ReflectKGC: Explainable Knowledge Graph Completion via a Plan-Act-Judge Agent

Anonymous ACL submission

Abstract

Knowledge Graph Completion (KGC) is crucial for addressing Knowledge Graph (KG) incompleteness, a key limitation for downstream applications. Existing KGC methods, including Large Language Model (LLM)-based approaches, often struggle with factual correctness and transparent reasoning for predictions. We introduce ReflectKGC, a novel, training-free Plan-Act-Judge agent framework designed to tackle these challenges. ReflectKGC employs LLMs across three stages for interpretable and accurate KGC: 1) Planning: An LLM profiles relations from example triples, inferring semantics and entity type constraints. 2) Acting: An Evaluator LLM assesses candidates, generating scores and human-readable rationales based on the profiled relation. 3) Judging: Critically, a Judge LLM scrutinizes the Evaluator's rationales, re-scoring or filtering candidates based on flawed reasoning, actively correcting predictions to enhance accuracy. This rationale-driven active correction enables ReflectKGC to deliver more accurate and trustworthy results. Experiments on standard benchmarks demonstrate ReflectKGC's state-of-the-art performance, yielding verifiable and accurate completions.

1 Introduction

Knowledge Graphs (KGs), representing facts as (head entity, relation, tail entity) triples like (egg, nutrient, Vitamin B5), underpin numerous applications such as question answering and recommendation systems. Knowledge Graph Completion (KGC) aims to automatically infer missing links, enriching KGs by analyzing existing facts—for instance, predicting the missing tail entity in a query like (egg, nutrient, ?). However, while methods ranging from triple-based approaches (Sun et al., 2019; Zhu et al., 2021), PLM-driven text-based models (Yao et al., 2019; Wang et al., 2021), to recent techniques leveraging Large Language Models (LLMs) (Wei et al., 2023; Liu et al., 2024; Li et al., 2025) have advanced KGC, many operate as "black boxes." This opacity hinders scrutiny, can propagate erroneous triples (e.g., incorrectly predicting (egg, nutrient, Vitamin C)), and critically limits trust in high-stakes domains like healthcare and finance, where understanding the "why" behind a prediction is paramount (Di Mauro et al., 2024). Explainability is thus evolving from a desirable feature to a fundamental requirement.

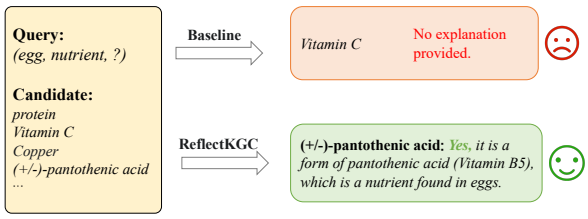

Figure 1: ReflectKGC yields interpretable, accurate completions over baselines. For `(egg, nutrient, ?)`, a baseline might wrongly predict 'Vitamin C' without explanation. ReflectKGC correctly identifies '(+/-)-pantothenic acid' (Vitamin B5) in eggs, providing a human-readable rationale for its prediction.

While LLMs bring powerful semantic understanding to KGC, they are not immune to producing erroneous or poorly justified outputs. Critically, existing research in explainable KGC has largely focused on generating post-hoc explanations for human understanding or subsequent model optimization (Bikaun et al., 2024; Chen et al., 2024; Chang et al., 2024; Di Mauro et al., 2024). The direct use of these explanations to actively refine KGC predictions for improved accuracy remains largely unexplored. This gap highlights the need for KGC methods that not only provide transparent reasoning but also leverage this interpretability to enhance the completion process itself.

To meet this demand for interpretable KGC with active correction, LLMs present a compelling avenue for generating the necessary nuanced explanations. However, the inherent fallibility of

LLMs—manifesting as potential factual inaccuracies, domain-specific knowledge gaps, or difficulties with complex structured reasoning (Bang et al., 2023; Yang et al., 2024; Sun et al., 2023)—necessitates a crucial second step: a mechanism to critically evaluate and refine these LLM-generated rationales to ensure the final accuracy of the KGC process. It is this dual imperative—to harness LLMs for explainable KGC while simultaneously ensuring the reliability of their interpretations—that motivates our work. We introduce ReflectKGC, a novel, training-free KGC framework structured around a Plan-Act-Judge agent paradigm. ReflectKGC distinctively employs LLMs at each stage, with a score-centric approach, to achieve both interpretability and enhanced accuracy through active, rationale-based correction: 1) Planning (Relation Profiling): An LLM analyzes example triples to summarize the relation's natural language description and infer entity type constraints. 2) Acting (Candidate Evaluation): An Evaluator LLM assesses each candidate entity against these constraints and semantic coherence, outputting a score and a detailed, human-readable rationale. 3) Judging (Rationale-based Correction): Crucially, a Judge LLM scrutinizes the Evaluator's rationale, correcting scores or filtering candidates based on flawed reasoning, thereby directly mitigating errors by leveraging these very explanations to improve final accuracy.

Our main contributions are:

- We introduce ReflectKGC, a novel training-free Plan-Act-Judge agent framework that uniquely integrates LLM-driven interpretability into the KGC process for enhanced accuracy.
- Our approach generates explicit, human-readable rationales that are actively employed by a Judge LLM for correction, directly improving the reliability and accuracy of KGC outputs.
- Extensive ablation studies validate the effectiveness of each component within our proposed framework.

