INJECTING KNOWLEDGE GRAPHS
INTO LARGE LANGUAGE MODELS

A PREPRINT

● Erica Coppolillo
University of Calabria
ICAR-CNR
erica.coppolillo@unical.it

ABSTRACT

Integrating structured knowledge from Knowledge Graphs (KGs) into Large Language Models (LLMs) remains a key challenge for symbolic reasoning. Existing methods mainly rely on prompt engineering or fine-tuning, which lose structural fidelity or incur high computational costs. Building on recent encoding techniques which integrate graph embeddings within the LLM input as tokens, we extend this paradigm to the KG domain by leveraging Knowledge Graph Embedding (KGE) models, thus enabling graph-aware reasoning. Our approach is model-agnostic, resource-efficient, and compatible with any LLMs. Extensive experimentation on synthetic and real-world datasets shows that our method improves reasoning performance over established baselines, further achieving the best trade-off in terms of accuracy and efficiency against state-of-the-art LLMs.

Keywords Knowledge Graphs · Large Language Models · Question Answering

1 Introduction

Large Language Models (LLMs) have demonstrated impressive performance across a range of natural language tasks, including machine translation [Gain, Bandyopadhyay, and Ekbal, 2025], question answering [Yue, 2025], summarization [Zhang, Yu, and Zhang, 2025], and dialogue generation [Chen et al., 2024]. However, they suffer from limitations such as hallucinations, lack of explicit memory, and difficulty in handling *structured knowledge*, such as the relational information encoded in Knowledge Graphs (KGs) [Ji et al., 2021]. Knowledge Graphs represent facts as networks of entities and relations, offering a form of symbolic knowledge that enables logical inference and precise factual retrieval.

Integrating the rich structured context of KGs into the generative reasoning abilities of LLMs is a promising direction, yet achieving effective integration remains challenging. Indeed, most existing approaches rely on *prompting* strategies [Biran and others, 2024, Chen and others, 2024, Wu and others, 2024], where graph facts are serialized into text and used as context. While simple to implement, these methods often lose relational structure and require careful template engineering. Other strategies [Zhao et al., 2023] use dedicated reasoning modules to traverse KGs and then guide the LLM through chain-of-thought prompting. Fine-tuning-based methods [Sun and others, 2023] adapt the LLM on graph-specific tasks but incur high computational costs and risk overfitting.

We propose a novel alternative: **injecting structured knowledge graph representations directly into a frozen LLM**. Our approach builds on recent work [Perozzi et al., 2024] which introduces GraphToken, a token-based encoding scheme for generic graphs. We extend this idea to the *knowledge graph* setting by leveraging Knowledge Graph Embedding (KGE) models to encode entities and relations as vectors. These vectors are transformed into structured tokens and injected into the LLM input alongside the natural language query.

Importantly, our method does not require any LLM fine-tuning or prompt engineering, is model-agnostic, and resource-efficient. The structured tokens are generated at inference time using trained KGE models and do not involve any weight updates to the LLM. As such, the technique can be integrated with any LLM that accepts input sequences.

We evaluate our method on both synthetic tasks and real-world datasets from the graph learning literature, providing a diverse testbed for our experimental evaluation.

Our main contributions are the following:

- We extend the GraphToken framework [Perozzi et al., 2024] to the knowledge graph domain, enabling structured injection of KG information into LLMs.
- Our method eliminates the need for LLM training or prompt engineering, operating directly with frozen LLMs. The only trainable component is the KGE model, ensuring efficiency and resource savings.
- We demonstrate the effectiveness and generalizability of the framework across both synthetic and real-world KG datasets and against several competitors, using a lightweight LLM as the underlying backbone.

The remainder of the paper is organized as follows: Section 2 reviews related work on enhancing LLM reasoning over graphs. Section 3 introduces our proposed methodology, followed by experimental setup in Section 4. The results are provided in Section 5. We conclude in Section 6, where we discuss limitations and future directions.

2   Related Work

KGQA   Question Answering over Knowledge Graphs (KGQA) has evolved significantly, transitioning from early rule-based systems to sophisticated models integrating deep learning and large language models (LLMs). Initial approaches focused on semantic parsing techniques to convert natural language questions into structured queries over knowledge graphs [Berant et al., 2013, Yih et al., 2015].

With the rise of neural networks, embedding-based methods gained prominence. Bordes et al. [Bordes et al., 2014] introduced a model that learns low-dimensional embeddings for entities and relations, facilitating question answering through vector space representations. Subsequent works like GraftNet [Sun et al., 2018] and PullNet [Sun et al., 2019a] employed graph neural networks to capture multi-hop reasoning paths within knowledge graphs.

LLMs for KG Reasoning   Recent advancements have seen the integration of LLMs with knowledge graphs to enhance reasoning capabilities. For instance, QA-GNN [Yasunaga et al., 2021] combines language models with graph neural networks to jointly reason over textual and structured data. Similarly, approaches like R3 [Toroghi et al., 2024] aim to ground LLM outputs in knowledge graphs, ensuring verifiable and factually accurate answers. Indeed, a common strategy is to *prompt* the LLM with KG-derived context. For example, Huang et al. [2023] showed that including relevant triples in prompts improves factual QA on long-tail queries. Other works use structured prompts or reasoning traces to further guide the model. Zhao et al. [2023] proposed KG-CoT, a method that traverses the KG to generate a step-by-step reasoning path, which is then provided to the LLM as a chain-of-thought prompt. Such methods have demonstrated improved performance on multi-hop and open-domain KGQA tasks without fine-tuning the LLM.

