PathGPT: Path-enhanced LLMs for Knowledge Graph Completion

Tao Huang[1,2], Junwei Zhang[1,*], Pengju Yan[1], Rui Yang[3], Xiaolin Li[1,2,*]

[1]Hangzhou Institute of Medicine, Chinese Academy of Sciences, Hangzhou, China
[2]University of Chinese Academy of Sciences, Beijing, China
[3]School of Nursing, Zhejiang Chinese Medical University, Hangzhou, China
[*]*Corresponding author: xiaolinli@ieee.org, zhangjunwei@him.cas.cn*

***Abstract*—Knowledge graph completion requires establishing credible reasoning chains between structure and text. However, existing methods are either confined to low-dimensional embeddings or neglect explicit graph structures, leading to hallucinations and insufficient explainability. To address this, this paper proposes PathGPT, a path-enhanced large language model framework. The core idea of PathGPT is to use key paths as evidence, textual paths as prompts, and adapters as bridges to achieve deep synergy between structural reasoning and semantic generation. Specifically, PathGPT first efficiently screens key paths from an exponential path space using a path model to obtain continuous embeddings of key paths. Subsequently, it maps the embeddings into the word vector space of the large language model through a lightweight path-text alignment adapter, and injects human-readable path text as a prefix prompt to guide the model's explainable prediction of triplets. Experiments on three benchmarks demonstrate that PathGPT significantly outperforms existing state-of-the-art methods in terms of accuracy and F1 score. Moreover, PathGPT possesses explainability and generalisation ability, effectively integrating structural path information and providing a new technical route for knowledge graph completion.**

***Index Terms*—Knowledge Graph Completion, Large Language Models, Knowledge Graphs.**

I. Introduction

Knowledge Graphs (KGs) represent real-world knowledge as structured triplets (head entity, relation, tail entity), serving as a critical infrastructure for AI systems to perform semantic reasoning. Its capacity to formalize domain-specific knowledge has facilitated advances in diverse AI applications including question answering [1] and recommender systems [2]. Large-scale KGs such as DBpedia [3], Freebase [4], and NELL [5] contain extensive amounts of observed knowledge. Yet their incompleteness severely limits the utility of knowledge graphs in decision-critical scenarios. To address this, Knowledge Graph Completion (KGC) aims to predict missing relational triples through reasoning over existing KG structures and auxiliary textual data [6].

Research in KGC primarily follows three paradigms: Embedding-based methods learn low-dimensional vector representations of entities and relations, scoring triplets through geometric operations. Although they effectively capture local graph structures, they struggle to model textual semantics. Path-based methods predict relations by enumerating and scoring multi-hop paths between entities. The third approach employs pre-trained language models using textual descriptions to evaluate the likelihood of triples. Recently, Large Language Models (LLMs) have demonstrated remarkable performance across various applications, particularly in knowledge reasoning tasks such as zero-shot reasoning and instruction tuning [7]. However, relying solely on individual triples and their descriptions may lead to certain limitations.

Although LLM-based KGC methods, like KG-LLaMA [8], achieve initial success via simple prompt engineering, they overlook the rich structural dependencies in knowledge graphs. Moreover, LLMs struggle to provide precise factual knowledge and often generate inaccurate or inconsistent information. Recent work [7] integrates the complex structural information

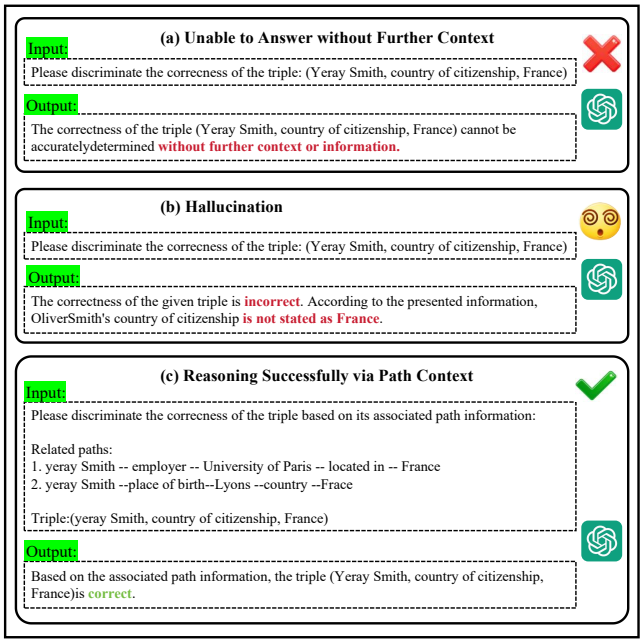

Fig. 1. A simple and intuitive example. By providing path context, LLM produces more grounded predictions and less hallucination.

This work is supported in part by funds from the National Key Research and Development Program of China (2022YFC3600902), Key Research and Development Program of Zhejiang (2025C01129) and the National Science Technology Major Project (No. 2023ZD0509706).

of KGs into LLM. This integration has been shown to enhance the model's understanding of KG. However, this approach has two limitations: (1) Modality misalignment between KG structure and LLM textual space, and (2) Path neglect, where triple-level integration fails to leverage relational paths for deeper reasoning.

To address this, we argue that explicit path integration is essential. Including inter-entity path information helps assess the validity of a triple. Fig. 1 shows a simple and intuitive example. Reasoning over KG paths enables LLMs to generate more accurate responses and reduce hallucinations. However, the exponential growth in the number of paths between entities presents a challenge when integrating path information within the limited input window of current LLMs.

