Enhancing Large Language Model for Knowledge Graph Completion via Structure-Aware Alignment-Tuning

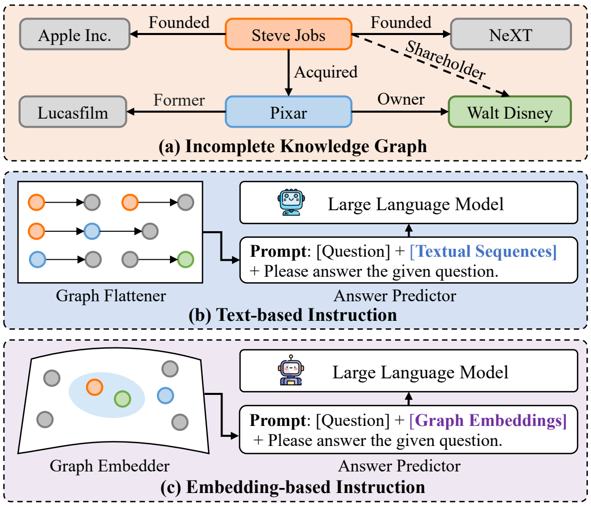
Yu Liu[1,2], Yanan Cao[1,2], Xixun Lin[1,2*], Yanmin Shang[1,2], Shi Wang[3], Shirui Pan[4]
[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[4]Griffith University, Queensland, Australia
`{liuyu2022,caoyanan,linxixun,shangyanmin}@iie.ac.cn`
`wangshi@ict.ac.cn, s.pan@griffith.edu.au`

Abstract

Knowledge graph completion (KGC) aims to infer new knowledge and make predictions from knowledge graphs. Recently, large language models (LLMs) have exhibited remarkable reasoning capabilities. LLM-enhanced KGC methods primarily focus on designing task-specific instructions, achieving promising advancements. However, there are still two critical challenges. First, existing methods often ignore the inconsistent representation spaces between natural language and graph structures. Second, most approaches design separate instructions for different KGC tasks, leading to duplicate works and time-consuming processes. To address these challenges, we propose SAT, a novel framework that enhances LLMs for KGC via structure-aware alignment-tuning. Specifically, we first introduce hierarchical knowledge alignment to align graph embeddings with the natural language space through multi-task contrastive learning. Then, we propose structural instruction tuning to guide LLMs in performing structure-aware reasoning over KGs, using a unified graph instruction combined with a lightweight knowledge adapter. Experimental results on two KGC tasks across four benchmark datasets demonstrate that SAT significantly outperforms state-of-the-art methods, especially in the link prediction task with improvements ranging from 8.7% to 29.8%[1].

1 Introduction

Knowledge graphs (KGs) organize world knowledge with structured relations between entities (Bordes et al., 2013; Chaudhri et al., 2022). In recent years, KGs have gained significant attention in various fields, such as information retrieval (Liu et al., 2018), question answering (Luo et al., 2024), and recommendation systems (Guo et al., 2020). Despite great achievements, real-world KGs often suffer from incompleteness, as shown in Figure 1(a), which inevitably limits their practical applications (Li et al., 2022a; Liu et al., 2024). Therefore, it is necessary to develop knowledge graph completion (KGC) (Wang et al., 2021) to automatically infer missing triples.



(a) Incomplete Knowledge Graph

(b) Text-based Instruction

(c) Embedding-based Instruction

Figure 1: Illustration of LLM-enhanced KGC methods. (a) Incomplete knowledge graph with missing triples. (b) Text-based methods flattening the KG into textual sequences. (c) Embedding-based methods integrating graph embeddings into LLMs.

Recently, large language models (LLMs) have demonstrated outstanding performance in various natural language processing (NLP) tasks (Meyer et al., 2023). LLM-enhanced KGC aims to leverage the generalizability of LLMs to make predictions over KGs. Existing methods can be broadly categorized into two lines: *text-based* and *embedding-based* methods, as illustrated in Figure 1(b) and (c). Text-based methods (Wei et al., 2023; Xu et al., 2024) verbalize relevant KG triples and append them to input prompts. While these approaches provide explicit paths, the flattening process destroys the underlying structure of KGs. In contrast,

[*]Corresponding author.

[1]Our source code is available at `https://github.com/liuyudiy/SAT`.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 20970–20984
November 4-9, 2025 ©2025 Association for Computational Linguistics

embedding-based methods (Ye et al., 2023; Zhang et al., 2024b; Tang et al., 2024) employ graph representation learning (Kipf and Welling, 2016; Lin et al., 2024) to generate graph embeddings for the retrieved subgraphs. By preserving the structures within KGs, these methods enable more robust reasoning and achieve impressive performance.

Despite the aforementioned success, embedding-based methods still face two critical challenges. First, while existing methods leverage graph embeddings to capture structural information, they often ignore the representational gap between graph structures and natural language, limiting LLMs' ability to fully comprehend and reason with structural knowledge. Second, current research predominantly depends on instruction tuning; however, most studies craft separate instructions for different KGC tasks (e.g., triple classification and link prediction), leading to redundancy and inefficiency. Therefore, it is essential to develop a new instruction-tuning paradigm that integrates various KGC tasks into a unified framework.

To overcome these challenges, we propose a novel framework named **SAT**, designed to enhance LLMs for KGC through **S**tructure-aware **A**lignment and **T**uning. Specifically, SAT first introduces a *hierarchical knowledge alignment* module to improve LLMs' understanding of graph structure encoding. We construct graph-text alignment datasets from a hierarchical perspective, considering both node-level and subgraph-level alignments, and employ multi-task contrastive learning to align graph embeddings with the natural language space. Subsequently, a *structural instruction tuning* module is proposed to guide LLMs in performing structure-aware reasoning. We design a new graph instruction and implement a lightweight tuning strategy, using a knowledge adapter to accommodate various KGC tasks.

In summary, our contributions are as follows:

- We present a new framework that seamlessly integrates graph structures into LLMs for KGC via structure-aware alignment-tuning.
- We propose hierarchical knowledge alignment to address the inconsistent representation spaces and structural instruction tuning to efficiently unify diverse KGC tasks.
- We evaluate SAT on two KGC tasks across four datasets. Extensive experiments show that our model surpasses state-of-the-art methods, particularly in the link prediction task.

2 Related work

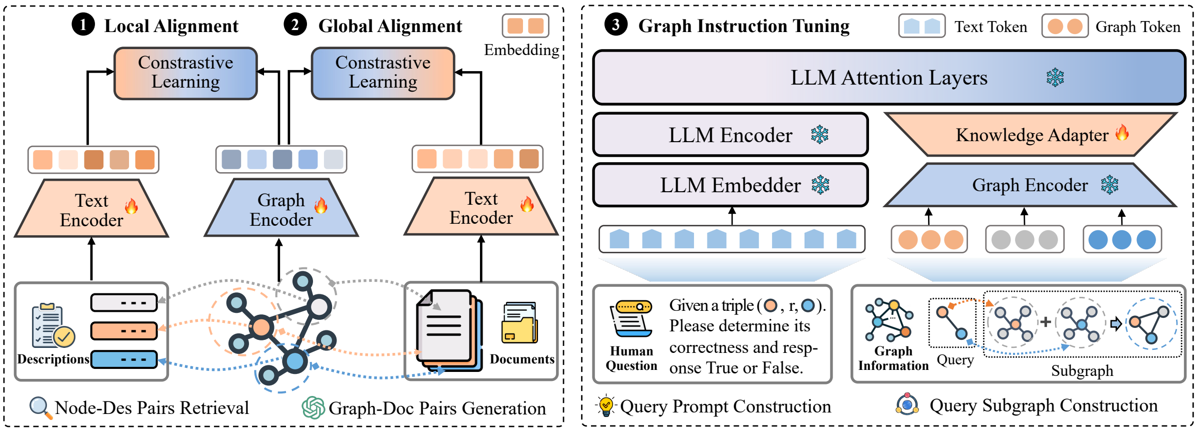
Nowadays, LLM-enhanced KGC has garnered increasing interest. Existing works can generally be categorized into two main directions.

