Information Sciences 755 (2026) 123818

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

An ego network-based approach to fine-tune large language models using knowledge graphs

Alessia Amelio [a], Christopher Buratti [b], Michele Marchetti [b], Davide Traini [b,c], Domenico Ursino [b,*], Luca Virgili [b]

[a] InGeo, University "G. D'Annunzio" of Chieti-Pescara, Pescara, Abruzzo, Italy
[b] DII, Polytechnic University of Marche, Ancona, Marche, Italy
[c] CHIMOMO, University of Modena and Reggio Emilia, Modena, Emilia-Romagna, Italy

ARTICLE INFO

Keywords:
Knowledge graphs
Large language models
Fine-tuning
Ego networks
Natural language processing

ABSTRACT

In this paper, we introduce EgoFine, a new approach for fine-tuning Large Language Models (LLMs) based on ego networks extracted from Knowledge Graphs (KGs). EgoFine first identifies the most informative nodes in the KG using degree centrality. It then extracts their ego networks and generates structured training data through random paths within them, thus enabling LLMs to learn domain-specific knowledge. It further constructs negative samples to explicitly model the absence of relationships between entities. We present an experimental campaign involving three KGs (PrimeKG, WN18RR, and YAGO3) and four LLMs (Minerva-350M, Llama3.2-1B, Qwen2-1.5B, and Ministral-3B). This campaign demonstrates that EgoFine outperforms traditional embedding-based methods (e.g., AutoSF, BoxE, NodePiece, PairRE, and TransE), two state-of-the-art approaches integrating KGs and LLMs (e.g., GNN-RAG and KG-Adapter), as well as a baseline approach operating on the same principle as EgoFine but without exploiting the contribution of ego networks. Compared with this last approach, EgoFine improves Hit@1 values by up to 47.37%, Mean Reciprocal Rank (MRR) values by up to 46.87%, F1-Score values by up to 36.59%, and Accuracy values by up to 30.91%. The paper also presents an ablation study devoted to evaluating several design choices underlying EgoFine, as well as an analysis of the EgoFine's behavior when applied to dynamic or noisy KGs. This way of proceeding makes EgoFine particularly beneficial for a variety of real-world applications, including understanding complex biological mechanisms, reasoning about the relationships between legislative sources and court cases, interpreting and explaining complex industrial maintenance and production processes, and understanding the connections between attacks, exploits, and countermeasures in the context of cybersecurity.

1. Introduction

Large Language Models (LLMs) are currently the most disruptive branch of Deep Learning [1]. They have transformed Natural Language Processing (NLP), accelerating progress in tasks such as sentiment analysis, text generation, text summarization, and domain-specific inference. As a result, they are now both major subjects of study in academia and essential problem-solvers in industry [2]. However, one problem that may hinder their adoption, especially in some application fields, is the fact that they can generate hallucinations [3]. To overcome this problem and enhance the quality of results provided by LLMs, researchers are exploring

* Corresponding author.
Email addresses: alessia.amelio@unich.it (A. Amelio), c.buratti@pm.univpm.it (C. Buratti), michele.marchetti@univpm.it (M. Marchetti), davide.traini@unimore.it (D. Traini), d.ursino@univpm.it (D. Ursino), luca.virgili@univpm.it (L. Virgili).

https://doi.org/10.1016/j.ins.2026.123818
Received 19 May 2025; Received in revised form 19 June 2026; Accepted 21 June 2026
Available online 24 June 2026

ways to incorporate external knowledge into them. In recent years, Knowledge Graphs (KGs) have been the most widely used form of external knowledge to support LLMs, as they have proven effective in multiple related tasks [4].

There are three lines of research in the literature related to the integration of LLMs and KGs [5]. The first line explores the feasibility of using LLMs to address various KG-related problems, such as completion, embedding, graph-to-text generation, construction, and question answering [6]. The second line uses KGs to improve LLMs' understanding of knowledge, support their training [7], and facilitate their inferences [8]. These approaches have proven effective in enhancing LLMs' reasoning skills, enabling them to solve difficult tasks. The third line of research involves KGs and LLMs sharing responsibilities and cooperating to enhance bidirectional reasoning skills [5], combining their strengths [9], and using joint representation learning models [10].

In this paper, we aim to provide a contribution to the second line of research by proposing EgoFine, a new approach that uses KGs to fine-tune an LLM. Unlike previous approaches, EgoFine does not use the entire KG for fine-tuning, but rather only appropriate subgraphs that represent uniform and compact domain information.

Specifically, EgoFine derives the KG subgraphs to be injected into the LLM by identifying nodes having the highest degree centrality, constructing their ego networks, and building random paths within them to generate training sequences that preserve the local topology and relational patterns of the KG. Moreover, it also generates negative training samples to explicitly represent the absence of relationships between two entities, thereby enhancing the model's ability to recognize whether a relationship exists or not. Using degree centrality ensures wide graph coverage while maintaining low computational cost, making it a preferable alternative to more expensive centrality measures, such as PageRank or betweenness centrality. Moreover, adopting random paths rather than deterministic traversals, such as Breadth-First Search (BFS) and Depth-First Search (DFS), captures heterogeneous relationship patterns and reduces structural bias toward densely connected regions. This enhances diversity and representational balance in fine-tuning data. Each path is converted into a chat-style prompt that naturally aligns with the conversational input–output format of instruction-tuned LLMs [11], facilitating knowledge integration. Additionally, EgoFine incorporates mechanisms that promote diversity and efficiency during LLM fine-tuning and prevent the prediction of non-existent KG relationships, which could reduce hallucinations.

EgoFine's approach enables a more precise and semantically meaningful transfer of domain knowledge from the KG to the LLM. This preserves meaningful knowledge structures stored in the KG and injects them into the LLM, which significantly improves the reliability and interpretability of the LLM's output. Additionally, EgoFine's mechanisms for promoting diversity during fine-tuning allow it to inject a wide range of relationship patterns into the LLM while focusing on knowledge areas semantically homogeneous within them. Finally, since EgoFine injects domain knowledge directly into the LLM, it enables faster response times and eliminates dependence on external retrieval components, such as Retrieval-Augmented Generations (RAGs), during inference. This improves efficiency and simplifies system architecture by eliminating the need for dynamic interactions with external KGs at query time.

From a practical standpoint, EgoFine provides an innovative and effective method for developing domain-specific expert systems in contexts where low hallucination rates are crucial. Contexts that can greatly benefit from EgoFine include biomedicine, law, industry, and cybersecurity. In particular, in the biomedical context, EgoFine enables LLMs to learn structured relationships among genes, diseases and drugs, which improves understanding of complex biological mechanisms. In the legal context, EgoFine enables LLMs to be fine-tuned on legal KGs that connect norms, judgments, and principles. This enhances the LLM's ability to reason about the relationships between legislative sources and court cases. Furthermore, EgoFine's approach improves the reliability and interpretability of an LLM's output. In the industry context, EgoFine can be used to fine-tune an LLM on relationships between components, sensors, and failure events, improving its ability to interpret complex maintenance or production processes. In the cybersecurity context, EgoFine can be used to fine-tune an LLM with KGs related to threats and vulnerabilities, improving its ability to understand the connections between attacks, exploits, and countermeasures.

We tested EgoFine on four LLMs (Minerva-350M,[1] Llama3.2-1B,[2] Qwen2-1.5B[3] and Ministral-3B[4]) and three KGs (PrimeKG [12], WN18RR [13], and YAGO3 [14]) in a knowledge completion task, specifically predicting the type of the relationship between two entities in a KG. Following the literature, we evaluated the behavior of the fine-tuned LLMs using Hit@K, Mean Reciprocal Rank (MRR), Precision, Recall, and F1-Score as metrics. We found that EgoFine outperforms traditional embedding-based methods (e.g., AutoSF [15], BoxE [16], NodePiece [17], PairRE [18], and TransE [13]), two state-of-the-art approaches integrating KGs and LLM (e.g., GNN-RAG [19] and KG-Adapter [20]), as well as a baseline approach operating on the same principle as EgoFine, but without exploiting the contribution of ego networks. In particular, compared to the baseline approach, EgoFine improves Hit@1 (resp., Hit@3, Hit@5) values by up to 47.37% (resp., 71.88%, 70.00%), MRR values by up to 46.87%, Precision values by up to 22.64%, Recall values by up to 48.48%, and F1-Score values by up to 36.59%. Finally, we tested EgoFine in a Multiple Choice Question Answering (MCQA) task and observed that it obtained a 30.91% higher accuracy than the corresponding baseline.