Graph Encodings for LLMs   Beyond prompting, another line of work explores how to encode graph structure into LLM-consumable formats. Fatemi, Halcrow, and Perozzi [2024] and Liu, Fatemi, and others [2024] investigated how different graph-to-text serializations affect reasoning performance, revealing that appropriate encodings can yield large accuracy gains. Other methods, such as soft prompts [Liu et al., 2023] represent the graph in vector form and append this to the LLM input. While effective, these methods often require either extensive prompt design or additional model training.

Our work departs from the above paradigms by injecting knowledge graphs into an LLM through structured embeddings, without any LLM training or prompt engineering. Extending the work of Perozzi et al. [2024] from generic, unlabeled graphs to KGs, we leverage Knowledge Graph Embedding (KGE) models to convert entities and relations into vector representations, which are then supplied to a frozen LLM. To the best of our knowledge, this is the first method to enable KG reasoning in LLMs using structured injection from KGE models.

3   Methodology

Reasoning Tasks   To evaluate the reasoning capabilities of the model, we consider the following node reasoning tasks: Existence (**E**), Counting (**C**), and Identification (**I**), progressively increasing the complexity of the query by supporting **0-Hop**, **1-Hop**, and **2-Hop** neighborhood. Specifically, we examine the following query categories:

- 0-Hop, Single Label (**SL**): query involves labeled nodes.

Table 1: Template of the considered queries for the identification task, embracing 0-, 1-, and 2-hop neighborhood questions, and Single, Double, and Triplet Label type.

Hop	Type	Template Query
0	SL	Which nodes are `[LABEL]`s in the graph?
1	SL	Which nodes have a `[LABEL]` as neighbor?
	DL	Which nodes are `[LABEL1]`s and have a `[LABEL2]` as neighbor?
2	SL	Which nodes have a neighbor who `[PROPERTY]`s `[LABEL]`s?
	DL	Which nodes are `[LABEL1]`s and have a neighbor who `[PROPERTY]`s `[LABEL2]`s?
	TL	Which nodes are `[LABEL1]`s and have a `[LABEL2]` as neighbor who `[PROPERTY]`s `[LABEL3]`s?

- 1-Hop, Single Label: query involves nodes having labeled neighbors.
- 1-Hop, Double Label (**DL**): query involves labeled nodes having labeled neighbors.
- 2-Hop, Single Label: query involves nodes having neighbors with a property regarding other labeled nodes.
- 2-Hop, Double Label: query involves labeled nodes having neighbors with a property regarding other labeled nodes.
- 2-Hop, Triple Label (**TL**): query involves labeled nodes having labeled neighbors with a property regarding other labeled nodes.

To ensure comprehension, Table 1 provides the query template for each considered category on the identification task. The instantiation is equivalent to the existence and counting tasks, trivially reformulating the query with "Are there . . . ?" and "How many . . . ?", respectively, instead of "Which . . . ?".

GraphToken   We here recall some key concepts of GraphToken, the encoding method introduced by Perozzi et al. [2024], to encode graph structures into continuous vector representations, which we extend to knowledge graphs. The GraphToken framework mainly consists of a Graph Encoder, which utilizes a Graph Neural Network (GNN) [Zhou et al., 2020] to process the input graph, capturing its structural information. Node features are derived using positional encodings, such as Laplacian eigenvectors or learned embeddings. For further technical details, please refer to the main paper.

KG Embedding   To compute the latent representation of the KG, we rely on Knowledge Graph Embedding (KGE) models [Ji et al., 2022], which aim to represent the entities and relations in a knowledge graph as low-dimensional vectors in a continuous vector space.

Given a set of entities $\mathcal{E}$ and relations $\mathcal{R}$, a knowledge graph $\mathcal{G}$ is typically represented as a set of triples $\mathcal{T} = \{(h, r, t)\}$, where h and t are the head and tail entities, and r is the relation.

Each entity $h, t \in \mathcal{E}$ and each relation $r \in \mathcal{R}$ is mapped to an embedding vector:

$$\mathbf{e}_h, \mathbf{e}_t, \mathbf{e}_r \in \mathbb{R}^d$$

A scoring function $f(h, r, t)$ is defined to measure the plausibility of a triple, where the form of f depends on the specific KGE model.

To train the model, a margin-based ranking loss is typically used [Wang et al., 2014, Ji et al., 2016, 2015, Bordes et al., 2013a, Lin et al., 2015]:

$$\mathcal{L} = \sum_{(h,r,t)\in\mathcal{T}} \sum_{(h',r,t')\in\mathcal{T}'} \max\big(0, \gamma + f(h', r, t') - f(h, r, t)\big) \tag{1}$$

where $\gamma > 0$ is the margin, and $\mathcal{T}'$ is the set of negative samples generated by corrupting either the head or tail entity.

The model parameters (entity and relation embeddings) are trained using stochastic gradient descent [Ruder, 2017] or variants (e.g., Adam [Kingma and Ba, 2017]). Embeddings are often normalized during training to enforce constraints (e.g., unit norm):

$$\mathbf{e}_h \leftarrow \frac{\mathbf{e}_h}{\|\mathbf{e}_h\|}, \quad \mathbf{e}_t \leftarrow \frac{\mathbf{e}_t}{\|\mathbf{e}_t\|}, \quad \mathbf{e}_r \leftarrow \frac{\mathbf{e}_r}{\|\mathbf{e}_r\|} \tag{2}$$

To our purpose, instead of aggregating the scalar scores as in Equation 1, we rewrite the aggregation function to preserve per-dimension contributions. More formally, let $\mathbf{x} \in \mathbb{R}^{n \times d}$ be a matrix representing n KG triplets with d features each. We define the new scoring vector $\mathbf{s} \in \mathbb{R}^d$ via the aggregation function $f'(\mathbf{x})$ as follows:

$$\mathbf{s} = f'(\mathbf{x}) = [f(\mathbf{x}_{:,1}), f(\mathbf{x}_{:,2}), \cdots, f(\mathbf{x}_{:,d})]. \tag{3}$$

Intuitively, in this formulation, while $f(\mathbf{x})$ performs row-wise aggregation across the feature dimension, producing a scalar per sample, $f'(\mathbf{x})$ performs column-wise aggregation across the sample dimension, yielding a single representative feature vector. This final vector can be interpreted as a dimension-wise decomposition of the graph's triples plausibility [Holtz, 2011].