Text-based methods verbalize relevant KGs to augment instructions for LLMs. KG-LLM (Yao et al., 2023) is the first to explore various LLMs for KGC. KICGPT (Wei et al., 2023) integrates LLMs with a triple-based retriever, while MPIKGC (Xu et al., 2024) employs LLMs to generate auxiliary texts for traditional KGC models. CP-KGC (Yang et al., 2024a) and GS-KGC (Yang et al., 2024b) further incorporate additional contextual constraints. Moreover, several methods (Jiang et al., 2024a; Chen et al., 2024) leverage LLMs for the enrichment of KG structures. Despite their success, these approaches still suffer from missing vital triples that make some questions unanswerable. Additionally, the flattening process may undermine the underlying structures within KGs.

Embedding-based methods incorporate structural knowledge into LLMs through graph representation learning. CSProm-KG (Chen et al., 2023) and PDKGC (Geng et al., 2023) investigate pretrained language models (PLMs) for KGC. More recently, KoPA (Zhang et al., 2024b) has utilized a prefix adapter to fuse structural embeddings into LLMs. MKGL (Guo et al., 2024a) further introduces a specialized KG language via KGL token embedding augmentation. GraphGPT (Tang et al., 2024) proposes graph instruction tuning, tailored specifically for citation networks. In addition, InstructGLM (Ye et al., 2023) and GRAG (Hu et al., 2024) also attempt to combine graph encoding into LLMs. However, these methods often ignore the space inconsistency and struggle to effectively integrate structures of KGs, limiting LLMs' ability to fully understand and perform KGC tasks.

3 Preliminary

Knowledge Graph Completion. Formally, a KG can be defined as $G = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E}$ and $\mathcal{R}$ are the sets of entities and relations, respectively, and $\mathcal{T} = \{(h, r, t) \mid h, t \in \mathcal{E}, r \in \mathcal{R}\}$ represents the set of relational triples. Moreover, entities are often accompanied by rich textual information. In this paper, our work focuses on KGC, specifically addressing two tasks: (i) triple classification, assessing the correctness of a given triple (h, r, t), and (ii) link prediction, identifying the most plausible tail entity t for a given query $(h, r, ?)$.



(a) Hierarchical Knowledge Alignment

(b) Structural Instruction Tuning

Figure 2: The overall framework of our **SAT**. (a) Hierarchical knowledge alignment, comprising local and global alignments, aligns graph structural representations with the natural language space. (b) Structural instruction tuning enables LLMs to perform structure-aware reasoning over KGs through a lightweight tuning strategy.

Large Language Models. LLMs have emerged as a powerful new paradigm for diverse tasks through prompting engineering. Let $\mathcal{M}$ denote the generative LLM, which takes a sequence $X = [x_1, ..., x_n]$ as the input prompt, and generates the response sequence $Y = [y_1, ..., y_m]$. Briefly, this process is represented as $Y = \mathcal{M}(X)$, where $X = [I; Q]$ consists of a task-specific instruction I and an input query Q. This paper aims to develop a structure-aware LLM that excels in various KGC tasks by effectively integrating structural information.

4 Methodology

The architecture of SAT is illustrated in Figure 2. Specifically, our model consists of two components: (1) hierarchical knowledge alignment and (2) structural instruction tuning. In the following sections, we provide a detailed explanation.

4.1 Hierarchical Knowledge Alignment

To enhance LLMs' understanding of graph encoding, we consider local and global alignments.

Local Knowledge Alignment. To ensure that LLMs effectively understand the semantics of individual node representations, we align each entity with its corresponding textual description.

Node-Description Pairs Construction. For each entity e in the KG G, we obtain its corresponding textual description D_e by extracting the first paragraph from its linked Wikipedia page. For entities without linked pages, we use their names as alternative descriptions. In this way, we construct a set of node-description pairs $\mathcal{P} = \{(e, D_e) \mid e \in \mathcal{E}, D_e \in \mathcal{D}\}$, where $\mathcal{D}$ is the set of textual descriptions.

Node-level Representation Alignment. We utilize a graph encoder, denoted as **GE**, to encode the graph G and obtain embeddings for entity nodes. Simultaneously, a text encoder, denoted as **TE**, encodes the corresponding entity descriptions $\mathcal{D}$. In practice, we employ a graph transformer (Yun et al., 2019) as the graph encoder and a vanilla transformer (Vaswani, 2017) as the text encoder. The detailed encoding process is as follows[2],

$$\mathbf{H} = \mathbf{GE}(G), \tag{1}$$

$$\mathbf{D} = \{\mathbf{d}_e \mid \mathbf{d}_e = \mathbf{TE}(D_e),\ e \in \mathcal{E}\}, \tag{2}$$

where $\mathbf{H} \in \mathbb{R}^{N\times d}$ and $\mathbf{D} \in \mathbb{R}^{N\times d}$ represent the embeddings of nodes and descriptions, with N denoting the number of nodes. We then apply L2 normalization to both $\mathbf{H}$ and $\mathbf{D}$. Finally, the node-level alignment is performed as follows,

$$\mathbf{\Lambda} = (\mathbf{H}\,\mathbf{D}^\top)\cdot \exp(\tau), \tag{3}$$

$$\mathcal{L}_{\text{local}} = \frac{1}{2}\left(\text{CE}(\mathbf{\Lambda}, \mathbf{Y}) + \text{CE}(\mathbf{\Lambda}^\top, \mathbf{Y})\right), \tag{4}$$

where $\mathbf{\Lambda} \in \mathbb{R}^{N\times N}$ is the similarity matrix, with each $\mathbf{\Lambda}_{i,j}$ representing the similarity of the i-th node embedding and the j-th description embedding. The label $\mathbf{Y} = \mathbf{I}_N$ is the identity matrix, indicating the i-th node embedding is closest to the i-th description embedding. $\text{CE}(\cdot)$ denotes the cross-entropy loss. Consequently, the loss $\mathcal{L}_{\text{local}}$ encourages bidirectional alignment between nodes and their corresponding descriptions. In addition, the scalar $\tau \in \mathbb{R}$ is a temperature parameter.

[2] Note that the input to **GE** is the graph structure G, with node and edge features randomly initialized. Similarly, we apply the same approach for the global alignment in Eq.(5).

Global Knowledge Alignment. To capture the global semantics conveyed by subgraphs, we focus further on global alignment that aligns subgraphs with their associated textual information.

Subgraph-Document Pairs Construction. Inspired by the powerful extraction abilities of LLMs, we employ GPT-4 to construct subgraph-document pairs. Initially, we sample a subset of the previously retrieved paragraphs to serve as input documents. For each document D_S, we leverage GPT-4 to extract triples[3], which are then organized into the subgraph, denoted as S. Through this process, we construct a set of subgraph-document pairs, represented as $\mathcal{P}' = \{(S, D_S) \mid S \in \mathcal{S}, D_S \in \mathcal{D}'\}$, where $\mathcal{S}$ is the set of subgraphs and $\mathcal{D}'$ is the set of corresponding documents.

