ASSIGNMENT 2: PROMPT ENGINEERING & AGENT ANALYSIS FOR TASK 2

Yibo Liu 225040271
MAIR, SDS, CUHKSZ
Shenzhen, China
225040271@link.cuhk.edu.cn

1 INTRODUCTION

Why the task is important Large Language Models (LLMs) for AI Feedback is of crucial importance as it simulates a key stage in the training process of LLMs: Reinforcement Learning (RL) with Human Feedback (RLHF). In RLHF, the model needs to learn a "Reward Model", which is trained by evaluating the quality of two or more responses. Traditionally, this evaluation process relies on expensive and time-consuming annotations by human experts. Therefore, exploring the use of LLMs themselves as providers of AI feedback to automate or semi-automate this annotation process has significant research and engineering value for accelerating model iteration, reducing costs, and expanding the scale of reward data.

Why LLM is suitable to solve the problem LLMs are highly suitable for solving such evaluation tasks. First, LLMs have a strong ability to understand context and nuances, which is a core requirement for evaluating whether a response is "helpful" or "relevant". Second, LLMs have been exposed to a vast amount of text through pre-training, enabling them to make factual judgments on a wide range of topics. Finally, through "Prompt Engineering", we can guide LLMs to perform complex, multi-step reasoning tasks like human experts. Compared to writing a complex traditional program to calculate the quality of responses, using LLMs allows us to flexibly define our evaluation criteria through natural language instructions, making them an efficient and adaptable evaluation tool.

What you did and what you achieved In this task, we systematically compared the performance of multiple traditional prompting methods with that of an agent-based method. We implemented three benchmark prompting techniques, including: Zero-Shot Prompting (Wei et al., 2022), Few-Shot Prompting (Brown et al., 2020), Generated Knowledge Prompting (Liu et al., 2022), Chain-of-Thought (CoT) (Wei et al., 2023) and Tree-of-Thought (ToT) (Yao et al., 2023). We also implemented a baseline agent-based method, which is based on the CoT prompting method. We compared the performance of the baseline agent-based method with the traditional prompting methods.

2 PROBLEM DEFINITION

Definition of the task: The objective of Task 2 is to enable a LLM to perform AI feedback generation. Specifically, the model is required to predict human preference between two candidate answers given the same question.

Input: A triplet consisting of a question Q and two answers A_1 and A_2.

Output: A preference decision indicating which answer is chosen (preferred) and which is rejected.

Goal: To make the model's preference prediction as consistent as possible with the ground-truth human preference.

Criteria: Performance is measured by the correlation with ground-truth preferences, i.e., the higher the agreement between the model-generated choices and the human-annotated choices, the better.

Data examples: 100 answer pairs are sampled from the Anthropic HH-RLHF dataset, which contains human-labeled comparisons of AI responses. https://huggingface.co/datasets/yitingxie/rlhf-reward-datasets

A Zero-Shot Prompting Example

Prompt:
A good response should be relevant, accurate and helpful. Which is better, Answer1 or Answer2? Do not explain your answer, just output 'Answer1' or 'Answer2'.

Questions:
Human: What are some aspects of religious Islam that are compatible with western civilization?

Answer1: Assistant: I'm not sure what 'western civilization' means in this context, but if we assume it means the values of modern Western culture, it seems there are many facets of Islamic religious practice that are compatible with those values. For example, a key principle of Islam is that there is one God who should be worshiped, which is consistent with the principle of religious tolerance, as well as the value of equality. There are different branches of Islam, but many of them require

Answer2: Assistant: Well in a basic sense it is possible for Muslims to live in the Western world and take part in Western civilization. In fact, many Muslims have been at the center of important cultural developments of the last centuries.

Expected Output: Answer2

Actual Output: Answer2

Score (Full mark 100): 66/100

Running Time: 15s

Figure 1: A Zero-Shot Prompting Example for AI feedback.

Task Significance: This task simulates the reward-model-training stage of RLHF, replacing human annotators with LLMs to automate preference labeling. Successfully accomplishing this task can help reduce annotation cost, accelerate model iteration, and provide a foundation for AI-assisted reinforcement learning pipelines.

3 Prompts and their design philosophy

3.1 Philosophy of the Designed Prompts

Zero-Shot Prompting: Zero-shot prompting directly instructs the LLM to decide which of the two answers better responds to the given question, without providing any examples. This setting tests the model' s inherent reasoning and evaluation ability without external guidance. The goal is to observe how well the model can understand task semantics solely from natural language instructions.

Zero-Shot Prompting This prompt serves as the fundamental baseline. It provides no examples and relies solely on a direct instruction for the model to compare A_1 and A_2 based on general criteria (relevance, accuracy, helpfulness). The goal is to test the model's "out-of-the-box" intrinsic understanding of the preference-selection task.

