Investigating the Effectiveness of Task-Agnostic Prefix Prompt for Instruction Following

Seonghyeon Ye[1], **Hyeonbin Hwang**[1], **Sohee Yang**[1,2], **Hyeongu Yun**[3], **Yireun Kim**[3], **Minjoon Seo**[1]
[1]KAIST
[2]UCL
[3]LG AI Research
seonghyeon.ye@kaist.ac.kr

Abstract

In this paper, we present our finding that prepending a Task-Agnostic Prefix Prompt (TAPP) to the input improves the instruction-following ability of various Large Language Models (LLMs) during inference. TAPP is different from canonical prompts for LLMs in that it is a *fixed* prompt prepended to the beginning of every input regardless of the target task for zero-shot generalization. We observe that both base LLMs (i.e. not fine-tuned to follow instructions) and instruction-tuned models benefit from TAPP, resulting in 34.58% and 12.26% improvement on average, respectively. This implies that the instruction-following ability of LLMs can be improved during inference time with a fixed prompt constructed with simple heuristics. We hypothesize that TAPP assists language models to better estimate the output distribution by focusing more on the instruction of the target task during inference. In other words, such ability does not seem to be sufficiently activated in not only base LLMs but also many instruction-fine-tuned LLMs.

Introduction

Large Language Models (LLMs) have demonstrated the ability to follow user instructions through approaches such as instruction tuning or reinforcement learning from human feedback (RLHF) (Sanh et al. 2021; Wei et al. 2021; Wang et al. 2022c; Ouyang et al. 2022; Min et al. 2022a; Chung et al. 2022; Ye et al. 2022; Bai et al. 2022; Askell et al. 2021). However, previous work mainly has focused on fine-tuning-based approaches to enhance the instruction-following ability of LLMs where the model is fine-tuned on various tasks with instructions, requiring multiple backpropagation processes and necessitating access to the model weights which limits its applicability to proprietary models.

In this paper, we present and analyze our finding that prepending a **T**ask-**A**gnostic **P**refix **P**rompt (TAPP) that is determined by simple heuristics during inference significantly enhances the instruction-following ability of LLMs across various tasks for both open-sourced and proprietary models (Zhang et al. 2022; Brown et al. 2020; Wang and Komatsuzaki 2021; Black et al. 2022). Specifically, TAPP consists of multiple cross-task demonstrations where each demonstration is a concatenation of an instruction, input, and output instance of a task. Note that TAPP is different from canonical task-specific prompts in that it is a fixed prefix prompt that is prepended regardless of the target task for zero-shot generalization.

We first observe that TAPP significantly enhances the instruction-following performance of various base LLMs that are not fine-tuned to follow instructions. Notably, even smaller LLMs with TAPP outperform much larger language models without TAPP, such as the 6B-sized GPT-J with TAPP outperforming 30 times larger 175B-sized GPT-3 Davinci without TAPP. Second, we show that applying TAPP on top of instruction-fine-tuned LLMs also improves the performance, boosting the performance of one of the strongest instruction-following LLMs (text-davinci-003) by 9.3%. This indicates that the effect of TAPP during inference is complementary to the effect of instruction fine-tuning. Moreover, we demonstrate that prepending TAPP to target task demonstrations also improves performance, implying that TAPP also enhances few-shot in-context learning during inference.

Our analysis shows that TAPP performs best when the prefix prompt consists of demonstrations of classification tasks that include explicit answer choice in the instruction (e.g., expression of *"agent" or "customer"* in Figure 1). This holds true even when the target task is a generation task, which contrasts with the findings of the previous studies that it is crucial to retrieve a set of prompts that are similar to the target task (Rubin, Herzig, and Berant 2021; Liu et al. 2022). We also observe that the performance does not degrade significantly even if the input distribution of each demonstration of TAPP is corrupted. Based on these two observations, we hypothesize that during inference of TAPP, LLMs learn the correspondence between the answer choice included in instruction and the output of each demonstration of TAPP. Through this hypothesis, we suggest that the role of TAPP is to help LLMs *focus* on the target instruction to better estimate the output distribution of the target task. This also implies that this ability does not seem to be sufficiently activated in both base LLMs and instruction-tuned LLMs, leaving further investigation as future work.

Related Works

Inference-time Task Adaptation In-context learning is one of the most widely known gradient-free task adaptation

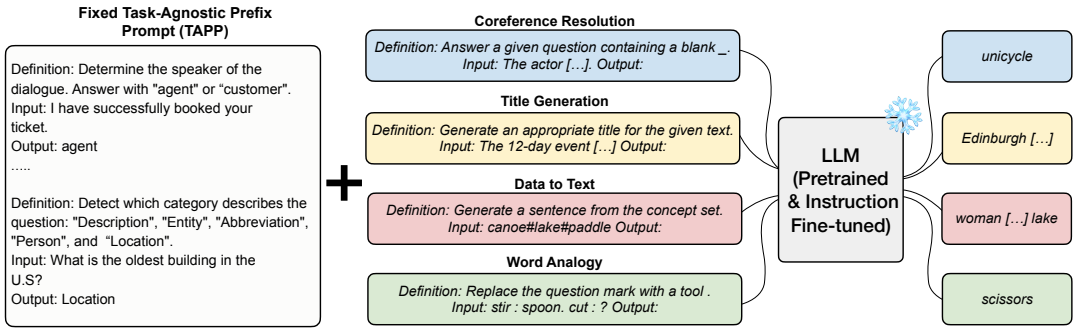


Figure 1: Overview of Task-Agnostic Prefix Prompt (TAPP). We construct a fixed set of demonstrations consisting of instruction, input, and output instances to evaluate base and instruction-fine-tuned LLMs for all tasks. The task categories included in the demonstrations are strictly held-out and from the tasks being evaluated, ensuring a zero-shot setting.