Subgraph-level Representation Alignment. Similarly, we utilize the graph encoder **GE** and the text encoder **TE** to encode subgraphs $\mathcal{S}$ and textual documents $\mathcal{D}'$ respectively. The subgraph embeddings $\mathbf{H}'$ and document embeddings $\mathbf{D}'$ are generated as follows,

$$\mathbf{H}' = \{\mathbf{h}_S \mid \mathbf{h}_S = \text{Pool}(\mathbf{GE}(S)),\ S \in \mathcal{S}\}, \quad (5)$$

$$\mathbf{D}' = \{\mathbf{d}_S \mid \mathbf{d}_S = \mathbf{TE}(D_S),\ S \in \mathcal{S}\}, \quad (6)$$

where $\mathbf{H}' \in \mathbb{R}^{M\times d}$, $\mathbf{D}' \in \mathbb{R}^{M\times d}$, and M is the number of subgraphs. $\text{Pool}(\cdot)$ represents the mean pooling operator. Then, we calculate the similarity matrix $\mathbf{\Lambda}'$ between $\mathbf{H}'$ and $\mathbf{D}'$ in the same way of Eq.(3), and apply the contrastive loss as follows,

$$\mathcal{L}_{\text{global}} = \frac{1}{2}\left(\text{CE}(\mathbf{\Lambda}', \mathbf{Y}) + \text{CE}(\mathbf{\Lambda}'^{\top}, \mathbf{Y})\right). \quad (7)$$

The final loss is defined as a joint training objective of the local loss and the global loss:

$$\mathcal{L}_{\text{HKA}} = \mathcal{L}_{\text{local}} + \mathcal{L}_{\text{global}}. \quad (8)$$

Overall, we align graph embeddings with the natural language space. Note that the parameters of the local and global alignments are shared.

4.2 Structural Instruction Tuning

To guide LLMs in structure-aware reasoning over KGs, we further introduce structural instruction tuning. Concretely, we design a unified graph instruction with a lightweight tuning strategy to support various KGC tasks.

Graph Instruction Tamplate

Human Question: # transform the input into a question
Graph Information: # construct the query subgraph from KG
Model Response: # generate answers for the given query

Triple Classification Task

Human Question: Given a triple <triple>\n Please determine the correctness of the input triple and response True or False.\n
Graph Information: Given a sequence of graph embeddings <graph> that represent a subgraph of the triple extracted from a knowledge graph.\n
Model Response: True or False

Link Prediction Task

Human Question: Given a question <question>\n Please answer the given question, and keep the answer as simple as possible.\n
Graph Information: Given a sequence of graph embeddings <graph> that represent a subgraph of the question extracted from a knowledge graph.\n
Model Response: Target Answers

Figure 3: Illustration of graph instruction design. Graph instruction template unifies various KGC tasks.

Graph Instruction Design. To unify KGC tasks, we formulate them as a generative question-answering problem, and design a graph instruction template comprising three parts: i) human question, ii) graph information, and iii) model response, as illustrated in Figure 3. Specifically, given an input query (e.g., a triple), we first convert it into a human question Q. Next, to provide structural context, we construct the query subgraph S_Q by extracting the k-hop neighborhoods around the anchor entities in the query from the KG. The final input prompt is defined as $X = [I; Q; \mathbf{EMB}(S_Q)]$, where I represents the system instruction, and $\mathbf{EMB}(\cdot)$ refers to the graph embeddings, obtained using the method introduced in Section 4.1. This instruction template not only effectively integrates both natural language and structural information, but also flexibly unifies different KGC tasks.

Lightweight Tuning Strategy. To optimize the fine-tuning process efficiently, we introduce a lightweight tuning strategy that incorporates a graph encoder and a knowledge adapter. The graph encoder is used to encode structural knowledge from the KG, while the adapter is designed to accommodate various KGC tasks. During training, we freeze the parameters of the LLM $\mathcal{M}$ and the graph encoder **GE**, focusing solely on optimizing the adapter's parameters. Once trained, the LLM is expected to generalize across various tasks. In practice, the adapter can be implemented as a simple projection layer. Note that the graph encoder **GE** is pre-trained as described in Section 4.1.

In general, the fine-tuning objective is to maximize the model's likelihood of generating the target response Y, conditioned on the prompt instruction X, as follows:

$$\mathcal{L}_{\text{SIT}} = \mathbb{E}_{(X,Y)\in(\mathcal{X},\mathcal{Y})}\left[-\log P_{\mathcal{M}}(Y \mid X)\right]. \quad (9)$$

[3] The extraction instruction is provided in Appendix G.1.

5 Experiments

5.1 Experimental Settings

Datasets. We evaluate SAT on two key KGC tasks: triple classification and link prediction, both of which are critical for completing incomplete KGs. Our experiments span four widely used benchmark datasets, categorized into two types: 1) small and hard (FB15k-237N, CoDeX-S) and 2) large and sparse (FB15k-237, YAGO3-10). The dataset statistics are presented in Table 1, and detailed descriptions can be found in Appendix A.

Table 1: Statistics of the benchmark datasets.

Dataset	#Nodes	#Edges	#Train	#Valid	#Test
FB15k-237N	13,104	93	87,282	7,041	8,226
CoDeX-S	2,034	42	32,888	1827	1,828
FB15k-237	14,505	237	272,115	17,535	20,466
YAGO3-10	123,182	37	1,079,040	5,000	5,000

Baselines. We compare SAT with representative baselines grouping into two categories. The details of each baseline are described in Appendix B.

- **Traditional KGC methods.** *i) Embedding-based methods:* TransE (Bordes et al., 2013), ComplEx (Trouillon et al., 2016), ConvE (Dettmers et al., 2018), RotatE (Sun et al., 2019), and QuatE (Sun et al., 2019). *ii) GNN-based methods:* CompGCN (Vashishth et al., 2019) and RED-GNN (Zhang and Yao, 2022). *iii) Transformer-based methods:* PKGC (Lv et al., 2022), SimKGC (Wang et al., 2022b), and KG-S2S (Chen et al., 2022).
- **LLM-enhanced KGC methods.** *i) Text-based methods:* KG-LLaMA (Yao et al., 2023), KERMIT (Li et al., 2023), MPIKGC (Xu et al., 2024), GS-KGC (Yang et al., 2024b), and KG-FIT (Jiang et al., 2024b). *ii) Embedding-based methods:* CSProm-KG (Chen et al., 2023), Structural-aware IT (Zhang et al., 2024b), and KoPA (Zhang et al., 2024b).

Evaluation Protocols. Following previous work, we evaluate triple classification using four standard metrics: accuracy, precision, recall, and F1-score, and assess link prediction with two metrics: Hits@1 and MRR. Accuracy reflects the overall correctness, while F1-score provides a balanced evaluation of precision and recall. Hits@1 measures the proportion of instances where the top-1 predicted answer is correct, and MRR calculates the average reciprocal rank of target answers.

Implementation Details. For SAT, we use Llama2-Chat-7B (Touvron et al., 2023) as the LLM backbone, fine-tuned on the training split of public datasets for 3 epochs. For each query, the top-3 answers are generated using a beam-search strategy. In the hierarchical knowledge alignment, we employ a graph transformer (Yun et al., 2019) as the graph encoder, and a vanilla transformer (Vaswani, 2017) as the text encoder. The pre-trained graph encoder and the learned graph embeddings are subsequently utilized in the structural instruction tuning module. During the tuning stage, the parameters of the LLM and the graph encoder are frozen, only the knowledge adapter is fine-tuned, implemented as a two-layer feed-forward neural network. Our model is implemented in Pytorch and trained on two Nvidia A800 GPUs. Detailed implementation settings are described in Appendix C.

5.2 Overall Comparison

To evaluate our SAT, we conduct comprehensive experiments on two KGC tasks across four datasets. **Triple Classification.** The overall results are presented in Table 2. SAT significantly outperforms previous state-of-the-art methods, demonstrating the effectiveness of our model. When compared to PKGC, SAT achieves an average relative F1-score improvement of 4.5% on FB15k-237N and 5.1% on CoDeX-S, highlighting its superiority over fully fine-tuned transformer-based models. In contrast with KoPA, our model shows average relative improvements of 2.8% and 2.9% on FB15k-237N and CoDeX-S, respectively. These results emphasize the importance of aligning graph structure encoding with the natural language space to enhance the understanding of graph structures by LLMs. Additional experiment results on FB15k-237 and YAGO3-10 are provided in Appendix E.1.