2 Related Works

Large Language Models in KGC LLMs have shown significant promise in KGC by leveraging their vast pre-trained knowledge. Key approaches aim to bridge the gap between structured KGs and the unstructured nature of LLMs, address grounding errors, and manage computational costs. These frameworks represent different strategies for LLM integration, from extensive context augmentation and dataset-specific fine-tuning to training-free prompt engineering. **ReflectKGC's Distinction:** ReflectKGC adopts a **training-free** philosophy, akin to KICGPT, by leveraging pre-trained LLMs for all its stages (Plan, Act, Judge) through carefully designed prompting and inter-LLM communication. This approach inherently offers **greater generalization capabilities across diverse knowledge graphs and relations without the need for dataset-specific fine-tuning**, which is a core component of methods like DIFT and KGR3's re-ranker. Consequently, ReflectKGC is designed to be more readily adaptable to new KGs with potentially lower setup overhead. Its core novelty, however, remains its explicit Plan-Act-Judge agentic structure, where the "Judge" LLM actively scrutinizes and corrects the "Act" LLM's outputs based on the quality of its generated rationale. This rationale-driven internal correction loop for enhancing accuracy and interpretability, built upon a training-free foundation, is a key differentiator.

LLM-as-a-Judge and Agentic Frameworks The "LLM-as-a-Judge" paradigm leverages LLMs to evaluate the quality of outputs against specified criteria, offering scalable automated assessment. This is relevant for assessing KGC prediction plausibility beyond standard metrics. Concurrently, LLM-based Agentic frameworks (e.g., ReAct (Yao et al., 2023)) are emerging, endowing LLMs with capabilities like planning, tool use, and reflection, often structured in plan-act-evaluate cycles. For KGC, an agent might plan information retrieval, execute queries, and refine its strategy. Recent work also explores LLM self-correction and critique, where models iteratively refine outputs based on internal or external feedback (Gu et al., 2024). **ReflectKGC's Distinction:** ReflectKGC explicitly adopts a Plan-Act-Judge agentic structure. The "Judge" stage directly embodies the LLM-as-a-Judge concept, but critically, it's not just for post-hoc evaluation. Instead, the Judge's assessment of the Evaluator's (Act stage) rationale is an integral part of the KGC pipeline, leading to active re-scoring and correction of the predictions before they are finalized. This tight integration of critique and refinement within the KGC process, forming a specialized agent for interpretable and ac-

curate completion, distinguishes ReflectKGC from general agentic frameworks or standalone LLM-as-a-Judge applications.

3 Methods

This section details the ReflectKGC framework, a novel Plan-Act-Judge agent approach designed for interpretable and accurate Knowledge Graph Completion (KGC). ReflectKGC systematically employs Large Language Models (LLMs) across three distinct stages: Relation Profiling (Plan), Candidate Evaluation and Rationale Generation (Act), and Rationale-based Correction and Verification (Judge). The core idea is to leverage LLM-generated rationales not just for human understanding, but as an active component for refining predictions and improving accuracy.

3.1 Overall Framework of ReflectKGC

Given a KGC query, typically in the form of an incomplete triple $(h, r, ?)$ where h is the head entity, r is the relation, and ? denotes the missing tail entity (or similarly for missing head entities $(?, r, t)$), and a set of candidate entities, ReflectKGC processes them through a sequential pipeline.

1. **Plan (Relation Profiling):** An LLM analyzes example triples sharing the same relation r from the knowledge graph. It profiles the relation by generating a natural language description (template) of its semantics and inferring plausible entity type constraints for its head and tail arguments. This stage provides crucial semantic context for subsequent evaluation.

2. **Act (Candidate Evaluation and Rationale Generation):** An Evaluator LLM assesses each candidate tail entity against the profiled relation (template and type constraints) and its semantic coherence with the head entity h. For each candidate, this stage outputs an initial score (e.g., on a Likert scale) and, critically, a detailed, human-readable textual rationale explaining the basis for this score.

3. **Judge (Rationale-based Correction and Verification):** A Judge LLM scrutinizes the rationales generated by the Evaluator LLM in the Act stage. Based on the factual correctness, logical soundness, and sufficiency of the rationale, the Judge LLM may confirm the initial score, correct it (Reflect), or even filter out the candidate if the rationale is found to be flawed or indicative of a misunderstanding. This stage outputs a verified score, a verification status (e.g., Correct, Unsure, Incorrect), and a judge's reason explaining any changes or confirmations.

This three-stage process, particularly the active correction by the Judge LLM based on rationales, is designed to produce more accurate and trustworthy KGC results, complete with explanations for each prediction. Figure 2 (as described in the user's initial draft) illustrates this overall workflow.

3.2 Stage 1: Plan - Relation Profiling and Query Formulation

The motivation for this stage is that LLMs may have an incomplete or incorrect understanding of specific relations, especially those involving domain-specific knowledge or nuanced semantic distinctions. Effective relation profiling is essential for guiding the subsequent evaluation of candidate entities.