Finally, to make this representation compatible with the token embedding space of the LLM, we apply a dense layer, i.e., a trainable linear projection:

$$\mathbf{g} = \mathbf{W}\mathbf{s} + \mathbf{b}, \quad \mathbf{g} \in \mathbb{R}^{d_{\text{LLM}}} \tag{4}$$

where $\mathbf{W} \in \mathbb{R}^{d_{\text{LLM}} \times d}$ and $\mathbf{b} \in \mathbb{R}^{d_{\text{LLM}}}$ are learnable parameters, and d_{LLM} is the dimensionality of the (frozen) LLM token embeddings.

The resulting vector $\mathbf{g}$ thus constitutes the final latent representation of $\mathcal{G}$.

Training Process   Similarly to the original work [Perozzi et al., 2024], the training dataset comprises pairs $(\mathcal{Q}, \mathcal{A})$, where $\mathcal{Q}$ includes the textual description of a knowledge graph $\mathcal{G}$ with the corresponding reasoning question; and $\mathcal{A}$ represents the ground truth answer.

During the forward pass, the final input encoding $\mathbf{t}$ is obtained by concatenating the tokenized representation of the query $\mathcal{Q}$ with the embedding of $\mathcal{G}$:

$$\mathbf{t} = \mathbf{q} \parallel \mathbf{g}$$

where $\mathbf{q}$ denotes the tokenized form of the query, and $\mathbf{g}$ is computed following Equation 4.

The objective is to minimize the negative log-likelihood loss [deAbril, Yoshimoto, and Doya, 2018] of the LLM's output conditioned on the augmented input $\mathbf{t}$:

$$\mathcal{L}(\mathcal{A} \mid \mathcal{Q}) = -\log \mathrm{P}_{\text{LLM}}(\mathcal{A} \mid \mathbf{t}) \tag{5}$$

During training, the parameters of the LLM are always kept **frozen**. Only the parameters of the underlying KGE model are updated, by back-propagating to minimize $\mathcal{L}(\mathcal{A} \mid \mathcal{Q})$. A sketch of the overall framework is depicted in Figure 1.

4   Experimental Setup

The experimental evaluation of our methodology aims at answering the following research questions:

RQ1: How does the proposed strategy compare on reasoning tasks against state-of-the-art methods and baselines?

RQ2: How do the results vary by changing the underlying KGE model?

RQ3: Is the final graph embedding effectively tailored to the given reasoning task?

Table 2: Knowledge graphs statistics from both synthetic and real-world datasets.

Dataset	#Graphs	Avg. #Nodes	Avg. #Edges
Synthetic	600	11.91	74.46
AIDS	2,000	15.59	32.39
MUTAG	187	18.03	39.8
AQSOL	9,833	17.6	35.8

Figure 1: Visual representation of the framework. During training, the KG embedding is concatenated to the token latent vectors of the frozen LLM, and the answer produced by the LLM is optimized.

Datasets   To assess the validity of our method, we perform the experiments by relying on both synthetic and real-world datasets, described as follows:

- **Synthetic**: We synthetically generate undirected, complete graphs of varying sizes, labeling nodes and edges with randomly chosen entities and relations, respectively.

- **AIDS**[1] [Riesen and Bunke, 2008, Morris et al., 2020]: The AIDS dataset consists of molecular graphs representing chemical compounds checked for evidence of anti-HIV activity. Nodes and edges represent chemical bonds and are provided with the corresponding features.

- **MUTAG**[2] [Debnath et al., 1991]: MUTAG is a dataset of nitroaromatic compounds, represented as graphs. Nodes are atoms (with labels such as C, N, O), and edges are chemical bonds. Each compound is labeled based on its mutagenic effect on a bacterium (Salmonella typhimurium).

- **AQSOL**[3] [Dwivedi et al., 2020]: AQSOL is a dataset of small molecules with their aqueous solubility values. Molecules are typically represented using SMILES strings and can be converted into molecular graphs.

Table 2 provides some statistics, including the number of graphs within each dataset, and the average number of nodes and edges.

KGE Models   In our experiments, we implement and adapt the following KGE models to compute the knowledge graph embedding:

- **TransE** [Bordes et al., 2013b]: Models relations as translations in the embedding space. For a triple (h, r, t), it enforces $\mathbf{e}_h + \mathbf{e}_r \approx \mathbf{e}_t$. The score function is $f(h, r, t) = -\|\mathbf{e}_h + \mathbf{e}_r - \mathbf{e}_t\|_p$, where p refers to the norm type used to compute the distance.

- **DistMult** [Yang et al., 2015]: Represents entities and relations as real-valued vectors and models interactions via element-wise multiplication. The score function is $f(h, r, t) = \langle \mathbf{e}_h, \mathbf{e}_r, \mathbf{e}_t \rangle = \sum_i e_h^i e_r^i e_t^i$.

