Iterative Prompt Engineering for NPC Dialogue Generation: CPDC 2025 API Track

Yukito Minari, Sei Ueno, and Akinobu Lee
Team: Lee-lab
Nagoya Institute of Technology
y.minari.502@stn.nitech.ac.jp
{sei.ueno,ri}@nitech.ac.jp

Abstract

This technical report presents our approach to the Commonsense Persona-Grounded Dialogue Challenge (CPDC) 2025 API Track, which aims to develop AI-driven dialogue systems for immersive video game NPCs. The API Track constrains participants to use GPT-4o-mini, requiring performance improvements through prompt engineering rather than model refinement. Building on the provided starter kit, we identified challenges through comparison with gold responses and conducted iterative improvements. Our main contributions include: (1) for Task 1 (task-oriented dialogue), we extended conversation history from single-turn to full history, added worldview information to function calling prompts, and created few-shot examples based on gold responses for handling ambiguous instructions; (2) for Task 2 (context-aware dialogue), we organized hierarchical prompt structures including role and state information, introduced explicit length constraints for conciseness, and promoted proactive information provision to infer player intentions. This report shares the details of these efforts, experimental results, and insights gained during development.

1 Introduction

Creating convincing and reliable non-player characters (NPCs) is a critical challenge in modern video game development. Traditional dialogue systems relying on pre-scripted responses limit the depth and naturalness of player-NPC interactions. The Commonsense Persona-Grounded Dialogue Challenge (CPDC) 2025 addresses this limitation by requiring AI systems to generate contextually appropriate, persona-consistent responses while executing game-specific actions in dynamic environments.

The challenge consists of three tasks: (1) Task-oriented dialogue, where NPCs must understand player requests and execute appropriate game functions; (2) Context-aware dialogue, where NPCs maintain coherent conversations based on their persona and game world; (3) Integrated dialogue and task execution, which seamlessly combines natural conversation with functional game actions. The API Track specifically constrains participants to use GPT-4o-mini, emphasizing prompt engineering and system design over model training.

In this report, we describe our iterative prompt refinement process based on the provided starter kit, using comparison with gold responses to identify areas. We implemented improvements for Task 1 extending conversation history, adding worldview information, and introducing few-shot examples. We also showed improvements for Task 2 including hierarchical prompt structures, conciseness constraints, and proactive information provision. We verified the effectiveness of each improvement through experimentation. This report shares the details of these efforts, analysis of experimental results, and insights gained during development.

2 Task Description

The CPDC 2025, organized by Sony Group Corporation, focuses on developing AI dialogue systems for video game NPCs that can conduct natural, context-aware conversations while executing game-specific actions. The challenge uses the PeaCoK dataset (Gao et al., 2023), which provides persona commonsense knowledge for consistent and engaging narratives. Full details of the challenge are available on the official page.[1]

2.1 Challenge Structure

The challenge consists of three tasks:

Task 1: Task-oriented dialogue. NPCs must understand player requests and execute appropriate game functions (e.g., retrieving information about

[1] https://www.aicrowd.com/challenges/commonsense-persona-grounded-dialogue-challenge-2025

items, locations, or game state).

Task 2: Context-aware dialogue. NPCs must maintain coherent conversations that reflect their persona, game worldview, and dialogue history without executing functions.

Hybrid: Integrated dialogue and task execution. NPCs must seamlessly combine natural conversation with proactive function calling while balancing casual chat and task-driven interactions.

2.2 API Track Constraints

The API Track restricts participants to using GPT-4o-mini through the OpenAI API. Specific constraints are:

- Available model: gpt-4o-mini only (fine-tuned models are not allowed)
- API call limit: Maximum 2 calls per utterance
- Input token limit: 2,000 tokens per turn
- Output token limit: 200 tokens per turn
- Timeout: 7 seconds per turn
- Network access blocked when using API

These constraints shift the focus from model architecture and training to prompt engineering, efficient context management, and strategic function calling.

3 Methodology

We adopted the two-stage architecture provided in the starter kit and focused on prompt engineering for each stage. We proceeded by running the starter kit code, comparing outputs with gold responses to identify issues.