3.2.1 Relation Profiling

To understand the semantics of a given relation R, we employ an LLM. For each distinct relation present in the KG, a sufficient number of example triples (e.g., up to 100) exhibiting this relation are sampled from the training set. The LLM is prompted to analyze these examples and perform two tasks:

1. **Generate a Natural Language Template:** The LLM summarizes the common semantic pattern of the relation into a natural language sentence template with placeholders for the head and tail entities. For example, for the relation `nutrient`, the template might be "[Head] contains <MASK> as one of its nutrients." This template captures the typical way the relation is expressed.

2. **Infer Entity Type Constraints:** Based on the example entities participating in the relation, the LLM infers the general types of entities that can plausibly act as the head and tail for this relation. For instance, for `nutrient`, the head entity type might be "Food items, Organisms" and the tail entity type "Nutrients, Chemical compounds."

These generated templates and type constraints for all relations are stored as a "relation profile memory" for the agent, to be used in subsequent stages.

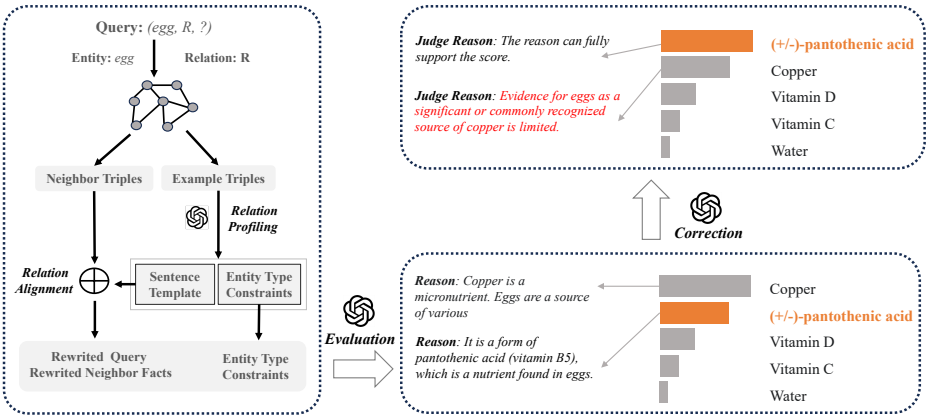

Figure 2: Overview of ReflectKGC. The framework processes a query and KGC candidates through Plan (Relation Profiling), Act (Initial Candidate Evaluation & Rationale Generation), and Judge (Rationale-based Correction & Verification) stages, yielding categorized completions (e.g., Correct, Unsure, Incorrect) with justifications.

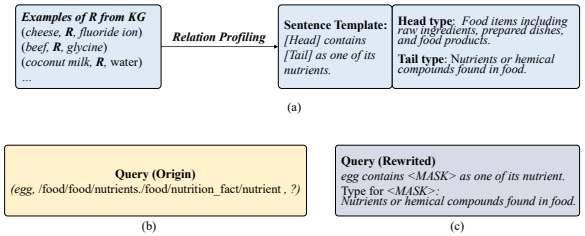

Figure 3: ReflectKGC Plan Stage. (a) Relation Profiling uses KG examples of relation R to derive a sentence template and entity type constraints. (b) Original KGC query. (c) The query is rewritten and enriched with type constraints for the Act stage.

This addresses challenges like understanding complex or composed relations, such as those found in datasets like FB15k-237 where intermediate CVT (Common Value Type) entities might obscure direct relationships. Figure 3 (a) depicts this process.

3.2.2 Query Formulation

Once a KGC query $(h, R, ?)$ is received, the precomputed profile for relation R (template and type constraints) is retrieved. The query is then formulated for the Act stage as follows:

1. The natural language template for R is instantiated with the given head entity h, leaving a placeholder for the tail entity. For example, if h is "egg" and the template is "[Head] contains <MASK> as one of its nutrients.", the formulated query sentence becomes "egg contains <MASK> as one of its nutrients."

2. The inferred tail entity type constraint for relation R is appended to this query sentence to guide the LLM. For example, "Type for <MASK>: Nutrients."

This formulated query, enriched with semantic and type information, is then passed to the Act stage. This process is illustrated in Figure 3 (b) and (c). This natural language rewriting is also applied to any neighbor facts used for context in the Act stage.

3.3 Stage 2: Act - Candidate Evaluation and Rationale Generation

The goal of the Act stage is to generate an assessment, comprising a score and a supporting rationale, for each candidate tail entity. This stage receives the rewritten query from the Plan stage (e.g., "egg contains <MASK> as one of its nutrient. Type for <MASK>: Nutrients.") and a list of candidate tail entities $C = \{c_1, c_2, ..., c_n\}$, which can be sourced from a base KGC model or a retrieval mechanism.

For each candidate entity c_j in C, an Evaluator LLM performs the following steps:

1. The LLM is prompted with the rewritten query, where the <MASK> is replaced by the candidate entity c_j. For example, "egg contains Vitamin C as one of its nutrient. Type for Vitamin C: Nutrients."

2. It is instructed to assess the likelihood or correctness of c_j being the true tail entity for h under relation R. This assessment considers the provided type constraints, the semantic coherence of the completed triple, and the LLM's internal knowledge, potentially augmented by retrieved neighbor facts of h.

