PrO-KGC: Prompt Optimization for LLM-Based Knowledge Graph Completion

Amel Gader[1],*, Alsayed Algergawy[1]

[1]*Chair of Data and Knowledge Engineering, University of Passau, Passau Germany*

Abstract

Knowledge Graphs (KGs) are integral to multiple applications which makes their completeness a key research focus. Large Language Models (LLMs) have been increasingly leveraged alongside embedding models to enhance Knowledge Graph Completion (KGC). However, effectively transforming contextual features from KGs into well-structured textual prompts for LLMs remains a challenge. In this work, we propose PrO-KGC, a framework for **Pr**ompt **O**ptimization in the context of LLM-Based **KGC**. PrO-KGC aims to enhance LLM-based KGC performance by refining and optimizing input prompts through multiple steps. This process includes enriching prompts with additional information, incorporating structural composition patterns and facts, and refining relation representations to be more LLM-friendly. Experimental results demonstrate the effectiveness of our approach, achieving improvements across three common benchmarks compared to vanilla models. Our code is available at https://github.com/amal-gader/PrO-KGC.

Keywords

Knowledge Graph Completion, Large Language Models, Knowledge Graph to Text

1. Introduction

Knowledge graphs form the backbone of the semantic web and are among the most widely used knowledge representation techniques across various domains. As a result, extensive research has been conducted to address the incompleteness of knowledge graphs through different tasks and settings like the link prediction task which consists of predicting a tail entity t for a given head entity h and a relation r (h, r, ?).

Knowledge graph completion was initially tackled using knowledge graph embedding (KGE) models such as TransE [1] and RotatE [2]. Later, embedding models evolved to incorporate semantics and textual information alongside structural connections in knowledge graphs [3]. With the rise of large language models (LLMs), tasks like link prediction were reformulated as sequence-to-sequence problems. Models like KG-BERT [4] and KGT5 [5] demonstrated the potential of LLMs in knowledge graph completion (KGC). However, research consistently finds that LLMs alone struggle to capture the intricate relationships and structured nature of knowledge graphs [6, 3]. Instead of viewing KGE models and LLMs as competing approaches, recent studies like [7, 8, 9, 10] have explored their synergy, leveraging LLMs in a second step to rerank and refine KGE model predictions.

In-context learning techniques were used to craft rich prompts by including entity descriptions, entity classes and relevant relations providing the model with structured hints about potential predictions [11, 12]. However, prompts can still be further optimized to maximize the integration of textual features and better adapt the task to LLMs' input processing patterns, especially that LLMs have shown reasoning capabilities [13]. One interesting aspect that was not well explored is the incorporation of composition patterns in the given prompt. A composition pattern is a set of relations where one relation logically implies another. In other terms, if A has *relation 1* with B and B has *relation 2* with C, then we can infer that A has *relation 3* with C. An intuitive example is illustrated in Figure 1: if *Dr. John Smith* works at *the University of Edinburgh* and *the University of Edinburgh* is in *Edinburgh, Scotland*, then

LLM-TEXT2KG 2025: 4th International Workshop on LLM-Integrated Knowledge Graph Generation from Text (Text2KG), June 01–07, 2025, Portoroz, Slovenia

*Corresponding author.

amel.gader@uni-passau.de (A. Gader); alsayed.algergawy@uni-passau.de (A. Algergawy)

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

we may reasonably infer that *Dr. John Smith* lives in *Edinburgh, Scotland*. These facts help resolve ambiguities and improve the model's ability to generate accurate predictions. However, not all multi-hop pattern relations hold consistently, which can sometimes be misleading. For example, the following path: *Bob is friends with Alice*, and *Alice is friends with Charlie* does not necessarily imply that *Bob is friends with Charlie*. This highlights the need for a meticulous approach in selecting composition facts from the knowledge graph to ensure reliability and avoid incorrect inferences.

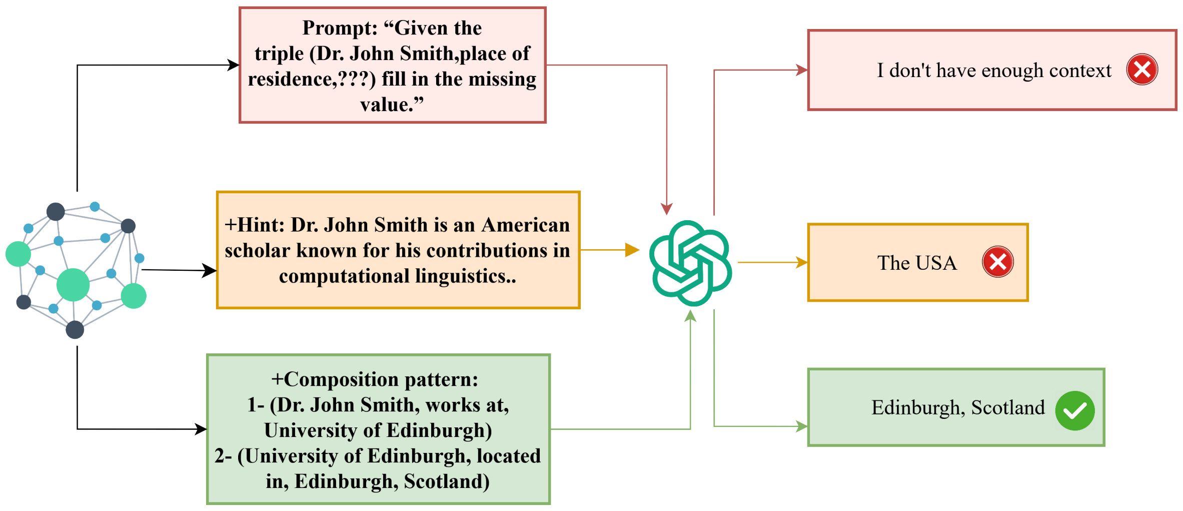
Prompt: "Given the triple (Dr. John Smith,place of residence,???) fill in the missing value."
+Hint: Dr. John Smith is an American scholar known for his contributions in computational linguistics..
+Composition pattern:
1- (Dr. John Smith, works at, University of Edinburgh)
2- (University of Edinburgh, located in, Edinburgh, Scotland)
I don't have enough context
The USA
Edinburgh, Scotland

Figure 1: This synthetic example reflects a realistic scenario where common names like *John Smith* require more context for an accurate response. At each step, additional context is introduced to refine the prediction.