This paper proposes PathGPT—a KGC framework for LLMs. The core idea is to explicitly introduce high-quality reasoning paths between entities and bridge the modality gap through a learnable alignment mechanism. This maximizes the structural reasoning capabilities of LLMs within a limited input window. PathGPT adopts a two-phase strategy for effectively incorporating KG paths into LLMs to enhance reasoning capabilities. The framework operates through:

- **Structural Path Sampling**: During pre-training, the model learns KG structural semantics by extracting multi-hop paths for each pair of entities awaiting relation completion. Using a scoring function, we selectively retain the top-N most discriminative paths, effectively filtering noise while preserving critical information.
- **Path-Aware Inference**: A lightweight adapter compresses the filtered paths into the LLM's embedding space, generating fixed-length path-aware virtual tokens. Notably, we simultaneously integrate the paths' textual descriptions into prompt templates. These processed tokens are prepended to the text prompt, creating a prefix-enhanced input sequence. For efficient tuning, we implement LoRA [9] fine-tuning that only updates the adapter and LoRA parameters, maintaining the frozen LLM backbone. This approach achieves profound structure-semantics integration while preventing catastrophic forgetting and preserving the model's inherent linguistic capabilities.

PathGPT effectively integrates path information from KGs with the language generation capabilities of LLMs, thereby achieving significant performance improvements in knowledge reasoning tasks. We conducted experiments on three public benchmarks: UMLS, CoDeX-S, and FB15K-237N. The results demonstrate that PathGPT significantly outperforms 18 baseline methods (including embedding, path-based, and pure LLM approaches) under the instruction fine-tuning setting, achieving the best scores in terms of accuracy and F1 score, and effectively suppressing hallucination.

In summary, this work makes two key contributions: (1) proposing a knowledge graph completion framework that effectively selects and integrates essential path information with LLMs; and (2) validating the framework's effectiveness through comprehensive experiments on three benchmark datasets.

II. Related Work

A. Knowledge Graph Completion

Knowledge Graph Completion (KGC) aims to infer missing triples by reasoning over structural and semantic information in existing knowledge graphs (KGs). This task mitigates the inherent incompleteness of real-world knowledge graphs and can be divided into subtasks such as triple classification [7], entity prediction [10] and Relation Prediction [11]. In this study, we primarily focus on the triple classification task.

KGC primarily employs three approaches: embedding-based methods, path-based methods, and methods based on pre-trained language models (PLMs). Embedding-based methods utilize structural information from KGs and employ negative sampling [12] to embed entities and relations into a continuous representation space.However, embedding-based methods [12], [13] largely ignore textual semantics within KGs.

Path-based methods, on the other hand, predict relations by considering the multi-hop paths between entities, offering interpretable reasoning chains. These methods are essential in KGC tasks due to their interpretability, inductive capacity, and high performance [10], [14]. They provide important insights into the relations between entities in KGs.

PLM-based approaches formulate KGC as text-based tasks by encoding triples or entity descriptions into contextualized embeddings to fine-tune pre-trained language models such as BERT [15]. KG-BERT [16] is one of the earliest methods in this paradigm, framing the task as binary text classification. Subsequent approaches like MTL-KGC [17] and StAR [18] extend this direction by incorporating additional training objectives and more sophisticated triple encoding mechanisms. Despite leveraging PLMs for semantic understanding, these methods often struggle to model the structural information inherent in KGs.

Embedding methods excel at capturing local structures but neglect text, while PLM methods do the opposite. PathGPT integrates both approaches by using structured path embeddings to provide explicit topological constraints for LLMs, thereby balancing semantics and structure.

B. Path-based Reasoning

Path-based methods are a popular approach for knowledge graph reasoning, as they leverage the relational paths connecting entities to infer new relations. Early methods such as Path Ranking [19] focused on collecting symbolic features in the form of relational paths for classification. Path-RNN [20] and PathCon [21] improve Path Ranking by learning the representations of paths with recurrent neural networks (RNN). These methods enumerated all possible paths between entities and extracted features from them to make predictions. DeepPath [22] and MINERVA [23] aimed to overcome the limitations of these early methods by using reinforcement learning to train an agent that could learn to collect meaningful

paths. These approaches faced challenges, primarily due to the sparse rewards. The training process was not efficient as the agent had to explore a vast search space of possible paths, making it difficult to converge.

To improve upon these challenges, subsequent works focused on enhancing the reward function [24] or search strategy [25]. They also explored the usage of multiple agents [26] or introduced a variational formulation [27] for path sampling. By engineering these aspects, researchers were able to train the agents more effectively and derive better results.

Another category of methods, such as NBFNet [28] and RED-GNN [29], both employ an algorithm similar to the Bellman-Ford algorithm to learn path representations from a single source entity to all other entities. This algorithmic adaptation enabled efficient learning of path representations for various entities in the knowledge graph. A*Net [10] is a novel method that addresses the scalability issue faced by existing dynamic programming methods. It introduces a priority function to guide the search process and only explores.

The early path-based methods relied on heuristic enumeration or reinforcement learning, which were inefficient and difficult to align with language models. PathGPT, through the efficient path sampling of A*Net and a learnable alignment adapter, injects paths into LLMs. This approach not only preserves the interpretability of paths but also avoids the exponential search costs.

C. LLMs for Knowledge Graph Completion

LLMs like GPT-4 [30] have revolutionized text understanding and generation through self-supervised pre-training on massive corpora, where they learn contextualized representations via next-token prediction [31].To ensure that LLMs adhere to human instructions and generate responses consistent with human values, techniques like instruction tuning [32] and human preference alignment are employed.