Link Prediction. The results for link prediction are shown in Table 3 and Table 4. In general, SAT substantially exceeds previous methods across all four datasets. Notably, SAT achieves significant relative improvements in Hit@1, surpassing the second-best model by 29.1% on FB15k-237N and 29.8% on CoDeX-S. Furthermore, compared to RED-GNN, our model demonstrates relative Hit@1 improvements of 13.1% and 8.7% on FB15k-237 and YAGO3-10, respectively. We attribute these substantial gains to two key factors: 1) our model effectively bridges the gap between graph structures and textual information through the hierarchical knowledge alignment, and 2) structural in-

Table 2: Experimental results on FB15k-237N and CoDeX-S datasets for triple classification. The best performances are highlighted in boldface, the second-best results are underlined, and "*" indicates result reproduction.

Model	FB15k-237N				CoDeX-S			
	Accuracy	**Precision**	**Recall**	**F1-score**	**Accuracy**	**Precision**	**Recall**	**F1-score**
TransE (Bordes et al., 2013)	0.697	0.708	0.671	0.689	0.721	0.719	0.724	0.722
DistMult (Yang et al., 2014)	0.587	0.590	0.568	0.579	0.668	0.697	0.595	0.642
ComplEx (Trouillon et al., 2016)	0.657	0.665	0.634	0.649	0.676	0.678	0.671	0.675
RotatE (Sun et al., 2019)	0.685	0.692	0.664	0.678	0.757	0.757	0.757	0.757
KG-BERT (Yao et al., 2019)	0.560	0.535	0.976	0.678	0.773	0.710	**0.924**	0.803
PKGC (Lv et al., 2022)*	0.796	0.784	0.865	0.795	0.802	0.743	0.923	0.823
GPT-4 (Achiam et al., 2023)*	0.708	0.817	0.535	0.647	0.822	0.825	0.703	0.798
Vanilla IT (Zhang et al., 2023)	0.735	0.659	0.975	0.786	0.812	0.770	0.889	0.825
KG-Alpaca (Yao et al., 2023)	0.699	0.627	**0.982**	0.766	0.803	0.794	0.817	0.805
KG-LLaMA (Yao et al., 2023)	0.748	0.674	0.962	0.793	0.794	0.787	0.807	0.797
Structural IT (Zhang et al., 2024b)	0.764	0.696	0.940	0.799	0.813	0.771	0.884	0.826
KoPA (Zhang et al., 2024b)	0.777	0.708	0.941	0.808	0.827	0.779	0.914	0.841
SAT (ours)	**0.827**	**0.823**	0.852	**0.831**	**0.856**	**0.834**	0.893	**0.865**

Table 3: Results of link prediction on two small datasets.

Model	FB15k-237N		CoDeX-S	
	Hit@1	**MRR**	**Hit@1**	**MRR**
TransE	0.152	0.255	0.219	0.354
ConvE	0.192	0.273	0.343	0.444
TuckER	0.228	0.312	0.339	0.444
CompGCN*	0.231	0.316	0.315	0.395
PKGC*	0.261	0.332	0.349	0.462
SimKGC*	0.286	0.343	0.216	0.321
KG-S2S	0.282	0.353	-	-
HaSa*	0.299	0.374	0.274	0.370
GPT-4*	0.241	0.264	0.196	0.232
SAT (ours)	**0.386**	**0.440**	**0.453**	**0.481**

Table 4: Results of link prediction on two large datasets.

Model	FB15k-237		YAGO3-10	
	Hit@1	**MRR**	**Hit@1**	**MRR**
RotatE	0.241	0.338	0.402	0.495
CompGCN	0.264	0.355	0.395	0.489
RED-GNN	0.282	**0.376**	0.483	0.559
SimKGC*	0.249	0.336	0.198	0.287
HaSa*	0.233	0.316	0.239	0.291
CSProm-KG	0.269	0.358	0.451	0.488
KERMIT	0.266	0.359	-	-
MPIKGC	0.267	0.360	-	-
KG-FIT	0.275	0.362	0.474	0.568
SAT (ours)	**0.319**	0.354	**0.525**	**0.616**

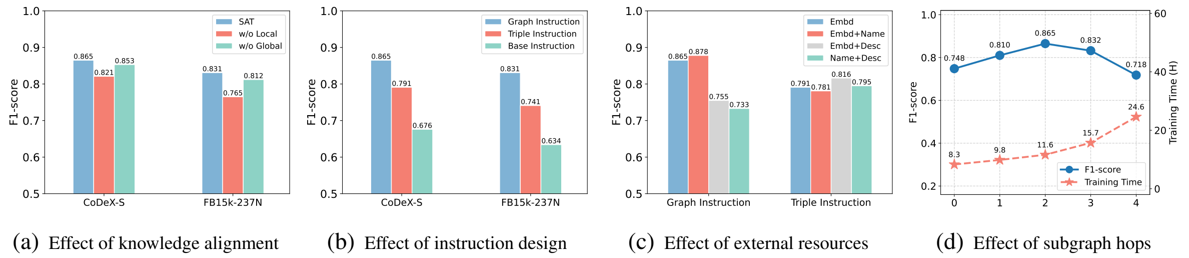
struction tuning enables LLMs to comprehend the underlying structures within KGs, thus enhancing their graph reasoning abilities.

5.3 Ablation Study

Effect of Hierarchical Knowledge Alignment. We conduct experiments to illustrate the necessity of aligning graph embeddings with the natural language space. As shown in Figure 4(a), the model without local alignment performs notably worse on FB15k-237N and CoDeX-S, which highlights the effectiveness of the node-level alignment. When global alignment is removed, the model struggles to understand global semantics expressed by subgraphs, resulting in a huge performance drop. This exhibits the critical importance of local and global alignment to achieve optimal results.

Effect of Graph Instruction Design. Well-crafted instructions are crucial for enhancing the adaptability of LLMs. To this end, we propose three instruction modes: base instruction (without graph information), triple instruction (with a single triple), and graph instruction (with subgraph information). As illustrated in Figure 4(b), the base instruction mode drops significantly, underscoring the importance of integrating graph information in KGC tasks. Furthermore, the graph instruction mode substantially outperforms the triple instruction mode, since query subgraphs provide richer contextual information than individual triples. Detailed instructions are provided in Appendix G.3.

Effect of External Resources Fusion. In addition to the graph information, we further enrich graph instruction by combining multiple external resources, such as graph embeddings, entity names, and text descriptions, as shown in Figure 4(c). Interestingly, graph instructions augmented with entity names achieve the best results. However, when further supplemented with corresponding descriptions, the F1-score on CoDeX-S unexpectedly declines. We speculate that the additional descriptions may introduce extraneous information, potentially diluting the focus on critical entities. Meanwhile,



(a) Effect of knowledge alignment (b) Effect of instruction design (c) Effect of external resources (d) Effect of subgraph hops

Figure 4: The comparative experiments of SAT on triple classification. (a) Knowledge alignment removes local and global alignment, respectively. (b) Instruction design is based on subgraphs or individual triples. (c) External resources, such as names and descriptions, are incorporated. (d) Query subgraph is constructed with different hops.

Table 5: Robustness analysis of SAT under conditions of limited or noisy textual information.

Description Type	**Linking Noise**	**FB15k-237N**		**CoDeX-S**	
		Hit@1	MRR	Hit@1	MRR
Name	0%	0.351	0.405	0.395	0.432
Paragraph	0%	0.386	0.440	0.453	0.481
Paragraph	5%	0.364	0.423	0.420	0.447
Paragraph	10%	0.328	0.397	0.372	0.413

Table 6: Efficiency analysis on YAGO3-10 during both the model training and inference stages.

Methods	**Language Model**	**Training**		**Inference**	
		Params	Time	Time	Hit@1
PKGC	RoBERTa	353.1MB	168H	50M	0.156
SimKGC	BERT	106.2MB	13H	10M	0.198
HaSa	SBERT[4]	218.0MB	37H	4M	0.239
Ours	LLaMa2-7B	563.8MB	70H	48M	0.525

this phenomenon also suggests that simply concatenating entity descriptions will disrupt semantic consistency and negatively impact model performance. Detailed descriptions of these instructions are illustrated in Appendix G.4.