This paper is organized as follows: in Section 2, we present the related literature. In Section 3, we provide a technical description of EgoFine. In Section 4, we show the experiments we conducted to evaluate its performance, an ablation study assessing the contribution of its components and design choices, and an analysis of its behavior when applied to dynamic or noisy KGs. In Section 5, we present a discussion of the results obtained, highlighting the advantages and limitations of EgoFine. Finally, in Section 6 we draw our conclusions and outline some possible future developments of this research.

[1] https://huggingface.co/sapienzanlp/Minerva-350M-base-v1.0.
[2] https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct.
[3] https://huggingface.co/Qwen/Qwen2.5-1.5B-Instruct.
[4] https://huggingface.co/ministral/Ministral-3b-instruct.

2. Related work

2.1. General overview

In recent years, researchers have extensively studied the integration of LLMs with KGs because KGs can provide LLMs with rich external knowledge, improving their inference capability and reducing the risk of hallucinations. Three research branches related to the integration of LLMs and KGs are found in the literature [5], namely: *(i) LLM-augmented KGs*, where LLMs are used to solve KG-related tasks, such as completion, embedding, graph-to-text generation, construction, and question–answering [6]; *(ii) KG-enhanced LLMs*, where KGs are used to improve the comprehension of the knowledge learned by LLMs or to support their training [7] and inference steps[8]; *(iii) Synergized LLMs + KGs*, where KGs and LLMs work together to improve bidirectional reasoning skills [5], combining their strengths [9] or using joint representation learning models [10].

KG-enhanced LLMs can be further divided into the following subcategories: *(i) KG-enhanced LLM pre-training*, in which KG knowledge is introduced into the LLM during training for solving different tasks, such as sentiment analysis [21], general-purpose NLP [22] and question–answering [23]; *(ii) KG-enhanced LLM inference*, in which KG knowledge is introduced into the LLM during inference to retrieve knowledge from external memory [24], reasoning chains [25], or graph structures [26]; and *(iii) KG-enhanced LLM interpretability*, in which KGs are used to improve the interpretability of the results provided by LLMs, explain their factual knowledge [27], or trace and visualize the reasoning paths behind their answers [28]. *KG instruction-tuning* is a subclass of *KG-enhanced LLM pre-training* that is still little investigated in the literature [5]. These methods use KG facts and structure to inject KG knowledge into the LLM while it is being fine-tuned. This subclass has been shown to enhance LLM knowledge, leading to excellent results on various tasks. Below, we describe relevant work in this subclass that has already been proposed in the literature. We categorize the approaches in this subclass based on their fine-tuning methods and dedicate a subsection to each category. Furthermore, in a separate subsection we also present related approaches focused on model alignment, efficiency, and preference learning with the aim of contextualizing EgoFine within the broader landscape of recent fine-tuning techniques for LLMs. In particular, we consider Low-Rank Adaptation (LoRA) [29], Quantized Low-Rank Adaptation (QLoRA) [30], Reinforcement Learning from Human Feedback (RLHF) [31], AI Feedback (RLAIF) [32], and Direct Preference Optimization (DPO) [33]. They address different dimensions of LLM optimization and can be combined with KG instruction-tuning methods, such as EgoFine, to efficiently adapt the LLM to the KG. Finally, we present a further subsection illustrating the innovations brought by EgoFine to KG instruction-tuning approaches.

2.2. Prompt-tuning and instruction injection approaches

These approaches fine-tune LLMs by injecting structured knowledge directly into prompts or instructions, often without updating the model's weights.

In [34], the authors propose Ontology-Enhanced Prompt-Tuning (OntoPrompt), a model-agnostic system for fine-tuning LLMs that aims to overcome the incompleteness, noise, and heterogeneity of injected external knowledge, which can be found in LLMs. OntoPrompt first enhances structural knowledge and then transforms it into text. Next, it reduces noise and filters informative knowledge through span-sensitive knowledge injection. After that, it optimizes representation through a collective training process. Tests on KG completion, relationship extraction, and event extraction show that OntoPrompt can achieve good performance.

In [35], the authors propose a knowledge prompting paradigm and a knowledge prompting-based pre-trained LLM architecture that aims to eliminate redundant and superfluous factual information from knowledge bases. In particular, their approach first creates a knowledge subgraph using knowledge bases related to heterogeneous contexts. Next, it converts this graph into natural language prompts by creating several continuous prompting rules. To make the most of the factual information contained in these prompts, it uses two new knowledge-aware self-supervised tasks called "prompt relevance inspection" and "masked prompt modeling". This approach has proven to be superior to the state-of-the-art in both full-resource and low-resource settings, as shown by several tests on a variety of Natural Language Understanding problems.

In [36], the authors propose Knowledge Graph Tuning (KGT), an innovative method that uses KGs to adapt the behavior of LLMs to individual users. KGT extracts factual knowledge triples from user queries and feedback. It dynamically updates KGs without altering the LLM's underlying parameters. Specifically, rather than updating the model weights, KGT injects structured knowledge into the model via an external KG that evolves based on user interactions. During future interactions, the KG is queried and relevant facts are injected into prompts or instructions to guide the LLM's responses. Experimental results using state-of-the-art LLMs show that KGT enhances personalization effectiveness while decreasing latency and reducing GPU memory consumption.

In [37], the authors introduce KnowledgeSmith, a comprehensive framework designed to systematically analyze the mechanisms by which LLMs update and manage knowledge. KnowledgeSmith uses a graph-based approach to simulate knowledge editing and unlearning by generating synthetic datasets composed of knowledge triples. These interventions are applied at entity, relation, and graph levels to evaluate the model's consistency, and robustness in knowledge retention and propagation.

2.3. Aligned reasoning and KG-based question–answering approaches

These methods enhance LLMs' reasoning and question–answering capabilities by fine-tuning them to align with structured logical patterns from KGs or by effectively combining generative and retrieval-based approaches for accessing KG information.

In [38], the authors propose Reasoning on Graphs (RoG), a framework to shield an LLM from hallucinations and lack of knowledge. Starting from a KG, RoG creates relationship pathways that are considered trustworthy plans. Then, it uses these pathways to create reasoning paths from the KG, which feed the LLM in inference and fine-tuning tasks. Experimental results on two Knowledge Base

Question–Answering (KBQA) benchmark datasets show that RoG achieves state-of-the-art performance in terms of reliable outcomes on KG reasoning tasks.

In [39], the authors propose Retrieve-Generate-Retrieve Knowledge Base Question–Answering (RGR-KBQA), a semantic parsing technique based on the Retrieve-Generate-Retrieve architecture. RGR-KBQA fine-tunes the LLM using factual knowledge from a KG. In this technique, the initial retrieval step improves the logical form generation accuracy of the LLM by strengthening its semantic comprehension skills. The next generation step aims to generate the logical form using the fine-tuned model with the goal of improving generation accuracy. Finally, the last retrieval step uses unsupervised retrieval of entities and relationships. The two retrieval steps mentioned above help to reduce the problem of hallucinations in LLM. Experiments on the ComplexWebQuestions (CWQ) and WebQuestionsSP (WebQSP) datasets show that RGR-KBQA demonstrates encouraging performance.