3. The LLM outputs a **Score** reflecting this assessment, using a Likert scale from 1 (definitely incorrect) to 7 (definitely correct), with 4 signifying "unsure."

4. Finally, the LLM generates a **Reason** (a human-readable rationale) explaining the basis for its assigned Score, adhering to principles of interpretability, sufficiency, conciseness, and faithfulness. For example, for `(egg, nutrient, (+/-)-pantothenic acid)`, a good Reason might include: "(1) (+/-)-pantothenic acid is a form of Vitamin B5. (2) Vitamin B5 is a nutrient found in eggs."

The output of this stage for each candidate is a `(Candidate, Score, Reason)` tuple. Figure 2 illustrates this output, which provides the first pass of evaluation and the raw explanations for scrutiny by the Judge LLM.

3.4 Stage 3: Judge - Rationale-based Correction and Verification

This stage embodies the core "Reflect" mechanism and is a key novelty of the ReflectKGC framework. Its primary objective is to critically evaluate the **Reason**s generated in the Act stage alongside their associated **Score**s. Based on this scrutiny, the Judge LLM corrects the Scores, thereby aiming to improve the overall accuracy and trustworthiness of the KGC predictions. This approach, inspired by "LLM-as-a-Judge" research (McAleese et al., 2024), allows ReflectKGC to actively mitigate potential errors, biases, or overconfidence from the Act stage by leveraging interpretability for enhanced accuracy.

The input to this stage consists of the `(Candidate, Score, Reason)` tuples produced by the Evaluator LLM in the Act stage. For each such tuple corresponding to a candidate c_j with its $\texttt{Score}_j$ and $\texttt{Reason}_j$, a distinct Judge LLM undertakes the following process:

1. The Judge LLM is prompted to scrutinize the provided $\texttt{Reason}_j$.

2. It is instructed to evaluate the factual correctness, logical soundness, and sufficiency of this $\texttt{Reason}_j$ in supporting the accompanying $\texttt{Score}_j$. The Judge assesses whether the provided Reason logically leads to the Score and if the evidence cited or implied within the Reason is valid and adequate.

3. Based on this critical assessment, the Judge LLM takes one of several actions:

    - **Confirm the Score:** If the Reason is deemed sound, factually correct, and fully supports the Score. In such cases, the Score might be affirmed, or confidence in it increased (e.g., for a candidate like `(+/-)-pantothenic acid` with a strong, verifiable Reason, its Score might be maintained or slightly increased).
    - **Correct the Score (Reflect):** If the Reason is identified as flawed (e.g., containing factual inaccuracies, logical fallacies), insufficient (e.g., relying on weak, irrelevant, or generalized evidence not specific enough to the query), or indicative of overconfidence from the Evaluator. The Judge LLM then adjusts the Score accordingly. For instance, if "Copper" received a high Score for the query `(egg, nutrient,?)` but the Reason was weak, its Score might be reduced.
    - **Mark as "Unsure":** If the evidence presented in the Reason is limited, ambiguous, or the reasoning itself is tenuous. The candidate might then be marked as "Unsure," often accompanied by a Score adjustment towards the middle of the scale (e.g., a Score of 4).
    - **Mark as "Incorrect":** If the Reason is found to be entirely nonsensical, demonstrates a clear misunderstanding of the query or relation, or is based on demonstrably false premises. In such scenarios, the candidate is typically marked as "Incorrect," and its Score is significantly lowered (e.g., to 1).

4. Finally, the Judge LLM outputs a **Verified Score**, a **Verification Status** (e.g., "Correct," "Unsure," "Incorrect" derived from the final score and assessment), and a **Judge Reason**. This Judge Reason explains any modifications made to the Score or confirms the original assessment, providing transparency into the Judge's decision-making (e.g., "The Reason provided by the Evaluator fully supports the Score." or "Evidence for eggs as a significant or commonly recognized source of copper is limited, hence the Score was reduced and marked Unsure.").

The right side of Figure 2 illustrates this crucial correction process. Upon completion of the ReflectKGC pipeline, candidate entities are categorized based on their verified scores and associated sta-

tus, yielding a refined and interpretable set of KGC predictions.

4 Experiments

4.1 Setup

Datasets. To assess the performance of our proposed method, we conduct experiments on two extensively utilized benchmark datasets: FB15k-237 and WN18RR. FB15k-237 is a subset of Freebase (Bollacker et al., 2008), a large-scale knowledge graph containing encyclopedic facts about a wide array of topics such as notable individuals, organizations, and cultural works. WN18RR, derived from WordNet (Miller, 1995), is a lexical knowledge graph that primarily captures information regarding English word meanings and relationships. Crucially, to ensure a fair evaluation and prevent issues of data leakage, both FB15k-237 and WN18RR are curated by removing inverse relations from their original knowledge bases. Further detailed statistics for these datasets are provided in Table 1.

Dataset	# Entities	# Relations	# Train	# Valid	# Test
FB15k-237	14,541	237	272,115	17,535	20,466
WN18RR	40,943	11	86,835	3,034	3,134

Table 1: Statistics of the benchmark datasets.