Effect of Query Subgraph Hops. To evaluate the impact of subgraph scale, we construct subgraphs with different hops. As shown in Figure 4(d), the model's F1-score improves consistently as the number of hops increases. However, when the hop count exceeds 2, we observe a noticeable decline in performance, likely due to the inclusion of distant or noisy nodes. These findings indicate a trade-off between contextual richness and information noise. Therefore, we set the hop to 2 for subgraph construction in all subsequent experiments. In addition, we also investigate the impact of in-context learning in Appendix F.

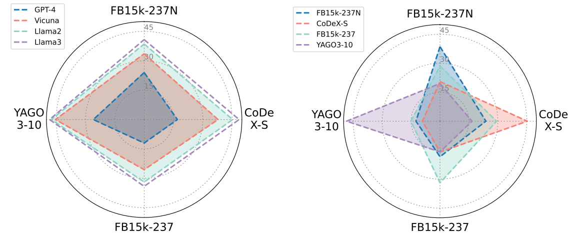
5.4 Robustness Analysis

In SAT, we rely on external textual information (e.g., Wikipedia), which may not always be available. To assess the robustness of our model under limited or noisy textual information, we present results in Table 5. (1) *Limited Textual Information*: We use only entity names as textual descriptions. Despite this reduction, SAT w/ name still achieves competitive inference performance, demonstrating strong generalization capabilities even in the absence of detailed contextual information. (2) *Noisy Textual Information*: We simulate different levels of noise in the entity descriptions (e.g., random substitution of paragraphs) to evaluate model stability. The results show that SAT maintains reliable performance under moderate noise. We attribute this to the inherent graph structure, where contextual signals from neighboring entities could mitigate the impact of erroneous textual information.

5.5 Efficiency Analysis

To evaluate model efficiency, we conduct experiments on the larger-scale KG YAGO3-10. From Table 6, we can observe that: (1) *Computational Cost*: SAT is more efficient than PKGC, primarily due to the use of graph embeddings that effectively reduce prompt length. While SimKGC and HaSa exhibit lower computational time, our method achieves substantially better performance, highlighting a trade-off between efficiency and effectiveness. (2) *Memory Usage*: SAT maintains the same order of magnitude (millions) as other methods. This is largely attributed to our lightweight fine-tuning strategy, in which only the adapter is updated. Overall, our method demonstrates acceptable time and space complexity. The specific parameters of our model are provided in Appendix E.2.

[4] SBERT (Sentence-BERT) is a modified version of the pretrained BERT network (Reimers and Gurevych, 2019).



(a) Transferability on different LLMs (b) Transferability on multiple domains

Figure 5: Transferability study of our SAT on different LLMs and cross-domain datasets.

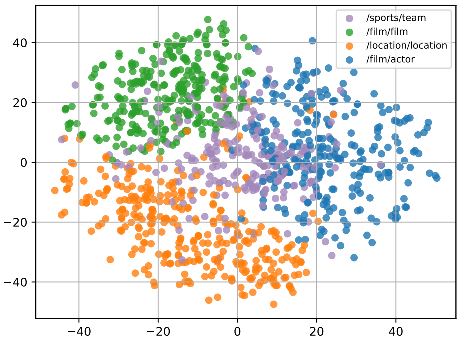


Figure 6: Visualization of graph embeddings.

5.6 Transferability Investigation

To assess the transferability of SAT across different LLMs and cross-domain datasets, we carry out experiments in Figure 5. (1) *Transferability Across LLMs*. We evaluate link prediction on four datasets using representative LLMs, including Vicuna-v1.5-7B, Llama2-7B, and Llama3.1-8B. SAT consistently demonstrates remarkable performance across these models, underscoring its robustness and generalizability. (2) *Transferability Across Domains*. SAT exhibits strong graph reasoning capabilities when trained and tested on the same dataset. However, its performance degrades when applied to unseen datasets without further training. Notably, the model trained on FB15k-237N transfers better to FB15k-237 than to YAGO3-10, indicating that SAT achieves higher transferability between more closely related domains.

5.7 Representation Visualization

We employ the t-SNE algorithm to visualize the learned graph embeddings, using a sample dataset comprising 1,000 entities across 4 classes from FB15k-237N. As shown in Figure 6, our method reveals more distinct class boundaries, particularly for */film/film*, */location/location*, and */film/actor*, demonstrating the superiority of graph embeddings learned by our model and further enhancing LLMs' understanding of graph structure encoding. More visualizations are included in the Appendix E.4.

Table 7: Case study of link prediction on FB15k-237N.

Task Instruction: You are an assistant specializing in large language models and knowledge graphs. Please follow the instructions carefully and provide your responses.
Text-based Instruction
Input: Given a question: **(Charles Dickens, influenced_by, ?)**. Given a textual sequence of graph: (Charles Dickens, profession, Author), (Paul Auster, influenced_by, Charles Dickens), (Leo Tolstoy, influenced_by, Charles Dickens), (Paul Auster, profession, Author), (Paul Auster, influenced_by, William Shakespeare) that represent a subgraph of the question extracted from a knowledge graph. Please answer the input question. **Output (GPT-4):** Paul Auster and Leo Tolstoy (✗)
Embedding-based Instruction
Input: Given a question: **(Charles Dickens, influenced_by, ?)**. Given a sequence of graph embeddings: Emb(Charles Dickens), Emb(Paul Auster), Emb(Leo Tolstoy), Emb(William Shakespeare), Emb(Author) that represent a subgraph of the question extracted from a knowledge graph. Please answer the input question. **Output (Ours):** William Shakespeare (✓)

5.8 Case Study

We present a case analysis on link prediction to further demonstrate the superiority of SAT. For the query (*Charles Dickens*, *influenced_by*, ?), we compare two different methods: text-based instruction and embedding-based instruction. As illustrated in Table 7, the embedding-based method effectively captures the implicit relationships between *Charles Dickens* and *William Shakespeare*, enabling accurate reasoning and predictions. Additionally, we test GPT-4 using the text-based instruction, but it incorrectly predicts two entities(i.e., *Paul Auster* and *Leo Tolstoy*), likely due to misleading triples in the instruction, such as (*Paul Auster*, *influenced_by*, *Charles Dickens*) and (*Leo Tolstoy*, *influenced_by*, *Charles Dickens*). This indicates that GPT-4 is more sensitive to the specific wording of instructions compared to our model.

6 Conclusion

In this paper, we introduce SAT, a novel framework designed to enhance LLMs for KGC tasks. Our model is comprised of two key components: hierarchical knowledge alignment and structural instruction tuning. Extensive experiments on two KGC tasks across four datasets demonstrate the superiority of our method, achieving significant improvements over existing state-of-the-art baselines.

Limitation

We analyze the limitations of our method and outline potential directions for future work. First, while our method effectively integrates graph embeddings into LLMs, we will explore more sophisticated strategies for combining textual and structural information to fully exploit the complementary strengths of both modalities. Second, although our SAT is a general framework adaptable to various KGC tasks, we plan to extend its applicability to a wider range of knowledge-intensive tasks and datasets, further enhancing its versatility.

Ethics Statement

This work does not raise any direct ethical concerns. Our focus is on LLM-enhanced KGC tasks. All experiments are conducted on publicly available datasets, and the findings and conclusions of this paper are reported accurately and objectively.

Acknowledgments

This work is supported by the National Key Research and Development Program of China (NO.2022YFB3102200), the National Natural Science Foundation of China (No.62402491), and the China Postdoctoral Science Foundation (No.2025M771524).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

Vinay Chaudhri, Chaitanya Baru, Naren Chittar, Xin Dong, Michael Genesereth, James Hendler, Aditya Kalyanpur, Douglas Lenat, Juan Sequeda, Denny Vrandečić, et al. 2022. Knowledge graphs: introduction, history and, perspectives. *AI Magazine*, 43(1):17–29.

