLMs in tabular data prediction, we narrowed down the potential areas to three main categories that can be listed as input formatting, prompting, and fine-tuning.

In order to feed tabular data as input into LLMs, we have to convert the structured data into a text format for training and inference purposes (Sui et al., 2024). The streaming format itself can have a direct impact on the performance of a model as its understanding will depend on the text being in free form, HTML table, CSV, JSON, XML, or YAML (Fang et al., 2024).

Researchers have been exploring how providing a Chain-of-Thought (CoT) may significantly improve the ability of LLMs in complex reasoning (Wei et al., 2023). They show how reasoning abilities emerge via a method called CoT prompting, where a chain of thought examples are provided in prompting.

Recently researchers have developed a framework called Hypotheses-to-Theories to help LLMs in reasoning tasks by teaching them how to learn from rules. It involves induction and deduction stages, resulting in a 20% accuracy improvement and transferable learned rules (Zhu et al., 2024) The learned rules are also transferable to different models and to different forms of the same problem.

In a new paradigm based on learning from rules, researchers proposed a method called rule distillation, where they extract knowledge from textual rules and explicitly encode them into the parameters of LLMs through in-context learning (ICL). The experiments show that this approach is more efficient and improves generalization ability compared to example-based learning (Yang et al., 2024).

Recent research suggests that using LLMs for the prediction of tabular data could be promising (Hegselmann et al., 2023). The researchers fine-tune GPT-3 in zero-shot and few-shot settings using serialized rows to make predictions on tabular data. Hegselmann et al. found that with a large number of shots, traditional methods including XGBoost, TabPFN (Hollmann et al., 2023), and logistic regression (LR) tend to surpass GPT-3 in AUC score. However, in zero to 8-shot settings, their TabLLM performs well against these common benchmarks.

4 Approach

Based on insights from related research on LLMs, we focused on three main components:

1. Input Stream Formatting
2. In-Context Learning (ICL)
3. Parameter Efficient Fine-Tuning (PEFT)

4.1 Input Stream Formatting

Since there has been a strong conviction in recent studies about the impact of the input format on LLMs, we converted the original tabular data into various input formats such as Text, HTML Table, DIV Table, CSV, JSON, XML, and YAML to compare their performance in training and inference.

4.2 In-Context Learning (ICL)

We implemented ICL in our prompts for the Llama-3-70B model, making API calls through Together AI. To optimize resource usage, we transitioned to using the Gemma-7B model for both prompting and fine-tuning. After observing Text and HTML Table as the top two performers in input streaming formats, we narrowed down our focus to these two formats when prompting with the Gemma-7B model. We employed three distinct prompting techniques to enhance performance.

4.2.1 Zero-Shot Prompting:

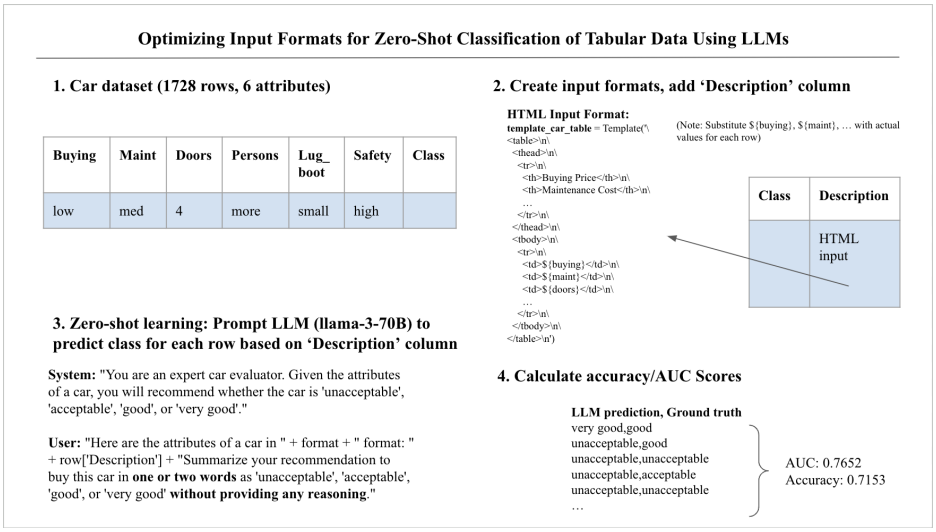

Figure 1: Optimizing Input Formats for Zero-Shot Classification of Tabular Data Using LLMs