---

[1]https://huggingface.co/datasets/graphs-datasets/AIDS
[2]https://huggingface.co/datasets/graphs-datasets/MUTAG
[3]https://huggingface.co/datasets/graphs-datasets/AQSOL

- **ComplEx** [Trouillon et al., 2016]: Extends DistMult to complex-valued embeddings, enabling the modeling of asymmetric relations. It uses the score function $f(h, r, t) = \Re(\langle \mathbf{e}_h, \mathbf{e}_r, \overline{\mathbf{e}}_t \rangle)$, where $\overline{\mathbf{e}}_t$ is the complex conjugate of $\mathbf{e}_t$, and $\Re$ denotes the real part.

- **RotatE** [Sun et al., 2019b]: Embeds entities in the complex space and represents relations as rotations. For each triple, it models $\mathbf{t} \approx \mathbf{h} \circ \mathbf{r}$, where $\circ$ is the element-wise complex multiplication and $\mathbf{r}$ lies on the unit circle. The score function is $f(h, r, t) = -\|\mathbf{e}_h \circ \mathbf{e}_r - \mathbf{e}_t\|$.

LLM   For the experiments, we adopt the 2B, fine-tuned version of Gemma [Team and et. al., 2024], a lightweight model based on Gemini[4], the LLMs family released by Google. However, we remark that the framework is independent from the underlying model since **any** LLM can be used.

Competitors   We compare our method against a comprehensive set of established baselines:

- **Zero-Shot**: The model receives only a task description and is immediately asked to generate the desired output, without any examples or demonstrations.

- **Few-Shot** [Brown et al., 2020]: The model is provided with a few example input-output pairs, embedded directly in the prompt. This enables the model to adapt during inference, without conventional training. In our experiments, the number of provided examples is equal to 2.

- **CoT** (**C**hain-**o**f-**T**hought) [Wei et al., 2022]: This prompting technique includes examples with step-by-step reasoning, encouraging the model to generate intermediate steps when solving new tasks.

- **Zero-CoT** [Kojima et al., 2022]: A variant of CoT prompting that requires no examples. Instead, it elicits reasoning by using a simple trigger phrase like "Let's think step by step".

- **CoT-BaG** [Wang et al., 2023]: An extension of CoT integrating **B**uilding-**a**-**G**raph prompting, which adds a statement to guide the model's reasoning on graph-related tasks.

- **Prompt Tuning** [Lester, Al-Rfou, and Constant, 2021]: A parameter-efficient fine-tuning technique where a small set of learnable embeddings (called *soft prompt*) is prepended to the model's input. Instead of updating the full model, only these prompt parameters are trained, allowing the LLM to adapt to new tasks with minimal computational cost.

Additionally, we test our reasoning framework against the following state-of-the-art LLMs, to evaluate their trade-off between accuracy and efficiency: **GPT-4o** [OpenAI, 2024], a powerful model produced by OpenAI in 2024, capable of processing and generating text, audio, and images; **o4-mini** [OpenAI, 2025], a compact, cost-efficient reasoning model released by OpenAI in 2025, and optimized for tasks in mathematics, coding, and visual understanding; and **DeepSeek-R1** [DeepSeek, 2025], a large-scale reasoning model developed by DeepSeek, trained via reinforcement learning for tasks in mathematics, coding, and reasoning.

Setting   For the experiments, we split the dataset into training, validation and test sets, using a $80/10/10$ ratio. We trained the models by performing early stopping on the validation set [Yao, Rosasco, and Caponnetto, 2007]. To optimize Equation 5, we adopt a Lion optimizer [Chen et al., 2023], with a learning rate equal to $10e-3$. Regarding the configuration of the KGE models, we set the hidden dimension to 64.

5 Results

Accuracy vs Efficiency   To address **RQ1**, we begin by validating our proposed method against the aforesaid established baselines. The results, summarized in Table 3, illustrate the performance across the considered datasets. In the table, the best-performing scores are highlighted in bold, while the percentage improvements are calculated relative to the second-best results.

Overall, our method consistently outperforms the competing approaches across all the examined reasoning tasks, with substantial improvement in most configurations. The highest accuracy is achieved on the existence task, which represents the simplest form of reasoning. Conversely, the counting and identification tasks present greater challenges, and here the performance tends to fluctuate depending on the specific dataset, reflecting the increasing complexity required for these tasks.

In addition to comparing against classical baselines, we further evaluate our method against state-of-the-art LLMs to understand the trade-off between accuracy and computational efficiency. This comparison is visualized in Figure 2,

---

[4]`https://deepmind.google/technologies/gemini/`

Table 3: Test accuracy over the considered datasets and reasoning tasks, against the selected established baselines. Best scores are highlighted in bold. Percentage improvement is computed with respect to the second-best score.