Chen Chen, Yufei Wang, Bing Li, and Kwok-Yan Lam. 2022. Knowledge is flat: A seq2seq generative framework for various knowledge graph completion. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4005–4017.

Chen Chen, Yufei Wang, Aixin Sun, Bing Li, and Kwok-Yan Lam. 2023. Dipping plms sauce: bridging structure and text for effective knowledge graph completion via conditional soft prompting. In *61st Annual Meeting of the Association for Computational Linguistics, ACL 2023*, pages 11489–11503. Association for Computational Linguistics.

Zhongwu Chen, Long Bai, Zixuan Li, Zhen Huang, Xiaolong Jin, and Yong Dou. 2024. A new pipeline for knowledge graph reasoning enhanced by large language models without fine-tuning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1366–1381.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Xiou Ge, Yun Cheng Wang, Bin Wang, and C-C Jay Kuo. 2023. Compounding geometric operations for knowledge graph completion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6947–6965.

Yuxia Geng, Jiaoyan Chen, Yuhang Zeng, Zhuo Chen, Wen Zhang, Jeff Z Pan, Yuxiang Wang, and Xiaoliang Xu. 2023. Prompting disentangled embeddings for knowledge graph completion with pre-trained language model. *Available at SSRN 4790015*.

Lingbing Guo, Zhongpu Bo, Zhuo Chen, Yichi Zhang, Jiaoyan Chen, Yarong Lan, Yangyifei Luo, Qian Li, Qiang Zhang, Wen Zhang, et al. 2024a. Mkgl: Mastery of a three-word language. In *38th Conference on Neural Information Processing Systems (NeurIPS 2024)*.

Lingbing Guo, Zhongpu Bo, Zhuo Chen, Yichi Zhang, Jiaoyan Chen, Yarong Lan, Mengshu Sun, Zhiqiang Zhang, Yangyifei Luo, Qian Li, et al. 2024b. Mkgl: Mastery of a three-word language. *arXiv preprint arXiv:2410.07526*.

Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3549–3568.

Jiabang He, Jia Liu, Lei Wang, Xiyao Li, and Xing Xu. 2024. Mocosa: Momentum contrast for knowledge graph completion with structure-augmented pretrained language models. In *2024 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE.

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

Pengcheng Jiang, Shivam Agarwal, Bowen Jin, Xuan Wang, Jimeng Sun, and Jiawei Han. 2023. Text-augmented open knowledge graph completion via pre-trained language models. In *61st Annual Meeting of the Association for Computational Linguistics, ACL 2023*, pages 11161–11180. Association for Computational Linguistics (ACL).

Pengcheng Jiang, Lang Cao, Cao Xiao, Parminder Bhatia, Jimeng Sun, and Jiawei Han. 2024a. Kg-fit: Knowledge graph fine-tuning upon open-world knowledge. *Advances in neural information processing systems*.

Pengcheng Jiang, Lang Cao, Cao Xiao, Parminder Bhatia, Jimeng Sun, and Jiawei Han. 2024b. Kg-fit: Knowledge graph fine-tuning upon open-world knowledge. *arXiv preprint arXiv:2405.16412*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Haotian Li, Bin Yu, Yuliang Wei, Kai Wang, Richard Yi Da Xu, and Bailing Wang. 2023. Kermit: Knowledge graph completion of enhanced relation modeling with inverse transformation. *arXiv preprint arXiv:2309.14770*.

Ren Li, Yanan Cao, Qiannan Zhu, Guanqun Bi, Fang Fang, Yi Liu, and Qian Li. 2022a. How does knowledge graph embedding extrapolate to unseen data: a semantic evidence view. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 5781–5791.

Rui Li, Jianan Zhao, Chaozhuo Li, Di He, Yiqi Wang, Yuming Liu, Hao Sun, Senzhang Wang, Weiwei Deng, Yanming Shen, et al. 2022b. House: Knowledge graph embedding with householder parameterization. In *International conference on machine learning*, pages 13209–13224. PMLR.

Xixun Lin, Wenxiao Zhang, Fengzhao Shi, Chuan Zhou, Lixin Zou, Xiangyu Zhao, Dawei Yin, Shirui Pan, and Yanan Cao. 2024. Graph neural stochastic diffusion for estimating uncertainty in node classification. In *41st International Conference on Machine Learning (PMLR)*. MLResearchPress.

Yu Liu, Yanan Cao, Shi Wang, Qingyue Wang, and Guanqun Bi. 2024. Generative models for complex logical reasoning over knowledge graphs. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 492–500.

Zhenghao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2018. Entity-duet neural ranking: Understanding the role of knowledge graph semantics in neural information retrieval. *arXiv preprint arXiv:1805.07591*.

Linhao Luo, Yuan-Fang Li, Reza Haf, and Shirui Pan. 2024. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations*.

Xin Lv, Yankai Lin, Yixin Cao, Lei Hou, Juanzi Li, Zhiyuan Liu, Peng Li, and Jie Zhou. 2022. Do pre-trained models benefit knowledge graph completion? a reliable evaluation and a reasonable approach. Association for Computational Linguistics.

Jesse G Meyer, Ryan J Urbanowicz, Patrick CN Martin, Karen O'Connor, Ruowang Li, Pei-Chen Peng, Tiffani J Bright, Nicholas Tatonetti, Kyoung Jae Won, Graciela Gonzalez-Hernandez, et al. 2023. Chatgpt and large language models in academia: opportunities and challenges. *BioData Mining*, 16(1):20.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. 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 3982–3992.

Tara Safavi and Danai Koutra. 2020. Codex: A comprehensive knowledge graph completion benchmark. *arXiv preprint arXiv:2009.07810*.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15*, pages 593–607. Springer.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*.

Zhaoxuan Tan, Zilong Chen, Shangbin Feng, Qingyue Zhang, Qinghua Zheng, Jundong Li, and Minnan Luo. 2023. Kracl: Contrastive learning with graph context modeling for sparse knowledge graph completion. In *Proceedings of the ACM Web Conference 2023*, pages 2548–2559.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. 2024. Graphgpt: Graph instruction tuning for large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 491–500.

Komal Teru, Etienne Denis, and Will Hamilton. 2020. Inductive relation prediction by subgraph reasoning. In *International Conference on Machine Learning*, pages 9448–9457. PMLR.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, pages 57–66.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2019. Composition-based multi-relational graph convolutional networks. *arXiv preprint arXiv:1911.03082*.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Changjian Wang, Xiaofei Zhou, Shirui Pan, Linhua Dong, Zeliang Song, and Ying Sha. 2022a. Exploring relational semantics for inductive knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 4184–4192.

Liang Wang, Wei Zhao, Zhuoyu Wei, and Jingming Liu. 2022b. 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.

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, James T Kwok, and Yu Zhang. 2024. Kicgpt: Large language model with knowledge in context for knowledge graph completion. *arXiv preprint arXiv:2402.02389*.

Yanbin Wei, Qiushi Huang, Yu Zhang, and James Kwok. 2023. Kicgpt: Large language model with knowledge in context for knowledge graph completion. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Derong Xu, Ziheng Zhang, Zhenxi Lin, Xian Wu, Zhihong Zhu, Tong Xu, Xiangyu Zhao, Yefeng Zheng, and Enhong Chen. 2024. Multi-perspective improvement of knowledge graph completion with large language models. *arXiv preprint arXiv:2403.01972*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Rui Yang, Jiahao Zhu, Jianping Man, Li Fang, and Yi Zhou. 2024a. Enhancing text-based knowledge graph completion with zero-shot large language models: A focus on semantic enhancement. *Knowledge-Based Systems*, 300:112155.

Rui Yang, Jiahao Zhu, Jianping Man, Li Fang, and Yi Zhou. 2024b. Exploiting large language models capabilities for question answer-driven knowledge graph completion across static and temporal domains. *arXiv preprint arXiv:2408.10819*.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Kg-bert: Bert for knowledge graph completion. *arXiv preprint arXiv:1909.03193*.

