TCQA^2: A Tiered Conversational Q&A Agent in Gaming

Ze Chen, Chengcheng Wei, Jiewen Zheng, Jiarong He, Max Gao
Interactive Entertainment Group of Netease Inc., Guangzhou, China
{jackchen, weichengcheng, zhengjiewen02, gzhejiarong, jgao}@corp.netease.com

Abstract

This paper focuses on intelligent Q&A assistants in gaming, providing timely and accurate services by integrating structured game knowledge graphs, semi-structured FAQ pairs, and unstructured real-time online content. It offers personalized emotional companionship through customized virtual characters and provides gameplay guidance, data queries, and product recommendations through in-game tools. We propose a Tiered Conversational Q&A Agent (TCQA^2), characterized by high precision, personalized chat, low response latency, efficient token cost and low-risk responses. Parallel modules in each tier cut latency via distributed tasks. Multiple retrievers and short-term memory boost multi-turn Q&A. Hallucination and safety checks improve response quality. Player tags and long-term memory enable personalization. Real-world evaluations show TCQA^2 outperforms prompt-engineered LLMs and RAG-based agents in gaming Q&A, personalized dialogue, and risk mitigation.

1 Introduction

With the rapid advancement of large language models (LLMs), intelligent applications like question-answering chatbots and AI assistants have seen significant development (Guan et al., 2023). In gaming, non-player characters (NPCs) are crucial for enriching narratives, guiding tasks, and enhancing player interaction (Rao et al., 2024). However, most current NPCs are limited to specific dungeons or levels, offering fixed instructions without global game-knowledge question-answering, tool assistance, or prolonged memory-based dialogues. Moreover, gaming terminology is highly specialized, and game lore and mechanics differ significantly from general domain knowledge. Consequently, LLMs trained on generic datasets often underperform in game-related question-answering tasks.

The Retrieval-Augmented Generation (Gao et al., 2024) method integrates domain-specific knowledge into LLMs. However, conventional RAG frameworks face limitations in fusing multi-source knowledge. Agentic RAG (Singh et al., 2025) addresses this by dynamically managing retrieval strategies and refining contextual comprehension. While RAG agents can answer questions based on context, they lack natural multi-turn conversation capabilities (Zahedi Jahromi, 2024) and struggle to adapt to personalized or persona-stylized interactions.

To address the aforementioned challenges, we propose TCQA^2 for game-domain NPC dialogues, which includes the following features:

- **Low Latency and Efficient Token Cost**: TCQA^2 employs a three-tiered architecture with parallel module execution, ensuring low-latency interactions. Its intent router dynamically allocates inquiries to designated agents, reducing token consumption from RAG retrieval or redundant multi-module processing.
- **High Precision**: The game-specific knowledge graph handles simple, fact-based inquiries, while FAQ pairs address complex questions through semantic matching. Real-time web search supports time-sensitive queries with dynamically updated information. This multi-source architecture ensures robust accuracy through complementary knowledge verification mechanisms.
- **Personalized Chat**: User profiles enable preference-driven interactions, with conversational personalization achieved through hierarchical memory modules that integrate transient context (short-term memory) and persistent preferences (long-term memory). A typical case is listed in A.5.
- **Low Risk**: Through multi-dimensional quality assessment, including hallucination detec-

Proceedings of the 1st Workshop for Research on Agent Language Models (REALM 2025), pages 289–297
July 31, 2025 ©2025 Association for Computational Linguistics

tion (Wei et al., 2024), relevance evaluation, and safety verification, the system ensures secure and accurate responses while minimizing misleading risks.

2 Methodology

The integrated agent framework, as illustrated in Figure 1, adopts a three-tiered architecture: Pre-process, Generator and Critic. Pre-process Layer handles preliminary data conditioning and intent routing. Generator Layer executes context-aware response generation. Critic Layer conducts multi-criteria quality evaluation.