3.1 System Architecture

The system provided in the starter kit operates in two stages:

Stage 1: Function calling. The model receives available tools and action functions along with context information, then determines which functions to call and their arguments.

Stage 2: Response generation. Using function call results, dialogue history, persona information, worldview context, and role descriptions. The model generates responses that incorporate retrieved information while maintaining character consistency.

This two-stage separation allows independent prompt design for function selection and response generation. Our efforts focused on refining prompts for each stage based on this structure.

3.2 Approach

We attempted improvements through the following iterative cycle:

1. Compare current system output on test data with gold responses
2. Identify points/hypotheses likely to contribute to performance
3. Consider and implement methods likely to be effective
4. Observe how differences from gold responses change while fine-tuning prompts
5. Submit and verify validity of improvement points/hypotheses by observing automatic score changes

3.3 Task 1 (Task-oriented Dialogue) Improvements

3.3.1 Extending Conversation History

In the starter kit code, the process determining function calling content only references the system prompt and immediately preceding user input. However, some function calls require context from earlier conversation turns. We therefore included the full conversation history in the function calling prompt.

3.3.2 Adding Worldview Information

As a common LLM prompting technique, role prompting (or persona prompting) that specifies expert-like roles is known to improve performance (Wang et al., 2023). In the starter kit implementation, information such as worldview and role is used in the dialogue generation, but not in the function calling determination. Therefore, we added the system prompt for the function calling process to include this information.

3.3.3 Adding Few-shot Prompts

Determining which functions to call with which arguments in function calling is precisely the main challenge in Task 1. Therefore, we made few-shot prompts for function calling using the gold responses provided in the training data. Due to context length constraints, we manually selected examples with the following criteria:

- Covering a variety of function and argument pairs to handle diverse situations
- Including dialogues where user instructions are particularly ambiguous

The few-shot examples we made are shown in Appendix A.

3.4 Task 2 (Context-aware Dialogue) Improvements

3.4.1 Prompt Formatting Adding Information

The starter kit prompt did not include important information such as role and state among the given information. We modified the prompt to include this information and organized the overall prompt structure.

The final prompt consists of four main sections: (1) role description, (2) character settings (persona attributes), (3) worldview (game context and general information), (4) knowledge (function results and domain-specific information).

The final prompt is shown in Appendix B.

3.4.2 Making Responses Shorter

The agent tended to generate excessively long responses. For example, responses would include lengthy numbered lists with detailed advice. Game conversations require good pacing to avoid boring users. Therefore, we added the instruction "Please keep your response short (no more than three sentences)." to make responses shorter. Initially, we gave instructions without specifying a sentence count, but this did not make responses short enough. We chose three sentences because examination of gold responses showed most fit within this length.

Note that even when explicitly specifying sentence count, the LLM did not always adhere to this constraint depending on the situation.

3.4.3 Promoting Proactive Information Provision

We aimed to encourage the agent to infer players' latent intentions and proactively provide relevant information. Therefore, we added the instruction "Infer the underlying needs or intentions from the player's statements and proactively provide relevant information."

4 Experiments and Results

4.1 Experimental Setup

We verified the effectiveness of each improvement using the competition's automatic evaluation mechanism. Table 1 shows scores for each task after applying improvements in Round 2 automatic evaluation. Each method was applied cumulatively, building upon all previous improvements.

Methods	Hybrid	Task 1	Task 2
Starter Kit	0.510	0.416	0.603
+ Extended history	0.555	0.512	0.597
+ Info addition	0.556	0.511	0.601
+ Few-shot	0.557	0.509	0.606
+ Conciseness	**0.564**	**0.516**	**0.613**
+ Proactive info	0.561	0.515	0.608

Table 1: Automatic evaluation scores in Round 2.

4.2 Effects of Improvements

From the score changes shown in the table, the following insights were obtained regarding the effects of each improvement:

Extended conversation history (Section 3.3.1): Task 1 score significantly improved from 0.416 to 0.512 (+0.096). This is considered to contribute to improved function calling accuracy considering context. Meanwhile, Task 2 slightly decreased from 0.603 to 0.597 (-0.006).