approaches during inference. Language models pretrained to predict the next token autoregressively possess the ability to adapt to the target tasks when conditioned on only a few task-specific training examples without gradient update (Brown et al. 2020; Chowdhery et al. 2022; Akyürek et al. 2022; von Oswald et al. 2022; Garg et al. 2022; Dai et al. 2022). However, few-shot in-context learning requires access to target task demonstrations for each task, implying that the user has to take the effort of generating the demonstrations for each task by themselves. To address this issue, Lyu et al. (2022) propose Zero-shot In-Context Learning method (Z-ICL), retrieving relevant sentences from an external corpus and assigning random labels to construct demonstrations for classification target tasks. However, Z-ICL is only applicable for single-sentence classification tasks and tasks that only have single-word answer choices. Also, Z-ICL assumes that the output distribution of the task is given. In contrast, our work observes the effect of task-agnostic prefix prompts without any restrictions on the type of the downstream task or the necessity of additional information about the task, which makes the approach applicable even for real-time scenarios.

Instruction-Following LLMs Recent works have shown that fine-tuning-based instruction learning, e.g., instruction tuning or RLHF, can boost the capability of LLMs to follow instructions or align to human preferences (Sanh et al. 2021; Wei et al. 2021; Wang et al. 2022c; Chung et al. 2022; Min et al. 2022a; Ye et al. 2022; Ouyang et al. 2022; Bai et al. 2022; OpenAI 2022). These works have demonstrated that the effect of instruction fine-tuning can be maximized by scaling the size of the base model or by training on a more diverse set of tasks. However, whether the instruction following ability of LLMs is newly obtained through instruction tuning or is already obtained during pretraining is under-explored. Wang et al. (2022b); Honovich et al. (2022) show that downstream tasks generated by LLMs themselves which contain noisy instances can actually be good training instances for instruction tuning, implying that LLMs are already somewhat aware of instructions. We extend this hypothesis that base LLMs already have the capability to follow instructions by showing that applying TAPP regardless of the target task without performing any backpropagation, i.e., using the base model checkpoint without any gradient update, improves the performance on the target downstream tasks.

Task-Agnostic Prefix Prompt

TAPP consists of cross-task demonstrations where each is a concatenation of instruction, input, and output instance, as shown in Figure 1. The exact prompt is provided in the Appendix. In this section, we explain the rules we have used to construct TAPP. Also, we mention the advantages of applying TAPP during the inference of LLMs for zero-shot task generalization.

TAPP Construction

We select K tasks as demonstrations for TAPP from a task pool containing a total of N tasks, with each task instance consisting of an instruction, input, and output[1]. We apply some simple heuristics to first filter the task set, randomly sample a single instance per filtered task set, and lastly, sample K instances all corresponding to different tasks. The rules are as follows:

1. Task Types: We only sample from classification tasks that explicitly include an answer choice in the instruction (e.g., *"agent" or "customer"* in Figure 1). We hypothesize that including the answer choice in the instruction might assist LLMs to follow instructions during inference.
2. Answer Choice Overlap: We ensure that the answer choices do not overlap between demonstrations. We expect that the overlap of answer choices leads to LLMs copying the labels of the demonstrations, similar to the copying effect during inference of LLMs (Lyu et al. 2022).
3. Maximum Length: We restrict the length of the concatenation of instruction, input, and output instance for each demonstration to 256 tokens by a maximum considering the maximum sequence length[2].
4. Ordering: We order the demonstrations by the number of answer choices for each task in ascending order. For demonstrations having the same number of answer

[1]Unless specified, we set $K = 8$ as default.

[2]Because we mainly experiment on 175B-sized GPT-3, we set the default maximum input sequence as 2048.

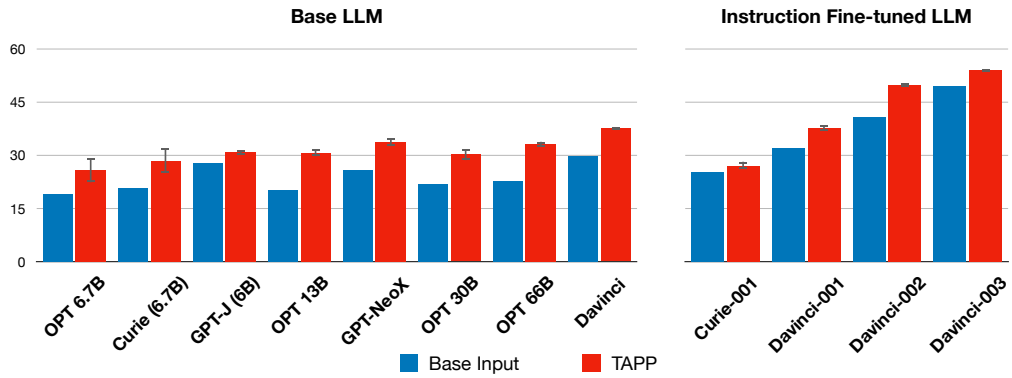


Figure 2: Average performance of 119 evaluation tasks on SUPERNI benchmark. TAPP is effective for both base and instruction-fine-tuned LLMs. We report the mean score of three random seeds for different demonstration sets for TAPP and the error bars of standard deviation. We also perform an evaluation using INSTRUCTSCORE (Xu et al. 2023) and provide the full demonstration sets in the Appendix.

choices, we sort by demonstration length in ascending order.