To determine whether the LLMs produce the correct logical forms and to fix any flaws, the approach of Banerjee [40] prompts the LLMs, fine-tunes them on specific datasets, and assesses their performance. Experiments show that both the creation of basic logical forms and their grounding to KG components yield encouraging results.

In [41], the authors introduce a Knowledge Graph Question–Answering (KGQA) model called Reasoning via Dynamic Planning on Graph (RDPG). RDPG enhances multi-hop question–answering over KGs by fine-tuning LLMs using an Adaptive Path Generation strategy.

In [23], the authors propose ChatKBQA, a generate-then-retrieve Knowledge Base Question–Answering system based on fine-tuned LLMs. ChatKBQA first fine-tunes the LLM using pairs of the form (logical form, natural language question) found in KBQA datasets. It then translates new natural language queries into the appropriate logical form using the fine-tuned LLM. Finally, to enhance generation and retrieval, it uses an unsupervised method to retrieve and replace entities and relationships. The results of applying ChatKBQA to the KBQA benchmark datasets are encouraging.

In [42], the authors propose GenTKG, a new retrieval-enhanced generation framework that solves the temporal Knowledge Graph (tKG) forecasting task in a generative forecasting setting, outperforming embedding-based, rule-based and In-Context Learning (ICL) methods. GenTKG aims to bring temporal knowledge forecasting into the generative forecasting setting. In order to generate a rule base, GenTKG first mines temporal logic rules from the tKG using a Temporal Logic Rule (TLR) based retrieval technique. It then exploits these rules to extract the most logically and chronologically relevant historical facts pertinent to the user's query. Next, it sequences these facts in plain English in ascending temporal order after inserting them into a particular prompt template for LLMs. Finally, it uses a few-shot, parameter-efficient instruction-tuning approach to achieve a trade-off between the high computational cost of fine-tuning LLMs and the huge amount of data in tKGs. The authors of Liao et al. [42] show that GenTKG outperforms traditional temporal relational forecasting methods with few computational resources, a negligible amount of training data, and acceptable in-domain and cross-domain generalizability.

In [19], the authors propose GNN-RAG, a hybrid model that integrates lightweight Graph Neural Networks (GNNs) and LLMs to enhance retrieval and reasoning. The GNN component assigns relevance scores to nodes and their neighbors based on the input. This allows the system to capture deeper graph context and identify meaningful paths.

In [43], the authors present a new subgraph-based retrieval-augmented approach for KGQA. It uses Chain-of-Thought reasoning to construct subgraphs, enriching the semantic context of candidate knowledge and providing the intermediate reasoning steps that are essential for multi-hop question–answering. These subgraphs help the retrieval model access relevant intermediary KG information. To reduce input redundancy while preserving the KG structure, the authors introduce a YAML-based representation format. They also design three KG-level tasks focused on entities, relationships, and graph structures for instruction tuning and pre-training. Finally, they generate synthetic reasoning data using larger open-source LLMs to further train their model, enhancing its reasoning capabilities.

2.4. Domain-specific KG fusion and forecasting approaches

These methods apply domain-specific supervised fine-tuning, often with KG embeddings or temporal graph encoding in specialized domains like healthcare, tourism, and industrial diagnostics.

In [44], the authors propose BioLORD-2023, an approach that integrates KG and LLM insights in the pre-training of the "Learning of Ontological Representations through Definitions and textual representation" (LORD) method. BioLORD-2023 first enriches the training corpus with new definitions for 400,000 concepts developed by LLMs. Next, it uses a new self-distillation technique to accelerate biological formation learning while preserving language comprehension. Finally, it uses a proven cross-lingual distillation technique to develop a state-of-the-art multilingual model for the biomedical domain. Through a comprehensive assessment of many downstream tasks, the authors show that BioLORD-2023 significantly outperforms the previous state-of-the-art techniques.

In [45], the authors propose a new method combining KGs and LLMs for classifying travel offers. This method first creates a Tourism Knowledge Graph (TKG) from the Tourism Analytical Ontology. It then applies Knowledge Graph Enhanced BERT (KGE-BERT), which combines a transformer model analyzing textual input, with a multilayer perceptron providing additional feature types. To simplify the knowledge input from TKG, KGE-BERT combines four types of features, namely textual, numerical, categories and linked entities. The authors also present a comparative analysis of four datasets related to housing offers in London to demonstrate the effectiveness of their approach.

In [46], the authors present a knowledge-enhanced joint model that integrates an aviation assembly KG into LLMs. This model uses graph-structured big data within KGs to prefix-tune LLMs. The subgraph embedding learning procedure enhances the joint model's domain-specific knowledge in aircraft assembly, especially in defect localization. The joint model can produce knowledge subgraphs, merge knowledge through retrieval augmentation, and produce knowledge-based reasoning replies in the context of aircraft assembly

functional testing. This model shows promising results for defect detection and troubleshooting procedures in real-world industrial scenarios.

In [47], the authors introduce the KG-SFT framework, a KG-driven approach to Supervised Fine-Tuning (SFT). This approach aims to enhance the reasoning and knowledge manipulation capabilities of LLMs in domain-specific, low-resource settings. It operates through three components: an extractor that identifies entities and retrieves relevant subgraphs, a generator that creates fluent explanations, and a detector that ensures semantic reliability.

2.5. Parameter-efficient fine-tuning approaches

These methods fine-tune LLMs by updating only small, targeted components, rather than retraining the entire model. Some of them use adapter-based or low-rank tuning techniques to incorporate KG without retraining the entire model.

In [29], the authors introduce LoRA, a method that fine-tunes LLMs by injecting low-rank trainable matrices into the existing layers of the transformer architecture, rather than updating full weight matrices. During training, only the low-rank matrices are updated, while the pretrained weights remain frozen. This reduces the number of trainable parameters for downstream tasks. Additionally, LoRA enables the seamless integration of trainable matrices with frozen weights during deployment, ensuring zero inference latency compared to a fully fine-tuned model. Experiments showed that LoRA reduces trainable parameters by a factor of 10,000 and cuts GPU memory usage by a factor of three compared to fine-tuning GPT-3 175B.

Similar to LoRA, QLoRA [30] inserts small trainable low-rank matrices into specific layers of the transformer architecture, instead of updating the full weight matrices during fine-tuning. However, QLoRA first quantizes the base model weights by converting the large pre-trained model's weights to 4-bit precision values using the NF4 (NormalFloat 4-bit) technique. Additionally, it performs double quantization to reduce memory usage by compressing the quantization constants and applies paged optimizers to manage memory spikes during training by offloading unused data to GPU memory. The quantized weights are frozen, while the low-rank adapters are trained during fine-tuning. This drastically reduces memory usage and allows huge models to run on devices with modest hardware. Experiments showed that QLoRA enables large-scale instruction tuning across more than 1000 models with between 80 million and 65 billion parameters. It drastically reduces memory overhead while matching 16-bit performance and maintaining general agreement in model rankings.

In [20], the authors introduce KG-Adapter, a parameter-level KG integration technique based on Parameter Efficient Fine-Tuning (PEFT) that aims to resolve knowledge conflicts and lack of structural information in KGs. KG-Adapter uses a novel adapter structure for decoder-only LLMs. It can engage in collaborative reasoning with LLMs to produce end-to-end replies after encoding KGs from both node-centered and relation-centered viewpoints. Experiments conducted on a variety of models and four datasets for two different tasks showed that KG-Adapter achieves significant performance. With only 28M learned parameters, it makes a 7B-parameter LLM perform better than the previous full-parameter fine-tuned state-of-the-art approaches and similarly to prompt-based ChatGPT methods.

In [48], the authors propose Graph-Aligned Language Model (GLAM), a fine-tuning approach that converts a KG into an alternative textual representation using labeled question–answer pairs. The goal of the conversion task is to facilitate LLM fine-tuning, since the training of LLMs relies heavily on textual materials. For each KG node, GLAM repeatedly divides a k-hop neighborhood subgraph and encodes its information into text sentences, which it later uses for fine-tuning the LLM. Thus, the integration of domain-specific KGs into LLMs is achieved directly through their fine-tuning.