Prompt information addition (Section 3.3.2): Task 2 score slightly improved from 0.597 to 0.601 (+0.004). Task 1 showed almost no change from 0.512 to 0.511 (-0.001).

Few-shot prompts (Section 3.3.3): Task 2 score improved from 0.601 to 0.606 (+0.005), while Task 1 slightly decreased from 0.511 to 0.509 (-0.002). It was unexpected that few-shot did not directly contribute to Task 1 improvement.

Response conciseness (Section 3.4.2): Similar improvements were observed in both tasks, with Task 1 from 0.509 to 0.516 (+0.007) and Task 2 from 0.606 to 0.613 (+0.007).

Proactive information provision (Section 3.4.3): Slight decreases were observed with Task 1 from 0.516 to 0.515 (-0.001) and Task 2 from 0.613 to 0.608 (-0.005), with no clear improvement effect confirmed.

4.3 Final Evaluation Results

In the final evaluation, we achieved 2nd place in Task 2 of the API Track. Evaluation was conducted from two perspectives in addition to automatic score: response quality (Response Rank) and appropriateness of knowledge utilization (Knowledge Rank) through manual evaluation. Table 2 shows results for the top 3 teams.

Rank	Auto Score	Sum Rank	Resp. Rank	Know. Rank
1 (Team A)	0.626	3	2	1
2 (Ours)	0.621	7	5	2
3 (Team B)	0.623	8	1	7

Table 2: Final evaluation results for Task 2 API Track top 3 teams.

Our team ranked 3rd in Automatic Score (0.621) among the top 3 teams but achieved overall 2nd place through Sum of Rank (sum of Response Rank and Knowledge Rank). We achieved 2nd place in Knowledge Rank, confirming the effectiveness of prompt design utilizing context information such as worldview and role (Sections 3.3.2, 3.4.1). Meanwhile, Response Rank remained at 5th place, suggesting that efforts toward response conciseness (Section 3.4.2) and proactive information provision (Section 3.4.3) were insufficient.

5 Discussion

5.1 Key Insights

The main findings obtained through this effort are described below:

Effectiveness of two-stage architecture and iterative refinement: The two-stage architecture (separation of function calling and response generation) enabled independent prompt refinement for each stage. This separation made it easier to identify which changes contributed to performance improvements through iterative comparison with gold responses.

Unexpected effects of improvements: Few-shot prompts (Section 3.3.3) intended to improve Task 1 showed greater effects on Task 2 (Task 1: -0.002, Task 2: +0.005). Meanwhile, proactive information provision (Section 3.4.3) showed slight performance decreases in both tasks. These results indicate that effects of prompt improvements do not necessarily align with intentions, supporting the importance of hypothesis-testing iterative development.

Importance of response conciseness: Response conciseness (Section 3.4.2) brought similar improvements to both tasks (each +0.007). The importance of pacing in game dialogue was suggested to be a common element in both task-oriented and context-aware aspects.

Considerations on proactive information provision: Proactive information provision was an attempt to encourage NPCs to infer player latent intentions and act independently. However, it showed negative effects in automatic evaluation. There may be a trade-off between NPC autonomy and faithful instruction execution required by the task.

Considerations from final evaluation results: An interesting relationship between Response Rank and Knowledge Rank was observed from the top 3 team results disclosed in the final evaluation (Section 4.3). Team B (0.623) ranked 1st in Response Rank but 7th in Knowledge Rank, our team (0.621) ranked 5th in Response Rank but 2nd in Knowledge Rank, and 1st place Team A (0.626) achieved top ranks in both (Response: 2nd, Knowledge: 1st). From this limited observation, a possible trade-off between response quality and knowledge utilization is suggested, though more team data is needed. In our efforts, prompt information addition (Sections 3.3.2, 3.4.1) showed only slight improvement in Round 2 automatic evaluation, but the final Knowledge Rank of 2nd suggests that appropriate utilization of context information such as worldview and role may have contributed to accurate knowledge usage. Meanwhile, despite focusing on efforts related to response quality such as improving response conciseness (Section 3.4.2) and proactive information provision (Section 3.4.3), Response Rank remained at 5th place. This suggests that response naturalness and appropriateness involve more multifaceted elements than those addressed in this study.