As illustrated in Figure 1, the general zero-shot prompting process involves iterating through each row of the dataset and creating various input formats (HTML, Text, JSON, YAML, etc.) that contain information about each attribute. This information is then added to a new "description" column (as shown in step 2). We then shuffle the rows, randomly select one, and prompt the LLM by providing it the "description" column in the following format:

System: You are an expert car evaluator. Given the attributes of a car, you will recommend whether the car is 'unacceptable', 'acceptable', 'good', or 'very good'.

User: Here are the attributes of a car in `format` format: `row['description']`. Summarize your recommendation to buy this car in one or two words as 'unacceptable', 'acceptable', 'good', or 'very good' without providing any reasoning.

For our system prompt, which serves as the fixed instruction to constrain the LLM's response, we specify the LLM's task to ensure it understands its role.

For our user prompt, which is the query that the user feeds the LLM, we provide the input format it should expect and the content of the 'description' column. We also include constricting instruction about the four classes and specify that the response should be "without providing any reasoning". This double measure addresses one of the issues found in the TabLLM paper, where the authors noted that GPT-3 often added statements like "This car is a good choice" in its responses.

4.2.2 Few-Shot Prompting:

Few-shot prompting followed a similar process to zero-shot prompting, with the key difference being the inclusion of multiple rows from the dataset in each prompt to the model. We tested few-shot learning using only the Text and Table formats. Within the prompt, we included several random rows from the table. Due to token limits, we could only include a maximum of 128 rows.

4.2.3 Rule-Based Prompting:

We introduced a novel prompting technique by using *Rules* that contained logical instructions rather than examples, aiming to improve the model's reasoning capabilities. Here are a few examples of the 15 rules used for training:

- If Persons are 'two' or Safety Score is 'low', then Recommendation should be 'unacceptable'.

- If Buying Price is 'very high' and Maintenance Cost is 'high' or 'very high', then Recommendation should be 'unacceptable'.
- If Buying Price is 'high' and Maintenance Cost is 'very high', then Recommendation should be 'unacceptable'.

4.3 Parameter Efficient Fine-Tuning (PEFT)

We used PEFT to achieve performance comparable to full fine-tuning while only having a small number of trainable parameters (Liu et al., 2022). In full fine-tuning, all the pre-trained model weights are updated during training, which can be computationally expensive, especially for Large Language Models. PEFT addresses this by only updating a small subset of the model's parameters.

We also used Low-Rank Adaptation (LoRA), which freezes the pre-trained model weights while injecting trainable matrices into each layer of the model architecture. This approach can reduce the number of trainable parameters by 10,000 times and the GPU memory requirement by 3 times (Hu et al., 2021). LoRA enables efficient and scalable fine-tuning by maintaining the core model weights and only adjusting the smaller, injected matrices.

We fine-tuned our Gemma-7B model three times on serialized rows of 16, 128, and 512 examples, and evaluated its performance on the remaining items in the original dataset of 1,728 rows. The specific experimental details for fine-tuning can be found in section **5.3**.

5 Experiments

5.1 Data

We used the Cars Dataset from the TabLLM paper, which contains 1728 rows and 6 attributes:

Buying: buying price, {vhigh, high, med, low}
Maint: price of maintenance, {vhigh, high, med, low}
Doors: number of doors, {2, 3, 4, 5more}
Persons: capacity in terms of persons to carry {2, 4, more}
Lug_boot: size of the luggage boot {small, med, big}
Safety: estimated safety of the car, {low, med, high}

The `Class` attribute indicates whether a car is acceptable to buy: "unacc" (70%), "acc" (22%), "good" (4%), "vgood" (4%).