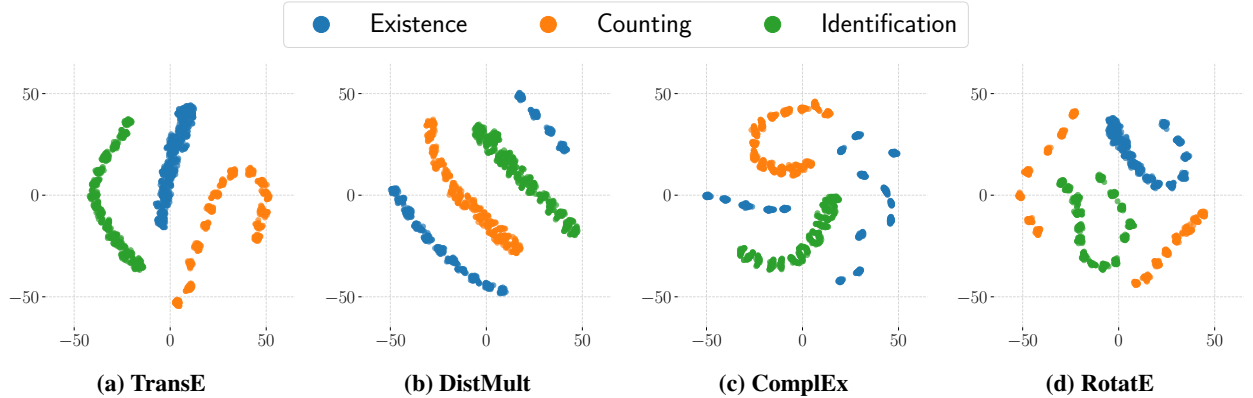| Data	Method	0-Hop			1-Hop						2-Hop								
		E-SL	C-SL	I-SL	E-SL	C-SL	I-SL	E-DL	C-DL	I-DL	E-SL	C-SL	I-SL	E-DL	C-DL	I-DL	E-TL	C-TL	I-TL
Synthetic	Zero-Shot	0.43	0.05	0.0	0.43	0.05	0.05	0.0	0.07	0.0	0.68	0.05	0.05	0.02	0.1	0.0	0.03	0.2	0.05
	Few-Shot	0.93	0.22	0.07	0.77	0.12	0.18	0.55	0.15	0.02	0.52	0.12	0.13	0.48	0.12	0.1	0.35	0.12	0.02
	CoT	0.9	0.13	0.07	0.65	0.12	0.17	0.52	0.13	0.02	0.45	0.08	0.12	0.43	0.15	0.08	0.48	0.22	0.03
	Zero-CoT	0.0	0.05	0.0	0.35	0.0	0.0	0.45	0.0	0.0	0.83	0.0	0.0	0.27	0.08	0.0	0.58	0.0	0.05
	CoT-BaG	0.78	0.13	0.07	0.65	0.12	0.17	0.52	0.13	0.02	0.57	0.12	0.13	0.53	0.12	0.1	0.57	0.25	0.03
	Prompt Tuning	0.77	0.17	0.03	0.52	0.15	0.35	0.3	0.1	0.0	0.28	0.02	0.12	0.0	0.07	0.03	0.03	0.12	0.02
	Ours	**1.0**	**0.7**	**0.08**	**1.0**	**0.88**	**0.93**	**1.0**	**0.63**	**0.13**	**1.0**	**0.88**	**0.93**	**0.97**	**0.67**	**0.13**	**0.95**	**0.7**	**0.18**
AIDS	Zero-Shot	0.45	0.08	0.01	0.45	0.12	0.07	0.0	0.28	0.04	0.71	0.12	0.07	0.0	0.07	0.0	0.08	0.45	0.18
	Few-Shot	0.93	0.26	0.02	0.71	0.16	0.05	0.69	0.36	0.01	0.72	0.17	0.04	0.71	0.14	0.16	0.62	0.3	0.0
	CoT	0.89	0.22	0.02	0.7	0.12	0.09	0.7	0.3	0.01	0.74	0.17	0.04	0.72	0.14	0.15	0.7	0.36	0.0
	Zero-CoT	0.0	0.08	0.0	0.34	0.02	0.02	0.46	0.0	0.04	0.68	0.02	0.02	0.11	0.1	0.02	0.19	0.0	0.18
	CoT-BaG	0.84	0.22	0.02	0.7	0.12	0.09	0.7	0.3	0.01	0.72	0.18	0.04	0.74	0.13	0.16	0.68	0.38	0.0
	Prompt Tuning	0.07	0.08	0.0	0.67	0.06	0.04	0.14	0.08	0.02	0.59	0.08	0.12	0.1	0.09	0.14	0.49	0.1	0.02
	Ours	**1.0**	**0.4**	**0.06**	**1.0**	**0.28**	**0.18**	**0.95**	**0.62**	**0.43**	**0.97**	**0.25**	**0.18**	**0.9**	**0.39**	**0.18**	**0.95**	**0.64**	**0.46**
MUTAG	Zero-Shot	0.37	0.0	0.0	0.37	0.0	0.05	0.0	0.16	0.0	0.32	0.0	0.05	0.0	0.05	0.0	0.0	0.0	0.0
	Few-Shot	0.89	0.11	0.05	0.63	0.0	0.0	0.53	0.16	0.05	0.63	0.11	0.0	0.63	0.16	0.16	0.63	0.16	0.05
	CoT	0.95	0.26	0.05	0.47	0.0	0.0	0.63	0.32	0.11	0.63	0.16	0.0	0.63	0.26	0.05	0.68	0.16	0.11
	Zero-CoT	0.0	0.05	0.0	0.42	0.0	0.0	0.16	0.0	0.0	0.32	0.0	0.0	0.0	0.0	0.0	0.32	0.0	0.0
	CoT-BaG	0.89	0.32	0.05	0.47	0.0	0.0	0.63	0.32	0.11	0.63	0.16	0.0	0.58	0.26	0.05	0.68	0.16	0.11
	Prompt Tuning	0.89	0.0	0.0	0.47	0.0	0.0	0.0	0.0	0.0	0.0	0.05	0.0	0.58	0.0	0.0	0.05	0.0	0.0
	Ours	**1.0**	**0.74**	**0.58**	**1.0**	**0.58**	**0.47**	**0.89**	**0.84**	**0.68**	**1.0**	**0.58**	**0.47**	**0.89**	**0.95**	**0.68**	**1.0**	**0.95**	**0.63**
AQSOL	Zero-Shot	0.53	0.06	0.01	0.51	0.15	0.02	0.03	0.33	0.0	0.73	0.15	0.02	0.17	0.07	0.0	0.08	0.37	0.06
	Few-Shot	0.92	0.18	0.05	0.74	0.16	0.05	0.69	0.28	0.01	0.66	0.15	0.05	0.69	0.16	0.1	0.58	0.3	0.02
	CoT	0.86	0.15	0.04	0.73	0.15	0.04	0.67	0.28	0.01	0.67	0.16	0.05	0.71	0.15	0.1	0.71	0.32	0.01
	Zero-CoT	0.0	0.05	0.0	0.01	0.02	0.12	0.74	0.0	0.01	0.51	0.02	0.12	0.48	0.02	0.01	0.22	0.0	0.12
	CoT-BaG	0.75	0.17	0.05	0.73	0.15	0.04	0.67	0.28	0.01	0.69	0.14	0.05	0.73	0.14	0.1	0.72	0.34	0.03
	Prompt Tuning	0.69	0.1	0.02	0.04	0.01	0.01	0.52	0.06	0.01	0.31	0.08	0.01	0.01	0.06	0.06	0.34	0.04	0.01
	Ours	**1.0**	**0.38**	**0.06**	**1.0**	**0.27**	**0.22**	**0.97**	**0.57**	**0.4**	**0.98**	**0.27**	**0.22**	**0.96**	**0.32**	**0.17**	**0.96**	**0.58**	**0.41**