The integration of LLMs with KGs [33] has gained considerable attention and is considered a crucial research area. KGs play a vital role in mitigating the issue of hallucination [34], [35] in LLMs by introducing factual knowledge. By incorporating KGs into LLMs, various KG-related tasks such as KGC, entity alignment, and KG question answering can be greatly enhanced.

Recent efforts to adapt LLMs for KGC fall into two categories. Methods like KG-LLaMA [8] treat triplets as plain text and fine-tune LLMs for sequence classification. While simple, this approach discards graph topology, leading to suboptimal performance on multi-hop reasoning. Methods like KoPA [7] map KG embeddings into LLM input spaces via adapter layers. However, gap between KG embeddings and textual semantics still remains. There is still a need for more profound research and systematic investigation in this area.

One approach to address this gap involves incorporating the path information of KGs into LLMs. This integration can enhance the models' understanding of KG facts, facilitating interpretable, more accurate and contextually appropriate completions. However, the number of paths in the sub-graph between entities is exponential, and bridging the gap between the path information and the textual semantic space of LLMs is still a challenge.

III. PathGPT: The Proposed Framework

We leverage multi-hop path information in KGs to enhance LLMs for triple completion. Directly converting paths to text introduces significant noise and rapidly exhausts the limited context window, as paths grow exponentially. To address this, we propose PathGPT, a two-stage framework that injects core KG paths into LLMs efficiently (Fig. 2).

- **Stage One: Structural Path Sampling**. We pre-train a model to learn the KG's structural semantics. For each target triple, we extract multi-hop paths, score their importance, and retain only the top-N most discriminative paths to reduce redundancy.
- **Stage Two: Path-Aware Inference**. A lightweight adapter embeds the selected paths into the LLM's word vector space, generating fixed-length virtual path tokens. We integrate these tokens into the prompt by prepending them to text embeddings, creating a prefix-enhanced input sequence. We fine-tune only the adapter and LoRA parameters, keeping the LLM's backbone frozen. This achieves structure-semantics fusion without catastrophic forgetting.

A. Structural Path Sampling

Initially, PathGPT employs a pre-trained path model to extract key paths between head entity h and tail entity t in the knowledge graph. For an incomplete triple $(h, ?, t)$, the model performs path reasoning leveraging the graph structure to identify the most relevant connecting paths. We adopt A*Net as the path model, which directly extracts multi-hop relational paths between h and t while filtering them by importance. These paths not only capture complex inter-entity relationships but are also automatically evaluated through a learned scoring module, enabling selection of the most representative key paths.

We pre-train the path model $\mathcal{A}^*$ using the graph data $\mathcal{G}$,

$$\begin{aligned}\mathcal{A}^* &= \mathcal{F}_{\text{A*Net}}(h, t, \mathcal{G}) \\ &= \arg\min_{\theta} \left(- \sum_{(h,r,t)\in\mathcal{G}} \log p(r|h, t; \theta) \right)\end{aligned} \tag{1}$$

Here, r represents the predicted relationship, and θ denotes the training parameters.

Once the path model has been pre-trained, we fix the parameters θ of the path model. Based on the input head entity h, tail entity t, and the predefined Top-N quantity N, we extract the embeddings $\mathcal{E}_{\text{Top-N}}$ of the key paths and their textual representations $\mathcal{T}_{\text{Top-N}}$ from the path model $\mathcal{A}^*$:

$$(\mathcal{E}_{\text{Top-N}}, \mathcal{T}_{\text{Top-N}}) = \mathcal{F}(\mathcal{A}^*(\theta), h, t, N) \tag{2}$$

where:

$$\mathcal{E}_{\text{Top-N}} = \{e(P) \mid P \in \mathcal{P}_{\text{Top-N}}\} \tag{3}$$