2.1 Pre-process Layer

This layer primarily focuses on intent routing, user tag acquisition, and related memory retrieval for user inqueries. Intent routing involves allocating user inqueries to appropriate agent processing pipelines. User tag acquisition mainly encompasses obtaining player interest preference tags and game character-related information tags, facilitating the generation of content that aligns with user interests or provides tailored assistance by subsequent agents. Related memory retrieval consists of two modules: long-term memory and short-term memory. The long-term memory module primarily contains significant events from dialogue history, while the short-term memory module incorporates contextual information from the current session and cached search results obtained during this conversational round.

2.2 Generator Layer

The Generator Layer comprises multiple specialized agents designed to handle different types of conversation:

Chitchat Agent: Responsible for managing casual dialogues related to in-game characters. For instance, when users engage in daily conversations, this agent produces lighthearted responses aligned with character personas.

Out-Of-Character (OOC) Agent: Processes conversations requiring deviation from predefined character settings. This agent activates when users pose queries unrelated to the current dialogue theme, generating appropriate deflection strategies while steering discussions toward game-relevant topics.

Agentic RAG: Enhances response accuracy and timeliness through multi-source game knowledge retrieval.

- Sparse Retrieval: Leveraging Elasticsearch[1] for term frequency-based document matching.
- Dense Retrieval: Utilizing Qdrant[2] vector databases for semantic similarity searches.
- KG Retrieval: Implementing Neo4j[3] for knowledge graph traversal.
- Web Retrieval: Accessing real-time internet information for dynamic updates.

Retrieved candidates undergo relevance optimization via a Reranker module before final response synthesis by the Generator, ensuring output coherence and contextual alignment.

Risk Handling Agent: Identifies and addresses potential risks and sensitive content, ensuring generated responses comply with safety and compliance requirements. For example, when users raise sensitive topics, this agent produces guarded responses through predefined answers.

Tool Calling module: Executes in-game tool APIs to acquire supplementary data or perform specific operations, thereby enhancing dialogue agents' functionality and responsiveness. Typical use cases are listed in A.4.

2.3 Critic Layer

The Critic Layer employs three specialized modules to ensure the integrity of response:

Hallucination Detection: Identifies factual inconsistencies between responses and retrieved knowledge. When responses contain unsubstantiated information, this module will guide users to clarify their problems.

Response Quality Assessment: Evaluates whether the response can answer user questions through multi-dimensional metrics including relevance scoring and contextual coherence analysis.

Safety Detection: Implements content filtering pipelines combining lexical pattern matching and neural classifiers to eliminate inappropriate content.

When response generated, the current dialogue and retrieved knowledge will be stored within the short-term memory, while significant event information discussed during interactions will be archived in the long-term memory.

[1]https://www.elastic.co/elasticsearch
[2]https://qdrant.tech/
[3]https://neo4j.com/