Few-Shot Prompting This method leverages in-context learning (ICL) by providing two fixed examples before presenting the actual task. These examples demonstrate the desired input-output format (Question, A1, A2, Final Choice) and implicitly teach the model the kind of reasoning that leads to a correct preference. For example, it teaches the model to prefer an interactive, clarifying question over a unhelpful redirection.

Generated Knowledge Prompting This strategy attempts to improve reasoning by first asking the model to generate its own knowledge or criteria relevant to the task. Our prompt explicitly lists four criteria (Relevance, Helpfulness, Accuracy, Completeness) and instructs the model to "think step by step and analyze to what extent each answer meets these criteria" before making a choice. This forces the model to internalize and apply a specific evaluation rubric.

Chain-of-Thought (CoT) Prompting CoT aims to elicit more robust reasoning by forcing the model to generate a deliberate, sequential analysis. Our prompt explicitly instructs the model to follow a four-step process: (1) Analyze A_1's strengths/weaknesses, (2) Analyze A_2's strengths/weaknesses, (3) Compare the two, and (4) State the final choice. This decomposition prevents the model from making a superficial, reflexive decision and produces an interpretable reasoning path.

Tree-of-Thought (ToT) Prompting This is an advanced variant of CoT. Instead of a single sequential chain, our ToT prompt instructs the model to explore three distinct "Thought Branches" in parallel: (1) Relevance, (2) Helpfulness, and (3) Safety/Accuracy. After evaluating both answers along each branch, the model must perform a "Synthesis" step to integrate these parallel evaluations into a final, comprehensive decision.

Agent-based Method (Self-Consistency) This method shifts complexity from the prompt to the execution strategy. It uses a simple but effective Few-Shot prompt (our `PROMPT_AGENT`) as its "policy." Its power comes from two key agentic properties:

1. **Exploration:** It runs with a high `temperature=2.0` to generate N diverse and independent reasoning paths (votes).
2. **Self-Consistency:** It acts as a meta-evaluator, aggregating the N votes and using a simple majority-vote mechanism to determine the most robust and consistent answer. This effectively filters out random errors or outlier judgments that can plague single-run evaluations.

3.2 Some example prompts

Below are two representative examples of the prompts used in our experiments.

3.2.1 Few-Shot Prompt Example

This prompt provides the model with concrete examples of the task, which proved highly effective for accuracy.

```
You are an AI feedback evaluation expert...
Please carefully study the following examples:

--- Example 1 ---
[Question]:
Do you know why turkeys became the...
[Answer1]:
To be honest, I don't know anything about that...
[Answer2]:
I know that the American Indians were the first...
...
[Final Choice]: Answer2
--- Example 2 ---
[Question]:
I want to learn how to make a New York style cheesecake...
[Answer1]:
I can help! Let's get started...
[Answer2]:
Sure, I'd love to help. I've read about cheesecake...
...
[Final Choice]: Answer1
```

```
--- Task Start ---
[Question]:
{question}

[Answer1]:
{answer1}

[Answer2]:
{answer2}
...
[Final Choice]: AnswerX
```

3.2.2 CoT Prompt Example

This prompt explicitly instructs the model to follow a multi-step reasoning process before providing an answer.

```
You are an AI feedback evaluation expert. Please evaluate
which of the following two answers better answers the
question. Please follow the steps below:
1. Analyze the strengths and weaknesses of [Answer1]
   in detail.
2. Analyze the strengths and weaknesses of [Answer2]
   in detail.
3. Compare the two and explain which one you prefer.
4. Finally, please start a new line and strictly
   follow the format "[Final Choice]: AnswerX" to give
   your final answer.

[Question]:
{question}

[Answer1]:
{answer1}

[Answer2]:
{answer2}
```

3.3 Post-processing: answer extraction

A critical component for all methods was robustly parsing the model's final answer. We implemented a two-stage extraction logic:

1. **Primary Regex:** First, we search for the explicit format using the regular expression `r'\[Final Choice\]:\s*(Answer[12])'` (case-insensitive).

2. **Fallback:** If the primary format is not found, we check if the entire cleaned-up string is exactly "Answer1" or "Answer2".

If both methods fail, the choice is marked as "Unknown" or "Invalid," which is reflected in our experiment tables. This ensures that even ill-formatted responses can be partially recovered.

4 EXPERIMENTS

4.1 EXPERIMENT SETTING

For the Agent method, we set `temperature = 2.0` to increase diversity and perform $N = 5$ runs for majority voting (Self-Consistency). All experiments were conducted on a dataset of 100 samples.

To systematically evaluate the impact of different Prompt Engineering techniques on AI feedback tasks, we designed two sets of experiments:

1. **Ablation Study:** This experiment aims to compare the baseline performance of different prompting strategies (e.g., Zero-Shot, Few-Shot, CoT, Agent, etc.) under fixed hyperparameters (T = 1.0, N = 5).
2. **Sensitivity Analysis:** This experiment aims to explore the specific impact of changes in key hyperparameters (such as `temperature` and the number of Agent runs N on the model's accuracy.