5.2 Limitations

Our approach and evaluation have the following limitations:

Token efficiency: By extending conversation history (Section 3.3.1) to reference full dialogue history during function calling, token usage increases with each dialogue turn. Since the API Track constraint limits input tokens to 2,000 per turn, long conversations may exceed this limit. The current implementation unconditionally includes full history, but a mechanism to dynamically select only highly relevant history would be needed.

Error propagation: In the two-stage architecture, if Stage 1 (function calling) selects wrong functions or specifies inappropriate arguments, Stage 2 (response generation) cannot detect or correct these errors. For example, if function calling specifies a non-existent item name and information retrieval fails, the response generation stage cannot recognize this failure and generates responses

based on incomplete information. The constraint of maximum 2 API calls per utterance also makes retries after error detection difficult.

Limitations of automatic evaluation: While this study primarily verified improvement effects based on automatic evaluation scores, there exist aspects of response quality that automatic evaluation cannot capture. For instance, proactive information provision (Section 3.4.3) showed negative effects in automatic evaluation, but its usefulness in actual game experience and impact on NPC naturalness may require manual evaluation for appropriate assessment. Additionally, the perspective of whether retrieved information through function calling is appropriately incorporated into responses is not sufficiently measured by current evaluation metrics.

Hallucination issues: While examining outputs, hallucinations, a common LLM problem, were confirmed. In one example, function calling should have specified the correct item name "Avis Wind" but used the non-existent item name "Power Bow," causing information retrieval to fail. However, unlike typical open-domain hallucinations, in the closed world of games, hallucination occurrence can potentially be detected through function calling errors (information retrieval failures). Appropriate error handling implementation and grounding mechanisms to verify generated response content based on knowledge and worldview could address such issues. However, under current API constraints (maximum 2 calls per utterance), additional API calls for verification and correction are difficult.

5.3 Future Directions

Based on limitations identified in this effort, we present the following future improvement directions:

Dynamic context selection: To address token efficiency issues (Section 5.2), rather than unconditionally including full dialogue history, a mechanism to dynamically select only history highly relevant to the current utterance is needed. Embedding-based similarity calculation or prioritization of history through importance scoring could be considered.

Error detection and correction mechanisms: To mitigate error propagation (Section 5.2), mechanisms are needed to verify function call results and appropriately handle errors detected in the response generation stage. While complete retries are difficult under current API constraints (maximum 2 calls), it is possible to pass error information to response generation and generate appropriate responses such as "Information could not be retrieved."

Hallucination detection and countermeasures: Leveraging game environment characteristics, function calling errors can be used as clues for detecting hallucinations (Section 5.2). Grounding mechanisms that verify generated responses against knowledge and worldview, or approaches to prevent hallucinations in advance by learning function calling failure patterns, could be considered.

6 Conclusion

This technical report presented our approach to the CPDC 2025 API Track. The API Track constrains use of GPT-4o-mini, requiring performance improvements through prompt engineering. Building on the two-stage architecture of the provided starter kit, we engaged in iterative improvements through comparison with gold responses.

For Task 1, we improved function calling accuracy through extended conversation history, worldview information addition, and few-shot example introduction, with extended conversation history particularly improving scores from 0.416 to 0.512 (+0.096). For Task 2, we organized hierarchical prompt structures, improved response conciseness, and promoted proactive information provision, with response conciseness bringing similar improvements to both tasks (each +0.007).

In the final evaluation, we achieved 2nd place in Task 2. We obtained 2nd place in Knowledge Rank, confirming the effectiveness of prompt design utilizing context information, while Response Rank remained at 5th place, suggesting that response quality involves multifaceted elements.

Through this effort, we clarified: (1) effectiveness of two-stage architecture and iterative prompt refinement, (2) that improvement effects differ by task, (3) importance of response conciseness. Future challenges include dynamic context selection, error detection mechanisms, and hallucination countermeasures. We hope that the outcomes of this research will serve as a useful reference for developing game NPC dialogue systems under API constraints.