We noticed the class distribution was uneven. Though we considered balancing it by reducing the "unacc" and "acc" examples, we decided to keep it as the authors of TabLLM did since there wouldn't be sufficient data for training or testing if we matched "good" and "very good" categories that had only 4% coverage.

For clarity, we replace terms like "vhigh" with "very high" and "unacc" with "unacceptable" when inputting values into the LLM. Using text over numbers (e.g., "five or more" instead of "5") ensures better comprehension by the model.

5.2 Evaluation Methods

For our evaluation metrics, we used AUC (Area Under the Curve) and accuracy scores to compare the LLM's `Class` predictions with the ground truth `Class` values from the dataset.

Accuracy Score: Accuracy measures the proportion of correct predictions out of all predictions. While straightforward, it may not fully capture performance in imbalanced datasets.

AUC (Area Under the Curve): AUC-ROC evaluates the model's ability to distinguish between classes. In a multiclass setting, AUC is extended using the one-vs-rest (OvR) approach by converting multiclass labels into binary labels for each class, calculating the ROC curve and AUC score for each class, and then averaging the AUC scores across all classes to obtain the Macro-AUC:

$$\text{Macro-AUC} = \frac{1}{N} \sum_{i=1}^{N} \text{AUC}_i$$

where N is the number of classes and AUC_i is the AUC score for the i-th class.

Using both AUC and accuracy provides a comprehensive evaluation of the LLM's performance, considering both overall correctness and discriminative ability across multiple classes.

5.3 Experimental Details

For prompting, we used VS Code as our IDE, and imported the 'together' library in order to make API calls using Llama-3-70B model available publicly on the Together AI's website.

The fine-tuning experiments were conducted with the 'google/gemma-7b-it' model on HuggingFace by using a Google Colab Pro account. The model preparation involved configuring LoRA with a rank of 6, alpha of 16, and dropout of 0.1, targeting the modules: 'q_proj', 'k_proj', 'v_proj', and 'out_proj'. Gradient checkpointing was enabled to manage memory constraints. The tokenizer used had its pad token set to the EOS token.

Model quantization was applied to optimize the model for faster inference and reduced memory usage, making it more efficient for deployment.

We conducted fine-tuning with few-shot learning, training the model on 16, 128, and 512 randomly selected rows, and testing on the remaining data. The TabLLM paper reported significant improvements at these few-shot levels, which guided our choice of training sizes.

The custom dataset for training was derived from CSV files containing "Question" and "Answer" columns, which were combined into a single text field. The training process utilized the 'SFTTrainer' from the 'trl' library. Key training parameters included: warmup steps of 100, batch size of 1, gradient accumulation steps of 1, learning rate of 1×10^{-4}, and a sequence length of 150 tokens. Model checkpoints and logs were saved every 10 steps, with weights stored in the './saved_weights' directory. Training was conducted on the NVIDIA L4 GPU due to the memory needs of the Cars dataset, while inference was run on the Tesla T4 GPU.

For inference, the fine-tuned model and tokenizer were loaded from the Hugging Face Hub. The model was evaluated on test data, and results were saved to CSV files. The evaluation process involved generating responses to the test prompts and comparing them to the ground truth, with detailed logging of performance metrics.[1]

5.4 Results

As shown in Figure 2, we compared several input streaming formats with zero-shot prompting on Llama-3-70B. The HTML Table format turned out to be the top performer followed by Text as the runner-up for inference. Since the other input formats didn't perform well, we focused on Text and HTML Table as the two input formats for the remaining experiments.

Format	Text	CSV	Table-Based HTML	Div-Based HTML	JSON	XML	YAML
AUC Score	**0.7183**	0.7115	**0.7652**	0.6971	0.7088	0.7173	0.6862
Accuracy	0.6522	0.6418	**0.7153**	**0.6696**	0.625	0.6649	0.6505

Figure 2: Llama-3-70B Zero-Shot Prompting AUC and Accuracy Scores

Figure 3 shows the comparison of Text and Table input formats with different number of examples used in few-shot prompting on Llama-3-70B. When we used the Text format, the accuracy increased from 65% to 74% with 128 examples in the prompt. However, when we used the Table format, the accuracy dropped from 67% to 64% with 100 examples. Due to the limitation of 8,192 tokens in

[1]Link to Github Repository: `https://github.com/sukara13/CS224N-Final-Project`.
Prompting: ICL/carllm.py Fine-tuning: Fine-tuning/CS224_Car_Gemma_Inference_ Fine_Tune.ipynb

the input message, we couldn't test with more than 128 examples in Text format and more than 100 examples in Table format. Even though the Table format performed better for inference, the Text format worked better for training purposes.