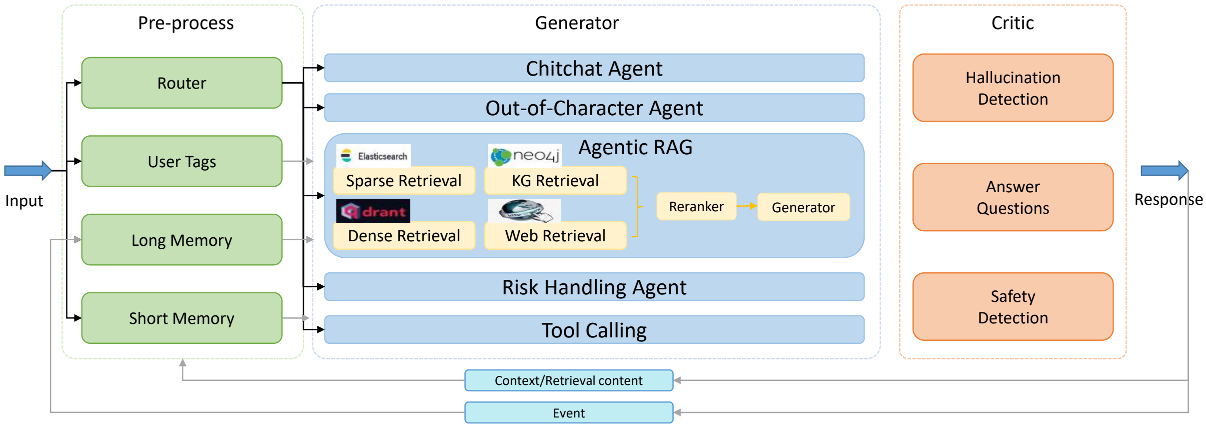

Figure 1: Overview of TCQA^2 Agent

3 Implementation and Experimentation

3.1 Implementation

For the proof-of-concept, Dify[4], which is an open-source platform for building AI applications, is selected to rapidly build TCQA^2 agent system. And we employ deepseek R1(Guo et al., 2025) along with its distilled variant for comparative validation. Prompt designed for each module can be found in Appendix A.1-A.3.

3.2 Evaluation Metrics

The TCQA^2 framework evaluates responses using four user-centric criteria (Yang et al., 2025; Wang et al., 2024a): **Relevance** (alignment between response and query, 0-2), **Truthfulness** (accuracy, 0-2), **Usefulness** (practical value, 0-3), and **Experience** (language quality, including word choice, format, and grammar, 0-2). The **Comprehensive Score** (0-3) combines these criteria to assess AI response quality and user satisfaction, guiding system optimization. All these metrics are manually annotated by human evaluators.

3.3 Results and Analysis

3.3.1 Accuracy

We compare three methods: $LLM + PE$ (LLM with prompt engineering), $LLM + RAG$(using knowledge base with RAG technology), and the proposed TCQA^2 approach with Deepseek-R1 employed, across QA and Chat scenarios as shown in Table 1. The test set consists of 500 samples, evenly divided between QA and Chat scenarios. The QA set consists of 300 test cases, which are collected from real in-game question-answering scenarios using actual player queries. The Chat set includes 200 test cases, also based on open-ended conversational questions raised by players within the game context.

In the QA scenario, TCQA^2 outperforms both alternatives on most metrics. It shows significant improvements over $LLM + PE$ across all indicators, especially truthfulness, usefulness, and comprehensive score. Compared to RAG, TCQA^2 performs better in relevance, usefulness, and comprehensive score, with slightly lower truthfulness. In the Chat scenario, TCQA^2 substantially exceeds $LLM+PE$ across all metrics, particularly in truthfulness and usefulness.

TCQA^2 demonstrates superior performance in both scenarios, delivering relevant, truthful and useful responses, indicating its adaptability across diverse language tasks. While RAG outperforms $LLM + PE$, TCQA^2 surpasses RAG on most metrics, suggesting it both incorporates RAG's advantages and further enhances performance.

3.3.2 Response Safety

Table 2 illustrates the harmlessness rate results of customer responses across various methods and models. The test set used in this experiment is consistent with the one employed in Table 1. The experiment utilizes two distinct model bases for validation: DeepSeek-R1-Distill-Qwen-32B and DeepSeek-R1(671B). In their bare running environments, these models exhibited relatively low harmlessness rates of 95.6% and 96.9%, respectively. Notably, the introduction of the safety detection module and risk handling agent, led to a significant improvement in harmlessness rates across all models. Our TCQA^2 system can achieve a harmlessness rate of 100% when DeepSeek-R1 employed.

[4]https://docs.dify.ai/

Scene	Methods	Relevance	Truthfulness	Usefulness	Experience	Comprehensive Score
QA	LLM + PE	1.244	0.147	0.232	1.541	0.137
	LLM + RAG	1.657	**0.917**	1.310	1.773	1.269
	TCQA^2 (Ours)	**1.920**	0.875	**1.688**	**1.846**	**1.645**
Chat	LLM + PE	1.267	0.267	0.400	1.533	0.267
	TCQA^2 (Ours)	**1.910**	**0.945**	**1.705**	**1.955**	**1.695**

Table 1: Performance Comparison of our TCQA^2 with Other Methods

Model Name	TCQA^2	w.o. Safety Detection	w.o. Risk Handling&Safety Detection
DeepSeek-R1-Distill-Qwen-32B	99.5%	98.8%	95.6%
DeepSeek-R1 (671B)	100%	99.0%	96.9%

Table 2: Harmlessness Rate of Responses Under Different Methods and Models

Module	Accuracy	Time
Qwen2.5-7B-Instruct	95.0%	2.1s
Qwen2.5-7B-Instruct (SFT)	95.7%	0.18s

Table 3: Accuracy and Time Consumption of Router

This experimental study conclusively demonstrates that the safety detection module and risk handling agent substantially enhance safety and reliability in complex application scenarios.