In this paper, we propose PrO-KGC a framework to systematically extract and integrate textual features from knowledge graphs like descriptions, composition patterns and related contextual facts into LLMs' prompts, transforming the task into an NLP problem that aligns more closely with LLMs' expected inputs. We also suggest methods to refine and adapt textual inputs bridging the gap between symbolic reasoning in knowledge graphs and the language-driven capabilities of LLMs. Our method consists of three main components: Composition Pattern Extraction (CPE), which extracts and validates relational patterns from the KG to retrieve supporting facts, Description Generation (DG), which generates missing entity descriptions and Relation Transformation (RT), which transforms no-verbal relations into more intuitive natural language forms.

2. Related Work

In this section we provide a brief survey of the latest state-of-the-art research combining Large Language Models (LLMs) and Knowledge Graph Completion (KGC) models and methods to improve the integration of structured knowledge with LLMs for KGC tasks.

2.1. State of the Art in KGC with LLMs

The KGC task has evolved significantly with the integration of Knowledge Graph Embeddings (KGE) and LLMs. Traditional KGE models such as TransE [1], RotatE [2], and TuckER [14] learn structural connections from the training data but remain limited by their inability to leverage textual information. On the other hand, LLM-based models rely on rich textual features overlooking the structured nature of knowledge graphs. Several recent approaches have tried to address this gap by combining the strengths of both paradigms. One such method, KICGPT [7], introduces a framework that leverages both structural and textual information side by side. KICGPT incorporates a two-stage process where a triple-based retriever model extracts structural knowledge, followed by an LLM-based reranker that iteratively refines the candidate predictions. This method requires training only the retriever module, while the reranking is handled by an external LLM, ensuring efficiency. However, the iterative reranking introduces additional computational costs. To reduce this overhead and hallucinations associated with LLMs, DIFT [10] proposes an alternative by finetuning an LLM specifically for the reranking task. Instead of treating the LLM as a generic reasoning engine, this approach injects structured embeddings of both the query and candidate entities into the model, making it more aligned with the KG's structure. Additionally, it employs a refined sampling strategy to filter and select only the most relevant contextual facts, thereby improving efficiency and reliability.

Another direction explored in KGR3 [9] is addressing the variability between the predefined semantics of a KG and the open-ended nature of generative LLM predictions. This method follows a three-step process where a retrieval module extracts structural information from the KG and generates a list of candidate answers, a reasoning module uses an LLM to infer additional candidates, and a reranking module to combine and rerank these results.

A related effort, KLR-KGC [8], aims to leverage analogical reasoning by extracting structural patterns and subgraphs from the KG. The extracted analogical knowledge helps filter KGE predictions based on similarities to known triples, while the subgraph information provides additional context for the reranking step. By introducing structural analogies, this approach ensures that predictions abide by the underlying KG structure.

2.2. Integrating Structured Knowledge into prompts for LLM-Based KGC

Beyond the integration of LLMs with KGE models, recent work has also explored optimizing textual representations to better guide LLMs in KGC tasks. One example is MPIKGC [11] which focuses on designing effective prompts that are adaptable across different LLMs. By expanding entity and relational descriptions both globally (from the KG perspective) and locally (from the triple perspective), it improves the contextual richness of the input. However, it only relies on keyword-based fact extraction.

To improve multi-hop link prediction, KG-LLM [15] formulates knowledge graph structures as natural language inputs. By extracting all possible paths using depth-first search (DFS) and categorizing them based on whether the first and last nodes are connected, it enables the model to learn logical chains of inference. However, the model may apply learned decision paths even when the pattern does not hold due to the absence of a filtering step leading to overgeneralization.

Another approach, MuKDC [12], aims to enhance few-shot KGC by generating additional knowledge for long-tailed entities. It utilizes an LLM to generate triplets, attributes, and decision paths based on a set of instructions and prompt templates, enriching entities with relevant contextual information. To mitigate potential inaccuracies introduced by the LLM, a consistency check is applied using a KGE model to score and validate new facts. However, this method is relying on LLMs for structural context generation, which may not always align with the KG's factual constraints.

Overall, these works offer valuable contributions and point out the ongoing efforts to integrate structured knowledge with LLMs' capabilities. While some remain loyal to structural patterns through retrievers and analogical reasoning, others focus on refining textual inputs to better fit LLMs' expectations. The synergy between structured KGE models and unstructured LLMs' reasoning remains a key area of research to get the best blend of both paradigms. In this work we focus on the latter approach and leave the exploration of the former for future work.

3. Methodology

In this section we explain in detail our *PrO-KGC* approach.

3.1. Problem Definition

A knowledge graph (KG) is a structured representation of knowledge, defined as $\mathcal{G} = (\mathscr{E}, \mathscr{R}, \mathscr{F}, \mathscr{D})$, where $\mathscr{E}$ is the set of entities, $\mathscr{R}$ is the set of relations, $\mathscr{F}$ is the set of factual triples, and $\mathscr{D}$ is the set of textual descriptions associated with entities. Each fact $f \in \mathscr{F}$ is represented as a triplet: $(h, r, t) \in \mathscr{E} \times \mathscr{R} \times \mathscr{E}$, where h, t are the head and tail entities respectively connected by the relation r. An entity may have a description denoted as $d \in \mathscr{D}$.