Figure 2: Accuracy-Efficiency Trade-off of our method compared to state-of-the-art LLMs. The X-axis represents the average score across all reasoning tasks, while the Y-axis reports the number of trainable parameters.

where the X-axis corresponds to the average score across all reasoning tasks, and the Y-axis indicates the number of trainable parameters. For this analysis, we select the synthetic dataset as a representative benchmark.

The figure reveals that while our method slightly underperforms relative to GPT-4o (0.79), o4-mini (0.81), and DeepSeek-R1 (0.72) — achieving an average score of 0.71 — it requires approximately 10^5 fewer active parameters. This remarkable efficiency underscores the practicality of our approach, offering a highly competitive performance at a fraction of the computational cost, which is particularly advantageous for deployment in resource-constrained environments.

Table 4: Test accuracy over the considered datasets and reasoning tasks, varying the underlying KGE model.

		0-Hop			1-Hop						2-Hop								
Data	Model	E-SL	C-SL	I-SL	E-SL	C-SL	I-SL	E-DL	C-DL	I-DL	E-SL	C-SL	I-SL	E-DL	C-DL	I-DL	E-TL	C-TL	I-TL
Synthetic	TransE	0.98	0.23	0.05	1.0	0.73	0.92	0.98	0.3	0.1	1.0	0.78	0.92	0.97	0.25	0.12	0.9	0.23	0.17
	DistMult	1.0	0.63	0.08	1.0	0.83	0.77	1.0	0.17	0.1	1.0	0.83	0.77	0.95	0.67	0.08	0.9	0.6	0.18
	ComplEx	1.0	0.7	0.08	1.0	0.68	0.78	1.0	0.63	0.08	1.0	0.68	0.78	0.95	0.67	0.13	0.9	0.67	0.18
	RotatE	1.0	0.6	0.05	1.0	0.88	0.93	1.0	0.57	0.13	1.0	0.88	0.93	0.97	0.55	0.1	0.95	0.7	0.18
AIDS	TransE	1.0	0.29	0.06	1.0	0.21	0.18	0.88	0.46	0.41	0.96	0.23	0.18	0.9	0.26	0.1	0.92	0.5	0.46
	DistMult	1.0	0.4	0.06	1.0	0.19	0.12	0.94	0.6	0.43	0.96	0.19	0.12	0.88	0.38	0.08	0.92	0.64	0.44
	ComplEx	1.0	0.36	0.06	0.99	0.28	0.18	0.87	0.59	0.4	0.96	0.25	0.15	0.87	0.3	0.18	0.95	0.63	0.46
	RotatE	1.0	0.28	0.06	1.0	0.21	0.14	0.95	0.62	0.41	0.97	0.21	0.14	0.88	0.39	0.1	0.95	0.57	0.46
MUTAG	TransE	1.0	0.37	0.37	1.0	0.58	0.16	0.89	0.84	0.68	1.0	0.42	0.05	0.89	0.58	0.58	1.0	0.74	0.63
	DistMult	1.0	0.68	0.58	1.0	0.58	0.47	0.79	0.68	0.63	0.95	0.58	0.47	0.89	0.89	0.68	1.0	0.74	0.63
	ComplEx	1.0	0.74	0.37	1.0	0.53	0.21	0.84	0.74	0.53	0.95	0.47	0.32	0.89	0.95	0.63	1.0	0.95	0.53
	RotatE	1.0	0.74	0.32	1.0	0.53	0.11	0.79	0.84	0.53	1.0	0.53	0.11	0.79	0.79	0.58	1.0	0.84	0.63
AQSOL	TransE	1.0	0.19	0.05	1.0	0.21	0.19	0.9	0.41	0.36	0.85	0.22	0.19	0.93	0.27	0.11	0.96	0.43	0.36
	DistMult	1.0	0.38	0.04	0.99	0.27	0.22	0.97	0.56	0.36	0.98	0.27	0.22	0.93	0.32	0.13	0.96	0.49	0.4
	ComplEx	1.0	0.33	0.04	0.99	0.25	0.2	0.96	0.57	0.36	0.97	0.24	0.2	0.96	0.26	0.15	0.96	0.58	0.36
	RotatE	1.0	0.38	0.06	0.99	0.21	0.21	0.96	0.52	0.4	0.97	0.21	0.21	0.91	0.28	0.17	0.96	0.52	0.41

● Existence ● Counting ● Identification

(a) TransE **(b) DistMult** **(c) ComplEx** **(d) RotatE**

Figure 3: t-SNE projection of the synthetic graph embeddings produced by each KGE model, trained on an Existence, Counting and Identification task.