Baselines. We compare our proposed method with several strong baseline methods spanning three main categories: triple-based, text-based, and large language model-based approaches. The triple-based methods include: TransE (Bordes et al., 2013), ComplEx (Trouillon et al., 2016), RotatE (Sun et al., 2019), TuckER (Balazevic et al., 2019), GIE (Cao et al., 2022), CompGCN (Vashishth et al., 2020), and NBF-Net (Zhu et al., 2021). The text-based methods include: KG-BERT (Yao et al., 2019), MEM-KGC (Choi et al., 2021), SimKGC (Wang et al., 2022), and CoLE (Liu et al., 2022). The large language model-based methods include ChatGPT, KICGPT (Wei et al., 2023) and MKGL (Guo et al., 2024); the results for ChatGPT (1-shot) are adopted from the KICGPT paper (Wei et al., 2023).

Metrics. For evaluating our proposed method, Mean Reciprocal Rank (MRR) and Hits@k (for $k = 1, 3$, and 10) are the standard metrics employed. MRR provides the average of the reciprocal ranks for the correct target entities over all test queries. Hits@k, on the other hand, indicates the proportion of test queries where the true target entity is successfully ranked within the top-k predicted entities. Consistently, higher values across all these metrics denote a more effective model performance.

Implementation Details. For initial candidate generation in ReflectKGC, we employ NBF-Net (Zhu et al., 2021), selected for its leading performance among triple-based methods (detailed in Table 2) and configured as per its original publication. NBF-Net provides the top 20 candidates per query. Our Plan, Act, and Judge modules then re-score these candidates, primarily utilizing the Qwen-72B API; the GPT-4o API was also used for comparative experiments reported in Table 4. To ensure deterministic outputs and reduce randomness, all LLM API calls used a temperature of 0. The final ranking sorts the LLM-generated scores and fuses this new order with NBF-Net's original candidate ranking using the Reciprocal Rank Fusion (RRF) (Cormack et al., 2009)algorithm. Our framework operates in a train-free manner, and all reported experimental results are the average of three runs. Approximate processing times are 30 minutes for the FB15k-237 dataset and 12 minutes for the WN18RR dataset. Detailed prompts for LLMs are available in Appendix A.

4.2 Main Results

Table 2 presents the main experimental results, underscoring the efficacy of our ReflectKGC framework. Notably, when compared to the KICGPT baseline, ReflectKGC demonstrates significant performance enhancements across both datasets. On WN18RR, ReflectKGC achieves absolute improvements of 0.114 in MRR and 0.129 in Hits@1 compared to KICGPT. On FB15k-237, our method also achieves improvements of 0.011 in MRR and 0.004 in Hits@1. These clear improvements highlight the effectiveness of our proposed approach.

It is also noteworthy that ReflectKGC achieves substantial improvements in Hits@3 and Hits@10 metrics when compared against traditional non-LLM baselines across both datasets; for instance, on WN18RR, it surpasses NBF-Net by up to 5.4% in Hits@3 and 4% in Hits@10. This particular strength underscores the efficacy of our Act module, where the Evaluator LLM precisely identifies target entities and assigns them high scores accompanied by detailed rationales, thereby ensuring robust recall of correct answers within the top can-

Methods	FB15k237				WN18RR			
	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
Triple-based Methods								
TransE	0.279	0.198	0.376	0.441	0.243	0.043	0.441	0.532
ComplEx	0.247	0.158	0.275	0.428	0.440	0.410	0.460	0.510
RotatE	0.338	0.241	0.375	0.533	0.476	0.428	0.492	0.571
TuckER	0.358	0.266	0.394	0.544	0.470	0.443	0.482	0.526
GIE	0.362	0.271	0.401	0.552	0.491	0.452	0.505	0.575
HittER	0.373	0.279	0.409	0.558	0.503	0.462	0.516	0.584
CompGCN	0.355	0.264	0.390	0.535	0.479	0.443	0.494	0.546
NBF-Net	0.415	0.321	0.450	0.599	0.551	0.497	0.573	0.666
Text-based Methods								
KG-BERT	-	-	-	0.420	0.216	0.041	0.302	0.524
MEM-KGC	0.346	0.253	0.381	0.531	0.557	0.475	0.604	0.704
CoLE	0.387	0.293	0.426	0.570	0.585	0.532	0.607	0.689
Large Language Model-based Methods								
ChatGPT (1-shot)	-	0.267	-	-	-	0.212	-	-
MKGL	0.415	**0.325**	0.454	0.591	0.552	0.500	0.577	0.656
KICGPT	0.410	0.321	0.430	0.581	0.564	0.478	0.612	0.677
ReflectKGC (Ours)	**0.421**	**0.325**	**0.477**	**0.614**	**0.589**	**0.532**	**0.627**	**0.706**

Table 2: Experiment results of the KGC task on FB15k-237 and WN18RR datasets. The best results are in **bold** and the second-best ones are underlined. All results of baseline methods are referred from corresponding original papers.

didates. While Hits@1 scores also see consistent enhancements, the pronounced gains in Hits@3 and Hits@10 suggest ReflectKGC excels in scenarios common in complex KGs, such as queries with multiple plausible entities like the (egg, nutrient, ?) example in Figure1. In these cases, our framework adeptly positions the ground truth entity prominently among the top-ranked candidates, demonstrating strong discriminative power even when several options are semantically fitting.