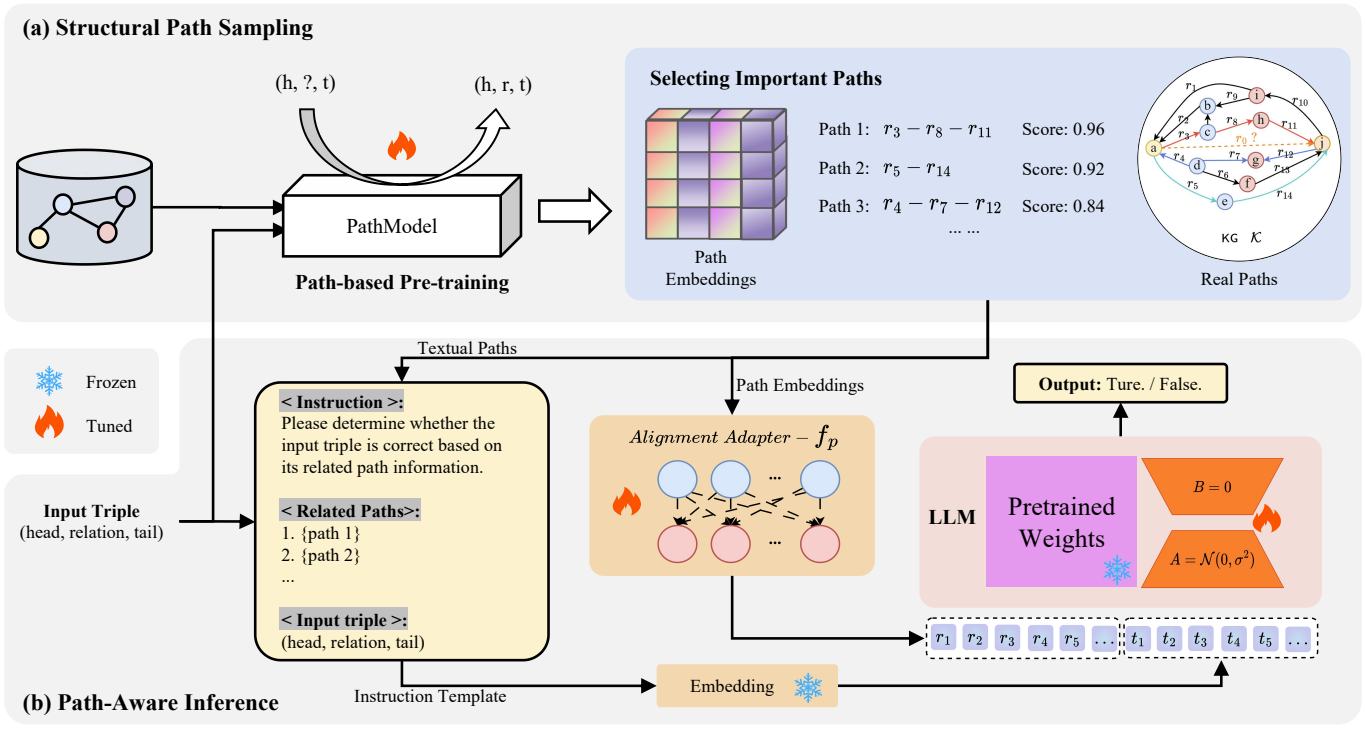

Fig. 2. An Overview of PathGPT. (a) Structural Path Sampling. By employing a pre-trained path model, multi-hop paths between two entities are extracted. Subsequently, a scoring function is utilised to filter out the most discriminative paths. (b) Path-Aware Inference. The filtered paths are embedded and mapped into the LLM word vector space. Before injecting the path tokens into the text prompt, fine-tuning is achieved through LoRA, thereby enabling a deep integration of structure and semantics.

$$\mathcal{T}_{\text{Top}-\text{N}} = \{T(P) \mid P \in \mathcal{P}_{\text{Top}-\text{N}}\} \tag{4}$$

The $\mathcal{P}_{\text{Top-N}}$ consists of the Top-N paths selected from all paths $\mathcal{P}_{h \rightsquigarrow t}$ from h to t, based on the path importance scores. The importance score of a path is calculated as the weighted sum of the priority function values of all nodes on the path, that is:

$$\mathcal{P}_{\text{Top-N}} = \text{Top-N}\left(\mathcal{P}_{h \rightsquigarrow t}, \sum_{v \in P} s_{h,t}(v)\right) \tag{5}$$

where $s_{h,t}(v)$ denotes the priority function value of node v in the reasoning path from the head entity h to the tail entity t in A*Net, representing the importance of node v.

B. Path-Aware Inference

In Phase One, PathGPT extracted the key paths between head entity h and tail entity t from the knowledge graph and generated the path embeddings $\mathcal{E}_{\text{Top}-\text{N}}$ and their textual representations $\mathcal{T}_{\text{Top}-\text{N}}$. In Phase Two, we further integrated this path information into the LLM to achieve a deep fusion of structure and semantics.

Firstly, to effectively incorporate the path embeddings $\mathcal{E}_{\text{Top}-\text{N}}$ extracted in Phase One into the word vector space of the LLM, we designed a lightweight adapter (MLP, Multi-Layer Perceptron). The role of this adapter is to compress and map the path embeddings to a dimension compatible with the LLM's word vector space. Specifically, for each path $P_i \in \mathcal{P}_{\text{Top}-\text{N}}$, its embedding $e(P_i) \in \mathcal{E}_{\text{Top}-\text{N}}$ is transformed by the adapter f_{MLP} into a fixed-length "virtual path token" v_{P_i}:

$$v_{P_i} = \text{MLP}(\mathcal{E}_{\text{Top}-\text{N}}) = \sigma(\mathbf{W}\mathcal{E}_{\text{Top}-\text{N}} + \mathbf{b}) \tag{6}$$

where $\mathbf{W}$ and $\mathbf{b}$ are the weight and bias parameters of the MLP, and σ is the nonlinear activation function.

Simultaneously, we integrated the textual representation of the paths $\mathcal{T}_{\text{Top}-\text{N}}$ into the prompt to enhance the semantic expression of the paths. Referring to Prefix Tuning [36], we placed the generated virtual path token v_{P_i} before the word embeddings of the text prompt to form a prefix-enhanced input sequence. The original word embedding sequence of the text prompt is $E_{\text{prompt}} = [e_1, e_2, \ldots, e_m]$, where e_i is the embedding of the i-th word, and m is the length of the text prompt. The input sequence enhanced by the path is:

$$E_{\text{enhanced}} = [v_{P_1}, v_{P_2}, \ldots, v_{P_N}, e_1, e_2, \ldots, e_m] \tag{7}$$

This prefix-enhanced approach enables the LLM to perceive the path information related to the knowledge graph structure first when processing the text prompt, thereby better utilising this structured knowledge in the reasoning process.

To optimise model performance, we adopted the efficient fine-tuning strategy of LoRA [9]. LoRA is a parameter-efficient fine-tuning method that adjusts the model's output by inserting low-rank matrices into each layer of the LLM without updating the entire model's parameters. Specifically, for the

TABLE I
COMPARASION AMONG LLM-BASED KGC METHODS IN FIVE DIMENSIONS.

