RoleRAG: Enhancing LLM Role-Playing via Graph Guided Retrieval

Yongjie Wang[1] and **Jonathan Leung**[1] and **Zhiqi Shen**[2]
[1]Alibaba-NTU Global e-Sustainability CorpLab (ANGEL), Nanyang Technological University
[2]College of Computing and Data Science, Nanyang Technological University
{yongjie.wang, jonathan.leung, zqshen}@ntu.edu.sg

Abstract

Large Language Models (LLMs) have shown promise in character imitation, enabling immersive and engaging conversations. However, LLMs often generate content that is irrelevant or inconsistent with a character's background. We attribute these failures to: 1) the inability to accurately recall character-specific knowledge due to entity ambiguity; and 2) a lack of awareness of the character's cognitive boundaries. This paper introduces RoleRAG, a retrieval-based framework that combines efficient entity disambiguation for knowledge indexing with a boundary-aware retriever to extract contextually appropriate content from a structured knowledge graph. We conducted extensive experiments on role-playing benchmarks and demonstrate that RoleRAG's calibrated retrieval enables both general LLMs and role-specific LLMs to exhibit knowledge that is more aligned with the given character and reduce hallucinated responses.

1 Introduction

The advent of Large Language Models (LLMs) has significantly enhanced the capabilities of conversational AI agents due to their proficiency in understanding and generation. To further promote user engagement and entrainment (Park et al., 2023), role-playing LLMs are designed to mimic the traits and experiences of specific characters, producing interactions that are role-consistent, emotionally deep, and contextually aware.

To improve imitation capabilities, recent studies (Shao et al., 2023; Tu et al., 2024; Tao et al., 2024; Lu et al., 2024; Zhou et al., 2024) have fine-tuned LLMs on datasets specifically curated for role-playing scenarios. However, due to the labor-intensive nature of data collection and the high computational costs associated with fine-tuning, an alternative line of research explores the use of in-context learning by, for example, providing few-shot examples (Li et al., 2023) or using static user profiles (Wang et al., 2024a), to provide role-related background information.

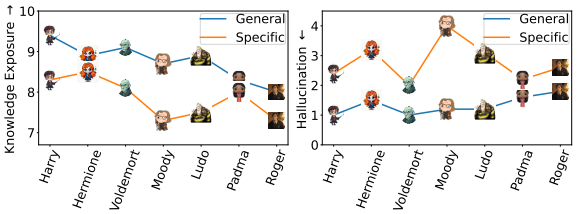

Figure 1: This figure illustrates that LLMs perform worse on role-specific questions, particularly when imitating lower-frequency characters.

However, LLMs still struggle to align accurately with a character's knowledge, often generating responses that lack appropriate traits or include fabricated content—particularly in certain role-playing scenarios where factual consistency is critical. As shown in Figure 1, we tasked GPT-4o-mini with playing seven characters from the Harry Potter series, selected based on their frequency of appearance. Each character was presented with 10 general questions (e.g., interests, attitudes) and 10 role-specific questions (e.g., experiences, activities). We then recruited human raters to assess whether the language models accurately reflected each character's traits and to rate the severity of hallucinations, using a ten-point scale. From these results, we observe that LLMs perform worse on role-specific questions that require detailed character knowledge, particularly for less frequent characters.

Our failure case analysis reveals two key factors contributing to these issues. (1) *Entity ambiguity during knowledge extraction:* a single character may be referred to by different names across different stages or contexts. For example, 'Anakin Skywalker' is also known as 'Darth Vader' or 'Lord Vader' in various installments of the Star Wars series. If not properly unified, such name variation can cause critical information to be missed during knowledge retrieval ; (2) *Character-related cognition-boundary unawareness:* LLMs encode

vast amounts of knowledge beyond the scope of the character they are portraying and often rely on LLM internal knowledge when responding to user queries. This can result in fabricated responses, particularly when the question falls outside the character's original knowledge boundaries. Such hallucinations are unique to the role-playing setting.