Format	1	2	4	8	16	32	64	128
Text	0.648	0.516	0.557	0.54	0.659	0.672	0.656	0.74
Table	0.674	0.493	0.64	0.555	0.619	0.624	0.614	0.636

Figure 3: Llama-3-70B Few-Shot Prompting Accuracy Scores

Figure 4A and 4B show how the accuracy and AUC scores change with the increasing number of examples in the prompt on Llama-3-70B. Both accuarcy and AUC scores of Text and Table formats increase with more than 32 examples in few-shot prompting.

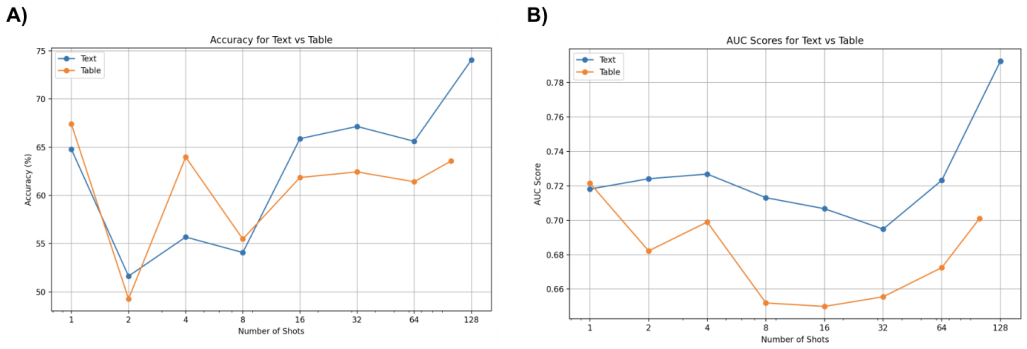

Figure 4: Llama-3-70B Few-Shot Prompting Accuracy and AUC Scores

After completing our tests with zero-shot and few-shot prompting on Llama-3-70B, we switched our focus to Gemma-7B for the same tests in the Text and Table formats. Since Gemma-7B didn't produce reliable results with zero-shot prompting, we moved onto few-shot prompting with at least one example. Similar to Llama-3-70B, we faced the same constraints with 8,192 tokens in the input message, and provided up to 128 examples at most.

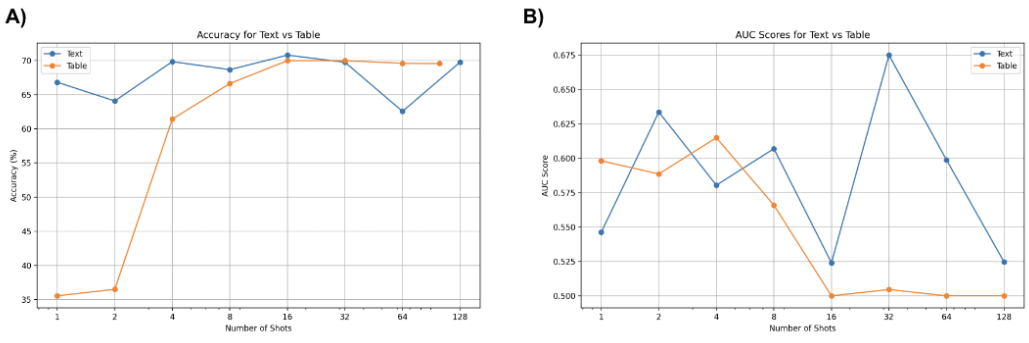

Figure 5: Gemma-7B Few-Shot Prompting Accuracy and AUC Scores

Figure 5A and 5B show the accuracy and AUC scores for few-shot prompting on Gemma-7B. The accuracy scores of Text and Table formats converge after 16 examples, but the AUC scores keep dropping with increasing number of examples. This could be explained with the imbalanced Cars dataset where the model may only predict the majority class of "unacceptable" well.