Method	Requires Fine-tuning	Extra KG Info	Include Path Info	KG Embedding	Interpretability
Zero-Shot Recognition	✗	✗	✗	✗	✗
In-Context Learning	✗	✓	✗	✗	✗
Vanilla IT	✓	✗	✗	✗	✗
Path Enhanced IT	✓	✓	✓	✗	✓
KoPA	✓	✓	✗	✓	✗
PathGPT	✓	✓	✓	✓	✓

weight matrix $\mathbf{W}$ of each layer of the LLM, we introduced two low-rank matrices $\mathbf{A}$ and $\mathbf{B}$ such that:

$$\mathbf{W}_{\text{new}} = \mathbf{W} + \mathbf{AB} \tag{8}$$

where the ranks of $\mathbf{A}$ and $\mathbf{B}$ are much lower than that of $\mathbf{W}$, significantly reducing the number of parameters that need to be updated. During fine-tuning, we only updated the parameters of the adapter (the $\mathbf{W}$ and $\mathbf{b}$ of the MLP) and the LoRA parameters (the $\mathbf{A}$ and $\mathbf{B}$), while keeping the main parameters of the LLM frozen. This strategy not only achieved a deep fusion of structure and semantics but also effectively avoided catastrophic forgetting, that is, it did not forget the original knowledge when introducing new knowledge.

We conducted a comprehensive comparison with other KGC methods based on LLMs, as illustrated in Tab. I. This comparison demonstrates the distinct advantages of PathGPT. Unlike the basic paradigms such as Zero-Shot Recognition, In-Context Learning, and Vanilla Instruction Tuning, PathGPT integrates path information embedding into the LLM framework. This integration effectively merges textual information with ternary relationship paths, thereby enriching the model's understanding. Additionally, PathGPT enhances the interpretability of the prompt content, making it more amenable to reasoning processes within LLMs.

IV. EXPERIMENTS

A. Datasets

We employed three publicly available knowledge graph benchmark datasets to validate our model: UMLS [16], CoDeX-S [40], and FB15K-237N [39]. UMLS is a medical knowledge graph, while CoDeX-S and FB15K-237N are encyclopaedic graphs extracted from Wikidata. These benchmark datasets provide challenging evaluations, incorporating difficult-to-distinguish negative samples. This design aims to prevent the occurrence of false negatives in the validation or test datasets [7].

B. Experimental Settings

1) Baseline Methods: In the experiment, we conducted a comprehensive comparison of one of the key subtasks in KGC, namely the triple classification task. To this end, we selected the following four categories of baseline models as the objects of comparison.

- **Embedding-based KGC methods**. We compare our approach with four traditional embedding-based KGC methods: TransE [12], DistMult [37], ComplEx [13], and RotatE [38]. To compute triple plausibility, these methods rely on learned structural embeddings in conjunction with a model-specific scoring function.
- **Path-based methods**. We select three recent path-based methods, including CURL [14], NBFNet [28], and A*Net [10]. Path-based methods aim to learn a representation to predict the triple based on all paths between a pair of entities.
- **PLM-based KGC methods**. KGC methods based on PLM are classic approaches that focus on the triple classification task, treating it as a binary text classification problem. We selected [16] and PKGC [39] as representative methods in this category.
- **LLM-based KGC methods**. LLM-based methods are divided into two categories: training-free and fine-tuning. There are KG-LLaMA [8], KG-Alpaca [8], Zero-shot [7] , ICL [7] and KoPA [7] to be the LLM-based KGC baseline.

In addition, we have proposed two novel baseline methods, namely **Path Prompt** and **Textual Paths**. Specifically, "Path Prompt" guides zero-shot reasoning of LLMs by constructing text contexts related to knowledge graph paths. "Textual" Paths involves transforming path information from the knowledge graph into textual form for instruction fine-tuning of LLMs.

2) Implementation and Detail Settings: We obtain and report the best results using the code and hyperparameters released by the authors of the baseline models. In addition to ChatGPT, the LLM utilised in our experiments is based on the LLaMa framework and has a parameter count of 7 billion. For path-based KGC methods, We reproduced CURL, NBFNet, A*Net based on the author's public code and evaluated it on the triple classification task. For the CURL, we set embedding dimension to 50, maximum path length to 3, learning rate to $1.0e-3$ and use the K-means algorithm to initialize 50 cluster. We selected the best entropy regularization constant β and moving average constant λ based on maximizing the accuracy on the validation set. For NBFNet and A* Net, we choose the optimal number of layers in the range $[1, 7]$ (the maximum path length is equal to the number of layers), and set embedding dimension to 32, learning rate to $5.0e-3$. To adapt Alpaca to the evaluated methods (KoPA, Textual Paths, PathGPT), we apply LoRA [41] with a rank of 32. The fine-tuning process involves a grid search over epochs $3, 4, 5$ and learning rates $1e-4, 3e-4, 5e-4$, optimized by AdamW ($\beta_1 = 0.9, \beta_2 = 0.99$) with a batch size of 12. This configuration results in a computational cost of roughly 200 Nvidia A100 GPU hours per model.

C. Main Results

We evaluated the proposed method using the triplet classification task [12], with the aim of determining whether a triplet (h, r, t) represents a true relationship. All test datasets were balanced in terms of labelling to ensure an equal number of true and false triplets. During the evaluation, we focused primarily on accuracy and the F1 score, as these two metrics

TABLE II
PERFORMANCE COMPARISONS OF DIFFERENT TRIPLE CLASSIFICATION MODELS USING DIFFERENT EVALUATION METRICS.