Liang Yao, Jiazhen Peng, Chengsheng Mao, and Yuan Luo. 2023. Exploring large language models for knowledge graph completion. *arXiv preprint arXiv:2308.13916*.

Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. 2023. Natural language is all a graph needs. *arXiv preprint arXiv:2308.07134*.

Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. 2019. Graph transformer networks. *Advances in neural information processing systems*, 32.

Honggen Zhang, June Zhang, and Igor Molybog. 2024a. Hasa: Hardness and structure-aware contrastive knowledge graph embedding. In *Proceedings of the ACM on Web Conference 2024*, pages 2116–2127.

Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019. Quaternion knowledge graph embeddings. *Advances in neural information processing systems*, 32.

Yichi Zhang, Zhuo Chen, Lingbing Guo, Yajing Xu, Wen Zhang, and Huajun Chen. 2024b. Making large language models perform better in knowledge graph completion. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 233–242.

Yichi Zhang, Zhuo Chen, Wen Zhang, and Huajun Chen. 2023. Making large language models perform better in knowledge graph completion. *arXiv preprint arXiv:2310.06671*.

Yongqi Zhang and Quanming Yao. 2022. Knowledge graph reasoning with relational digraph. In *Proceedings of the ACM web conference 2022*, pages 912–924.

Zhanqiu Zhang, Jianyu Cai, Yongdong Zhang, and Jie Wang. 2020. Learning hierarchy-aware knowledge graph embeddings for link prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3065–3072.

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

A Statistical information of datasets

In our experiments, we evaluate SAT on four popular benchmark datasets. The detailed descriptions of these datasets are provided below:

- **FB15k-237N** (Lv et al., 2022) is an improved version of FB15k-237 derived from Freebase, a large-scale KG containing factual knowledge.
- **CoDeX-S** (Safavi and Koutra, 2020) is an encyclopedic KG extracted from Wikidata with three versions, among which we focus on CoDeX-S.
- **FB15k-237** (Toutanova and Chen, 2015) is a subset of FB15k (Bordes et al., 2013) with inverse relations removed to avoid data leakage.
- **YAGO3-10** (Suchanek et al., 2007) is a subset of YAGO, which is a large knowledge base derived from multiple external sources like Wikipedia.

B Detailed Descriptions of Baselines

The compared baselines can be divided into two categories: traditional KGC methods and LLM-enhanced KGC methods. For all compared methods, we provide a brief overview for reference.

- **Traditional KGC methods.** *i) Embedding-based methods:* TransE (Bordes et al., 2013) models relations as translations on entity embeddings. DistMult (Yang et al., 2014) introduces bilinear models for learning representations. ComplEx (Trouillon et al., 2016) represents vectors in the complex space, while RotatE (Sun et al., 2019) extends this by modeling each relation as a rotation. Furthermore, several variants have been proposed to improve KGC performance, including QuatE (Zhang et al., 2019), HAKE (Zhang et al., 2020), HousE (Li et al., 2022b), and CompoundE (Ge et al., 2023). *ii) GNN-based methods:* GNN-based approaches, such as R-GCN (Schlichtkrull et al., 2018), GraiL (Teru et al., 2020), and CFAG (Wang et al., 2022a), leverage the inherent structural patterns of KGs. More recently, NBFNet (Zhu et al., 2021) solves path formulation using the neural Bellman-Ford network. RED-GNN (Zhang and Yao, 2022) introduces a novel relational directed graph for KG reasoning. *iii) Transformer-based methods:* Most methods take advantage of PLMs by integrating textual information, such as KG-BERT (Yao et al., 2019), PKGC (Lv et al., 2022), and TAGREAL (Jiang et al., 2023). Besides, SimKGC (Wang et al., 2022b), HaSa (Zhang et al., 2024a), KRACL (Tan et al., 2023), and MoCoSA (He et al., 2024) enhance PLMs through contrastive learning techniques.
- **LLM-enhanced KGC methods.** *i) Text-based methods:* Recently, LLMs have been introduced to further improve KGC performance. KG-LLaMA (Yao et al., 2023) first investigates KGC tasks using various LLMs. KICGPT (Wei et al., 2024) provides explicit instructions to guide the behavior of LLMs based on retrieved triples. KERMIT (Li et al., 2023) employs LLMs for generating predictive descriptions, while MPIKGC (Xu et al., 2024) prompts LLMs to generate auxiliary texts from various perspectives. Additionally, GS-KGC (Yang et al., 2024b) and KG-FIT (Jiang et al., 2024b) leverage LLMs for graph structure augmentation. *ii) Embedding-based methods:* CSProm-KG (Chen et al., 2023) investigates PLMs for KGC through structural soft prompts. Structural-aware IT (Zhang et al., 2024b) integrates structural knowledge into LLMs. KoPA (Zhang et al., 2024b) incorporates structural encoding into LLMs through a prefix adapter. MKGL (Guo et al., 2024b) further introduces a specialized KG language via KGL token embedding augmentation.

C Implementation Settings

In our implementation, we utilize Llama2-Chat-7B as the LLM backbone, fine-tuned on the training split of public datasets for 3 epochs. For hierarchical knowledge alignment, we employ a graph transformer as the graph encoder and a vanilla transformer as the text encoder. The local alignment is trained first, followed by the global alignment. During training, we set the learning rate to 1e-4, the dropout rate to 0.1, and use Adam optimizer. The dimensions for graph embedding and word embedding are set to 128, and the maximum text sequence length is limited to 256 tokens. In structural instruction tuning, the parameters of the pre-trained graph encoder and the LLM are frozen, while the knowledge adapter is fine-tuned exclusively during the instruction tuning stage. The training epoch is set to 3, the batch size is 2, and the learning rate is 2e-3, with a warmup ratio of 3e-2. Additionally, we sample 2-hop neighborhoods to build subgraphs, and the maximum input length for the LLM is capped at 2048 tokens. Our model is implemented in Pytorch and trained on two Nvidia A800 GPUs.

D Reproducibility Explanation

In the paper, we compare our method with representative baselines. To ensure a fair comparison, we report results from their original publications when available. For PKGC, CompGCN, SimKGC, and HaSa, we retrain the models using their official codes under the same experimental settings, as the original papers did not include results on some of the datasets used in our study. In addition, we evaluate the performance of GPT-4 (gpt-4-turbo) through OpenAI's publicAPI.

E Experimental Results

E.1 Triple Classification on Large Datasets

Table 8 presents the results for triple classification on FB15k-237 and YAGO3-10. As shown, SAT achieves relative F1 improvements of 1.3% and 2.4% over PKGC, respectively. Additionally, when compared to GPT-4, our model also demonstrates 14.9% and 5.8% relative improvements. These findings further emphasize the effectiveness of our model across diverse datasets.

Table 8: Results of triple classification on large datasets.

Model	FB15k-237		YAGO3-10	
	Acc	F1	Acc	F1
PKGC*	0.823	0.846	0.807	0.831
GPT-4*	0.726	0.615	0.781	0.643
SAT (ours)	0.834	0.822	0.826	0.795

E.2 Model Parameter Supplement

We provide a detailed supplement of model parameters in Table 9. As shown, compared to the knowledge alignment phrase (GNN+PLM), the instruction tuning phrase (LLM) results in a substantial increase in training parameters (525M, 8.0%) and time (23H). This underscores that tuning the LLM is a significant computational bottleneck.

E.3 Global Alignment Data Statistics

The statistics for global alignment data are summarized in Table 10. It comprises 12,000 pairs, with an average processing time of 2.41 seconds per document. Note that FB15k-237N and FB15-237 share the same pairs. On average, each document contains approximately 490 tokens, and the corresponding subgraph includes around 12 triples.

Table 9: Model parameters of SAT during training stage.