To address these issues, we introduce RoleRAG, a retrieval-based framework specifically designed for role-playing tasks. Our approach is built on knowledge graph–enhanced retrieval, motivated by the observation that answering a single question may require reasoning over a broad range of dispersed textual knowledge. In the context of role-playing, the knowledge graph is constructed from character-centric corpora such as Wikipedia profiles and books. Each node in the graph represents an entity (e.g., character, location), and each edge encodes a semantic relationship between two entities (e.g., interactions between characters). To normalize duplicated names referring to the same entity, we propose an efficient semantic entity normalization algorithm. It first links name variants based on their local context, then clusters them into groups representing the same entity. Finally, an LLM is prompted to generate a unified canonical name for each group. The knowledge graph is then constructed using these normalized entities.

Our retrieval module, built on this graph-based indexing system, is designed to extract both specific and general entities mentioned in user queries while rejecting those that fall outside the scope of the character's knowledge. Subsequently, information relevant to the designated role is retrieved from the knowledge graph and provided to the LLM, equipping it with detailed contextual information to generate accurate responses. In contrast, out-of-scope questions are encouraged to be rejected to prevent the model from generating hallucinated or fabricated content.

Our contributions can be summarized as follows:

- We propose an efficient entity normalization algorithm that merges duplicated names referring to the same entity, thereby facilitating high-quality graph-based indexing over large character corpora.
- We introduce an effective retrieval module that not only retrieves both general concepts and character-specific details, but also helps reject out-of-scope questions to reduce hallucinations.
- Extensive experiments that demonstrate that RoleRAG outperforms relevant baselines by exhibiting aligned character knowledge and reducing hallucinations.

2 Related Works

LLM-based Role-Play enables LLMs to embody user-specified characters, enhancing engagement through conversation. Existing research falls into three main directions: (1) Fine-tuning-based approaches (Shao et al., 2023; Tu et al., 2024; Wang et al., 2024a; Zhou et al., 2024; Lu et al., 2024) involve training open-source LLMs on curated character corpora. The training data is either synthetic—generated specifically for character conditioning (Shao et al., 2023; Tu et al., 2024; Wang et al., 2024a)—or extracted from real-world datasets using LLMs (Zhou et al., 2024). (2) Retrieval-based approaches (Salemi et al., 2024; Weir et al., 2024; Zhou et al., 2024) fetch relevant documents from a character corpus to serve as contextual input to the LLM, thereby enhancing its ability to generate accurate and character-specific responses. The performance of these methods heavily depends on the quality and relevance of the retrieved content. (3) Plugin-based methods (Liu et al., 2024) freeze the LLM while encoding each user's characteristics using a lightweight plugin model. The resulting user embedding is then concatenated with the embedding of the user's query to guide the LLM in generating personalized responses. A comprehensive comparison of the three categories is provided in the Appendix A. In this work, we follow retrieval-based approaches, aiming to provide character-relevant content while reducing hallucinations in responses.

Persona-based Dialogue. Persona-based dialogue tasks require LLMs to exhibit general human-like traits such as humor, empathy, or curiosity, rather than adhering to specific role characteristics. Unlike role-playing, the focus is on consistent personality expression. Personas can be assigned via Big Five trait prompts (Jiang et al., 2023), character profiles (Tu et al., 2024; Zhou et al., 2024), or dialogue history (Zhong et al., 2022). Evaluation is typically conducted through personality assessments or interviews (Wang et al., 2024b). A comprehensive comparison between role-playing and persona-based dialogue refer to (Tseng et al., 2024). Our work focuses on enabling role-playing LLMs to produce character-faithful responses.

Retrieval-Augmented Generation (RAG) enhances LLMs by retrieving external knowledge

to support informed, accurate, and contextually grounded responses (Lewis et al., 2020; Liu et al., 2022; Zhuang et al., 2023; Li et al., 2024). Standard RAG struggles with capturing complex inter-entity relationships across multiple chunks (Guo et al., 2024) and often fails on general queries requiring comprehensive understanding of large knowledge bases (Edge et al., 2024). To address these limitations, recent work (Edge et al., 2024; Sarmah et al., 2024; Wu et al., 2024; Guo et al., 2024) leverages LLMs to construct knowledge graphs (KGs), where nodes represent entity attributes and edges encode inter-entity relationships.