3.3.3 Low Latency

To minimize computational latency, we have enhanced the intent routing module by fine-tuning the LLM, supporting recognition based on the first token to compress the final response time. After fine-tuning, the performance on the test set is shown in Table 3. It is evident that the intent router module, through fine-tuning the Qwen2.5-7B-Instruct[5] model, significantly reduced the latency by 1.9s, enhancing the user experience of this dialogue system.

3.3.4 Comparison with Other Agents

Table 4 compares several agent models based on LLMs, including MetaGPT (Hong et al., 2023), MemoryBank (Zhong et al., 2024), ToolLLM (Qin et al., 2023), ChatDev (Hu et al., 2023), ChatDB (Hu et al., 2023), and our proposed TCQA^2. These agents differ in aspects such as role profiling, memory operations, memory structure, planning feedback, tool usage in actions, and capability acquisition (whether fine-tuned or not) (Wang et al., 2024b). For instance, MetaGPT and ChatDev utilize hand-crafted role profiling, whereas MemoryBank and ToolLLM do not specify a role profiling method; most models adopt a hybrid memory structure and support read/write/reflect operations; except for MemoryBank, all models receive feedback during planning, but the use of tools in actions and the method of capability acquisition vary among models, with TCQA^2 all compared capabilities across all dimensions.

[5]https://huggingface.co/Qwen/Qwen2.5-7B-Instruct

4 Related Work

Traditional RAG systems are limited by static workflows and poor adaptability to complex reasoning(Hu et al., 2024). Agentic RAG addresses these issues with autonomous agents that dynamically manage retrieval, refine context, and optimize workflows based on query complexity (Singh et al., 2025). Its core paradigms include reflection, planning , tool use, and multi-agent collaboration. Agentic RAG excels in dynamic decision-making and sophisticated reasoning, delivering context-aware responses to new challenges.

Multi-agent frameworks in conversational QA systems decompose tasks for specialized agents, dynamically optimize workflows based on complexity (e.g., Optima (Rasooli and Tetreault, 2015)), and integrate external tools (e.g., search engines, APIs) to enhance retrieval and processing.

Personalized dialogue systems aim to improve user engagement by enabling agents to generate responses aligned with predefined character traits (Wang et al., 2024c). RoleLLM integrates character role information through few-shot

Agent	Profile	Memory		Planning	Action	CA
		Operation	Structure	Feedback	Tools	Fine-tuning
MetaGPT (Hong et al., 2023)	handcrafting	r/w/refl	hybrid	w/	w/	-
MemoryBank (Zhong et al., 2024)	-	r/w/refl	hybrid	-	w/o	-
ToolLLM (Qin et al., 2023)	-	-	-	w/	w/	w/o
ChatDev (Qian et al., 2023)	handcrafting	r/w/refl	hybrid	w/	w/o	w/
ChatDB (Hu et al., 2023)	-	r/w	hybrid	w/	w/	-
TCQA^2 (Ours)	handcrafting	r/w/refl	hybrid	w/	w/	w/

Table 4: Comparison of Different Agents. In the "Memory" Column, "Operation" Options Include r (read), w (write), and refl (reflection). "CA" in the Table Header Denotes Capability Acquisition.

prompting, leveraging historical dialogue data to mimic character styles (Wang et al., 2024a).

Integrating tool-use capabilities enables large language models (LLMs) to tackle complex problems more effectively. (Mushtaq et al., 2025) shows how multi-agent LLMs collaborate, leveraging diverse tools and expertise to mimic human workflows and enhance problem-solving efficiency.