TAPP for Zero-Shot Task Generalization

After randomly sampling K tasks from a set of tasks that satisfy the criteria and ordering them by the criterion, we construct a fixed set of demonstrations $M = [M_1, M_2, ..., M_K]$ (TAPP) and prepend it on the concatenation of instruction (I_t) and i-th input instance (x_{ti}) of the target task t. The response (y_{ti}) of the model parameterized by θ is calculated as follows:

$$\arg\max P(y_{ti}|M, I_t, x_{ti}; \theta) \quad (1)$$

where M is invariant regardless of the target task t and K is the number of demonstrations. We ensure that the K tasks comprising the demonstration set of TAPP are strictly held-out from the target task T in order to measure the effect of TAPP for zero-shot task generalization.

It is worth noting that TAPP is different from canonical task-specific prompts which usually vary depending on the target task. TAPP is a fixed prefix prompt that can be prepended to any target task without any restriction, being easily reproducible and widely applicable. Also, TAPP does not require any additional information during inference such as the task category information or the output distribution of the target task, unlike task-specific prompts such as few-shot prompting or the approach of Lyu et al. (2022).

Experiments

Experimental Setup

We construct the demonstrations for TAPP by utilizing English training tasks of SUPER-NATURALINSTRUCTIONS (SUPERNI) benchmark (Wang et al. 2022c) as the task pool, which includes 756 tasks in total. To evaluate the effectiveness of TAPP, we use the held-out tasks from SUPERNI for testing, which consists of 119 tasks across 12 different categories, including free-form generation, word relation reasoning, and various classification tasks. We select SUPERNI as our evaluation benchmark because it offers a diverse set of tasks with varying levels of complexity. Each task has 100 instances, and we exclude instances that exceed the maximum sequence length, resulting in a total of 11,802 instances. We use different evaluation metrics for each task, such as Exact Match for classification or single-word prediction tasks and ROUGE-L for free-form generation tasks, following the metric used in Wang et al. (2022c). We provide the list of 12 evaluation task categories and more detailed evaluation settings in the Appendix.

Model Types We evaluate 4 LLMs with various model sizes: 1) GPT-3 (Brown et al. 2020), 2) OPT (Zhang et al. 2022), 3) GPT-NeoX (Black et al. 2022), and 4) GPT-J (Wang and Komatsuzaki 2021)[3] For GPT-3, we evaluate not only the base LLM but also evaluate LLMs that are fine-tuned to follow instructions and aligned to human preferences through reinforcement learning (Ouyang et al. 2022). We evaluate the performance of GPT-3 models with sizes of 6.7B and 175B. For OPT, we evaluate models with 6.7B, 13B, and 30B parameters, while for GPT-NeoX and GPT-J, we evaluate models with 20B and 6B parameters, respectively[4].

Results

Various base LLMs benefit from TAPP. As shown in the left part of Figure 2, Task-Agnostic Prefix Prompt (TAPP) consistently improves the performance of base LLMs (i.e not fine-tuned with instructions) across all model scales, resulting in over 50% performance increase for OPT-13B. Using this fixed prompt, smaller models can outperform much larger models without TAPP (Base Input). Specifically, the 6B-sized GPT-J model with TAPP exceeds 30 times larger

[3]From preliminary experiments, we observe that applying TAPP harms the performance for OPT-IML (Iyer et al. 2022) and FLAN-T5 (Chung et al. 2022) due to the characteristics of each model. We provide more discussion in the Appendix.

[4]We do not evaluate on GPT-3.5 (OpenAI 2022) or GPT-4 (OpenAI 2023) since the model details such as the model size or architecture are not known.