Knowledge Graph Completion (KGC) includes tasks such as link prediction, relation prediction and instance completion aiming all to enrich KGs with new inferred facts. Traditional triple-based models do not leverage the set of descriptions $\mathscr{D}$ and other potential textual features, while LLM-based methods often overlook the structural connections within the KG. The *PrO-KGC* framework addresses this limitation and suggests a set of steps to preserve structual information and optimize the LLM input prompt for KGC tasks.

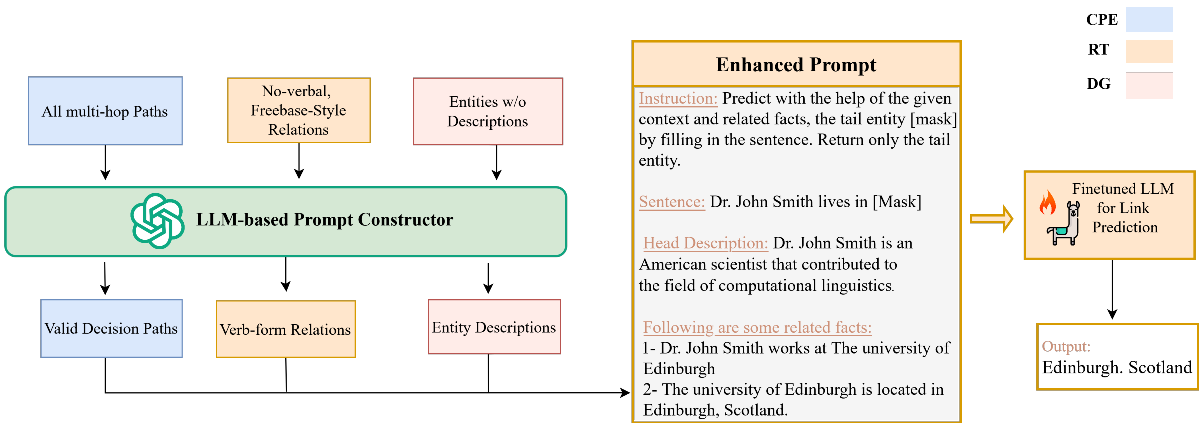
CPE
RT
DG
All multi-hop Paths
No-verbal, Freebase-Style Relations
Entities w/o Descriptions
LLM-based Prompt Constructor
Valid Decision Paths
Verb-form Relations
Entity Descriptions
Enhanced Prompt
Instruction: Predict with the help of the given context and related facts, the tail entity [mask] by filling in the sentence. Return only the tail entity.
Sentence: Dr. John Smith lives in [Mask]
Head Description: Dr. John Smith is an American scientist that contributed to the field of computational linguistics.
Following are some related facts:
1- Dr. John Smith works at The university of Edinburgh
2- The university of Edinburgh is located in Edinburgh, Scotland.
Finetuned LLM for Link Prediction
Output:
Edinburgh. Scotland

Figure 2: This figure depicts the proposed PrO-KGC framework. The process begins at the top left, where three LLM-powered operations are performed. CPE (Composition Pattern Extraction), RT (Relation Transformation), and DG (Description Generation) to construct an enhanced prompt, which is then fed to a fine-tuned LLM.

3.2. Overview

Our approach consists of two main steps, with the first being the primary focus of this work.

Step 1: Prompt Generation. The goal of this step is to extract and integrate the most relevant knowledge from the KG into the prompt to enhance the performance of the language model. First, if a head entity description is available in the KG, we incorporate it directly. Otherwise, we generate it using a backbone Large Language Model (LLM). Next, we extract composition patterns from the KG, validating them with the same LLM to extract relevant supporting facts. If no composition patterns are available, we retrieve a fact associated with a neighboring entity that shares the same relation. If such a fact is also unavailable, we select any fact involving the same relation. To address cases where the relation is ambiguous or difficult to interpret, we transform it into a more intuitive verb-based form using the LLM. All these components are then combined into a structured prompt, compensating for the language model's lack of direct access to the structural and relational information in the KG. All of these steps are detailed in the following subsections and depicted in Figure 2.

Step 2: Link Prediction. We evaluate the effectiveness of our framework on the Link Prediction task. We use the constructed prompt as the input for a finetuned LLM.

Link prediction focuses on inferring missing entities in triples of the form (h,r,?) or (?,r,t), where the goal is to predict the missing tail t or head h, respectively. In this work, we focus on tail prediction, noting that the two cases are structurally symmetric.

3.3. Prompt Generation

Following we detail the steps of the proposed *PrO-KGC* approach to generate a clear and informative prompt for the model.

Composition Pattern Extraction (CPE) To provide the model with a structured subgraph containing relevant facts that support the inference of implicit knowledge, we extract all possible continuous paths of length l using a depth-first search (DFS) algorithm, as detailed in Algorithm 1. The hyperparameter l determines the maximum allowed path length and can be set empirically. A multi-hop path of length l consists of a sequence of l interconnected nodes and relations, where each consecutive pair of nodes is linked by a relation. These paths connect a subject to an object and are grounded in actual entities and facts from the knowledge graph.