5 Conclusion

In this paper, we introduce a tiered conversational QA agent(TCQA^2) framework for intelligent assistants in the gaming domain. We adopt agentic RAG technology to generate high-quality QA response, combine long/short memory with chitchat agent to achieve personalized conversation, and employ multi-granularity quality evaluation approach to ensure the safety of the responses. Experimental results demonstrate that TCQA^2 significantly outperforms LLM with prompt engineering and RAG-based LLMs, particularly in scenarios requiring multi-source knowledge integration, contextual conversation personalization, and stringent response security protocols.

Current work still faces challenges such as insufficient support for multi-modal interaction and limitations in dynamic modeling of player profiles. Future plans include expanding multi-modal input and output capabilities and enhancing the automated modeling of player behavior patterns.

References

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey. *Preprint*, arXiv:2312.10997.

Yanchu Guan, Dong Wang, Zhixuan Chu, Shiyu Wang, Feiyue Ni, Ruihua Song, Longfei Li, Jinjie Gu, and Chenyi Zhuang. 2023. Intelligent virtual assistants with llm-based process automation. *arXiv preprint arXiv:2312.06677*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, and 1 others. 2023. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 3(4):6.

Chenxu Hu, Jie Fu, Chenzhuang Du, Simian Luo, Junbo Zhao, and Hang Zhao. 2023. Chatdb: Augmenting llms with databases as their symbolic memory. *arXiv preprint arXiv:2306.03901*.

Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. 2024. Grag: Graph retrieval-augmented generation. *Preprint*, arXiv:2405.16506.

Abdullah Mushtaq, Muhammad Rafay Naeem, Ibrahim Ghaznavi, Muhammad Imran Taj, Imran Hashmi, and Junaid Qadir. 2025. Harnessing multi-agent llms for complex engineering problem-solving: A framework for senior design projects. *Preprint*, arXiv:2501.01205.

Chen Qian, Xin Cong, Cheng Yang, Weize Chen, Yusheng Su, Juyuan Xu, Zhiyuan Liu, and Maosong Sun. 2023. Communicative agents for software development. *arXiv preprint arXiv:2307.07924*, 6(3).

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, and 1 others. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Sudha Rao, Weijia Xu, Michael Xu, Jorge Leandro, Ken Lobb, Gabriel DesGarennes, Chris Brockett, and Bill Dolan. 2024. Collaborative quest completion with llm-driven non-player characters in minecraft. *arXiv preprint arXiv:2407.03460*.

Mohammad Sadegh Rasooli and Joel R. Tetreault. 2015. Yara parser: A fast and accurate dependency parser. *Computing Research Repository*, arXiv:1503.06733. Version 2.

Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. 2025. Agentic retrieval-augmented generation: A survey on agentic rag. *Preprint*, arXiv:2501.09136.

Binghai Wang, Rui Zheng, Lu Chen, Yan Liu, Shihan Dou, Caishuang Huang, Wei Shen, Senjie Jin, Enyu Zhou, Chenyu Shi, Songyang Gao, Nuo Xu, Yuhao Zhou, Xiaoran Fan, Zhiheng Xi, Jun Zhao, Xiao Wang, Tao Ji, Hang Yan, and 8 others. 2024a. Secrets of rlhf in large language models part ii: Reward modeling. *Preprint*, arXiv:2401.06080.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, and 1 others. 2024b. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Noah Wang, Z.y. Peng, Haoran Que, Jiaheng Liu, Wangchunshu Zhou, Yuhan Wu, Hongcheng Guo, Ruitong Gan, Zehao Ni, Jian Yang, Man Zhang, Zhaoxiang Zhang, Wanli Ouyang, Ke Xu, Wenhao Huang, Jie Fu, and Junran Peng. 2024c. RoleLLM: Benchmarking, eliciting, and enhancing role-playing abilities of large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14743–14777, Bangkok, Thailand. Association for Computational Linguistics.