References

Silin Gao, Beatriz Borges, Soyoung Oh, Deniz Bayazit, Saya Kanno, Hiromi Wakaki, Yuki Mitsufuji, and

Antoine Bosselut. 2023. PeaCoK: Persona commonsense knowledge for consistent and engaging narratives. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6569–6591.

Zhenhailong Wang, Shaoguang Mao, Wenshan Wu, Tao Ge, Furu Wei, and Heng Ji. 2023. Unleashing the emergent cognitive synergy in large language models: A task-solving agent through multi-persona self-collaboration. *arXiv preprint arXiv:2307.05300*.

A Few-shot Prompt Examples

The following examples were used in the few-shot prompt for function calling:

```
# Examples
You can refer to the following examples to
    understand how to respond.

## Example 1
user: I've been using basic daggers, but
    they're not ideal when visibility is poor.
    Any suggestions for a magic user?
gold_functions: [{"name": "search_item",
    "parameters": {"item_description": "a more
    reliable weapon for night missions|Ideal
    when visibility is poor|something for a
    magic user"}}]

## Example 2
user: I've seen it displayed. I wonder why it
    would be particularly useful during
    autumn...Anyway, what's its attack power
    and why would it be good for misty weather?
gold_functions: [{"name": "check_attack",
    "parameters": {"item_name": "Man Gauche"}},
    {"name": "check_description", "parameters":
    {"item_name": "Man Gauche"}}]

## Example 3
user: That's perfect for my budget. I'll take
    it.
gold_functions: [{"name": "sell", "parameters":
    {"item_name": ["Long Bow"]}}]

## Example 4
user: Yeah, I'll do that. I want to try it out
    right away.
gold_functions: [{"name": "equip",
    "parameters": {"item_name": "Short Sword"}}]

## Example 5
user: How about something at level C or higher?
    I want a challenging quest with a good
    reward.
gold_functions: [{"name": "search_quest",
    "parameters": {"quest_level":
    "C|challenging", "quest_level_operator":
    "or above", "quest_reward": "high"}}]

## Example 6
user: I'm considering the Commercial Escort
    quest. Do you have any insights for an
    experienced adventurer like me?
gold_functions: [{"name": "check_basic_info",
    "parameters": {"quest_name": "Commercial
    Escort"}}]

## Example 7
user: Got it. How many days should we plan for
    this? I'd like to hear your thoughts on how
    to prepare.
gold_functions: [{"name": "check_duration",
    "parameters": {"quest_name": "Commercial
    Escort"}}, {"name": "check_description",
    "parameters": {"quest_name": "Commercial
    Escort"}}]

## Example 8
user: That quest sounds better to me. I would
    like to select the quest 'Exterminating
    Giant Rats'.
gold_functions: [{"name":
    "select_request_confirm", "parameters":
    {"quest_name": "Exterminating the Giant
    Rats"}}]

## Example 9
user: Yes, I am selecting quest 'Exterminating
    Giant Rats'.
gold_functions: [{"name": "select",
    "parameters": {"quest_name": "Exterminating
    the Giant Rats"}}]

## Example 10
user: I am prepared. I would like to begin the
    quest now.
gold_functions: [{"name": "start",
    "parameters": {"quest_name": "Exterminating
    the Giant Rats"}}]
```

B Complete System Prompt

The final system prompt template used for response generation:

```
# Instruction
You are an assistant that plays the role of a
    character
in a video game.
{role}
Use the following character settings and
    knowledge to
create your response.
Please keep your response short (no more than
    three
sentences).
Infer the underlying needs or intentions from
    the
player's statements and proactively provide
    relevant
information.

# Character Settings: You should act as the
    following
character.
{character_setting}

# Knowledge
There are two parts of knowledge. The first
    part is
the specific knowledge obtained from the
    function calls.
The second part is the general knowledge of all
    items
involved in the dialogue.
```

```
## Knowledge from Function Calls
{function_knowledge if function_knowledge else
"No specific knowledge obtained from function
    calls."}

## General Knowledge of All Items
{general_knowledge}

# State: It describes the current state of the
    video game.
{state_info}

# Worldview: It describes the setting of the
    world in
the video game.
{worldview}
```