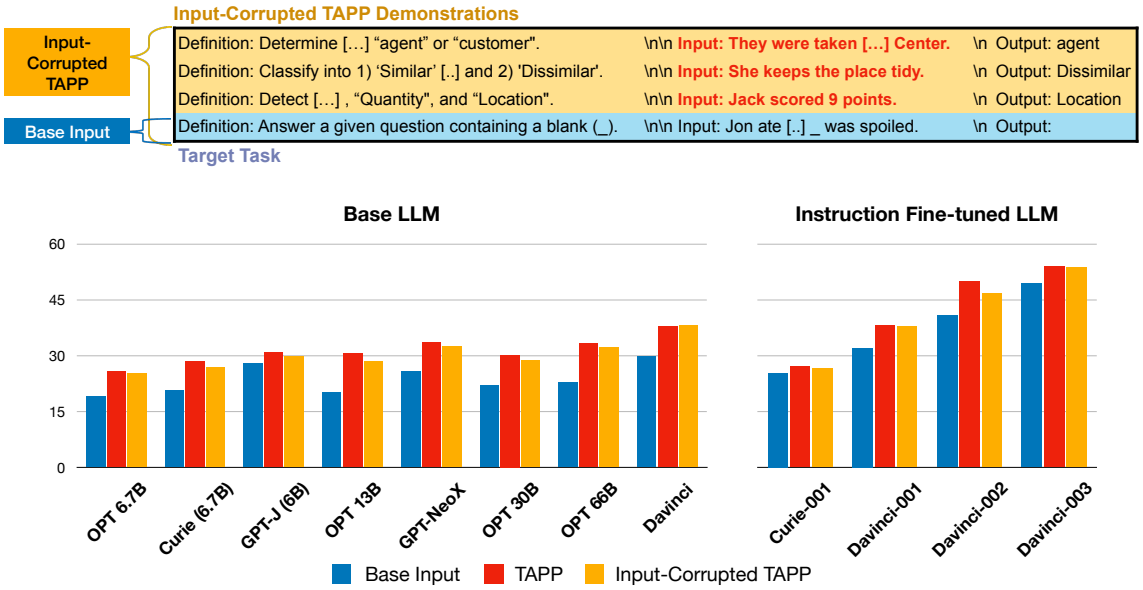


Figure 3: (Top) Example of Input-corrupted TAPP, where we corrupt the input instance distribution of the demonstrations. (Bottom) Comparison with Base Input, TAPP, and Input-corrupted TAPP. For most of the models, input distribution corruption does not harm the performance much. We report the mean score of three random seeds for different demonstration sets for TAPP. We report a result of a single seed for 175B-sized models due to inference costs. We provide the full demonstration sets in the Appendix.

175B-sized GPT-3 model without TAPP. This shows that TAPP can improve the ability of base LLMs to follow instructions without fine-tuning or backpropagation. Moreover, we observe the gain from TAPP during inference can outperform the gain from instruction tuning by comparing the performance of TAPP applied to GPT-3 model without instruction tuning (davinci) and the base input setting of the instruction-tuned GPT-3 model (text-davinci-001).

The gain from TAPP is complementary to instruction fine-tuning. As shown in the right part of Figure 2, we observe that TAPP improves the performance of LLMs fine-tuned through instruction tuning or RLHF, especially for models over 100B parameters. This implies that instruction fine-tuning alone might be sometimes insufficient for larger models and pretending a fixed prefix prompt can improve the instruction following ability orthogonally. In particular, we observe a significant performance improvement for text-davinci-002 (175B), outperforming an RLHF-tuned model text-davinci-003 with base input. Also, we demonstrate that the most powerful model (text-davinci-003) also benefits from TAPP by 9.3%, achieving the best performance. We leave detailed analysis on more diverse instruction-fine-tuned models as future work.

Input Corruption of TAPP does not harm the performance much. In Figure 3, we observe that corrupting the distribution of input instances for each demonstration for TAPP does not harm the performance much, similar to the observation in Min et al. (2022b) for few-shot in-context learning. Instead of perturbing the input-output correspondence, done in Min et al. (2022b), we perturb the input distribution *itself*, which is a setting where there are more corruptions as shown at the top of Figure 3. Following Min et al. (2022b), we use CC-News (Hamborg et al. 2017) as an external corpus to replace the ground truth input instance with random sentences that have a similar length to the original input instance. As shown in the bottom of Figure 3, corrupting the input instance distribution of each demonstration does not harm the performance much across most model scales. This is in line with the observations made in previous works that LLMs do not make full use of all the information provided to them (Min et al. 2022b; Webson and Pavlick 2021; Madaan and Yazdanbakhsh 2022; Wang et al. 2022a). Interestingly, unlike few-shot in-context learning where corrupting the input distribution itself leads to significant performance degradation, we demonstrate that not only the input-output correspondence does not matter, but also the input instance distribution matters little for TAPP.

Analysis

We additionally investigate the factors that make TAPP effective. We evaluate base GPT-3 175B checkpoint on a single task per task category, a total of 12 tasks due to API costs[5].

Additional Experiments

Comparison with Task-specific Prefix Prompts We compare the performance of TAPP with other prefix prompts that

[5]We select a single task per task category with a significant discrepancy between the lower bound and upper bound performance across davinci, text-davinci-001, 002, 003 models to see the tendency more clearly.

	Category	Output	Task	AVG
Base (No PP)	✗	✗	✗	29.66
TAPP	✗	✗	✗	**44.24**
Nearest PP	✗	✗	✗	44.16
Category PP	✓	✗	✗	42.43
Output PP	✗	✓	✗	34.34
Few-shot ICL	✓	✓	✓	56.65
+ TAPP	✓	✓	✓	**60.21**

Table 1: We compare the performance of TAPP with different strategies to construct task-specific Prefix Prompts (PP): Nearest PP, Category PP, and Output PP. Unlike other approaches, TAPP is fixed regardless of the target task and does not require any information about the task category, output, or target task. Additionally, we observe prepending TAPP to target task demonstrations (Few-shot ICL) enhances the performance.

are task-specific, meaning that the prefix prompt depends on the target task rather than being fixed. We compare with three approaches: Nearest PP, Category PP, and Output PP. First, for Nearest PP, we construct the prefix prompt by retrieving top-K similar instances for each target task from training tasks of SUPERNI using SimCSE (Gao, Yao, and Chen 2021) search tool, similar to the setting of Lyu et al. (2022)[6]. Second, for Category PP, we construct the prefix prompt by randomly sampling demonstrations from the task that belongs to the same task category (e.g., question answering), from the evaluation tasks of SUPERNI benchmark but excluding the same task, assuming that the task category information is provided during inference. Third, for Output PP, we utilize the output label of few-shot demonstrations of the target task while corrupting the input distribution of each demonstration, following the input corruption setting of Min et al. (2022b). This setting is equivalent to providing only the output distribution through demonstrations.