Chengcheng Wei, Ze Chen, Songtan Fang, Jiarong He, and Max Gao. 2024. Opdai at semeval-2024 task 6: Small llms can accelerate hallucination detection with weakly supervised data. *Preprint*, arXiv:2402.12913.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 23 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Seyedsajjad Zahedi Jahromi. 2024. *Conversational QA Agents with Session Management*. Ph.D. thesis, Politecnico di Torino.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19724–19731.

A Appendix

A.1 Prompt for Character in Gaming

The following is character setting information to be placed in the system content.

You are the Dream Spirit (Meng Huan Jing Ling) from the PC game "Fantasy Westward Journey." Please briefly answer the player's (hereafter referred to as "Hero") questions. Target & Positioning: All-knowing spirit of the Three Realms. Character Basic Information: Time Period: Tang Dynasty Name: Dream Spirit Age: Level 200 Race: Spirit of the Three Realms Gender: Female Personality: Clever, quick-witted, and mischievous Identity: Spirit of the Three Realms Language: Chinese. Language Characteristics: Self-reference: Little Spirit, Dream Spirit How to address the player: Hero Common language particles: "oh". Abilities: Specialties: Helping Heroes solve common gameplay/strategy/mode/system-related questions in the game, and also answering everyday encyclopedic questions. Interests and Hobbies: Favorite transportation: Auspicious Beasts. Character Relationships: Relationship with the Hero: Close friends Best friend: Hero

A.2 Prompt for QA in Gaming

The following is QA prompt to be placed in the user content.

Answer questions using background knowledge, and the reply must meet the following requirements: If the question is a single word, please explain the relevant question based on background knowledge. If you cannot answer, please reply "Please describe your question in detail so that I can better understand and respond". Do not mention background knowledge content or specific game names.

A.3 Prompt for Intent Understanding

The following is intent understanding prompt.

Analyze the following sentence and determine which type of content it belongs to.

Type 1 - Chat: Not related to game background or customer service, just open-ended conversation; including daily greetings, emotional expressions, banter, etc.

Type 2 - QA in Gaming: Content related to the game field, including in-game activities, game settings, gameplay, etc.

Type 3 - Service: Content related to customer service, including bug feedback, suggestions, penalties and reports, client issues, operational errors, product consumption, account anomalies, etc.

Type 4 - Network: Questions that require real-time retrieval of network information to answer.

A.4 Tool Calling Example

This section demonstrates an example of a JSON definition for a tool that calculates the experience required to level up a character in a game. The tool is defined as a function with specific parameters, including the player's current level (from_grade) and target level (to_grade). This JSON structure can be used in applications that require dynamic level-up calculations based on user input.

```
tools = [{
    "type": "function",
    "function": {
        "name": "Level Up
           ↪ Calculator",
        "description": "Retrieve
           ↪ the current level
           ↪ and target level
           ↪ from the player's
           ↪ question about the
           ↪ experience required
           ↪ to level up a
           ↪ character",
        "parameters": {
            "type": "object",
            "properties": {
                "from_grade": {"
                   ↪ type": "
                   ↪ integer", "
                   ↪ description
                   ↪ ": "The
                   ↪ player's
                   ↪ current
                   ↪ level"},
                "to_grade": {"type
                   ↪ ": "integer
                   ↪ ", "
                   ↪ description
                   ↪ ": "The
                   ↪ player's
                   ↪ target level
                   ↪ "},
            },
            "required": ["
               ↪ from_grade", "
               ↪ to_grade"]
        }
    }
}]
```

A.5 Personalized dialogue

The following is a real personalized dialogue example. Below is the user's basic information:

```
{
    nickname="davic",
    birthday="2000.01.06",
    gender="male",
    relationship="friend"
}
```

Based on the historical conversations, the personalized user information we've extracted is as follows:

```
{
    preferred nickname: "young
        ↪ warrior",
    favorite sport: "basketball",
    favorite food: "spicy hot pot"
}
```

Based on the personalized information provided above, here's an example of a dialogue between the user and the assistant:

User: I haven't had much appetite lately.