Furthermore, ReflectKGC establishes a new state-of-the-art when compared against prior LLM-based KGC methods. For instance, while KICGPT also operates on a training-free basis, its performance does not consistently surpass leading traditional triple-based or text-based models; KICGPT's MRR of 0.410 on FB15k-237 trails NBF-Net (0.415), and its 0.564 MRR on WN18RR is substantially lower than that of SimKGC (0.671). We attribute ReflectKGC's more effective LLM utilization to its comprehensive Plan-Act-Judge architecture: the Plan module provides crucial relational context and type constraints derived from the graph structure, guiding the Act module's sophisticated candidate evaluation, which is then critically refined by the Judge module's rationale-based correction. This structured approach ensures a deeper and more reliable application of LLM capabilities. In contrast, methods like MKGL rely on LoRA-based fine-tuning, entailing a complex training process, and critically, their outputs lack explanatory rationales, which can diminish the persuasiveness of their results. ReflectKGC not only delivers stronger or comparable predictive performance (e.g., achieving an MRR of 0.421 on FB15k-237 versus MKGL's 0.415) but does so as a training-free framework that inherently provides crucial interpretability.

	MRR	Hits@1	Hits@3	Hits@10
ReflectKGC	**0.421**	**0.325**	**0.477**	**0.614**
w/o entity type constraints	0.395	0.301	0.440	0.605
w/o relation alignment	0.394	0.296	0.450	0.604
w/o neighbor triples	0.403	0.322	0.436	0.585
w/o judge	0.405	0.308	0.452	0.596

Table 3: Ablation results on **FB15k-237**.

4.3 Ablation Studies

To ascertain the individual contributions of key components within our ReflectKGC framework, we conducted a series of ablation studies on the FB15k-237 dataset. The detailed results of these experiments are presented in Table 3.

First, we examined the impact of **entity type constraints**, an output of our Plan module's Relation Profiling. Removing these constraints led to a notable performance decline across all metrics, with MRR dropping from 0.421 to 0.395 and Hits@3 decreasing from 0.477 to 0.440. This underscores their importance, particularly for a dataset like FB15k-237 where relations such as

"/film/film/film_festivals" can be ambiguous. While literally suggesting a link between a film and a festival, such relations often involve entities like individuals or awards associated with the festival. Without explicit type constraints, the Act module's LLM may misjudge the typicality of correct entities, assigning them unduly low scores.

Next, we ablated the **relation alignment template**, also part of the Plan module, which aids the LLM in interpreting complex relation identifiers. Its removal resulted in a significant performance drop across all metrics, with Hits@1 decreasing from 0.325 to 0.296 and MRR from 0.421 to 0.394. Many relations in FB15k-237, such as *"/food/food/nutrients./food/nutrition_fact/nutrient"* (as exemplified in Figure 2), are syntactically complex. Indeed, a substantial portion of the dataset exhibits this characteristic, with 144 out of the 237 distinct relations in FB15k-237 presenting similar structural intricacies, underscoring the prevalence of this challenge. The relation alignment template is designed to normalize or clarify these convoluted identifiers, and its absence can lead to LLM confusion and a poorer understanding of the relational context, thereby impairing overall KGC performance.

We then investigated the role of **neighbor triples**, which inform the Act module's pattern-guided evaluation by providing contextual evidence from the knowledge graph. Removing access to these triples resulted in a significant drop in Hits@3 (from 0.477 to 0.436) and Hits@10 (from 0.614 to 0.585), while Hits@1 remained relatively stable (0.325 to 0.322). The stability in Hits@1 can be attributed to the presence of numerous common-sense facts in FB15k-237 (e.g., (*"Titanic","/film/film/language","English"*)), for which the LLM's internal knowledge often suffices. However, for more complex queries requiring specific factual evidence (e.g., (*"52nd Annual Grammy Awards","/award/award_honor/award_winner","50 Cent"*)), the absence of relevant neighbor triples makes it challenging for the model to make informed judgments, thus impacting broader recall.

Finally, removing the **Judge module** led to a discernible decrease across all metrics, with MRR falling to 0.405 and Hits@1 to 0.308. While the performance drop might appear less substantial than other ablations, it highlights the Judge's crucial role in refining the outputs from the Act module. This suggests that while the Act module's Evaluator LLM often generates consistent scores and rationales, the Judge module is vital for identifying and correcting residual errors or flawed reasoning from the Evaluator. This final verification step is indispensable for enhancing the overall reliability and accuracy of ReflectKGC, ensuring that the explanations actively contribute to the trustworthiness of the final predictions.

Settings	MRR	Hits@1	Hits@3	Hits@10
Qwen-72B	0.421	0.325	0.477	0.614
GPT4o	0.420	0.325	0.479	0.617

Table 4: Ablation Experiments on FB15k-237 dataset with different LLM.