Rule-based prompting generated remarkable results as shown in Figure 6. Similar to few-shot prompting, both accuracy and AUC scores improved in Llama-3-70B, while only accuracy improved in Gemma-7B. We achieved 84% accuracy on Llama-3-70B and 71% on Gemma-7B in Table format.

Model	Format	Accuracy	AUC Score
LLama-70B	Text	0.814	0.802
LLama-70B	Table	0.842	0.811
Gemma-7B	Text	0.675	0.563
Gemma-7B	Table	0.708	0.522

Figure 6: Gemma-7B Rule-Based Prompting Accuracy and AUC Scores

Finally, we applied fine-tuning to the pre-trained Gemma-7B model with 16, 128, and 512 examples. As shown in Figure 7, both accuracy and AUC scores consistently increased with more examples to train the model with. We were expecting these improvements based on the TabLLM paper that we used as our baseline benchmark (Hegselmann et al., 2023). They were able to achieve the AUC scores of 86%, 98%, and 100% by fine-tuning GPT-3 with 16, 128, and 512 examples respectively. Our corresponding scores with Gemma-7B were relatively lower than those baselines but were still the best among all three approaches that we tried in this project.

Number of Shots	Accuracy	AUC Score
16	0.609	0.607
128	0.859	0.782
512	0.910	0.891

Figure 7: Gemma-7B Fine-Tuning Accuracy and AUC Scores

6 Analysis

Our initial experiment to test different input streaming formats proved our hypothesis correct as the Table format performed better for inference with zero-shot prompting. To our surprise, the Text format performed better for training through ICL with few-shot prompting. Even though LLMs understand HTML Table format intrinsically, the ICL process seems to work better with text.

In our few-shot prompting experiments, the accuracy and AUC scores of Llama-3-70B improved significantly with more examples, while the performance of Gemma-7B didn't improve. This can be attributed to the smaller size of Gemma-7B compared to Llama-3-70B that is almost ten-fold in the number of parameters.

Rule-based prompting showed improvements in both models. Similar to few-shot prompting, due to their nature of the same ICL approach, Llama-3-70B performed much better than Gemma-7B because of their size difference. If we compare the few-shot and rule-based prompting results within Llama-3-70B, we will quickly notice the accuracy score of 66% for few-shot with 16 examples versus 84% for rule-based with 15 rules to have a relatively fair comparison.

Out of the three different approaches we experimented with, fine-tuning performed the best. This outcome was expected as we essentially trained the model with a large set of examples. Unlike ICL, fine-tuning didn't have any limitations on the number of examples. It is also a clean approach as we didn't have to use any few-shot training after fine-tuning. Therefore, we could save time and money in our inferences with smaller number of tokens in the input message. However, fine-tuning comes with its own disadvantages. Training a model is an expensive and time-consuming process that requires a lot of resources in terms of memory and GPU. Since we're training an existing model with additional data, we may potentially face unexpected results such as catastrophic forgetting when we pose questions outside the Cars dataset (Luo et al., 2024).

7 Conclusion

Among the tested input streaming formats, the HTML Table performs the best for inference, while Text is the best choice for training through ICL and fine-tuning. However, the Table format requires a larger number of tokens due to the HTML tags needed for each row and column. Besides, additional template creation is required to convert the original text into the Table format.

Even though few-shot prompting shows some improvements in large models, it usually requires more than 32 examples that may not be practical. It also creates a lot of overhead for each request, which increases the time and cost due to excessive amount of tokens.

Fine-tuning is the best approach for smaller models as few-shot and rule-based prompting don't introduce much improvement. Rule-based prompting would be the ideal choice for larger models due to the high cost of fine-tuning and the small number of rules required for training. However, rule generating requires an additional step to identify the patterns in the dataset.

One of the limitations of our work was the 8,192 tokens allowed for input messages. Without this limitation, we would be able to test with more examples and make a fair comparison between ICL and fine-tuning on larger models.