2.6. Preference-aligned fine-tuning approaches

These methods use comparative feedback to fine-tune LLMs and guide them toward safer, more useful, or more aligned behavior to produce outputs that better reflect human or AI preferences.

In [31], the authors propose the Reinforcement Learning from Human Feedback (RLHF) approach. It begins with supervised training using curated prompts and responses to establish a baseline. Then, human annotators rank multiple outputs of the same prompt to create data that trains a reward model to predict responses most aligned with human judgment. Next, the LLM is optimized using reinforcement learning to generate output scored by the reward model. This process updates the LLM's parameters to maximize the predicted reward. Although RLHF produces highly capable models for conversation and coding, the process is much more complex than standard supervised learning. It requires multiple language models, continuous sampling during training, and substantial computational resources.

To reduce costs and complexity while maintaining strong alignment performance, the authors of Bai et al. [32] introduce the Reinforcement Learning with Artificial Intelligence Feedback (RLAIF) approach. It is a scalable alternative to RLHF that uses AI-generated preferences instead of human feedback to train LLMs. Instead of relying on human annotators to rank LLM outputs, RLAIF uses an existing, high-quality LLM, such as GPT-4 or Claude, to generate preference labels. These AI-generated rankings are then used to train the reward model, which guides the reinforcement learning process, just as in RLHF.

To fine-tune an LLM to align directly with human preferences and bypass the steps of reward modeling and reinforcement learning, the authors of Rafailov et al. [33] introduce the Direct Preference Optimization (DPO) approach. Unlike RLHF, DPO leverages pairwise preference data directly to circumvent these steps. The LLM is fine-tuned using a contrastive objective that increases the probability of preferred responses over less preferred ones based on human-annotated or synthetic preference pairs. Optimization is performed via a log-likelihood ratio loss, by which the model assigns higher probabilities to the preferred output while staying close to the original pre-trained policy through a regularization term. Experimental results show that DPO performs similarly to established approaches, such as RLHF, for preference-driven tasks including controllable language generation, summarization, and dialogue generation.

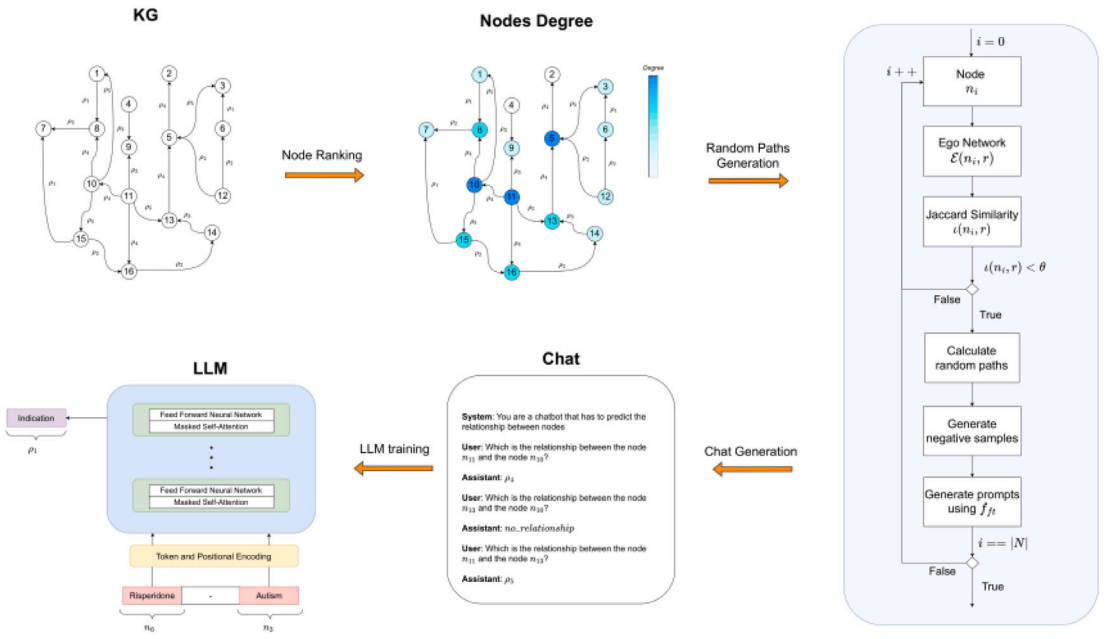

Fig. 1. Schematic workflow of EgoFine, illustrating its main steps: node ranking, random path generation, chat generation, and LLM training.

2.7. Current limitations and our contribution

Despite their widespread use in the literature and their main advantages, the approaches that integrate KGs and LLMs also have drawbacks [5]. In particular, as for LLM-enhanced KGs, they are used to enhance KGs, instead of enhancing the reasoning skills of LLMs; as a result, they are unable to mitigate the hallucination problem characterizing LLMs. Furthermore, the latter problem cannot be completely solved by KG-enhanced LLMs. Indeed, in order to partially solve this problem, it is necessary to integrate them with other models supporting them in this task. Finally, as for Synergized LLMs + KGs, sophisticated and expensive techniques, including multimodal learning, GNNs and continuous learning, are needed to suitably combine LLMs and KGs.

EgoFine is classified as a KG instruction-tuning approach. Indeed, the process of fine-tuning an LLM involves finding specific subgraphs in the KG and injecting the information they convey into the LLM, rather than injecting the KG as a whole. Using an ego network extraction algorithm, EgoFine selects substructures from the KG that reflect distinct and consistent subsets of the knowledge represented by the KG itself. EgoFine preserves the most meaningful connections between the entities of the KG, and thus introduces contextual and more relevant knowledge into the LLM. This can mitigate the problem of hallucinations in an LLM without requiring additional methods to detect and evaluate them, as long as it has been previously fine-tuned by EgoFine. Finally, as will become clear later, EgoFine has a mechanism that allows an LLM fine-tuned by it to specify that a relationship does not exist between two entities. This feature is not so obvious, since, if not appropriately trained, an LLM always predicts the presence of a relationship between two entities.

3. Proposed approach

EgoFine aims to extract knowledge from a given KG and represent it in the form of training samples for fine-tuning an LLM. The central idea is to identify relevant subgraphs centered on high-degree nodes of the KG and to explore them through random paths while ensuring heterogeneity among the explored subgraphs. Fig. 1 shows an overview of the EgoFine workflow and highlights its main components. As shown in the figure, EgoFine takes a KG as input and processes it through four main stages. The workflow begins with the Node Ranking step, which computes the degree centrality of each node in the KG and sorts them in descending order, ensuring that the most structurally relevant entities are prioritized from the outset. Building on this ranking, the Random Paths Generation step constructs ego networks centered on the highest-degree nodes and explores them through random traversals, accounting for both existing relationships and the absence of connections between uncorrelated entities. The resulting paths are then passed to the Chat Generation step, which transforms each traversal into a sequence of natural language prompts. Finally, the LLM Training step uses these prompts to fine-tune the LLM, producing the fine-tuned LLM as output. This workflow shows that EgoFine focuses on ego-centric regions around structurally relevant entities, enabling targeted exploration of the KG while preserving its essential relationship structure and eliminating the need for exhaustive traversal.

Formally speaking, let $\mathcal{G}$ be the KG of interest. It can be represented as a triple:

$$\mathcal{G} = \langle N, A, R \rangle \tag{3.1}$$

Table 1
List of abbreviations, corresponding names and descriptions used in the paper.