However, knowledge graph–enhanced methods overlook the entity ambiguity issue, where multiple names refer to the same character, and their retrieval process typically ignores the character's knowledge boundary, leading to responses that go beyond the intended role and produce out-of-character content.

3 RoleRAG

Our overall framework for RoleRAG is illustrated in Figure 2 and consists of two novel modules specifically designed for the role-playing task: (1) an entity normalization module that removes semantically duplicated entities, and (2) a retrieval module that fetches question-relevant information while rejecting out-of-scope queries.

3.1 Entity and Relation Extraction

In role-playing, character corpora often originate from novels, TV series, or biographies, with descriptions that exceed LLM token limits. Following prior work (Edge et al., 2024; Wu et al., 2024; Guo et al., 2024), we split descriptions into chunks $\{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_n\}$, process them independently, and aggregate the results into a unified character profile.

For each chunk $\mathcal{D}_i$, we employ LLMs to meticulously extract entities, adhering to a predefined data structure: {*name*, *type*, *description*}, denoted by $\mathbf{n}_i$. Furthermore, we prompt LLMs to identify structural relations between two entities, specifically, {*source*, *target*, *description*, *strength*}, denoted by $\mathbf{r}_i$, where *description* and *strength* denote the textual relationship and its intensity between the source and target nodes, respectively. After all chunks are processed, all entities and relations are stored in global databases $\mathcal{N}$ and $\mathcal{R}$.

To enable semantic retrieval, each entity $\mathbf{n}_i$ is encoded into a high-dimensional vector $\mathbf{v}_i$ using a text embedding model applied to both its name and description. The resulting pairs $\mathbf{n}_i, \mathbf{v}_i$ are stored in a vector database $\mathcal{V}$ for efficient similarity-based retrieval. We define the retrieval interface as $f_k(\mathcal{V}, \mathbf{n})$, which returns the top k entities most similar to a query entity.

Algorithm 1 Entity Normalization Algorithm

Require: Entity Database $\mathcal{N}$.
Ensure: a unified name for each name group.

Initialize empty entity graph $\mathcal{G}$.
Initialize empty vector database $\mathcal{V}$.
for $\mathbf{n}_i \in \mathcal{N}$ **do**
  if $\mathbf{n}_i \in \mathcal{V}$ **then**
    continue; ▷ node exists
  else
    $\mathcal{N}_k = f_k(\mathbf{n}_i, \mathcal{V})$
    Insert $\mathbf{n}_i$ to $\mathcal{V}$
    Insert $\mathbf{n}_i$ to $\mathcal{G}$
  end if
  for $\mathbf{n}_j \in \mathcal{N}_k$ **do**
    if $\mathbf{n}_i == \mathbf{n}_j$ **then** ▷ LLM prompt
      Insert $\mathbf{n}_j$ to $\mathcal{G}$
      Connect $\mathbf{n}_i$ and $\mathbf{n}_j$ in $\mathcal{G}$
    else
      continue
    end if
  end for
end for
Count the number of connected components in $\mathcal{G}$
for each connected components G in $\mathcal{G}$ **do**
  Select the unified name in G ▷ LLM prompt
end for

3.2 Entity Normalization

To mitigate entity ambiguity, we introduce a semantic entity normalization procedure, detailed in Algorithm 1. Given all extracted entities, our algorithm iterates through each entity, retrieving the k most semantically similar entities from the entity vector database. Next, we present these entity pairs, along with their names and descriptions, to the LLM, prompting it to determine if they refer to the same character. If the LLM identifies two entities as the same individual, we connect their corresponding nodes with an edge in the entity graph $\mathcal{G}$. After processing all entities, we partition the entity graph into multiple connected subgraphs, each representing a distinct individual, as illustrated in