Sensitivity Analysis Further, we conduct a sensitivity analysis aimed at evaluating the robustness of our method when different KGE models are employed (**RQ2**). The full performance breakdown across all reasoning tasks is presented in Table 4.

Our findings show no absolute winner across the considered embedders, but the performance varies depending on the dataset and the task at hand. Indeed, the results on the existence task remain relatively stable across different KGE models, indicating that simpler reasoning tasks are less sensitive to the choice of underlying representation. However, in the more demanding counting and identification tasks, the performance varies significantly depending on the selected embedder.

We hypothesize that this variability is due to several interacting factors: (i) the increased complexity inherent to the counting and identification tasks, (ii) the structural properties and difficulty levels embedded within the datasets, and (iii) the differing representational capacities of the various KGE models. These insights suggest that careful selection of the underlying embedding model is crucial when targeting more complex forms of reasoning.

Latent Representation Lastly, we address **RQ3** by examining how the final knowledge graph embeddings vary according to the specific reasoning task. To this end, we visualize the learned graph representations across different tasks. Figure 4 illustrates the t-SNE projections [van der Maaten and Hinton, 2008] of the embeddings obtained using each KGE model trained on an Existence, Counting, and Identification task, respectively.

This analysis reveals that: first, although the underlying graphs remain the **same** across tasks, their latent representations exhibit consistent adaptations depending on the reasoning objective; and secondly, such adaptations persist regardless of the underlying KGE model used. This behavior underscores the ability of our method not only to align the intrinsic representation of the knowledge graph with the latent space of the LLM, but also to flexibly encode task-relevant features, adjusting the structure of the embedding space to better support different types of reasoning.

6 Conclusions and Future Work

In this work, building upon prior research, we introduced a novel technique for the direct integration of Knowledge Graph (KG) representations into frozen Large Language Models (LLMs), eliminating the need for prompt engineering or costly LLM fine-tuning. Through extensive experimentation on both synthetic and real-world datasets, we evaluated our approach against a variety of baselines and state-of-the-art LLMs. The results consistently demonstrate the effectiveness, generalizability, and computational efficiency of our method across diverse reasoning tasks. Our findings validate the potential of integrating external knowledge representations to enhance LLMs' reasoning capabilities without sacrificing scalability or flexibility.

Despite these promising results, there remain several avenues for further improvement and extension. First, while our evaluation encompasses a wide range of reasoning tasks, it primarily focuses on node-related queries. Extending the framework to cover edge-centric and graph-level reasoning tasks would broaden its applicability and provide a more comprehensive validation of its generalization capabilities. Second, while our method shows robust performance across most tasks, certain complex reasoning scenarios, such as the Identification task, pose greater challenges. To address this, future work could explore the integration of more sophisticated learning techniques during training, such as task-specific inductive biases, or hybrid approaches combining symbolic and neural representations. Finally, we envision that adapting the method to support dynamic or evolving knowledge graphs, as well as exploring tighter coupling mechanisms between the KG embeddings and the LLM latent space, could further enhance performance and unlock new capabilities for knowledge-intensive applications.

References

Berant, J.; Chou, A.; Frostig, R.; and Liang, P. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of EMNLP*, 1533–1544.

Biran, O., et al. 2024. Hopping through knowledge graphs with llms. In *ACL*.

Bordes, A.; Usunier, N.; Garcia-Durán, A.; Weston, J.; and Yakhnenko, O. 2013a. Translating embeddings for modeling multi-relational data. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, 2787–2795. Red Hook, NY, USA: Curran Associates Inc.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013b. Translating embeddings for modeling multi-relational data. In Burges, C.; Bottou, L.; Welling, M.; Ghahramani, Z.; and Weinberger, K., eds., *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Bordes, A.; Weston, J.; Usunier, N.; and Chopra, S. 2014. Question answering with subgraph embeddings. In *Proceedings of EMNLP*, 615–620.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20. Curran Associates Inc.

Chen, Z., et al. 2024. A new era for knowledge graph reasoning: Benchmarking large language models. *arXiv preprint arXiv:2402.00086*.

Chen, Z.; Bansal, Y.; Zhang, X.; Darrell, T.; and Song, D. 2023. Symbolic discovery of optimization algorithms. *arXiv preprint arXiv:2302.06675*.

Chen, Y.-P.; Nishida, N.; Nakayama, H.; and Matsumoto, Y. 2024. Recent trends in personalized dialogue generation: A review of datasets, methodologies, and evaluations. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*. ELRA and ICCL.

deAbril, I. M.; Yoshimoto, J.; and Doya, K. 2018. Connectivity inference from neural recording data: Challenges, mathematical bases and research directions. *Neural Networks* 102.

Debnath, A. K.; Lopez de Compadre, R. L.; Debnath, G.; Shusterman, A. J.; and Hansch, C. 1991. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry* 34(2):786–797.

DeepSeek. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.

Dwivedi, V. P.; Joshi, C. K.; Laurent, T.; Bengio, Y.; and Bresson, X. 2020. Benchmarking graph neural networks. *CoRR* abs/2003.00982.

Fatemi, B.; Halcrow, J.; and Perozzi, B. 2024. Talk like a graph: Encoding graphs for large language models.

Gain, B.; Bandyopadhyay, D.; and Ekbal, A. 2025. Bridging the linguistic divide: A survey on leveraging large language models for machine translation.

Holtz, M. 2011. *Dimension-wise Decompositions*. Springer Berlin Heidelberg. 11–27.

Huang, Z.; Wang, X.; Shen, Y.; Zhang, H.; Liu, W.; and Zhang, Y. 2023. Can llms benefit from knowledge graphs? an empirical evaluation. *arXiv preprint arXiv:2309.01528*.