Abbreviation	Name	Description
BFS	Breadth-First Search	An algorithm to traverse or search trees or graphs
DFS	Depth-First Search	An algorithm to traverse or search trees or graphs
GNN	Graph Neural Network	–
ICL	In-Context Learning	A technique where LLMs learn by processing examples within their prompt
KBQA	Knowledge Base Question Answering	A subfield of NLP focused on answering questions by querying information stored in a knowledge base
KG	Knowledge Graph	–
KGQA	Knowledge Graph Question Answering	A subfield of NLP focused on answering questions using the structured information present in a KG
LLM	Large Language Model	–
LoRA	Low-Rank Adaptation	A parameter-efficient method for fine-tuning LLMs based on low-rank weight updates
MCQA	Multiple Choice Question Answering	A subfield of NLP focused on answering multiple choice questions
MRR	Mean Reciprocal Rank	A metric used in Machine Learning
NLP	Natural Language Processing	–
PEFT	Parameter Efficient Fine-Tuning	A technique for fine-tuning LLMs
PrimeKG	Precision Medicine Knowledge Graph	A benchmark dataset
RAG	Retrieval-Augmented Generation	–
SFT	Supervised Fine-Tuning	A technique for fine-tuning LLMs
WN18RR	WordNet 18 Relations Reduced	A benchmark dataset
YAGO3	Yet Another Great Ontology 3	A benchmark dataset

where N is the set of nodes (or entities), A is the set of directed arcs (or relationships) between nodes, and R is the set of possible relationship types. Each directed arc $a_{ij} = (n_i, n_j) \in A$ is annotated with a label $\rho \in R$, which specifies its type (and consequently its semantics).

The behavior of EgoFine can be modeled through the following five steps:

1. Node ranking based on degree centrality;
2. Ego network extraction and random path generation;
3. Filtering out overly similar ego networks based on the Jaccard similarity coefficient;
4. Construction of negative samples;
5. Prompt construction and dataset generation.

In the following subsections, we describe each of these steps in detail. Then, we provide an algorithm illustrating the behavior of EgoFine. Before proceeding, to improve the clarity of our proposal, in Table 1 we list the abbreviations used throughout the paper, along with their corresponding names and, where necessary, brief descriptions.

3.1. Step #1: node ranking

The first step of EgoFine is the computation of the degree centrality of each node $n \in N$. This is given by the number of arcs incoming into or outgoing from n. It is defined as:

$$deg(n) = \left|\{a | a = (n, n') \in A \vee a = (n', n) \in A\}\right| \tag{3.2}$$

Once this computation is done, EgoFine sorts all nodes of N in descending order of degree centrality, obtaining a ranked list $\mathcal{N} = [n_1, n_2, \ldots, n_{|N|}]$ such that $deg(n_i) \geq deg(n_{i+1})$, $1 \leq i \leq |N| - 1$.

Nodes having high degree centrality tend to act as local hubs within $\mathcal{G}$ because they are involved in a large number of relationships. This makes them good candidates for constructing subgraphs (in particular, ego networks) from them, since their neighborhoods are typically rich in structural and semantic information. By prioritizing nodes having high degree and using them as starting points for constructing subgraphs, EgoFine increases the probability of capturing diverse and informative substructures in the KG. Furthermore, since high-degree nodes are likely to appear in multiple subgraphs, using them as sources for subgraph construction helps reduce the number of subgraphs required to cover a significant portion of the KG.

3.2. Step #2: ego network extraction and random path generation

To capture significant parts of the KG, EgoFine extracts ego networks centered on the nodes having the highest degree centrality. An ego network provides a localized view of the KG; thus, the use of ego networks allows EgoFine to focus on subgraphs where entities are densely connected and likely to belong to the same conceptual domain, which is particularly beneficial for domain-specific fine-tuning.

Formally speaking, by traversing the nodes of the list $\mathcal{N}$ from the first to the last, EgoFine considers a node n_i, $1 \leq i \leq |N|$, and constructs its ego network of radius r.[5] This is defined as the subgraph $\mathcal{E}(n_i, r)$ containing all nodes reachable from n_i in at most r hops and all arcs connecting these nodes. It can be represented as:

$$\mathcal{E}(n_i, r) = \langle N_i^r, A_i^r, R_i^r \rangle \tag{3.3}$$

where N_i^r is the set of nodes belonging to the neighborhood of radius r of n_i (and therefore reachable from n_i in at most r hops), A_i^r and R_i^r are the corresponding arcs and relationship types.[6]

Within each ego network, EgoFine extracts random paths to generate training sequences. Random paths are well-suited to preserving the local topology and relationship patterns of the ego network. By sampling the paths through a random traversal, EgoFine captures a variety of connections and relationship types between entities, allowing the LLM to learn from various structures derived from the KG.

More specifically, given an ego network $\mathcal{E}(n_i, r)$, EgoFine constructs a random path rp_j of fixed length δ, starting from a randomly selected node and traversing its neighborhood by uniformly sampling outgoing arcs. This path can be expressed as $rp_j = [(n_0, \rho_1, n_1), (n_1, \rho_2, n_2), \ldots, (n_{\delta-1}, \rho_\delta, n_\delta)]$, where the triple $(n_i, \rho_{i+1}, n_{i+1})$, $1 \leq i \leq \delta - 1$, indicates that there is an arc from n_i to n_{i+1} whose corresponding relationship type is ρ_{i+1}. The parameter r defines the semantic breadth of each ego network. Intuitively, if l is the average shortest path length of $\mathcal{G}$, then two arbitrary entities can be connected with approximately l hops on average. Sampling neighborhoods up to half this distance ($r \approx \left\lceil \frac{l}{2} \right\rceil$) strikes a balance between coverage and noise. In fact, r should be large enough to include multi-step relational patterns that typically link relevant entities, yet small enough to avoid introducing weakly related nodes that add noise. Therefore, setting $r = \left\lceil \frac{l}{2} \right\rceil$ aligns the neighborhood radius with the graph's intrinsic connectivity. We estimate the length l of the average shortest path via sampled single-source distances on the largest connected component. As shown in our experiments, this procedure consistently yields $r = 2$ across all datasets.

Following this reasoning, in an ego network of radius r, the maximum possible length of the shortest path between two nodes (i.e., the diameter of the ego network) is $2 \cdot r$. This value is obtained when the path links the farthest nodes in the ego network. Consequently, we can set the length δ of rp_j to $2 \cdot r$ so that rp_j is in principle able to represent the maximum shortest path in the ego network.

Given the number $|N_i^r|$ of nodes in the r-hop ego network $\mathcal{E}(n_i, r)$ and the length δ of the random path, the number of random paths to generate can be approximated as $\left\lceil \frac{|N_i^r|}{\delta} \right\rceil$. This ratio serves as a simple yet effective heuristic to estimate the number of random paths necessary to achieve approximate coverage of $\mathcal{E}(n_i, r)$ and include most of its nodes in the fine-tuning dataset. Intuitively, δ represents the average number of distinct nodes visited by a single random path, while $|N_i^r|$ denotes the total number of nodes in the ego network. Thus, their ratio approximates the number of random paths needed to visit most nodes at least once. Furthermore, the term $\left\lceil \frac{|N_i^r|}{\delta} \right\rceil$ naturally scales with the size of the ego network; larger neighborhoods generate more random paths, whereas smaller neighborhoods generate fewer.

The combination of ego network extraction and random path sampling balances locality and coverage; in fact, the adoption of ego networks ensures the semantic focus of data, while random paths increase the variability in the generated dataset. It is important to note that in EgoFine random paths are not sampled arbitrarily across the entire KG, but rather they are limited to ego networks centered on nodes having the highest degree centrality values. This significantly reduces the risk of overlooking critical structures because ego networks already concentrate the densest and most informative portions of the graph. Nodes having the highest degree centrality values tend to act as conceptual hubs, connecting different parts of the graph and encompassing essential domain knowledge. Consequently, random paths generated within these ego networks are not only diverse, due to the stochastic nature of the traversal, but also intrinsically representative, because they explore subgraphs where relevant structural patterns are most likely to emerge. In practice, combining the most important ego networks, associated with the nodes having the highest degree, with randomly selected paths within them ensures breadth and depth in exploration. On the one hand, the selection of ego networks ensures that the sampled regions cover areas of the KG that are rich in structure and semantics. On the other hand, the random selection of paths introduces variability within these regions, allowing the model to encounter different configurations of entities and relationships. Together, these two mechanisms allow EgoFine to preserve local topology and expose the LLM to a wide range of patterns. This makes the generated training sequences sufficiently representative for effective fine-tuning.