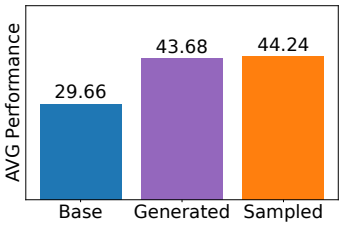


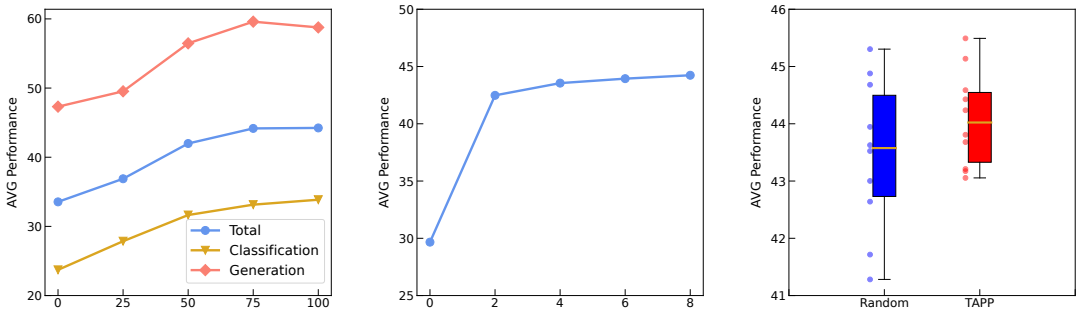
Figure 4: The result of TAPP using demonstrations generated by ChatGPT (OpenAI 2022). Machine-generated demonstrations show comparable performance to demonstrations sampled from SUPERNI benchmark.

[6]Note that the original setting of Lyu et al. (2022) is only applicable for single sentence classification tasks and for tasks that have single word answer choices. Therefore, the method cannot be directly compared to benchmarks that include a diverse collection of tasks such as SUPERNI.

Results in Table 1 show that TAPP is comparable to or outperforms other task-specific prefix prompts. First, we find that Nearest PP does not outperform TAPP. This indicates that using an external search tool to find similar demonstrations for each target task might not help much. Second, Category PP slightly underperforms TAPP because constructing cross-task demonstrations that are similar to the target task sometimes leads to copying the output of the demonstration, similar to the copying effect observed in Lyu et al. (2022). Lastly, we observe that Output PP significantly underperforms TAPP. Although Output PP outperforms TAPP for classification tasks (36.83 vs 43.67), it significantly underperforms for generation tasks (51.93 vs 24.98). We hypothesize that this is because while the correspondence between input and output for each demonstration is less crucial for classification tasks (Min et al. 2022b), the correspondence is important for generation tasks, giving a distracting signal to the LLM if the correspondence is not matched (Shi et al. 2023). Through these results, we observe that TAPP shows comparable or better performance compared to task-specific prefix prompts while not requiring any additional information or search tools.

Orthogonality with Few-shot In-Context Learning From the result of Figure 2, we have observed that TAPP enhances the performance of instruction-fine-tuned LLMs. Here, we investigate if TAPP also enhances few-shot in-context learning, which assumes that in addition to category and output information, the target task information is provided through demonstrations of the target task. We use 4-shot few-shot demonstrations for Few-shot ICL and prepend 4 demonstrations for TAPP to fit the input into the maximum sequence length. As shown in Table 1, prepending TAPP to Few-shot ICL boosts the performance, implying that TAPP can also enhance few-shot in-context learning through a fixed prefix prompt. Additionally, we observe that prepending 4-shot TAPP to 4-shot Few-shot ICL setting largely reduces the performance gap between 8-shot Few-shot ICL without TAPP (60.21 vs 61.71). This suggests the advantage of TAPP in real-time LLM serving scenarios: while the users can save the effort of manually constructing twice more task-specific demonstrations for each task, they can achieve similar performance by simply prepending TAPP to the input.

Comparison with Machine-Generated Prefix Prompts We explore if TAPP shows effectiveness for machine-generated demonstrations instead of sampling from the task pool (SUPERNI). We use ChatGPT (OpenAI 2022) for demonstration generation by specifying the rules used to construct the demonstration set for TAPP. As shown in Figure 4, TAPP is also effective for machine-generated demonstrations, showing comparable performance to TAPP with demonstrations from SUPERNI and significantly outperforming the result without any prefix prompt. This finding suggests that TAPP is effective even without a sampling process from benchmarks that consist of diverse instructions, indicating that the performance enhancement is not from demonstration construction through sampling, but is from the construction rule and the format of TAPP. We provide an example of a machine-generated demonstration set in the Appendix.