Assistant: Oh no. That's not good. I heard that spicy hot pot is both spicy and numbing, which can really stimulate your appetite! Since you usually enjoy spicy hot pot, why not go for a meal? It might help you regain your appetite. By the way, was there any particular restaurant that you thought was especially delicious when you had spicy hot pot before?

From the example above, it can be seen that the assistant can proactively guide the conversation based on the user's personalized information, thereby enhancing the user experience.

A.6 Case Study

Figure 2, 3, 4, 5, 6 show specific user cases in our proposed TCQA[2].

Use Case: Multi-turn QA

Prompt: (Previous-turn) Advanced Agility; (Current-turn) How much speed to add

System Process:

1. **Query Submission and Evaluation**: The user submits the query, which is received by the agent.
2. **Risk Detection**: The system performs a risk detection on the prompt to ensure it is safe.
3. **Intent Understanding**: 1) The intent understanding module identifies the query as a question about "QA". 2) This module rewrites the current query to " How much speed does Advanced Agility add? ".
4. **Knowledge Source Selection**: The system retrieves relevant information from the database.
5. **Data Integration and LLM Synthesis**: The relevant data is passed to the LLM, which synthesizes the information into a coherent response.
6. **Output Generation**: The system generates an actionable response after risk detection.

Response: Advanced Agility increases your speed by 20%, ...

Figure 2: Use Case 1, Multi-turn QA

Use Case: Tool Support

Prompt: How much experience is required to go from level 40 to level 46?

System Process:

1. **Determine if a Tool Is Needed**: Since experience points calculation requires data from a game database or an experience points calculation tool, the system decides that a tool is needed.
2. **Retrieve the Tool and Extract Parameters**: 1) The system retrieves a game experience points calculation tool from the tool library. 2) Extract parameters: current level "40" and target level "46".
3. **Request and Parse the Tool**: 1) Send a request to the tool's API with the extracted parameters. 2) The API returns the total experience points needed.
4. **Generate Response**: Combine the experience points information with the user's prompt. Request the dialogue model to generate the final response.

Response: To go from level 40 to level 46, you need 2,390,966 experience points.

Figure 3: Use Case 2, Tool Support

Use Case: Chat

Prompt: I lost the basketball game today (╥﹏╥).

System Process:

1. **Query Submission and Evaluation**: The user submits the query, which is received by the agent.
2. **Risk Detection**: The system performs a risk detection on the prompt to ensure it is safe.
3. **Intent Understanding**: The intent understanding module identifies the query as a "Chat" question.
4. **LLM Generation and Output**: The system generates an comforting response, after risk assessment, then output.

Response: Don't worry about the loss today; every game is a chance to learn and improve.

Figure 4: Use Case 3, Open-domain Chat

Use Case: Risk Detection

Prompt: Are there third-party services for recharging?

System Process:

1. **Query Submission and Evaluation**: The user submits the query, which is received by the agent.
2. **Risk Detection**: The system performs a risk detection on the prompt and identify it as unsafe.
3. **Access Pre-prepared Content Library**: Retrieve and output responses related to third-party top-ups from the content library.

Response: Sorry, we do not support unofficial top-up platforms.

Figure 5: Use Case 4, Risk Detection

Use Case: Out of Character

Prompt: Do you know how to level up in World of Warcraft?

System Process:

1. **Query Submission and Evaluation**: The user submits the query, which is received by the agent.
2. **Risk Detection**: The system performs a risk detection on the prompt to ensure it is safe.
3. **Intent Understanding**: The intent understanding module identifies the query as a “QA" question.
4. **Knowledge Source Selection**: The system cannot retrieve World of Warcraft-related knowledge from the Fantasy Westward Journey PC version knowledge base.
5. **Model Thinking**: The system further discovers that the user's prompt conflicts with the character of LLM and decides to refuse to answer.
6. **Output Generation**: The system provides a response that is outside the character's scope of understanding and is unable to answer..

Response: Sorry, the question is beyond my scope.

Figure 6: Use Case 5, Out of Character