3.3. Step #3: computation of ego network similarities

To ensure the diversity of the areas explored within the KG, EgoFine compares the ego networks associated with the nodes of $\mathcal{N}$. Specifically, let n_i, $1 \leq i \leq |N|$, be a node of $\mathcal{N}$, let N_i^r be the set of nodes of its ego network $\mathcal{E}(n_i, r)$ of radius r, and let $\overline{N^r}$ be the set of nodes of the ego networks of radius r such that: *(i)* their ego is a node preceding n_i in $\mathcal{N}$, and *(ii)* they have not been discarded

[5] Thus, n_i coincides with the first node of $\mathcal{N}$ the first time, with the second node of $\mathcal{N}$ the second time, and so on.

[6] We point out that $\mathcal{E}(n_i, r)$ also contains n_i, because each node is reachable from itself in 0 hops.

(see below). EgoFine calculates the Jaccard coefficient $\imath(n_i, r) = J(N_i^r, \overline{N^r})$ between N_i^r and $\overline{N^r}$ as follows:

$$\imath(n_i, r) = J(N_i^r, \overline{N^r}) = \frac{|N_i^r \cap \overline{N^r}|}{|N_i^r \cup \overline{N^r}|} \tag{3.4}$$

$\imath(n_i, r)$ quantifies the overlap between the sets of nodes in $\mathcal{E}(n_i, r)$ and the ego networks corresponding to nodes preceding n_i in $\mathcal{N}$ and not discarded; it serves as a proxy for their structural and semantic similarity. If $\imath(n_i, r)$ exceeds a threshold θ, EgoFine discards $\mathcal{E}(n_i, r)$. For the remainder of this paper, whenever we refer to the similarity between ego networks, we are always referring to Jaccard similarity.

This policy allows EgoFine to avoid redundant sampling of highly overlapping subgraphs. To make θ adaptive without exhaustive tuning, we link it to the relationship density of the KG. This density is defined as the ratio of the number of triples to the number of distinct relationship types:

$$\mu = \frac{|A|}{|R|}$$

Intuitively, graphs with a high relationship density contain many arcs per relationship type, resulting in stronger overlap and greater redundancy among ego networks. Therefore, they have a greater pruning potential and require a smaller value of θ. Conversely, graphs with a low relationship density contain few arcs per relationship type, resulting in weaker overlap and smaller redundancy among ego networks. Therefore, they have a smaller pruning potential and require a larger value of θ. This inverse relationship between θ and μ provides a simple criterion for initializing θ without any exhaustive search. Indeed, based on this reasoning, we have identified the following heuristic for θ:

$$\theta = \begin{cases} 0.001 & \text{if } \mu \leq 10^4 \\ 0.0001 & \text{if } \mu > 10^4 \end{cases} \tag{3.5}$$

The values of μ for PrimeKG and YAGO3 are 2.25×10^5 and 4.14×10^4, respectively. Therefore, θ is set to 0.0001 for these two KGs. Conversely, since μ is 6.20×10^3 for WN18RR, it follows that θ is equal to 0.001 for this KG. A similar way of proceeding (i.e., computation of μ and application of Eq. (3.5) to obtain the value of θ) can be adopted to any new KG of interest.

Combining this filtering mechanism with the random path sampling strategy significantly improves the variety and representativeness of the extracted data. Since EgoFine favors ego networks centered on highly connected nodes, the regions of the graph that are explored tend to be those where multiple types of relationships, which often belong to different semantic domains, converge. This favors the extraction of sequences that reflect the semantic complexity of the KG. At the same time, filtering based on the Jaccard similarity coefficient avoids the repetition of subgraphs with similar structures, encouraging the exploration of new portions of the graph and relationships not yet considered. Thus, the combined action of sampling and Jaccard filtering preserves the local consistency typical of denser regions and the global diversity among the sampled subgraphs. Consequently, the resulting dataset encompasses a broader and more balanced set of relationships that more accurately represent the semantic richness of the original KG.

3.4. Step #4: construction of negative samples

LLMs are designed to provide an answer when prompted, even when no correct relationship exists between two entities. Without a specific training to recognize and appropriately handle this latter case, an LLM would be biased toward predicting an existing relationship for any input pair. This behavior could artificially introduce hallucinations by generating spurious triples that are not represented in the original KG structure. To mitigate this risk, EgoFine incorporates negative samples into the fine-tuning process, explicitly teaching the LLM to identify and return a "no relationship" response when appropriate. This strategy strengthens the LLM's ability to distinguish between valid and invalid connections, thereby improving its reliability and factual consistency.

To reach this objective, EgoFine extends each random path with negative samples derived from pairs of entities that are structurally correct but semantically invalid. Given a random path $rp_j = [(n_0, \rho_1, n_1), (n_1, \rho_2, n_2), \ldots, (n_{\delta-1}, \rho_\delta, n_\delta)]$, EgoFine first extracts all possible pairs of nodes (n_i, n_k) such that n_i and n_k are present in rp_j, $i < k$, and there is no arc (n_i, ρ_k, n_k) in rp_j (the latter condition is equivalent to saying that $i < k + 1$). For each of these pairs, EgoFine checks whether there is an arc between n_i and n_k in the original graph $\mathcal{G}$. If there is no arc, the pair is a candidate negative sample.

However, to avoid generating negative samples that could potentially conflict with the implicit hierarchical semantics of the KG, EgoFine introduces an additional constraint. Specifically, given the candidate negative sample pair (n_i, n_k), it considers the relationship types of all the arcs forming the subpath from n_i to n_k in rp_j. If all these arcs have the same relationship type ρ_q, it means that the pair (n_i, n_k) corresponds to two entities that could participate in a transitive hierarchical structure. Consequently, EgoFine discards such a pair to avoid the contradiction of considering as negative sample a pair whose nodes are transitively connected by a hierarchy in which all arcs have the same relationship type. This procedure not only defines how negative pairs are selected, but also determines the nature of the signal the model receives during fine-tuning. Since the pairs are extracted from the same ego networks used to generate positive samples, each negative sample arises in a structurally consistent context where a relationship could theoretically exist, but it is not actually present. This makes the learning process more meaningful since the model must distinguish between entities that are close in the graph but not semantically connected. Thus, the model develops a more subtle and realistic ability to discriminate. Applying additional topological and semantic filters further improves the quality of these examples. Pairs that could introduce ambiguity, such as those connected through homogeneous or transitively equivalent relationships, are excluded to prevent implicitly valid connections from being treated as negative. Thanks to this approach, the negative samples obtained by EgoFine fall

Table 2
Examples of how the function f_{ft} works when applied to some triples of $\mathcal{G}$ for a link prediction task on the PrimeKG dataset [12].

Input	$f_{ft}(Input)$
"Bosentan", "synergistic interaction", "Diethylpropion"	Which is the relationship type between the node "Bosentan" and the node "Diethylpropion"?
"Tetany", "side effect", "Calcitonin porcine"	Which is the relationship type between the node "Tetany" and the node "Calcitonin porcine"?
"TEK", "expression present", "midbrain"	Which is the relationship type between the node "TEK" and the node "midbrain"?

between completely unrelated and potentially ambiguous cases. They are neither so trivial nor so similar as to generate uncertainty. Instead, they represent plausible yet incorrect associations that prompt the model to refine its understanding of the absence of relationships within realistic graph contexts.