Dataset	**UMLS**				**CoDeX-S**				**FB15K-237N**			
Metric	**Acc**	**P**	**R**	**F1**	**Acc**	**P**	**R**	**F1**	**Acc**	**P**	**R**	**F1**
Embedding-based methods												
TransE [12]	84.49	86.53	81.69	84.04	72.07	71.91	72.42	72.17	69.71	70.80	67.11	68.91
DistMult [37]	86.38	87.06	86.53	86.79	66.79	69.67	59.46	64.16	58.66	58.98	56.84	57.90
ComplEx [13]	90.77	89.92	91.83	90.87	67.64	67.84	67.06	67.45	65.70	66.46	63.38	64.88
RotatE [38]	92.05	90.17	94.41	92.23	75.68	75.66	75.71	75.69	68.46	69.24	66.41	67.80
Path-based methods												
CURL [14]	81.69	86.43	75.19	80.42	68.57	62.96	90.21	74.16	63.86	64.69	61.03	62.80
NBFNet [28]	92.58	89.81	96.06	92.83	79.34	79.42	79.21	79.32	71.79	67.14	85.32	75.15
A*Net [10]	91.60	86.47	**98.64**	92.15	79.18	72.37	94.42	81.93	69.98	64.33	89.68	74.92
PLM-based methods												
KG-BERT [16]	77.30	70.96	92.43	80.28	77.30	70.96	92.43	80.28	56.02	53.47	97.62	67.84
PKGC [39]	-	-	-	-	-	-	-	-	79.60	-	-	79.50
LLM-based training-free methods												
Zero-shot(Alpaca) [7]	52.64	51.55	87.69	64.91	50.62	50.31	**99.83**	66.91	56.06	53.32	97.37	68.91
Zero-shot(GPT-3.5) [7]	67.58	88.04	40.71	55.67	54.68	69.13	16.94	27.21	60.15	**86.62**	24.01	37.59
ICL [7]	55.52	55.85	52.65	54.21	50.62	50.31	**99.83**	66.91	59.23	57.23	73.02	64.17
Path Prompt(GPT-3.5)	54.92	54.16	63.99	58.66	63.29	65.20	57.00	60.82	60.24	70.36	35.38	47.09
LLM-based fine-tuning methods												
KG-LLaMA [8]	85.77	87.84	83.05	85.38	79.43	78.67	80.74	79.69	74.81	67.37	96.23	79.25
KG-Alpaca [8]	86.01	94.91	76.10	84.46	80.25	79.38	81.73	80.54	69.91	62.71	**98.28**	76.56
Textual Paths (Ours)	86.83	88.83	84.26	86.49	70.76	**94.59**	44.03	60.09	75.16	73.66	78.32	75.92
KoPA (LLaMa 2) [7]	91.45	**95.66**	86.83	91.04	81.94	75.63	94.25	83.92	77.42	71.82	90.26	79.99
KoPA (Alpaca) [7]	92.58	90.85	94.70	92.70	82.74	77.91	91.41	84.11	77.65	70.81	94.09	80.81
PathGPT (Ours)	**94.32**	92.59	96.39	**94.44**	**83.89**	82.72	85.66	**84.17**	**79.51**	75.56	87.24	**80.98**

provide a comprehensive reflection of the model's overall performance on a balanced dataset. Nevertheless, we also included data on precision and recall in the results table to facilitate a thorough analysis and comparison of different models. The main experimental results are shown in Tab. II.

In experiments on multiple datasets, PathGPT consistently outperformed all 18 baseline models. For example, on the UMLS dataset, PathGPT improved accuracy and F1 score by 1.74%. This improvement is primarily attributed to PathGPT's innovative use of pre-trained A*Net embeddings. By effectively integrating these embeddings, PathGPT not only enhanced its performance but also outperformed traditional path-based methods on the larger and more challenging FB15K-237N dataset. Additionally, PathGPT's overall architecture and optimization techniques played a crucial role in achieving these results.

Unlike fine-tuned LLMs, generic LLMs struggle to understand KG information. Despite GPT-3.5-turbo's (175B parameters) strong capabilities, its zero-shot performance on triple classification remains poor. Adding path information to the Path Prompt provides no significant performance improvement. Moreover, training-free methods show biased predictions, frequently resulting in either perfect accuracy or complete failure.

TABLE III
ABLATION STUDY RESULTS ON UMLS.

UMLS	Acc	F1
PathGPT	94.32	94.44
Unidirectional Paths	92.05	92.25
Textual Paths	86.83	86.49
w/o Path Embedding	81.07	82.36
w/ NBFNet	94.17	94.27

In contrast, fine-tuning LLMs enables path information integration, significantly boosting performance. Although Textual Paths augment input prompts with path information, they are less effective than PathGPT. This suggests that PathGPT captures broader semantic information than text-based auxiliary prompts.

D. Ablation Study

To verify the effectiveness of the design, we perform an ablation study of PathGPT on UMLS with different configurations in Tab. III. Compared to the completed model PathGPT, 'Unidirectional Paths' means that only one-way paths from the first entity to the last entity are used, and 'Textual Paths' means using textual paths for instruction tuning. We find that replacing path prefix embeddings with textual paths or unidirectional path embeddings leads to performance decline. Moreover, removing path embedding will cause significant performance degradation. The last variant replaces the A*Net [10] with NBFNet [28].

E. Influence of the Number of Paths

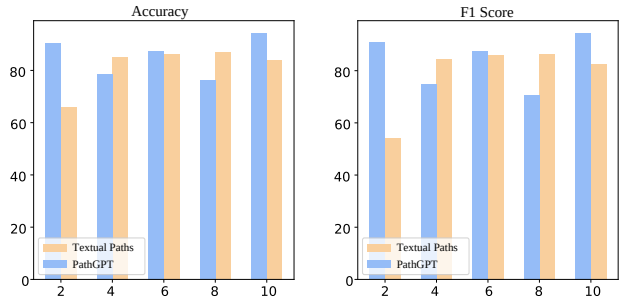

Fig. 3. Results of ACC (**a**) and F1 (**b**) with different number of paths on UMLS. We set all hyperparameters to their optimum values and change the number of paths within [0, 10] in 2-step increments.