(a) Classification Ratio (%) (b) Number of Demonstrations (c) Ordering of Demonstrations

Figure 5: (a) shows that the average performance increases as the ratio of classification tasks that are used as demonstrations for TAPP increases, even for generation target tasks. (b) shows that the performance increases as the number of demonstrations increases for TAPP. (c) shows that ordering the demonstration set by the number of answer choices reduces the variance on 10 demonstration sets.

	Inst.	Input	Output	AVG
TAPP	✓	✓	✓	**44.24**
Random Inst.	✗	✓	✓	31.18
Random Input	✓	✗	✓	**44.27**
Random Output	✓	✓	✗	38.30

Table 2: Corrupting the distribution of each component (instruction, input, output) of the demonstration of TAPP by replacing it with random words or sentences. An example of each demonstration corruption is shown in the Appendix.

Ablation Studies

Instruction and output distribution of the demonstrations of matters. We further analyze the effectiveness of each component of the demonstrations for TAPP by corrupting the distribution of each component: instruction, input, and output instance. For instruction corruption, we replace the ground truth sequences with random sequences from an external corpus, which is similar to how we corrupt the input distribution. For output corruption, we replace ground truth labels with random English words, following Min et al. (2022b). The results are shown in Table 2. Unlike input distribution corruption results of Figure 3, corrupting the distribution of the instruction or the output instance of each demonstration significantly harms the performance. In particular, corrupting the instruction distribution shows little improvement compared to base input without any prefix prompts (31.18 vs 29.67). This suggests that, unlike input instances, the distribution of instruction and output instances significantly affects the performance of TAPP.

Constructing the demonstration set with classification tasks is important. We analyze the heuristic of constructing the demonstration set from only classification tasks in TAPP by varying the ratio of classification tasks consisting of the demonstration set. As shown in Figure 5a, the average performance increases as the ratio of classification tasks increases. Interestingly, we observe that constructing the demonstration set with classification tasks also enhances generation (non-classification) target tasks. This finding contrasts with few-shot in-context learning setting, where retrieving demonstrations similar to the target query enhances the few-shot performance (Rubin, Herzig, and Berant 2021; Liu et al. 2021) [7].

Increasing the number of demonstrations improves the performance. We study the impact of the number of demonstrations that consist TAPP. Results are shown in Figure 5b. As expected, the mean performance improves as the number of demonstrations increases. Notably, the instruction-following ability significantly improves even with 2 demonstrations, implying that using only a small set of prefix prompts can still improve the performance of LLMs. This also suggests that for settings where efficient computation during inference is crucial, reducing the number of demonstrations that consist TAPP might be an optimal approach since the performance degradation is not severe.

Ordering the demonstrations by the number of answer choices reduces the variance. To examine the impact of different orderings of the demonstration set, we compare the ordering of demonstrations that consist TAPP based on the number of answer choices with a random ordering. Figure 5c shows the result of 10 different demonstration sets by sampling them with 10 different random seeds. Although the mean performance does not show a significant difference between the two settings, we observe that applying ordering based on the number of answer choices reduces the variance and improves the worst-case accuracy.

Answer choice overlap between demonstrations harms the performance. We analyze the effect of answer choice overlap between demonstrations, which is one of the rules used to construct the demonstration set. We compare the demonstration set used for TAPP with the demonstration set that has the same answer choice for all demonstrations. The result is demonstrated in Table 3. We observe that the

[7] Note that the classification ratio of 0% in Figure 5a corresponds to constructing the demonstration set solely from generation (non-classification) tasks.

	Classification	Generation	Total
Overlap	**35.14**	52.32	42.30
No Overlap	33.86	**58.77**	**44.24**

Table 3: Effect of answer choice overlap between demonstrations. The demonstration set that has an overlap underperforms the set without overlap on average, especially for generation tasks.

demonstration set with answer choice overlap underperforms the demonstration set without overlap on average, especially for generation tasks. We find that the demonstration set with answer choice overlap tends to make the model generate short sequences for long text generation or predict the output by copying one of the labels of the demonstration set, leading to poor performance.

Discussion

From previous sections, we have observed that TAPP significantly boosts the performance of both base and instruction-fine-tuned LLMs. Also, we have demonstrated that corrupting the input distribution does not harm the performance much and analyzed that constructing the demonstration set from classification tasks is crucial for performance improvement. In this section, we suggest the role of TAPP based on the findings from the previous sections.

Why is constructing the demonstration set from classification tasks important? Figure 5a shows that constructing the demonstration set with classification tasks is important for TAPP. Then, what is the difference between classification and generation (non-classification) tasks? Because one of our heuristics for demonstration construction is to only consider classification tasks that include an answer choice in the instruction (e.g. *"agent" or "customer"* in Figure 1), these demonstrations have more *explicit* cues about the output distribution. We hypothesize that during inference, LLMs learn the correspondence between answer choice in the instruction (e.g. Determine the speaker of the dialogue, "agent" or "customer".) and the label (e.g. agent) from demonstrations. Especially, because the label word appears in the instruction for classification tasks, it would be easy to exploit this relationship for LLMs. We observe that adding only a sentence that includes answer choices for corrupted instruction demonstrations in Table 2 leads to an increase in the performance of TAPP ($31.18 \rightarrow 38.92$), supporting the hypothesis.