Ji, G.; He, S.; Xu, L.; Liu, K.; and Zhao, J. 2015. Knowledge graph embedding via dynamic mapping matrix. In Zong, C., and Strube, M., eds., *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 687–696. Beijing, China: Association for Computational Linguistics.

Ji, G.; Liu, K.; He, S.; and Zhao, J. 2016. Knowledge graph completion with adaptive sparse transfer matrix. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, 985–991. AAAI Press.

Ji, S.; Pan, S.; Cambria, E.; Marttinen, P.; and Yu, P. S. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems* 33(2):494–514.

Ji, S.; Pan, S.; Cambria, E.; Marttinen, P.; and Yu, P. S. 2022. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems* 33(2):494–514.

Kingma, D. P., and Ba, J. 2017. Adam: A method for stochastic optimization.

Kojima, T.; Gu, S. S.; Reid, M.; Matsuo, Y.; and Iwasawa, Y. 2022. Large language models are zero-shot reasoners. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22. Curran Associates Inc.

Lester, B.; Al-Rfou, R.; and Constant, N. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; and Zhu, X. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, 2181–2187. AAAI Press.

Liu, Y.; Zheng, W.; Zhang, Y.; Ding, K.; Sun, Y.; Xu, Z.; He, Y.; and Zhao, W. X. 2023. Graphprompt: Graph-structured soft prompts for llm reasoning on graph data. *arXiv preprint arXiv:2310.07952*.

Liu, H.; Fatemi, E.; et al. 2024. Graphqa: A benchmark for reasoning over graph-structured data. *arXiv preprint arXiv:2402.03633*.

Morris, C.; Kriege, N. M.; Bause, F.; Kersting, K.; Mutzel, P.; and Neumann, M. 2020. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*.

OpenAI. 2024. Gpt-4o system card.

OpenAI. 2025. Introducing openai o3 and o4-mini.

Perozzi, B.; Fatemi, B.; Zelle, D.; Tsitsulin, A.; Kazemi, M.; Al-Rfou, R.; and Halcrow, J. 2024. Let your graph do the talking: Encoding structured data for llms.

Riesen, K., and Bunke, H. 2008. Iam graph database repository for graph based pattern recognition and machine learning. In da Vitoria Lobo, N.; Kasparis, T.; Roli, F.; Kwok, J. T.; Georgiopoulos, M.; Anagnostopoulos, G. C.; and Loog, M., eds., *Structural, Syntactic, and Statistical Pattern Recognition*. Springer Berlin Heidelberg.

Ruder, S. 2017. An overview of gradient descent optimization algorithms.

Sun, J., et al. 2023. Can llms benefit from knowledge graphs? an empirical evaluation. *arXiv preprint arXiv:2309.01528*.

Sun, H.; Dhingra, B.; Zaheer, M.; Mazaitis, K.; Salakhutdinov, R.; and Cohen, W. W. 2018. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of EMNLP*, 4231–4242.

Sun, H.; Dhingra, B.; Mazaitis, K.; Zaheer, M.; Cohen, W. W.; and Salakhutdinov, R. 2019a. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. In *Proceedings of EMNLP-IJCNLP*, 2380–2390.

Sun, Z.; Deng, Z.-H.; Nie, J.-Y.; and Tang, J. 2019b. Rotate: Knowledge graph embedding by relational rotation in complex space.

Team, G., and et. al., T. M. 2024. Gemma: Open models based on gemini research and technology.

Toroghi, A.; Guo, W.; Abdollah Pour, M. M.; and Sanner, S. 2024. Right for right reasons: Large language models for verifiable commonsense knowledge graph question answering. *arXiv preprint arXiv:2403.01390*.

Trouillon, T.; Welbl, J.; Riedel, S.; Éric Gaussier; and Bouchard, G. 2016. Complex embeddings for simple link prediction.

van der Maaten, L., and Hinton, G. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research* 9(86):2579–2605.

Wang, Z.; Zhang, J.; Feng, J.; and Chen, Z. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, 1112–1119. AAAI Press.

Wang, H.; Feng, S.; He, T.; Tan, Z.; Han, X.; and Tsvetkov, Y. 2023. Can language models solve graph problems in natural language? In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23. Curran Associates Inc.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E. H.; Le, Q. V.; and Zhou, D. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22. Red Hook, NY, USA: Curran Associates Inc.

Wu, J., et al. 2024. Thinking with knowledge graphs enhancing llm reasoning. *arXiv preprint arXiv:2402.02371*.

Yang, B.; tau Yih, W.; He, X.; Gao, J.; and Deng, L. 2015. Embedding entities and relations for learning and inference in knowledge bases.

Yao, Y.; Rosasco, L.; and Caponnetto, A. 2007. On early stopping in gradient descent learning. *Constructive Approximation* 26.

Yasunaga, M.; Ren, X.; Bosselut, A.; Liang, P.; and Leskovec, J. 2021. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. In *Proceedings of NAACL-HLT*, 535–546.

Yih, W.-t.; Chang, M.-W.; He, X.; and Gao, J. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of ACL*, 1321–1331.

Yue, M. 2025. A survey of large language model agents for question answering. *arXiv preprint arXiv:2503.19213*.

Zhang, H.; Yu, P. S.; and Zhang, J. 2025. A systematic survey of text summarization: From statistical methods to large language models. *ACM Comput. Surv.*

Zhao, J.; Li, Y.; Guo, H.; Lin, C.-Y.; and Zhang, Y. 2023. Kg-cot: Injecting knowledge graph in chain-of-thought for multi-hop question answering. *arXiv preprint arXiv:2305.14045*.

Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; and Sun, M. 2020. Graph neural networks: A review of methods and applications. *AI Open* 1:57–81.