Additionally, we evaluate the effect of the number of paths on UMLS. We set all hyperparameters to their optimum values and change the confidence threshold within $[2, 10]$ in 2-step increments. Fig. 3 displays the accuracy and F1 values obtained by PathGPT and Textual Paths. We make the following observations: (1) When the number of paths is 2, the effect of text paths drops significantly, while the effect of PathGPT is still good. It shows that PathGPT has good robustness to the number of paths. (2) When the number of paths is 10, PathGPT is also significantly better than text paths and achieves the best results, indicating that PathGPT can use more paths for reasoning.

F. Case Study

Fig. 4 shows the important paths learned by PathGPT for two test sample in UMLS. Given the query *(anatomical abnormali, result of, health care activity)*, we can see two paths *anatomical abnormali* $\xleftarrow{associated\ with}$ *health care activity*, and *anatomical abnormali* $\xrightarrow{affects}$ *archaeon* $\xleftarrow{process\ of}$ *Cell or molecular dysfunction* $\xrightarrow{result\ of}$ *health care activity* are consistent with human cognition. This proves that the reasoning capability of LLMs can be effectively improved by injecting path information embedding into LLMs.

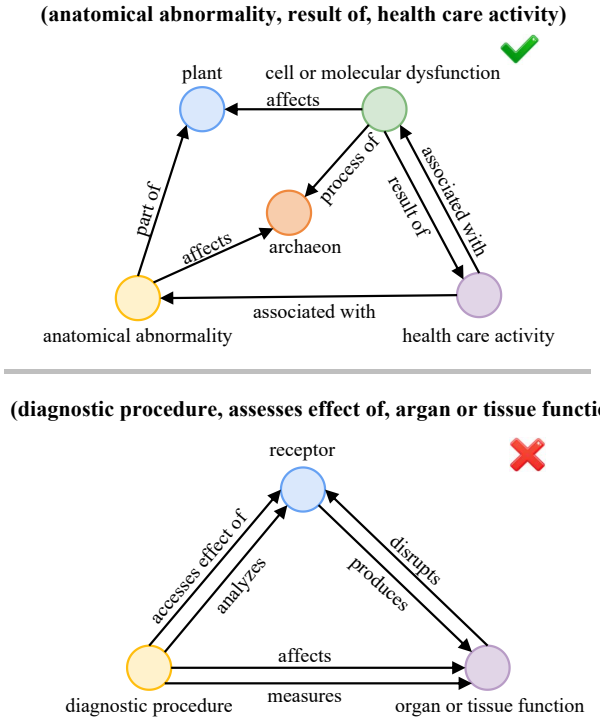

Fig. 4. Case study of different samples, including important learned paths of samples. "✓" indicates that the label of the sample is true, and "✗" indicates false.

V. Conclusion

We propose PathGPT, a framework designed to enhance the knowledge reasoning capabilities of LLMs through the integration of path information. The primary objective of PathGPT is to achieve a deep integration of path information from knowledge graphs with LLMs. This approach generates virtual path markers to enrich input prompts, thereby improving the text decoding process for more logically consistent and accurate predictions. Experimental results on triple classification demonstrate that PathGPT outperforms baseline methods significantly.

VI. Limitations

There are some limitations to our work. Currently, we have not attempted to combine LLM with more path-based reasoning methods. In the future, we plan to further investigate knowledge reasoning of path-augmented LLM and design a more unified framework that can seamlessly combine various LLMs and path-based reasoning methods. In addition, we will explore knowledge graphs to complement LLM-based downstream tasks, making LLM more knowledgeable with additive relevant facts and less prone to hallucinations.

References

[1] M. Yasunaga, H. Ren, A. Bosselut, P. Liang, and J. Leskovec, "QA-GNN: reasoning with language models and knowledge graphs for question answering," in *NAACL-HLT*, pp. 535–546, Association for Computational Linguistics, 2021.

[2] R. Sun, X. Cao, Y. Zhao, J. Wan, K. Zhou, F. Zhang, Z. Wang, and K. Zheng, "Multi-modal knowledge graphs for recommender systems," in *CIKM*, pp. 1405–1414, ACM, 2020.

[3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data," in *international semantic web conference*, pp. 722–735, Springer, 2007.

[4] K. D. Bollacker, C. Evans, P. K. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *SIGMOD Conference*, pp. 1247–1250, ACM, 2008.

[5] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. Hruschka, and T. Mitchell, "Toward an architecture for never-ending language learning," in *Proceedings of the AAAI conference on artificial intelligence*, **24**(1), pp. 1306–1313, 2010.

[6] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE transactions on neural networks and learning systems* **33**(2), pp. 494–514, 2021.

[7] Y. Zhang, Z. Chen, L. Guo, Y. Xu, W. Zhang, and H. Chen, "Making large language models perform better in knowledge graph completion," in *Proceedings of the 32nd ACM international conference on multimedia*, pp. 233–242, 2024.

[8] L. Yao, J. Peng, C. Mao, and Y. Luo, "Exploring large language models for knowledge graph completion," *CoRR* **abs/2308.13916**, 2023.

[9] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, *et al.*, "Lora: Low-rank adaptation of large language models.," *ICLR* **1**(2), p. 3, 2022.