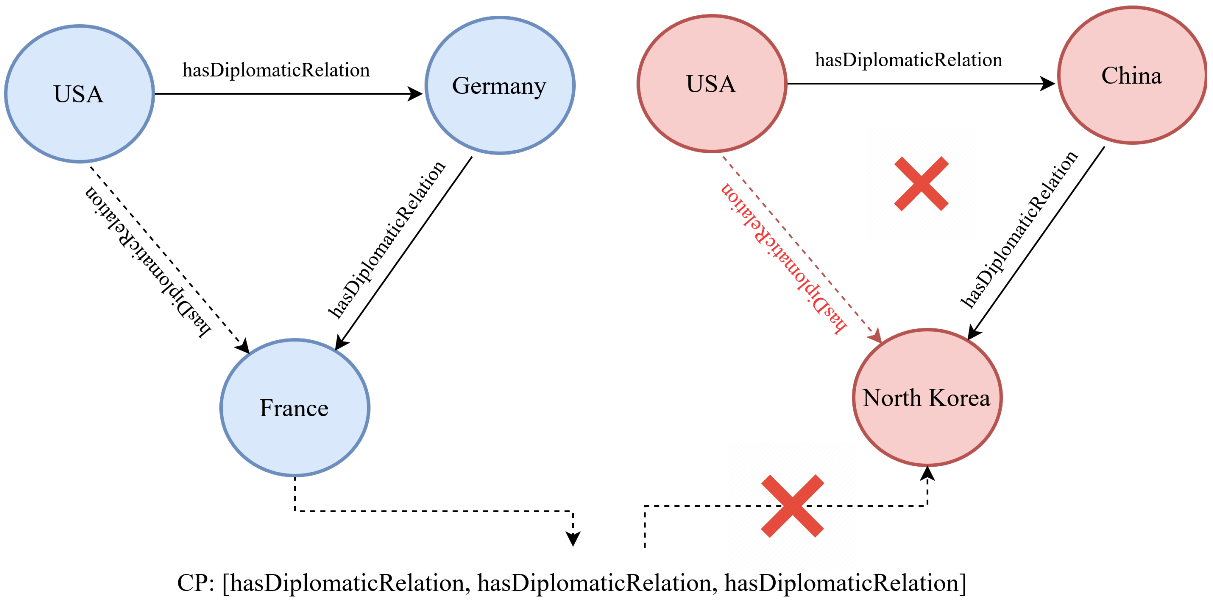

Figure 3: An example of a misleading composition pattern extracted from valid multi-hop paths

A composition pattern (CP), on the other hand, is defined as the ordered sequence of relations that interconnect the nodes within a multi-hop path. It represents an abstract relational template derived from the structure of the graph. However, not all extracted composition patterns are semantically valid or reliable. For example, if the USA has a diplomatic relation with Germany, and Germany has a diplomatic relation with France, then it might be reasonable to assume that the USA also has a diplomatic relation with France (which is also a valid fact). In this case, the algorithm would extract the composition pattern (hasDiplomaticRelation, hasDiplomaticRelation, hasDiplomaticRelation) as depicted in Figure 3. The same pattern could misleadingly suggest that the USA has a diplomatic relation with North Korea, given that the USA has a diplomatic relation with China, and China has a diplomatic relation with North Korea.

Since our goal is to improve the quality of the prompt rather than introduce misleading information, we employ an LLM to validate and filter out erroneous patterns that could negatively impact the model's reasoning. The prompt used is reported in Appendix A.1.

Descriptions Generation (DG) The head entity's description plays a crucial role in disambiguating entity types and names, ensuring clarity and reducing ambiguity. It also allows the Large Language Model (LLM) to retrieve and reinforce relevant knowledge about the entity from its internal knowledge base. Modern knowledge graphs often include descriptions for entities, but some may be missing. Given that LLMs have been trained on extensive knowledge bases and hence have broad general knowledge, we leverage them as an external resource to generate missing descriptions. We address cases where an entity ENT lacks a description by prompting our backbone model with the following instruction. If an entity is not recognized, we instruct the model to return a blank response to reduce post-processing efforts and limit the risk of hallucinations:

Prompt:
You are a knowledge graph. Provide a concise description of the given entity. Return only the description without any additional text. If the entity is unknown, return a blank response: {ENT}

Algorithm 1 Extraction of Composition Patterns from a Knowledge Graph

```
1: Initialize pattern dictionary $P \leftarrow \emptyset$ to store relation sequences and examples.
2: For each entity node $n \in G$, perform DFS to explore paths of length $l$.
3: If a valid path is found:
   Let the path be: $(e_1, r_1, e_2), (e_2, r_2, e_3), ..., (e_{l-1}, r_{l-1}, e_l)$
   Check if a direct relation exists between $e_1$ and $e_l$: $(e_1, r_f, e_l) \in G$
if $r_f$ exists then
      Form pattern: $(r_1, r_2, ..., r_{l-1} \rightarrow r_f)$
      Store two examples of supporting facts:
         Observed: $(e_i, r_i, e_{i+1})$ for $i = 1, ..., l-1$
         Inferred: $(e_1, r_f, e_l)$
end if
4: Repeat for all nodes.
5: Until no more paths exist.
6: Validate extracted patterns using an LLM:
   $P_{\text{validated}} \leftarrow \text{LLM}(P)$
7: Return $P_{\text{validated}}$.
```

Relation Transformation (RT) In some datasets, relations are expressed in a structured or a non-natural language format, making them difficult to interpret and incorporate into an LLM-friendly prompt. For example in the Freebase-extracted FB15K-237 dataset [16], relations are represented as a chain of concatenated categories like: */sports/sports_position/players./sports/sports_team_roster/position* which could be simplified to *plays as*. To ensure prompt readability and clarity, we introduce a step to simplify relation representation and semantics.

This is achieved by instructing our backbone LLM with the following prompt:

Prompt:
You will get a relation from the Freebase knowledge graph along with an example. Your task is to convert the relation into a simple verb form while preserving its meaning and key terms. Return only the converted relation. Example: Input: $person/born\_in\_city$ Output: $born\ in$ Relation: {REL}

Contextual Facts Retrieval (CFR) In this step, we leverage the composition patterns extracted and validated in the first stage. If the relation in the given triplet matches one of these patterns, we retrieve the associated nodes from the graph following the same relational structure. For more details refer to the algorithm in Appendix A.3. When no matching pattern is found, we instead retrieve a neighboring fact that shares the same relation. In such cases, neighboring nodes often reveal similar properties, which can support inference. If neither approach is applicable, we include a fact containing the same relation to provide contextual hints about entity categories to reduce ambiguity. It's worth noting that during the training phase, we strictly limit the CFR module to access only the training and validation splits to prevent any risk of data leakage.

4. Experiments

In this section, we present the experimental setup, including datasets, evaluation metrics, and results.