4.4 Analysis on different LLM

To assess framework generalizability, we evaluated ReflectKGC on FB15k-237 using both Qwen-72B and GPT-4o as the backbone LLM for all modules. As shown in Table 4, performance remained remarkably consistent across these distinct models, indicating that ReflectKGC's efficacy is driven by its robust Plan-Act-Judge architecture and rationale-based mechanisms rather than reliance on a specific LLM's idiosyncratic knowledge. This highlights the architectural soundness and adaptability of our approach.

5 Conclusion

We introduced ReflectKGC, a novel training-free Plan-Act-Judge agent framework that enhances Knowledge Graph Completion by improving factual correctness and transparent reasoning where existing methods often fall short. ReflectKGC systematically employs LLMs for relation profiling (Plan), candidate evaluation with rationale generation (Act), and critically, rationale-based correction by a Judge LLM, enabling it to mitigate LLM fallibility and deliver more accurate, trustworthy results. Demonstrated state-of-the-art performance on standard benchmarks underscores ReflectKGC's effectiveness in producing verifiable completions, with future work poised to extend this reflective framework.

Limitations

Despite its strengths, ReflectKGC's performance is inherently linked to the capabilities of the underlying LLMs, and its sequential Plan-Act-Judge pipeline can introduce latency, potentially impacting suitability for real-time applications. The efficacy of the critical Judge module also depends on nuanced prompt engineering to ensure accurate critique of the Act module's rationales without being overly restrictive or lenient. Finally, the system's overall recall is constrained by the quality of the initial candidate set provided by the external KGC model.

References

Ivana Balazevic, Carl Allen, and Timothy Hospedales. 2019. TuckER: Tensor factorization for knowledge graph completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5185–5194, Hong Kong, China. Association for Computational Linguistics.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, and 1 others. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *arXiv preprint arXiv:2302.04023*.

Tyler Bikaun, Michael Stewart, and Wei Liu. 2024. Cleangraph: Human-in-the-loop knowledge graph refinement and completion. *arXiv preprint arXiv:2405.03932*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 1247–1250, New York, NY, USA. Association for Computing Machinery.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26*, volume 26. Curran Associates, Inc.

Zongsheng Cao, Qianqian Xu, Zhiyong Yang, Xiaochun Cao, and Qingming Huang. 2022. Geometry interaction knowledge graph embeddings. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(5):5521–5529.

Heng Chang, Jiangnan Ye, Alejo Lopez-Avila, Jinhua Du, and Jia Li. 2024. Path-based explanation for knowledge graph completion. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 231–242.

Weijian Chen, Yixin Cao, Fuli Feng, Xiangnan He, and Yongdong Zhang. 2024. Hogrn: Explainable sparse knowledge graph completion via high-order graph reasoning network. *IEEE Transactions on Knowledge and Data Engineering*.

Bonggeun Choi, Daesik Jang, and Youngjoong Ko. 2021. Mem-kgc: Masked entity model for knowledge graph completion with pre-trained language model. *IEEE Access*, 9:132025–132032.

Gordon V Cormack, Charles LA Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 758–759.

Antonio Di Mauro, Zhao Xu, Wiem Ben Rim, Timo Sztyler, and Carolin Lawrence. 2024. Generating and evaluating plausible explanations for knowledge graph completion. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12106–12118.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, and 1 others. 2024. A survey on llm-as-a-judge. *arXiv preprint arXiv:2411.15594*.

Lingbing Guo, Zhongpu Bo, Zhuo Chen, Yichi Zhang, Jiaoyan Chen, Lan Yarong, Mengshu Sun, Zhiqiang Zhang, Yangyifei Luo, Qian Li, Qiang Zhang, Wen Zhang, and Huajun Chen. 2024. Mgkl: Mastery of a three-word language. In *NeurIPS*.

Muzhi Li, Cehao Yang, Chengjin Xu, Xuhui Jiang, Yiyan Qi, Jian Guo, Ho-fung Leung, and Irwin King. 2025. Retrieval, reasoning, re-ranking: A context-enriched framework for knowledge graph completion. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4349–4363, Albuquerque, New Mexico. Association for Computational Linguistics.

Yang Liu, Zequn Sun, Guangyao Li, and Wei Hu. 2022. I know what you do not know: Knowledge graph embedding via co-distillation learning. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM '22, page 1329–1338, New York, NY, USA. Association for Computing Machinery.

Yang Liu, Xiaobin Tian, Zequn Sun, and Wei Hu. 2024. Finetuning generative large language models with discrimination instructions for knowledge graph completion. *Preprint*, arXiv:2407.16127.

Nat McAleese, Rai Michael Pokorny, Juan Felipe Ceron Uribe, Evgenia Nitishinskaya, Maja Trebacz, and Jan Leike. 2024. Llm critics help catch llm bugs. *arXiv preprint arXiv:2407.00215*.

George A. Miller. 1995. Wordnet: a lexical database for english. *Commun. ACM*, 38(11):39–41.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel M Ni, Heung-Yeung Shum, and Jian Guo. 2023. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. *arXiv preprint arXiv:2307.07697*.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Eric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2071–2080, New York, New York, USA. PMLR.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2020. Composition-based multi-relational graph convolutional networks. In *International Conference on Learning Representations*.