[10] Z. Zhu, X. Yuan, M. Galkin, S. Xhonneux, M. Zhang, M. Gazeau, and J. Tang, "A* net: A scalable path-based reasoning approach for knowledge graphs," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[11] S. K. Alqaaidi and K. J. Kochut, "Relations prediction in knowledge graph completion using large language models," in *Proceedings of the 2024 8th International Conference on Information System and Data Mining*, pp. 122–127, 2024.

[12] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *NIPS*, pp. 2787–2795, 2013.

[13] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *ICML*, *JMLR Workshop and Conference Proceedings* **48**, pp. 2071–2080, JMLR.org, 2016.

[14] D. Zhang, Z. Yuan, H. Liu, H. Xiong, *et al.*, "Learning to walk with dual agents for knowledge graph reasoning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, **36**(5), pp. 5932–5941, 2022.

[15] Y. Lan, S. He, K. Liu, X. Zeng, S. Liu, and J. Zhao, "Path-based knowledge reasoning with textual semantic information for medical knowledge graph completion," *BMC medical informatics and decision making* **21**, pp. 1–12, 2021.

[16] L. Yao, C. Mao, and Y. Luo, "KG-BERT: BERT for knowledge graph completion," *CoRR* **abs/1909.03193**, 2019.

[17] B. Kim, T. Hong, Y. Ko, and J. Seo, "Multi-task learning for knowledge graph completion with pre-trained language models," in *COLING*, pp. 1737–1743, International Committee on Computational Linguistics, 2020.

[18] B. Wang, T. Shen, G. Long, T. Zhou, Y. Wang, and Y. Chang, "Structure-augmented text representation learning for efficient knowledge graph completion," in *WWW*, pp. 1737–1748, ACM / IW3C2, 2021.

[19] M. Gardner and T. Mitchell, "Efficient and expressive knowledge base completion using subgraph feature extraction," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1488–1498, 2015.

[20] R. Das, A. Neelakantan, D. Belanger, and A. McCallum, "Chains of reasoning over entities, relations, and text using recurrent neural networks," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pp. 132–141, 2017.

[21] H. Wang, H. Ren, and J. Leskovec, "Relational message passing for knowledge graph completion," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1697–1707, 2021.

[22] W. Xiong, T. Hoang, and W. Y. Wang, "Deeppath: A reinforcement learning method for knowledge graph reasoning," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, ACL, (Copenhagen, Denmark), September 2017.

[23] R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. Smola, and A. McCallum, "Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning," in *International Conference on Learning Representations*, 2018.

[24] X. V. Lin, R. Socher, and C. Xiong, "Multi-hop knowledge graph reasoning with reward shaping," in *EMNLP*, 2018.

[25] Y. Shen, J. Chen, P.-S. Huang, Y. Guo, and J. Gao, "M-walk: Learning to walk over graphs using monte carlo tree search," *Advances in Neural Information Processing Systems* **31**, 2018.

[26] M. Hildebrandt, J. A. Q. Serna, Y. Ma, M. Ringsquandl, M. Joblin, and V. Tresp, "Reasoning on knowledge graphs with debate dynamics," in *Proceedings of the AAAI Conference on Artificial Intelligence*, **34**(04), pp. 4123–4131, 2020.

[27] W. Chen, W. Xiong, X. Yan, and W. Y. Wang, "Variational knowledge graph reasoning," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1823–1832, 2018.

[28] Z. Zhu, Z. Zhang, L. A. C. Xhonneux, and J. Tang, "Neural bellman-ford networks: A general graph neural network framework for link prediction," in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 29476–29490, 2021.

[29] Y. Zhang and Q. Yao, "Knowledge graph reasoning with relational digraph," in *Proceedings of the ACM Web Conference 2022*, pp. 912–924, 2022.

[30] OpenAI, "GPT-4 technical report," *CoRR* **abs/2303.08774**, 2023.

[31] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *NeurIPS*, 2020.

[32] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. Lowe, "Training language models to follow instructions with human feedback," in *NeurIPS*, 2022.

[33] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, "Unifying large language models and knowledge graphs: A roadmap," *CoRR* **abs/2306.08302**, 2023.

[34] Y. Zhang, Y. Li, L. Cui, D. Cai, L. Liu, T. Fu, X. Huang, E. Zhao, Y. Zhang, Y. Chen, L. Wang, A. T. Luu, W. Bi, F. Shi, and S. Shi, "Siren's song in the AI ocean: A survey on hallucination in large language models," *CoRR* **abs/2309.01219**, 2023.

[35] L. Yang, H. Chen, Z. Li, X. Ding, and X. Wu, "Chatgpt is not enough: Enhancing large language models with knowledge graphs for fact-aware language modeling," *CoRR* **abs/2306.11489**, 2023.

[36] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 4582–4597, 2021.

[37] B. Yang, W. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in *ICLR (Poster)*, 2015.

[38] Z. Sun, Z. Deng, J. Nie, and J. Tang, "Rotate: Knowledge graph embedding by relational rotation in complex space," in *ICLR (Poster)*, OpenReview.net, 2019.

[39] X. Lv, Y. Lin, Y. Cao, L. Hou, J. Li, Z. Liu, P. Li, and J. Zhou, "Do pre-trained models benefit knowledge graph completion? A reliable evaluation and a reasonable approach," in *ACL (Findings)*, pp. 3570–3581, Association for Computational Linguistics, 2022.

[40] T. Safavi and D. Koutra, "Codex: A comprehensive knowledge graph completion benchmark," in *EMNLP (1)*, pp. 8328–8350, Association for Computational Linguistics, 2020.

[41] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," in *ICLR*, OpenReview.net, 2022.