As a consequence, the final set of negative samples consists of entity pairs such that:

- they do not share a direct relationship in $\mathcal{G}$;
- they do not transitively inherit a relationship type through a subpath of rp_j;
- they do not violate structural patterns (e.g., transitivity, general-to-specific transitions) suggested by the original sequence of relationships.

Each valid negative pair (n_i, n_k) is then used to construct a fictitious (i.e., negative) triple $(n_i, \tilde{\rho}, n_k)$, where $\tilde{\rho}$ is a placeholder relationship (e.g., "unknown" or "no-relationship") indicating the absence of a real connection from n_i to n_k in $\mathcal{G}$. All valid negative triples are added to the resulting prompt, allowing the LLM to learn from both positive and negative patterns during fine-tuning.

3.5. Step #5: prompt construction and dataset generation

Each random path can be transformed into a sequence of prompts using a function f_{ft}, defined according to the downstream task. In the case of link prediction, f_{ft} generates prompts that ask the LLM to infer the relationship type between two entities. Examples of how f_{ft} works in this context are given in Table 2.

Following this way of proceeding, EgoFine builds the input for fine-tuning the LLM. This process starts with the definition of the LLM prompt, which sets the context for the chat-based interaction with the LLM:

$$p_s = \text{"You are a chatbot that has to predict the relationship type between nodes"}$$

At this point, EgoFine applies f_{ft} to each random path to convert it into a prompt sequence; then, it couples such a sequence with p_s to form a chat session. An example of such a session is shown in Fig. 2.

Each of these chats corresponds to a single training instance for the LLM. The resulting dataset consists of all these chats, which represent structurally diverse and semantically rich samples extracted from the KG. At this point, EgoFine injects all the chats of this dataset into the LLM in order to fine-tune it.

3.6. Algorithmic description of EgoFine

The pseudocode algorithm describing the behavior of EgoFine is shown in Algorithm 1.

As can be seen from it, EgoFine starts by computing the degree centrality of all nodes in $\mathcal{G}$ and sorting them in descending order. The nodes having the highest degree centrality are prioritized since they most likely represent structurally relevant and semantically rich regions of $\mathcal{G}$.

Starting from the top-ranked nodes, EgoFine constructs ego networks of a predefined radius r. To ensure diversity among the selected ego networks, it filters out those ego networks whose set of nodes has a Jaccard coefficient with the set of nodes of previously accepted ego networks higher than a specific threshold θ. For each accepted ego network, EgoFine constructs a set of random paths of fixed length δ; each random path ultimately generates a sequence of entities and relationships.

Starting from each random path, EgoFine extracts positive samples. Next, it also generates negative samples by selecting pairs of nodes that are not directly connected in $\mathcal{G}$ and that do not violate structural constraints (e.g., pairs of nodes that do not create conflicts due to transitivity in hierarchical relationships). Then, it uses the function f_{ft} to transform the positive and negative samples into natural language prompts that can be exploited to fine-tune the LLM $\mathcal{L}$ of interest. All these prompts are collected in a final database $\mathcal{D}$, which therefore contains all training samples derived from the KG. As a last step, EgoFine injects the prompts of $\mathcal{D}$ into $\mathcal{L}$ to fine-tune it.

To ensure EgoFine's scalability, we have optimized each stage of the preprocessing pipeline. Calculating degree centrality involves scanning all $|A|$ arcs of $\mathcal{G}$ and has a time complexity of $O(|A|)$. The subsequent sorting of nodes by degree has a time complexity of $O(|N| \cdot log|N|)$, but this step is performed only once per graph. Therefore, its cost is negligible compared to the next stages of ego network extraction and random path sampling. Jaccard filtering compares each r-hop ego set N_i^r with the global coverage set $\overline{N^r}$, which is stored as a hashed set. Computing the intersection $|N_i^r \cap \overline{N^r}|$ requires $O(|N_i^r|)$ time via membership checks, while $|\overline{N^r}|$ is

System: You are a chatbot that has to predict the relationship type between nodes.

User: Which is the relationship between the node 'GZF1' and the node 'joint laxity, short stature, and myopia'?

Assistant: no relationship

User: Which is the relationship between the node 'joint laxity, short stature, and myopia' and the node 'hereditary connective tissue disorder'?

Assistant: parent-child

User: Which is the relationship between the node 'GZF1' and the node 'chondroectodermal dysplasia with night blindness'?

Assistant: no relationship

User: Which is the relationship between the node 'hereditary connective tissue disorder' and the node 'chondroectodermal dysplasia with night blindness'?

Assistant: parent-child

User: Which is the relationship between the node 'chondroectodermal dysplasia with night blindness' and the node 'Hyperconvex fingernails'?

Assistant: phenotype present

- -

System: You are a chatbot that has to predict the relationship type between nodes.

User: Which is the relationship between the node 'Signaling by phosphorylated juxtamembrane, extracellular and kinase domain KIT mutants' and the node 'KRAS'?

Assistant: interacts with

User: Which is the relationship between the node 'Signaling by phosphorylated juxtamembrane, extracellular and kinase domain KIT mutants' and the node 'TEK'?

Assistant: no relationship

User: Which is the relationship between the node 'KRAS' and the node 'cardiac atrium'?

Assistant: expression present

User: Which is the relationship between the node 'Signaling by phosphorylated juxtamembrane, extracellular and kinase domain KIT mutants' and the node 'midbrain'?

Assistant: no relationship

User: Which is the relationship between the node 'cardiac atrium' and the node 'TEK'?

Assistant: expression present

User: Which is the relationship between the node 'TEK' and the node 'midbrain'?

Assistant: expression present

Fig. 2. Two chat examples generated by EgoFine to represent the triples of random paths.

updated incrementally. Ego extraction considers only a small neighborhood ($r = 2$ in our setup), resulting in a per-ego cost of $O(|A_i^r|)$, where $|A_i^r| \ll |A|$. Finally, the time complexity of random path generation is $O(\lceil |N_i^r|/\delta \rceil \cdot \delta)$ per ego network. Each path explores a fixed-length sequence of $\delta = 2 \cdot r$ hops, and the number of paths scales linearly with the size of the corresponding ego network. Since r and δ are small constants in our setup ($r = 2$, $\delta = 4$), this step remains computationally lightweight, even for very large graphs. Each step of the random path is generated in constant time ($O(1)$) using efficient neighbor sampling. Finally, since paths are independent, they can be generated in parallel without synchronization overhead.

4. Experiments

This section presents the experiments we performed to evaluate the effectiveness of EgoFine. Specifically, in Section 4.1 we describe the experimental setup in detail. In Section 4.2, we present a sensitivity analysis that empirically validates the heuristic settings of EgoFine's hyperparameters. In Section 4.3 we compare the results of EgoFine with related approaches proposed in the literature and with a baseline approach. In Section 4.4, we analyze EgoFine's computation time and resource requirements across different KGs. Then, in Section 4.5, we evaluate the performance of EgoFine on an MCQA task. In Section 4.6, we conduct an ablation study to analyze the contribution of each design choice underlying EgoFine. In Section 4.7, we evaluate the behavior of EgoFine on dynamic KGs. Finally, in Section 4.8 we evaluate the EgoFine's robustness to noisy KGs.

Algorithm 1 The pseudocode algorithm describing the behavior of EgoFine.