What does the result of input-corrupted TAPP imply? From Figure 3 and Table 2, we observe that the input distribution of demonstrations for TAPP does not matter much, while instruction and output distribution matter significantly. This observation bolsters the above hypothesis that LLMs learn the correspondence of answer choice in the instruction and the label of the demonstrations during TAPP. Instead of relying on complex correspondence such as the relationship between instruction, input, and output altogether, LLMs tend to focus on simple correspondence such as string matching between the instruction including answer choices and the label. Previous work also demonstrates similar findings that LLMs *takes less effort* to adapt to a task, similar to shortcut learning (Webson and Pavlick 2021; Min et al. 2022b).

What is the role of TAPP? If LLMs learn the correspondence of the answer choice in the instruction and the label of the demonstrations during TAPP, then how does this assist the instruction-following ability? During TAPP, we hypothesize that the demonstrations give a signal that assists LLMs *focus* on the instruction to more accurately estimate the output distribution, making LLMs better follow instructions. We suggest that this hypothesis explains why constructing the demonstration set from classification tasks also improves the performance of generation target tasks. As a preliminary experiment, we calculate the ratio of how much the first output token attends to the target task instruction compared to the target task input and observe that TAPP leads to increased attention to the task instruction in the Appendix. Meanwhile, our observations imply that such ability does not seem to be sufficiently activated for both base LLMs and instruction-fine-tuned LLMs. Although instruction fine-tuning also assists the signal of focusing on the instructions, we hypothesize that TAPP directly enforces the correspondence between the instruction and the label of the demonstrations during inference.

Limitations

First, although TAPP leads to significant performance gain across various LLMs, it suffers from increased computation during inference due to the increased number of input sequences. However, since we use a fixed prefix prompt for all tasks, the inference cost can be minimized through caching. Second, our evaluation is mainly based on heuristic metrics such as ROUGE and Exact Match scores. We leave investigating the effect of TAPP using qualitative evaluation settings by recruiting human evaluators as future work. Third, our interpretation of the role of TAPP is hypothetical, whereas further interpretation of the role of TAPP can be conducted by analyzing the inner operation inside the model (Lieberum et al. 2023; Grosse et al. 2023).

Conclusion

In this paper, we explore the effectiveness of Task-Agnostic Prefix Prompt (TAPP) for instruction-following during inference of LLMs. We observe that prepending TAPP that is determined through simple heuristics significantly enhances the performance of both base and instruction-fine-tuned LLMs. TAPP differs from task-specific prompts in that it is a fixed prompt that can be prepended to any target task. Through detailed analysis, we hypothesize that the effect of TAPP comes from learning the correspondence between answer choice in the instruction and the label of the classification task demonstrations consisting of TAPP, leading LLMs to better focus on the instruction. To this end, our work demonstrates the effect of task-agnostic prefix prompts for a diverse set of tasks and suggests research direction for exploring various approaches that further activate the instruction-following ability of LLMs.

Acknowledgments

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2022-0-00113, Developing a Sustainable Collaborative Multi-modal Lifelong Learning Framework, 80%; No.2019-0-00075, Artificial Intelligence Graduate School Program (KAIST), 20%).

References

Akyürek, E.; Schuurmans, D.; Andreas, J.; Ma, T.; and Zhou, D. 2022. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*.

Askell, A.; Bai, Y.; Chen, A.; Drain, D.; Ganguli, D.; Henighan, T.; Jones, A.; Joseph, N.; Mann, B.; DasSarma, N.; et al. 2021. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*.

Bai, Y.; Jones, A.; Ndousse, K.; Askell, A.; Chen, A.; DasSarma, N.; Drain, D.; Fort, S.; Ganguli, D.; Henighan, T.; et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Black, S.; Biderman, S.; Hallahan, E.; Anthony, Q.; Gao, L.; Golding, L.; He, H.; Leahy, C.; McDonell, K.; Phang, J.; et al. 2022. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*.

Chowdhery, A.; Narang, S.; Devlin, J.; Bosma, M.; Mishra, G.; Roberts, A.; Barham, P.; Chung, H. W.; Sutton, C.; Gehrmann, S.; et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Chung, H. W.; Hou, L.; Longpre, S.; Zoph, B.; Tay, Y.; Fedus, W.; Li, E.; Wang, X.; Dehghani, M.; Brahma, S.; et al. 2022. Scaling Instruction-Finetuned Language Models. *arXiv preprint arXiv:2210.11416*.

Dai, D.; Sun, Y.; Dong, L.; Hao, Y.; Sui, Z.; and Wei, F. 2022. Why Can GPT Learn In-Context? Language Models Secretly Perform Gradient Descent as Meta Optimizers. *arXiv preprint arXiv:2212.10559*.

Gao, T.; Yao, X.; and Chen, D. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Garg, S.; Tsipras, D.; Liang, P.; and Valiant, G. 2022. What can transformers learn in-context? a case study of simple function classes. *arXiv preprint arXiv:2208.01066*.

Grosse, R.; Bae, J.; Anil, C.; Elhage, N.; Tamkin, A.; Tajdini, A.; Steiner, B.; Li, D.; Durmus, E.; Perez, E.; et al. 2023. Studying Large Language Model Generalization with Influence Functions. *arXiv preprint arXiv:2308.03296*.

Hamborg, F.; Meuschke, N.; Breitinger, C.; and Gipp, B. 2017. news-please: A Generic News Crawler and Extractor. In *Proceedings of the 15th International Symposium of Information Science*, 218–223.