Table 1
Statistics of the datasets used in our experiments.

Dataset	# Ent	# Rel	# Train	# Valid	# Test
WN18RR	40,943	11	86,835	3,034	3,134
FB15k-237	14,541	237	272,115	17,535	20,466
CoDEx-m	17,050	51	185,584	10,310	10,310

4.1. Setups

Datasets We evaluate our approach on three widely used benchmark datasets: FB15k-237 by [16], an improved iteration of FB15k [16] that removes inverse relations to prevent data leakage. WN18RR [17] is as well a refined version of WN18 [16], a dataset based on WordNet. And CoDEx-M [18] which is extracted from Wikidata and Wikipedia. Detailed statistics of these datasets are provided in Table 1.

Baselines To assess the impact of our framework's components on KGC tasks, particularly link prediction, we establish a baseline using the vanilla model (i.e., before integrating our framework enhancements). Additionally, we benchmark our approach against leading triple-based models, including TuckER [14], TransE [1], and ComplEx[19], as well as the earliest implemented LLM-based methods: KG-BERT [4], KGT5[5], and GenKGC[20].

Implementation Details We conduct our experiments using two different GPUs: an NVIDIA A100 (80GB) and an NVIDIA RTX A6000 (50GB).

For the knowledge base LLM in the **DG** module, we use LLaMA 3.1 70B[1]. In the **CPE** filtering step and **RT** module, we employ GPT-4-Turbo via the OpenAI API[2], with a temperature of 1 and top-p set to 1. The size of composition patterns is set to 3. For the link prediction task, we use two different models: T5-Large[3] from [21]. and LLaMA 3 (8B) from Unsloth[4], which enables 2× faster fine-tuning, 70% lower memory usage, and 2× faster inference.

To optimize GPU memory usage and efficiency, we apply LoRA [22] and QLoRA [23] in the fine-tuning process. As decoding techniques, we use top-k sampling (`top_k=50`) and beam search (`num_beams=10`) to get the top-10 predictions. The remaining hyperparameters are reported in Appendix A, Table 5.

Metrics To evaluate the link prediction task, we use standard evaluation metrics: Hits@k (for $k \in \{1, 3, 10\}$), which measures the proportion of cases where the correct entity appears in the top-k predictions. Additionally, we use the Mean Reciprocal Rank (MRR), which computes the average of the inverse ranks of the correct predictions.

4.2. Main Results

The main results of our experiments are reported in Table 2. We observe that our enhanced model competes with PLM-based models; however, KGT5 continues to outperform on WN18RR. This can be explained by the fact that KGT5 was trained exclusively on KGC benchmarks. Pretrained weights were discarded and the objective function was replaced with a link prediction one. This probably reduced potential hallucinations, as the model encounters entity names for the first time within the KG. In

[1]LLaMA3.1 70B
[2]OpenAI API
[3]T5-large Checkpoint
[4]Unsloth

Table 2
Comparison between our approach and some existing methods on three different benchmarks. Results on CoDEx are copied from [18], † denotes that results are copied from [24]. Other results are taken from the original papers.

Model	FB15k-237				WN18RR				CoDEx			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
Triple-based Models												
TransE [1] †	–	0.198	0.376	0.441	–	0.043	0.441	0.532	0.303	0.223	0.3363	0.454
TuckER [14]	0.358	0.266	0.394	0.544	0.470	0.443	0.482	0.526	0.328	0.259	0.3599	0.458
ComplEx [19] †	–	0.194	0.297	0.450	–	0.409	0.469	0.530	0.337	0.262	0.3701	0.476
PLM-based Models												
KG-BERT [4]	–	–	–	0.420	–	0.041	0.302	0.524	–	–	–	–
KGT5 [5]	0.276	0.210	–	0.414	0.508	0.487	–	0.544	–	–	–	–
GenKGC [20]	–	0.192	0.355	0.439	–	0.287	0.403	0.535	–	–	–	–
Our Models												
Vanilla Model	0.271	0.227	0.294	0.382	0.304	0.256	0.341	0.406	0.359	0.293	0.398	0.514
+PrO-KGC	0.293	0.241	0.317	0.403	0.400	0.353	0.432	0.500	0.415	0.331	0.434	0.541

contrast, LLMs, which were initially pretrained on large text corpora, may show biases toward the expected answer format from their pretraining. Particularily, relations and taxonomy in WN18RR might differ from those in the generic pretraining datasets used for LLMs. For example: according to the WordNet Glossary[5], the relation "derivationally related form" refers to terms in different syntactic categories that share the same root form and are semantically related. However, in WN18RR, we find entities connected through this relation which are semantically related but do not share the same root. e.g: *("cover", "derivationally related form", "spread over")*

Regarding triple-based models, we observe that our model outperforms them on CoDEx with a 6.9% Hits@1 improvement, while TuckER achieves better results on WN18RR and FB15k-237, which might be explained by the inherent nature of these datasets. CoDEx, based on Wikidata, primarily contains binary relations, unlike FB15k-237, which is derived from Freebase, where relations are n-nary and require an understanding of the KG structure [18], such as */media_common/netflix_genre/titles.* Deterministic binary relations help LLMs in making more accurate predictions.

According to [25], WN18RR is the most structured KG among these datasets, which could explain the strong performance of triple-based models on it.

Our overarching goal is to improve LLM-based KGC performance. The results show that our approach outperforms the vanilla models across all three benchmarks, with a notable 9.7% improvement in Hits@1 on WN18RR, 3.8% on CoDEx, and a slight enhancement on FB15k-237 of 1.4% in Hits@1 and 2.1% in Hits@10. This confirms our hypothesis that enriching the prompt with structured knowledge enhances LLM-based KGC performance.