Model	Time	Params(%)	Params(#)
Ours (Tune GNN+PLM)	5H	1.0%	65,360,896
Ours (Tune LLM)	23H	8.0%	525,893,632
Ours (Tune All)	28H	8.9%	591,254,528
Ours (Pretrain All)	OOM	100%	6,609,987,584

Table 10: Statistical analysis of global alignment data.

Datasets	Pairs(#)	Tokens(#)	Triples(#)	Time(H)
CoDeX-S	1,000	518	12	0:42:37
FB15k-237	6,000	481	11	3:55:12
YAGO3-10	5,000	495	12	3:23:20

E.4 More Representation Visualizations

To visualize the aligned representation space, we further incorporate more visualizations, using a new sample dataset comprising about 500 entities across 3 classes from FB15k-237N.

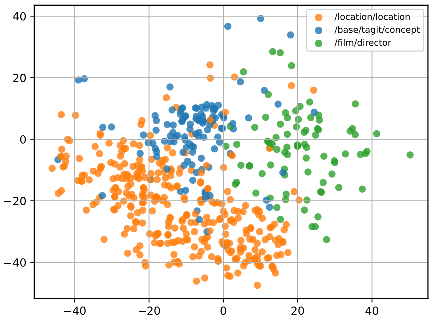
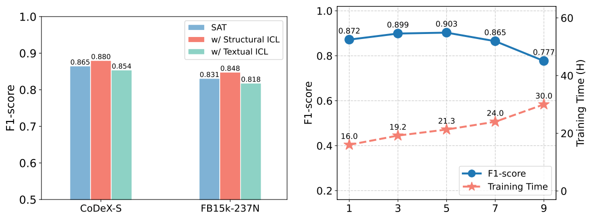


Figure 7: Visualization of graph embeddings.

F In-Context Learning

Recently, in-context learning (ICL) has emerged as a powerful technique that incorporates task examples directly into the prompts. However, the effectiveness of ICL is significantly dependent on the selection of relevant examples. Unlike previous methods that rely primarily on the textual semantics of the query, we propose the structural ICL, which retrieves structure-related examples. Specifically, we build a candidate pool $\mathcal{C} = \{C_1, C_2, ...\}$, where each candidate is represented as $\mathcal{C}_i = (Q_i, S_i)$. We then select the top-k relevant examples based on the similarity φ between the current query subgraph S_q and each candidate subgraph S_i, as follows,

$$\mathcal{K}_{\text{top-k}} = \underset{\{C_i\}_{i=1}^{k}}{\text{argmax}} \sum_{i=1}^{k} \varphi\left(\boldsymbol{h}_{S_q}, \boldsymbol{h}_{S_i}\right). \quad (10)$$



(a) Effect of structural ICL (b) Effect of example numbers

Figure 8: Additional experiments on ICL.

Effect of Structural ICL. To assess the impact of structural ICL, we implement experiments with different retrieval mechanisms. As shown in Figure 8(a), we observe a noticeable improvement when structural ICL is introduced. Moreover, structural ICL outperforms textual ICL, which we attribute to the fact that textual similarity is insufficient for capturing graph structures. In contrast, structural ICL can effectively retrieve structure-related examples. Note that these experiments are conducted using 3-shot examples.

Effect of Example Numbers. As known, the effectiveness of ICL is significantly influenced by the number of retrieved examples. To determine the optimal number of examples, we perform various experiments on CoDeX-S, as shown in Figure 8(b). Our findings reveal that while the training time increases linearly with the number of instances, the performance does not exhibit a corresponding upward trend. To balance effectiveness and efficiency, we set the optimal ICL number to 3.

G Instruction Templates

G.1 Instruction for Data Construction

We design an instruction to prompt GPT-4 to extract subgraphs from the provided documents for the construction of global alignment data.

G.2 Graph Instruction for KGC tasks

We present instruction templates for two key KGC tasks: triple classification and link prediction.

G.3 Graph Instruction with Different Graph Information

We design three modes for graph instruction: base instruction (without graph information), triple instruction (with a single triple), and graph instruction (with subgraph information). Templates for triple classification are provided, with templates for link prediction developed using the same way.

G.4 Graph Instruction with Multiple External Resources

In addition to the graph structure, we further enrich graph instruction by incorporating external resources, such as entity names and text descriptions. Again, specific templates for triple classification are provided, while templates for link prediction are adapted from this framework.

Instruction templates for global data construction

Task Instruction: Given a document, please extrapolate as many relationships as you can from the document and generate triples like (source, relation, target).

Example

Input: Steven Paul Jobs was an American businessman, inventor, and investor best known for co-founding the technology giant Apple Inc. Jobs was also the founder of NeXT and chairman and majority shareholder of Pixar. He was a pioneer of the personal computer revolution of the 1970s and 1980s, along with his early business partner and fellow Apple co-founder Steve Wozniak.

Output: (Steven Paul Jobs, nationality, American); (Steven Paul Jobs, occupation, businessman); (Steven Paul Jobs, occupation, inventor); (Steven Paul Jobs, co-founder of, Apple Inc.); (Steven Paul Jobs, founder of, NeXT); (Steven Paul Jobs, chairman of, Pixar); (Steven Paul Jobs, business partner, Steve Wozniak);

Input: Original Document
Output: Generated Triples

Instruction templates for two KGC tasks

Task Instruction: You are an assistant specializing in large language models and knowledge graphs. Please follow the instructions carefully and provide your responses.

Triple Classification

Input: Given a triple {*triple*} that consists of a head entity, a relation, and a tail entity. Given a sequence of graph embeddings {*graph*} that represent a subgraph of the triple extracted from a knowledge graph. Please determine the correctness of the input triple and response True or False.

Output: True or False

Link Prediction

Input: Given a question:{*question*} that represents a natural language question. Given a sequence of graph embeddings {*graph*} that represent a subgraph of the question extracted from a knowledge graph. Please answer the input question, and keep the answer as simple as possible.

Output: Target Answers

Instruction templates for graph information

Task Instruction: You are an assistant specializing in large language models and knowledge graphs. Please follow the instructions carefully and provide your responses.

Base Instruction
Input: Given a triple {*triple*} that consists of a head entity, a relation, and a tail entity. Please determine the correctness of the input triple and response True or False.
Output: True or False

Triple Instruction
Input: Given a triple {*triple*} that consists of a head entity, a relation, and a tail entity. Given a sequence of graph embeddings {*graph*} that represent the anchor entities of the triple. Please determine the correctness of the input triple and response True or False.
Output: True or False

Graph Instruction
Input: Given a triple {*triple*} that consists of a head entity, a relation, and a tail entity. Given a sequence of graph embeddings {*graph*} that represent a subgraph of the triple extracted from a knowledge graph. Please determine the correctness of the input triple and response True or False.
Output: True or False

Instruction templates for external sources

Task Instruction: You are an assistant specializing in large language models and knowledge graphs. Please follow the instructions carefully and provide your responses.

Entity Names
Input: Given a triple {*triple*} that consists of a head entity, a relation, and a tail entity. Given a sequence of graph embeddings {*graph*} that represent a subgraph of the triple extracted from a knowledge graph. Each graph node contains an entity name. Here is a list of entity names: {*name*}. Please determine the correctness of the input triple and response True or False.
Output: True or False

Text Descriptions
Input: Given a triple {*triple*} that consists of a head entity, a relation, and a tail entity. Given a sequence of graph embeddings {*graph*} that represent a subgraph of the triple extracted from a knowledge graph. Each graph node contains an entity description. Here is a list of entity textual descriptions: {*description*}. Please determine the correctness of the input triple and response True or False.
Output: True or False

Entity Names + Text Descriptions
Input: Given a triple {*triple*} that consists of a head entity, a relation, and a tail entity. Given a subgraph of the triple extracted from a knowledge graph. Each graph node contains an entity name and its textual description information. Here is a list of entity information: {*name*} and {*description*}. Please determine the correctness of the input triple and response True or False.
Output: True or False