Honovich, O.; Scialom, T.; Levy, O.; and Schick, T. 2022. Unnatural Instructions: Tuning Language Models with (Almost) No Human Labor. *arXiv preprint arXiv:2212.09689*.

Iyer, S.; Lin, X. V.; Pasunuru, R.; Mihaylov, T.; Simig, D.; Yu, P.; Shuster, K.; Wang, T.; Liu, Q.; Koura, P. S.; et al. 2022. OPT-IML: Scaling Language Model Instruction Meta Learning through the Lens of Generalization. *arXiv preprint arXiv:2212.12017*.

Lieberum, T.; Rahtz, M.; Kramár, J.; Irving, G.; Shah, R.; and Mikulik, V. 2023. Does Circuit Analysis Interpretability Scale? Evidence from Multiple Choice Capabilities in Chinchilla. *arXiv preprint arXiv:2307.09458*.

Liu, J.; Shen, D.; Zhang, Y.; Dolan, B.; Carin, L.; and Chen, W. 2022. What Makes Good In-Context Examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*. Association for Computational Linguistics.

Liu, X.; Zheng, Y.; Du, Z.; Ding, M.; Qian, Y.; Yang, Z.; and Tang, J. 2021. GPT understands, too. *arXiv preprint arXiv:2103.10385*.

Lyu, X.; Min, S.; Beltagy, I.; Zettlemoyer, L.; and Hajishirzi, H. 2022. Z-ICL: Zero-Shot In-Context Learning with Pseudo-Demonstrations. *arXiv preprint arXiv:2212.09865*.

Madaan, A.; and Yazdanbakhsh, A. 2022. Text and patterns: For effective chain of thought, it takes two to tango. *arXiv preprint arXiv:2209.07686*.

Min, S.; Lewis, M.; Zettlemoyer, L.; and Hajishirzi, H. 2022a. MetaICL: Learning to Learn In Context. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Min, S.; Lyu, X.; Holtzman, A.; Artetxe, M.; Lewis, M.; Hajishirzi, H.; and Zettlemoyer, L. 2022b. Rethinking the Role of Demonstrations: What Makes In-Context Learning Work? *arXiv preprint arXiv:2202.12837*.

OpenAI. 2022. Chatgpt: Optimizing language models for dialogue. https://openai.com/blog/chatgpt/.

OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C. L.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*.

Rubin, O.; Herzig, J.; and Berant, J. 2021. Learning To Retrieve Prompts for In-Context Learning. *arXiv preprint arXiv:2112.08633*.

Sanh, V.; Webson, A.; Raffel, C.; Bach, S. H.; Sutawika, L.; Alyafeai, Z.; Chaffin, A.; Stiegler, A.; Scao, T. L.; Raja, A.; et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Shi, F.; Chen, X.; Misra, K.; Scales, N.; Dohan, D.; Chi, E. H.; Schärli, N.; and Zhou, D. 2023. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, 31210–31227. PMLR.

von Oswald, J.; Niklasson, E.; Randazzo, E.; Sacramento, J.; Mordvintsev, A.; Zhmoginov, A.; and Vladymyrov, M. 2022. Transformers learn in-context by gradient descent. *arXiv preprint arXiv:2212.07677*.

Wang, B.; and Komatsuzaki, A. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

Wang, B.; Min, S.; Deng, X.; Shen, J.; Wu, Y.; Zettlemoyer, L.; and Sun, H. 2022a. Towards Understanding Chain-of-Thought Prompting: An Empirical Study of What Matters. *arXiv preprint arXiv:2212.10001*.

Wang, Y.; Kordi, Y.; Mishra, S.; Liu, A.; Smith, N. A.; Khashabi, D.; and Hajishirzi, H. 2022b. Self-Instruct: Aligning Language Model with Self Generated Instructions. *arXiv preprint arXiv:2212.10560*.

Wang, Y.; Mishra, S.; Alipoormolabashi, P.; Kordi, Y.; Mirzaei, A.; Arunkumar, A.; Ashok, A.; Dhanasekaran, A. S.; Naik, A.; Stap, D.; et al. 2022c. Benchmarking generalization via in-context instructions on 1,600+ language tasks. *arXiv preprint arXiv:2204.07705*.

Webson, A.; and Pavlick, E. 2021. Do Prompt-Based Models Really Understand the Meaning of their Prompts? *arXiv preprint arXiv:2109.01247*.

Wei, J.; Bosma, M.; Zhao, V. Y.; Guu, K.; Yu, A. W.; Lester, B.; Du, N.; Dai, A. M.; and Le, Q. V. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Xu, W.; Wang, D.; Pan, L.; Song, Z.; Freitag, M.; Wang, W. Y.; and Li, L. 2023. INSTRUCTSCORE: Explainable Text Generation Evaluation with Finegrained Feedback. arXiv:2305.14282.

Ye, S.; Kim, D.; Jang, J.; Shin, J.; and Seo, M. 2022. Guess the Instruction! Making Language Models Stronger Zero-Shot Learners. *arXiv preprint arXiv:2210.02969*.

Zhang, S.; Roller, S.; Goyal, N.; Artetxe, M.; Chen, M.; Chen, S.; Dewan, C.; Diab, M.; Li, X.; Lin, X. V.; et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.