4.3. Ablation Study

To gauge the impact of the different components on the model's performance, we conduct an ablation study. We start with the vanilla model where we only feed the model with the query including the head entity and the relation and we instruct the model to predict the tail entity. Then, we gradually add the different components starting with the Description Generation (DG) component and then the Contextual facts Retrieval (CFR) component. The results of this study are reported in Table 3 and show that the more components we add the better the model's performance gets.

In fact, the incorporation of textual descriptions in CoDEx boosts performance by 3.4% in Hits@10, while the addition of structural information further increases this improvement to 4.7%. On WN18RR, we observe a 9.4% increase in Hits@10 with the addition of contextual facts, comparing the vanilla

[5]WordNet Glossary

model to the improved version. As aforementioned, WN18RR is a typical structured network, which probably contributes to this boost. For FB15k-237, however, incorporating descriptions does not significantly enhance performance, while adding contextual facts improves Hits@10 by 2.1%. One possible explanation for the limited impact of descriptions is their length: 141.7 words on average per description [26]. These lengthy descriptions may contain too much information, which may cause important facts to lose their relevance.

Table 3
Ablation study results (MRR, Hits@1, Hits@3, Hits@10) on different datasets.

Model Variant	**FB15k-237**				**WN18RR**				**CoDEx**			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
+PrO-KGC (Full Model)	0.293	0.241	0.317	0.403	0.400	0.353	0.432	0.500	0.415	0.331	0.434	0.561
w/ Desc, w/o CP	0.272	0.226	0.290	0.390	0.341	0.293	0.375	0.441	0.402	0.307	0.418	0.548
w/o CP and DG (Vanilla)	0.271	0.227	0.294	0.382	0.304	0.256	0.341	0.406	0.359	0.293	0.398	0.514

To better understand the influence of multi-hop decision paths on model performance, we measure accuracy on a subset of CoDEx test triples that are supported by such paths, and compare it to the overall performance on the full test dataset. As shown in Table 4, the performance is significantly better on this subset, with Hits@1 and Hits@3 increased by 14.5% and 15.9%, respectively. These results indicate that the presence of structured decision paths plays a significant role in guiding the model's predictions.

Table 4
Performance comparison of the full model on the full CoDEx test set vs. a subset with supporting decision paths.

Setting	**Hits@1**	**Hits@3**	**Hits@10**	**MRR**
Full Test Set	0.331	0.434	0.561	0.415
Subset w/ Decision Paths (2,343 triples; 22.7% of the test set)	0.476	0.593	0.704	0.548

4.4. Model Comparison

In this subsection, we compare the performance of the different models fine-tuned for the link prediction task based on two key dimensions: Accuracy and Efficiency, as depicted in Figure 4. The models used are, as aforementioned, T5-large and LLaMA 3 8B, with only a small percentage of their parameters fine-tuned: 4.7M (0.64%) for T5 and 42M (0.92%) for LLaMA.

We observe that, overall, LLaMA outperforms T5 in terms of accuracy and is more efficient in training time for WN18RR and CoDEx. However, on FB15k-237, T5 is 70% faster (21 vs. 29 hours). One possible explanation for this difference is the nature of the dataset and the complexity of its n-nary relations. In this case, decoder-only models like LLaMA may struggle with complex multi-entity relationships, requiring more updates to learn meaningful patterns.

Regarding performance, LLaMA has more parameters and was trained on larger datasets, which could explain its superior results, particularly in Hits@1. For example, on CoDEx, T5 achieves 0.212, whereas LLaMA reaches 0.331, an 11.9% improvement. However, in terms of Hits@10, the T5-based model outperforms LLaMA (0.584 vs. 0.561, a 2.3% difference). One possible reason for this is the difference in decoding methods; for LLaMA, beam search is not available in the used library, so sampling was employed instead.

In terms of the impact of additional prompt components, we observe that LLaMA's performance on CoDEx and WN18RR significantly decreases when prompt components are ablated (9.4% and 4.7% reductions, respectively), whereas T5 remains relatively stable (2.4% and 2.5% reductions). One possible explanation is T5's 512-token context length limit, prompts exceeding this limit are truncated, which could lead to information loss.

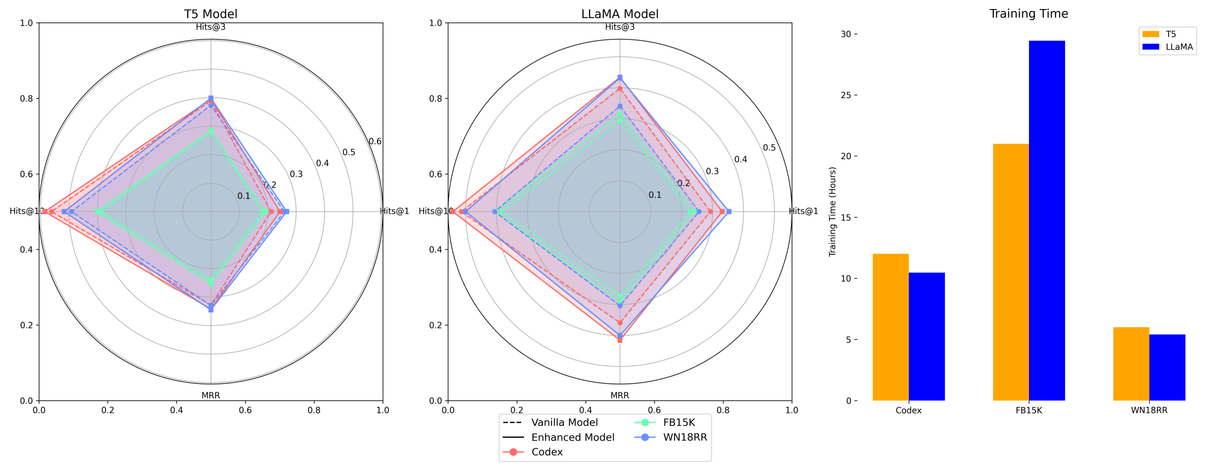

Figure 4: Performance and training time comparison of T5 and LLaMA for link prediction. The bar chart on the right illustrates the training time required by both models across three different datasets. The radar chart on the left presents performance metrics for various model variants on the same datasets.