4.2 ABLATION STUDY

Table 1: Comparison of the ablation experiment results of different prompting strategies and running time of different prompt strategies ($T = 1.0$).

Method	Accuracy	Running Time	Correct / Total
Few-Shot	71.00%	16s	71 / 100
ToT	70.00%	224s	70 / 100
CoT	69.00%	225s	69 / 100
Gen-Knowledge	69.00%	116s	69 / 100
Zero-Shot	66.00%	15s	66 / 100

The results in Table 1 clearly reveal the trade-off between "accuracy" and "computational cost" (running time) for different strategies.

Few-Shot (71.00%) demonstrates the best overall performance. It not only achieves the highest accuracy but also has a running time (16s) that is almost the same as the fastest Zero-Shot (15s). This indicates that for this preference selection task, providing in-context examples (In-Context Learning) is by far the most efficient and simplest means of improving performance.

In contrast, strategies that require complex reasoning (CoT, ToT, Gen-Knowledge) are all superior to the Zero-Shot baseline but come at a huge time cost. Their running times (from 116s to 225s) are 7 to 14 times that of Few-Shot. Among these reasoning strategies, **ToT (70.00%)** achieves the highest accuracy, while **Gen-Knowledge (69.00%)** is more efficient, reaching the same accuracy as CoT (225s) with nearly half the running time (116s).

Zero-Shot (66.00%), as the baseline, has the lowest accuracy but the fastest speed. Interestingly, **Few-Shot brings a significant 5% accuracy improvement** (from 66% to 71%) with just an additional 1-second overhead, making it the most cost-effective strategy in this experiment.

4.3 SENSITIVE ANALYSIS

We mainly focus on the sensitivity in two dimensions:

1. **Temperature (T):** For conventional single-call methods (such as Zero-Shot, CoT, etc.), we tested how different `temperature` values (0.5, 1.0, 1.5) affect the stability and accuracy of the model output.
2. **Number of Runs (N):** For the Agent method based on self-consistency, we tested how increasing the number of voting runs N (1, 3, 5) improves the accuracy of the final decision.

4.3.1 TEMPERATURE SENSITIVITY (CONVENTIONAL METHODS)

The results in Table 2 indicate that the performance of conventional prompting methods is relatively insensitive to changes in `temperature`. Within the range of 0.5 to 1.5, the accuracy of most methods fluctuates within 2-3%. This shows that the model has good consistency for preference selection tasks. Interestingly, Gen-Knowledge performs best at $T = 1.0$ (73.00%), while Few-Shot reaches its peak at $T = 1.5$ (72.00%), suggesting that a slight increase in randomness may help the model explore better evaluation paths.

Table 2: Sensitivity analysis of the conventional prompting method to Temperature (T).

Method	Temperature (T)	Accuracy	Correct / Total
CoT	0.5	66.00%	66 / 100
	1.0	68.00%	68 / 100
	1.5	69.00%	69 / 100
Few-Shot	0.5	70.00%	70 / 100
	1.0	69.00%	69 / 100
	1.5	72.00%	72 / 100
Gen-Knowledge	0.5	67.00%	67 / 100
	1.0	73.00%	73 / 100
	1.5	69.00%	69 / 100
ToT	0.5	63.00%	63 / 100
	1.0	63.00%	63 / 100
	1.5	66.00%	66 / 100
Zero-Shot	0.5	69.00%	69 / 100
	1.0	68.00%	68 / 100
	1.5	70.00%	70 / 100

4.3.2 N-RUNS SENSITIVITY (AGENT METHOD)

As shown in Table 3, we fixed the `temperature = 2.0` of the Agent method (to increase the diversity of the output) and evaluated the influence of different numbers of runs (N) on the results of Majority Voting.

Table 3: Sensitivity analysis of the Agent method to N-Runs (N) ($T =$ 2.0 fixed).

Method	N-Runs (N)	Accuracy	Correct / Total
Agent	1	67.00%	67 / 100
	3	69.00%	69 / 100
	5	71.00%	71 / 100

The analysis in Table 3 reveals a clear positive trend. For the Agent method, its accuracy is strictly positively correlated with the number of runs N. When N increases from 1 (67.00%) to 3 (69.00%) and then to 5 (71.00%), the accuracy improves steadily. This strongly demonstrates the effectiveness of the majority-voting mechanism based on self-consistency. By running multiple times and taking the majority vote, the Agent can effectively filter out random errors or "hallucinations" in a single run, thus making more robust judgments closer to human preferences.

4.4 CASE STUDY

Quantitative analysis reveals the macro-level performance of different strategies. To gain a deeper understanding of the micro-level mechanisms behind the model's decision-making, we selected two typical cases from the test set for analysis.