Another limitation was the lack of GPU and memory due to the high cost associated with those resources. Because of those constraints, we couldn't train the model with more than 512 examples during fine-tuning to compare with our baseline benchmark results of TabLLM.

In future research, hybrid approaches can be experimented to improve the performance of LLMs even further. For example, rules can be used to train the model for fine-tuning rather than examples from the original dataset.

Another future work could be to develop an approach to generate rules from a dataset with the use of LLMs, and feed those rules into either ICL or fine-tuning. This would be invaluable as it will avoid the additional step required for rule-based prompting.

8 Ethics Statement

The use and training of LLMs pose ethical challenges and societal risks, such as replicating biases in sensitive areas like income, health, race, gender, sexual orientation, religion, or political affiliation. To mitigate this, we prioritized diverse and inclusive data selection and training. Notably, our dataset choice, the Cars dataset, was used by the authors of TabLLM and is not prone to harmful stereotypes or biases.

LLMs' resource-intensive nature also creates exclusive research environments and significant environmental impacts. To address this, we fine-tuned a smaller model, Gemma-7B, recognizing the high costs and potential environmental harm associated with fine-tuning. Additionally, we used parameter-efficient fine-tuning (PEFT), which further reduces resource usage and carbon emissions. We aim for transparency in our decision-making process and seek sustainable computing solutions.

By incorporating these strategies, we strive to minimize the ethical and environmental risks associated with LLMs, ensuring responsible and inclusive AI research practices.

References

Borisov, Vadim, et al. “Deep Neural Networks and Tabular Data: A Survey” arXiv:2110.01889 [cs.LG] [Submitted on 5 Oct 2021 (v1), last revised 29 Jun 2022 (this version, v3)]

Fang, Xi, et al. “Large Language Models (LLMs) on Tabular Data: Prediction, Generation, and Understanding - A Survey” arXiv:2402.17944 [cs.CL] [Submitted on 27 Feb 2024 (v1), last revised 1 Mar 2024 (this version, v2)]

Hegselmann, Stefan, et al. “TabLLM: Few-shot Classification of Tabular Data with Large Language Models.” arXiv:2210.10723v2 (cs.CL) [Submitted on 17 Mar 2023]

Hollmann, Noah, et al. “TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second” arXiv:2207.01848 [cs.LG] [Submitted on 5 Jul 2022 (v1), last revised 16 Sep 2023 (this version, v6)]

Hu, Edward, et al. "LoRA: Low-Rank Adaptation of Large Language Models" arXiv:2106.09685 [cs.CL] [Submitted on 17 Jun 2021 (v1), last revised 16 Oct 2021 (this version, v2)]

Liu, Haokun, et al. “Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning.” arXiv:2205.05638v2 (cs.LG) [Submitted on 11 May 2022 (v1), last revised 26 Aug 2022 (this version, v2)]

Luo, Yun, et al. "An Empirical Study of Catastrophic Forgetting in Large Language Models During Continual Fine-tuning" arXiv:2308.08747 [cs.CL] [Submitted on 17 Aug 2023 (v1), last revised 2 Apr 2024 (this version, v3)]

Sui, Yuan, et al. “Evaluating and Enhancing Structural Understanding Capabilities of Large Language Models on Tables via Input Designs.” arXiv:2305.13062v1 (cs) [Submitted on 22 May 2023 (this version), latest version 17 Feb 2024 (v4)]

Wei, Jason, et al. “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models” arXiv:2201.11903 [cs.CL] [Submitted on 28 Jan 2022 (v1), last revised 10 Jan 2023 (this version, v6)]

Yang, Wenkai, et al. "Enabling Large Language Models to Learn from Rules" arXiv:2311.08883 [cs.CL] [Submitted on 15 Nov 2023 (v1), last revised 16 Feb 2024 (this version, v2)]

Zhu, Zhaocheng, et al. "Large Language Models can Learn Rules" arXiv:2310.07064 [cs.AI] [Submitted on 10 Oct 2023 (v1), last revised 24 Apr 2024 (this version, v2)]