4.5. Limitations

In this section, we discuss the major limitations of our work from our perspective.

Firstly, we believe that applying the framework to other Knowledge Graph Completion (KGC) tasks beyond link prediction, such as triple classification and relation prediction, may reveal additional insights and further validate the effectiveness of our approach.

For composition patterns, the length l was fixed at 3, but we believe that longer paths could lead to greater improvements, particularly for multi-hop relations. Additionally, composition patterns cannot be applied to all samples; only a small percentage of queries can leverage them (e.g., in the FB15K-237 test set, only 10% of queries benefit from contextual paths), which may obscure their true impact. A potential solution could be utilizing an LLM to generate decision paths when compositions are unavailable.

Another limitation is that the reported results in Table 2 were not extracted under the same settings, which is not ideal for a fair comparison. Additionally, the extraction of contextual facts could benefit from refinements, such as selecting the most semantically similar facts to the query.

A further arguable point is the limited improvement of our framework on the FB15k-237 dataset and the overall performance lagging behind knowledge graph embeddings. This could be attributed to the reduction of trainable parameters to 42M (0.92%), which may limit the LLM's ability to memorize newly learned facts or patterns. This reinforces the idea that LLMs alone cannot fully replace KGC models but should rather be optimized to complement KGE models.

5. Conclusion

In this work, we introduced *PrO-KGC*, a framework for prompt optimization in LLM-based Knowledge Graph Completion (KGC). *PrO-KGC* follows a structured approach to extracting and integrating structural context from the KG into prompts, leveraging LLMs to bridge the gap between structured knowledge and LLM capabilities.

We first extract composition patterns, validate them using a knowledge base LLM, and harness them to support link prediction queries with relevant subgraphs. In addition, we generate new descriptions when needed and refine relation representations to make the prompt more natural, language-friendly, and semantically rich.

Through a set of experiments, we demonstrated the effectiveness of our approach across three different benchmarks. However, we also conclude that LLMs are still far from replacing embedding models; rather, they should be used jointly. Our work contributes to fostering a more effective synergy between LLMs and Knowledge Graph Embedding (KGE) models.

For future work, we plan to assess the impact of varying composition pattern lengths on model performance and explore transforming the *PrO-KGC* textual input into embeddings as an initial input for embedding-based models. Another promising direction is evaluating our approach on multiple KGC tasks and investigating new strategies to further optimize prompt design.

6. Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT in order to: spelling check, paraphrase and reword. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, Advances in neural information processing systems 26 (2013).

[2] Z. Sun, Z.-H. Deng, J.-Y. Nie, J. Tang, Rotate: Knowledge graph embedding by relational rotation in complex space, arXiv preprint arXiv:1902.10197 (2019).

[3] L. Wang, W. Zhao, Z. Wei, J. Liu, Simkgc: Simple contrastive knowledge graph completion with pre-trained language models, arXiv preprint arXiv:2203.02167 (2022).

[4] L. Yao, C. Mao, Y. Luo, KG-BERT: BERT for knowledge graph completion, CoRR abs/1909.03193 (2019). URL: http://arxiv.org/abs/1909.03193. arXiv:1909.03193.

[5] A. Saxena, A. Kochsiek, R. Gemulla, Sequence-to-sequence knowledge graph completion and question answering, 2022. URL: https://arxiv.org/abs/2203.10321. arXiv:2203.10321.

[6] X. Wang, T. Gao, Z. Zhu, Z. Zhang, Z. Liu, J. Li, J. Tang, Kepler: A unified model for knowledge embedding and pre-trained language representation, 2020. URL: https://arxiv.org/abs/1911.06136. arXiv:1911.06136.

[7] Y. Wei, Q. Huang, Y. Zhang, J. Kwok, Kicgpt: Large language model with knowledge in context for knowledge graph completion, in: Findings of the Association for Computational Linguistics: EMNLP 2023, Association for Computational Linguistics, 2023, p. 8667–8683. URL: http://dx.doi.org/10.18653/v1/2023.findings-emnlp.580. doi:10.18653/v1/2023.findings-emnlp.580.

[8] S. Ji, L. Liu, J. Xi, X. Zhang, X. Li, Klr-kgc: Knowledge-guided llm reasoning for knowledge graph completion, Electronics 13 (2024). URL: https://www.mdpi.com/2079-9292/13/24/5037. doi:10.3390/electronics13245037.

[9] M. Li, C. Yang, C. Xu, X. Jiang, Y. Qi, J. Guo, H. fung Leung, I. King, Retrieval, reasoning, re-ranking: A context-enriched framework for knowledge graph completion, 2024. URL: https://arxiv.org/abs/2411.08165. arXiv:2411.08165.

[10] Y. Liu, X. Tian, Z. Sun, W. Hu, Finetuning generative large language models with discrimination instructions for knowledge graph completion, 2024. URL: https://arxiv.org/abs/2407.16127. arXiv:2407.16127.

[11] D. Xu, Z. Zhang, Z. Lin, X. Wu, Z. Zhu, T. Xu, X. Zhao, Y. Zheng, E. Chen, Multi-perspective improvement of knowledge graph completion with large language models, arXiv preprint arXiv:2403.01972 (2024).

[12] Q. Li, Z. Chen, C. Ji, S. Jiang, J. Li, Llm-based multi-level knowledge generation for few-shot knowledge graph completion, in: Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, volume 271494703, 2024.

[13] J. Huang, K. C.-C. Chang, Towards reasoning in large language models: A survey, 2023. URL: https://arxiv.org/abs/2212.10403. arXiv:2212.10403.