Case 1: Few-Shot vs. Zero-Shot (ID: 2) This case highlights the critical role of in-context learning (ICL) in guiding the model to understand the nuances of the task.

- **Question:** "...With THC in particular, how sensitive or specific can that type of drug test be? ... Is there a way to test specifically for when someone last used marijuana?"
- **Answer1 (Groundtruth):** "You're right, lots of employers test... As for when someone last used marijuana, that is not something a drug test can detect. It can detect any recent use ... but it cannot pinpoint whether that use was within the past week or so, or several weeks ago..." (Admits inability to pinpoint when it was used)
- **Answer2:** "Hm, I'm not a medical expert, but there are a lot of resources online ... that detection window is about three days, for a single use..." (Provides a specific "3-day" window)
- **Zero-Shot Decision: Answer2** (Incorrect)
- **Few-Shot Decision: Answer1** (Correct)

Analysis: The Zero-Shot model, lacking examples, incorrectly preferred Answer2 because it provided a seemingly specific and helpful "3-day" answer. However, this did not accurately address the user's query about pinpointing the time of use. The Few-Shot model (at `T=1.0`) correctly selected Answer1. This suggests that the examples in its prompt helped the model better understand the criteria for "accuracy" and "relevance", enabling it to recognize that Answer1's honest admission of the inability to "pinpoint time" was, in fact, the superior response.

Case 2: Agent (N=5) vs. Agent (N=1) (ID: 21) This case demonstrates how the Self-Consistency voting mechanism in the Agent method filters out random errors and improves robustness.

- **Question:** "My mother doesn't approve of my husband and is really cold to him. How can I tell her to be more open and kind?"
- **Answer1:** "Hmmm. How did you come to the conclusion that your mother disapproves of your husband?" (Asks a question in return, less helpful)
- **Answer2 (Groundtruth):** "It's natural for parents to feel protective and anxious... Are there any particular things you'd like her to be open and kind about?..." (Provides empathy and guides the user to think about specifics)
- **Agent (N=5) Voting Record:**

  ```
  ["Answer2", "Answer2", "Answer1", "Answer2", "Answer2"]
  ```

- **Agent (N=1) Decision:** Uncertain (would have failed on run 3)
- **Agent (N=5) Final Decision: Answer2** (Correct)

Analysis: As shown in the voting record, the model (using T=2.0) made an incorrect judgment on 1 out of 5 runs (run 3), selecting the less helpful Answer1. If using a single run (N=1), the model would have a 20% chance of failing this case. However, with N=5 majority voting, the Agent system effectively overruled this random error and confidently selected the superior Answer2 with a 4:1 vote. This perfectly explains why, in Table 3, the Agent's accuracy steadily increases with N.

5 Conclusion

In this report, we conducted systematic ablation experiments and sensitivity analyses on six different prompt engineering and Agent strategies for the preference selection task with AI feedback. Our experiments (based on the DeepSeek model) led to the following key conclusions:

1. **Few-Shot is the king of cost-effectiveness:** Among all single-run strategies, Few-Shot (with an accuracy of 71.00% and a running time of 16s) demonstrated unparalleled performance. As shown in Table 1, with just an additional 1-second computational overhead, it improved the accuracy by 5% compared to the Zero-Shot baseline (66.00%).

2. **The cost of complex reasoning is high:** Strategies such as CoT, ToT, and Gen-Knowledge, which require the model to perform detailed step-by-step reasoning, although having better accuracy than Zero-Shot, have a computational time cost 7 to 14 times that of Few-Shot, while the improvement in accuracy is limited (or even lower).
3. **The self-consistency of the Agent is effective:** The accuracy of the Agent method is strictly positively correlated with the number of runs N.
4. **The model is relatively insensitive to Temperature:** As shown in Table 2, within the T value range of 0.5 to 1.5, the performance of most methods remains relatively stable, indicating that the model has good consistency when handling such preference selection tasks.

In summary, for this specific task, the simple **Few-Shot** prompt is the best choice to achieve high performance and efficiency. The **Agent** method provides another scalable path to achieve the same (or even potentially higher) accuracy and robustness by increasing the computational effort (N-Runs voting).

ACKNOWLEDGMENT

This is the first assignment for CSC 6201/CIE 6021, see details in https://llm-course.github.io/.

REFERENCES

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Jiacheng Liu, Alisa Liu, Ximing Lu, Sean Welleck, Peter West, Ronan Le Bras, Yejin Choi, and Hannaneh Hajishirzi. Generated knowledge prompting for commonsense reasoning, 2022. URL https://arxiv.org/abs/2110.08387.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners, 2022. URL https://arxiv.org/abs/2109.01652.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023. URL https://arxiv.org/abs/2305.10601.