Liang Wang, Wei Zhao, Zhuoyu Wei, and Jingming Liu. 2022. SimKGC: Simple contrastive knowledge graph completion with pre-trained language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4281–4294, Dublin, Ireland. Association for Computational Linguistics.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.

Yanbin Wei, Qiushi Huang, Yu Zhang, and James Kwok. 2023. KICGPT: Large language model with knowledge in context for knowledge graph completion. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8667–8683, Singapore. Association for Computational Linguistics.

Linyao Yang, Hongyang Chen, Zhao Li, Xiao Ding, and Xindong Wu. 2024. Give us the facts: Enhancing large language models with knowledge graphs for fact-aware language modeling. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3091–3110.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Kg-bert: Bert for knowledge graph completion.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. 2021. Neural bellman-ford networks: A general graph neural network framework for link prediction. In *Advances in Neural Information Processing Systems*, volume 34, pages 29476–29490. Curran Associates, Inc.

A Appendix

Prompt Template

```
<Role>
You are a knowledge graph expert, skilled in analyzing entity relationships and extracting semantic information.
</Role>

<Task>
I will provide you with a series of triples in the format: head_entity\trelation\ttail_entity
All these triples share the same relation: {relation}
Please analyze these triples and complete the following tasks:
1. Summarize the types of head entities (there may be multiple types)
2. Summarize the types of tail entities (there may be multiple types)
3. Provide a concise one-sentence description of this relation.
4. Write a natural sentence template using [Head] and [Tail] that clearly expresses the meaning and direction of the relation. Keep
it concise and intuitive. Do not use parentheses or examples.
</Task>

<Format>
Please return your response in JSON format as follows:
{
"head_entity_type": "type of head entities in one sentence or a python enumerate list",
"tail_entity_type": "type of head entities in one sentence or a python enumerate list",
"short_description": "brief description of the relation (one sentence)",
"template": "natural language template with [Head] and [Tail]"
}
<\Format>

<Input>
Here are the triple data: {triples}
Please provide the final answer in JSON format:
<\Input>
```

Figure 4: Prompt fo Relation Profiling

Prompt Template

```
<Role>
You are a knowledge graph expert skilled in evaluating the validity of candidate entities in knowledge graph completion tasks.
</Role>

<Task>
- Question: Based on the provided context information and your general knowledge, does the candidate entity logically fit into the
sentence and context? Choose a score from 1 to 7:
- 1: Completely Incorrect (creates a clear contradiction with known facts or context)
- 4: Uncertain (not sure)
- 7: Correct (logically consistent and fits perfectly with context and knowledge)
</Task>

<Rule>
Explain the provided score. Your explanation must be:

* **Readable:** Clear, straightforward, and easy for the target audience to understand.
* **Sufficient yet Concise:** Cover all crucial aspects and implications of the score thoroughly, while remaining brief and to the
point, avoiding jargon or unnecessary details.
* **Faithful:** Accurately and truthfully represent the score's intended meaning, how it is derived (if applicable), and its
correct interpretation within its given context.
</Rule>

<Format>
Please output your evaluation in the following JSON format:
{
"entity": "candidate_entity_name",
"reason": "Explanation for this score",
"score": score (1-7)
}
</Format>

<Input>
Triple to be completed: {triple}
Candidate entity: {candidate_entities}
Provided context: {context}
Please provide the final answer in JSON format:
<\Input>
```

Figure 5: Prompt fo Evaluation

Prompt Template

```
<Role>
You are a knowledge graph expert skilled in evaluating candidate entities based on previous judgments.
</Role>

<Task>
**Objective:**
Re-evaluate a candidate entity's fit for a triple, using a previously assigned score/reason and new `context`.

**Inputs You Will Receive:**
* `context`
* `current_score` (A 1-7 score from a previous judgment)
* `current_reason` (The explanation for the `current_score`)

**Reference 1: Scoring Scale (1-7)**
This scale indicates how well the candidate entity fits the triple:
* 1: **Completely Incorrect** (clear contradiction with known facts or `context`)
* 4: **Uncertain** (not sure)
* 7: **Correct** (logically consistent and fits perfectly with `context` and knowledge)
(Intermediate scores denote varying degrees of fit.)

**Reference 2: Reason Definition**
The `reason` explains the justification for a given `score`.

**Your Evaluation Steps:**
1.  **Context-Reason Analysis:** Does the provided `context` adequately support the `current_reason`?
2.  **Reason-Score Analysis:** Does the `current_reason` logically justify the `current_score` (based on the 1-7 scale defined
above)?
3.  **Final Judgment & Explanation:** Based on your analysis in Steps 1 and 2:
    * **If revision is needed:** Provide a `new_score` (1-7) and a clear, comprehensive `new_reason` explaining your revised
judgment.
    * **If no revision is needed:** Briefly explain why the `current_score` and `current_reason` remain appropriate and are well-
supported by the `context`.
</Task>

<Format>
Please output your evaluation in the following JSON format:
{
"entity": "candidate_entity_name",
"judgment": {
"score": score (1-7),
"reason": "Explanation based on the provided context, reason and score",
"final_assessment": "Whether this is a good candidate or not"
}
}
</Format>

<Input>
. . .
Please provide the final answer in JSON format:
</Format>
```

Figure 6: Prompt fo Correction