[14] I. Balažević, C. Allen, T. M. Hospedales, Tucker: Tensor factorization for knowledge graph completion, arXiv preprint arXiv:1901.09590 (2019).
[15] D. Shu, T. Chen, M. Jin, C. Zhang, M. Du, Y. Zhang, Knowledge graph large language model (kg-llm) for link prediction, arXiv preprint arXiv:2403.07311 (2024).
[16] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: C. Burges, L. Bottou, M. Welling, Z. Ghahramani, K. Weinberger (Eds.), Advances in Neural Information Processing Systems, volume 26, Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper_files/paper/2013/file/1cecc7a77928ca8133fa24680a88d2f9-Paper.pdf.
[17] T. Dettmers, P. Minervini, P. Stenetorp, S. Riedel, Convolutional 2d knowledge graph embeddings, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence, AAAI'18/IAAI'18/EAAI'18, AAAI Press, 2018.
[18] T. Safavi, D. Koutra, CoDEx: A Comprehensive Knowledge Graph Completion Benchmark, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Online, 2020, pp. 8328–8350. URL: https://www.aclweb.org/anthology/2020.emnlp-main.669. doi:10.18653/v1/2020.emnlp-main.669.
[19] T. Trouillon, C. R. Dance, J. Welbl, S. Riedel, Éric Gaussier, G. Bouchard, Knowledge graph completion via complex tensor factorization, 2017. URL: https://arxiv.org/abs/1702.06879. arXiv:1702.06879.
[20] X. Xie, N. Zhang, Z. Li, S. Deng, H. Chen, F. Xiong, M. Chen, H. Chen, From discrimination to generation: Knowledge graph completion with generative transformer, in: Companion Proceedings of the Web Conference 2022, WWW '22, ACM, 2022. URL: http://dx.doi.org/10.1145/3487553.3524238. doi:10.1145/3487553.3524238.
[21] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer, 2023. URL: https://arxiv.org/abs/1910.10683. arXiv:1910.10683.
[22] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, Lora: Low-rank adaptation of large language models, 2021. URL: https://arxiv.org/abs/2106.09685. arXiv:2106.09685.
[23] T. Dettmers, A. Pagnoni, A. Holtzman, L. Zettlemoyer, Qlora: Efficient finetuning of quantized llms, 2023. URL: https://arxiv.org/abs/2305.14314. arXiv:2305.14314.
[24] D. Nathani, J. Chauhan, C. Sharma, M. Kaul, Learning attention-based embeddings for relation prediction in knowledge graphs, in: A. Korhonen, D. Traum, L. Màrquez (Eds.), Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Florence, Italy, 2019, pp. 4710–4723. URL: https://aclanthology.org/P19-1466/. doi:10.18653/v1/P19-1466.
[25] S. Faralli, A. Lenzi, P. Velardi, A benchmark study on knowledge graphs enrichment and pruning methods in the presence of noisy relationships, J. Artif. Int. Res. 78 (2024). URL: https://doi.org/10.1613/jair.1.14494. doi:10.1613/jair.1.14494.
[26] X. Wang, Q. He, J. Liang, Y. Xiao, Language models as knowledge embeddings, 2023. URL: https://arxiv.org/abs/2206.12617. arXiv:2206.12617.

A. Appendix

A.1. Prompt used for Pattern Validation

Pattern Validation Prompt

Instruction: Given a relation pattern (`r1, r2, r3`) and two examples of paths using these relations, determine whether this represents a **valid** or **invalid** composition pattern.
A valid composition pattern means that `Fact1` and `Fact2` necessarily imply `Fact3`. If this implication does not universally hold or if there are exceptions, classify it as **invalid**.
Your response should be in the following format: `Valid/Invalid. Explanation:`
Pattern: `(diplomatic relation, member of, member of)`
Example 1:
```
Fact1: (Senegal, diplomatic relation, Germany)
Fact2: (Germany, member of, International Telecommunication Union)
Fact3: (Senegal, member of, International Telecommunication Union)
```
Example 2:
```
Fact1: (Senegal, diplomatic relation, Germany)
Fact2: (Germany, member of, African Development Bank)
Fact3: (Senegal, member of, African Development Bank)
```
Response: `Invalid.` **Explanation:** If a country has a diplomatic relation with another that is a member of an organization, it does not necessarily imply that it is also part of that organization.

A.2. Hyperparameters of Models used for Link Prediction

Table 5
Hyperparameters of finetuned models

Hyperparameter	**LLaMA**	**T5**
Maximum Sequence Length	2048	512
Learning Rate	2e-4	1e-3
Batch Size	4	16
Optimizer	paged_adamw_8bit	AdamW
Gradient Accumulation Steps	4	8
LoRA Rank	16	16
LoRA Alpha	16	8
LoRA Dropout	0	0.05
Number of epochs	3	3
Gradient Accumulation Steps	4	–

A.3. Extraction of supporting facts from the KG

Algorithm 2 Extract Supporting Facts from a Graph

```
Input: A query $(h, r, ?)$, a set of composition patterns $patterns$, and a knowledge graph $G$.
Output: A supporting triplet or triplet pair.
Identify patterns where relation $r$ appears.
for each matching pattern do
    Determine the position of $r$ in the pattern.
    if $r$ appears as the last relation then
        Retrieve the preceding relations $(r_1, r_2)$: $(r_1, r_2, r) \leftarrow patterns$.
        Find entities $n_1$ connected to $h$ via $r_1$ in $G$.
        for each $n_1$ do
            Store $(h, r_1, n_1)$ as a potential supporting fact.
            Check if $n_1$ is linked to another entity $n_2$ via $r_2$.
            if $(n_1, r_2, n_2)$ exists in $G$ then
                Store the pair $((h, r_1, n_1), (n_1, r_2, n_2))$.
            end if
        end for
    end if
end for
if $\exists (h, r_1, n_1), (n_1, r_2, n_2) \in G$ then return randomly chosen $((h, r_1, n_1), (n_1, r_2, n_2))$.
else if $\exists (h, r_1, n_1) \in G$ then return randomly chosen $(h, r_1, n_1)$.
elsereturn None.
end if
```