Input

- $\mathcal{G}$: Knowledge Graph
- $\mathcal{L}$: LLM to fine-tune
- r: a positive integer representing the radius of the ego networks
- θ: a real number in the interval $[0, 1]$ representing the threshold of the Jaccard coefficient

Output

- $\mathcal{L}$: the fine-tuned LLM

Compute the degree centrality $deg(n)$ of each node $n \in N$
Compute $\delta = 2 \cdot r$
Sort nodes in descending order of degree to obtain the list $\mathcal{N} = [n_1, n_2, \dots, n_{|N|}]$
Initialize the set of previously accepted nodes $\overline{N^r} \leftarrow \emptyset$
Initialize the fine-tuning dataset $\mathcal{D} \leftarrow \emptyset$
for each node n_i in $\mathcal{N}$ **do**
    Extract the ego network $\mathcal{E}(n_i, r)$
    Compute the Jaccard coefficient $\iota(n_i, r)$
    if $\iota(n_i, r) < \theta$ **then**
        $\overline{N^r} \leftarrow \overline{N^r} \cup N_i^r$
        $\max\_rp = \frac{|N_i^r|}{\delta}$
        for $j = 0$ to $\max\_rp$ **do**
            Construct a random path rp_j of length δ within $\mathcal{E}(n_i, r)$
            Generate positive samples from rp_j
            Generate negative samples from rp_j by selecting node pairs present in it that are not directly connected in $\mathcal{G}$ and do not violate structural constraints
            Transform each sample into a prompt using f_{ft}
            Add the resulting prompt to $\mathcal{D}$
Inject the prompts of $\mathcal{D}$ into $\mathcal{L}$ to fine-tune it
return $\mathcal{L}$

Table 3
Statistics of the KGs employed for evaluating EgoFine.

Statistic	PrimeKG [12]	WN18RR [13]	YAGO3 [14]
Number of nodes	129,375	32,374	122,640
Number of arcs	4,049,642	68,230	744,694
Number of relationship types	18	11	18
Density	$0.484 \cdot 10^{-3}$	$0.130 \cdot 10^{-3}$	$0.099 \cdot 10^{-3}$

4.1. Experimental setup

We evaluated EgoFine on three different widely used KGs, each characterized by a different domain and structure. Such KGs are:

- *PrimeKG* [12]: This is a large-scale biomedical KG designed for precision medicine applications. It integrates information from 18 high-quality biomedical resources to describe 17,080 diseases through more than 4 million relationships spanning 10 major biological scales. In addition to its structured graph format, PrimeKG includes textual clinical guidelines for drugs and diseases, enabling multimodal analysis.
- *WN18RR* [13]: This is a refined link prediction benchmark dataset derived from the original WN18 subset of WordNet. It handles 11 relationship types. It addresses the inverse relationship leakage issue present in WN18 by removing redundant inverse triples, thus providing a more challenging and realistic evaluation setting.
- *YAGO3* [14]: This is a large, automatically constructed multilingual knowledge base that combines information from Wikipedia, WordNet, and GeoNames. It includes spatial and temporal annotations, allowing for spatio-temporal queries, and has a high extraction quality. It covers a wide range of general-domain knowledge across multiple languages.

Each dataset was filtered and adapted to our objective. The corresponding statistics are reported in Table 3.

Among the many LLMs available in the literature, we selected Minerva-350M, Llama3.2-1B, Qwen2-1.5B and Ministral-3B. Minerva and Ministral are both based on the Mistral architecture,[7] which uses sliding window attention, a rolling buffer cache, and pre-fill chunking. Minerva-350M has 350 million parameters, while Ministral-3B has 3 billion parameters. Llama3.2-1B has 1 billion parameters and belongs to the Llama3 family, which was developed by Meta.[8] This family is characterized by efficient transformer blocks, rotary positional embeddings, and instruction tuning for general-purpose reasoning and dialogue. Qwen2-1.5B has 1.5 billion parameters and is part of the Alibaba Cloud's[9] Qwen series. It is designed for multilingual understanding and reasoning; it features optimized attention mechanisms and extensive pretraining on various web and code corpora.

To assess the effectiveness of EgoFine, we evaluated the performance of the LLMs fine-tuned by means of it on the link prediction task [48]. Regarding this, we observe that the number of relationship types in the three KGs varies from 11 to 18; this makes the link prediction task (which is essentially a classification task where the model has to predict the correct arc label) extremely difficult, as there are from 11 to 18 classes that constitute the possible outputs of the process. For performance evaluation, we used standard evaluation metrics such as Hit@K and Mean Reciprocal Rank (MRR) [48]. Hit@K measures the proportion of queries for which the correct answer appears among the top-K predictions. The higher the value of Hit@K, the better the ability of the LLMs to return the correct answer within the top-K outputs. Meanwhile, MRR computes the average of the reciprocal ranks of the correct answers in the prediction list. A correct answer ranked closer to the top will yield a higher reciprocal value; thus, a higher MRR reflects better ranking performance. In addition, we also computed Precision, Recall, and F1-Score to evaluate the LLM's ability to correctly predict the label of the arc between two entities. Together, these metrics provide a complementary perspective by evaluating performance in terms of correctness and completeness of the predicted arc labels.

To fine-tune the LLMs and accurately assess their performance, we first extracted the test set by removing a subset of connections from the full KG; the held out portion is represented as a collection of chats. We then applied EgoFine to the remaining KG to extract conversational paths. Finally, we split the extracted paths into 80% for training and 20% for validation, which we used to fine-tune the LLMs.

For comparison, we selected the following baseline approaches: AutoSF [15], BoxE [16], NodePiece [17], PairRE [18], and TransE [13]. All of them perform link prediction, like EgoFine; however, unlike EgoFine they do not belong to the category of KG-enhanced LLMs. Specifically, while EgoFine maps triples of entities and relationships of a KG into a sequence of prompts, these approaches encode triples as embeddings in a latent space for link prediction. As a consequence, they do not use an LLM to support the prediction task. They are also baseline approaches because they do not use specific criteria, such as node importance or structural properties of the KG, to extract the triples. We implemented each approach using the hyperparameter values proposed in the corresponding paper. We also evaluated two recent state-of-the-art approaches that integrate KGs with LLMs, i.e., GNN-RAG [19] and KG-Adapter [20]. In addition to these approaches, we considered another one that uses an LLM to support the link prediction task. It is a baseline approach because it generates random paths by uniformly sampling the KG arcs without considering node importance or structural properties. Therefore, we call it Baseline-LLM. The total number of random paths in Baseline-LLM and their lengths match the corresponding values for random paths in EgoFine. Negative samples are also selected and injected into the LLM in the same way as in EgoFine. We conducted all experiments on an NVIDIA DGX A100 workstation equipped with an AMD EPYC 7742 CPU, 128 GB of RAM, and two NVIDIA A100 GPUs with 40 GB of memory each.

The interested reader can find the source code and the employed datasets in the GitHub repository at the web address https://github.com/christopherburatti/EgoFine.

4.2. Setting the hyperparameters of EgoFine

We have seen that EgoFine has two hyperparameters, namely the radius r of the ego networks constructed from the nodes of the KG (see Eq. (3.3)) and the threshold θ relative to the Jaccard coefficient (see Eq. (3.4)). Although EgoFine includes these two hyperparameters, their exhaustive tuning is not required in practice. In Sections 3.2 and 3.3, we presented some heuristics for setting both parameters based on the structural properties of the input KG, allowing for an out-of-the-box configuration. In this section, we perform a sensitivity analysis to empirically validate these heuristics. Specifically, we carry out a grid search on the WN18RR dataset using the Minerva-350M model, with $r \in \{1, 2, 3, 4\}$ and $\theta \in \{0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1\}$. We assess performance using Hit@1, Hit@3, Hit@5, MRR, Precision, Recall, and F1-Score. We summarize the corresponding results in Table 4.

From the analysis of this table, we can see that increasing the radius r from 1 to 2 leads to consistent improvements in all metrics. However, further increasing r from 2 to 3 generally leads to a deterioration in performance, especially for the Hit@K and MRR metrics, suggesting that a neighborhood that is too large introduces noise rather than useful information. This is even more evident when $r = 4$. As for the threshold θ of the Jaccard coefficient, lower values tend to produce higher values of Recall and F1-Score. However, when values are lower than 0.001, there is no longer any improvement in the metrics; rather, there is a deterioration. On the other hand, a slightly higher threshold (i.e., 0.01) yields slightly better values of Hit@1 and MRR for $r = 2$. Taking into account all the available metrics, we observe that the configuration predicted by our heuristics, namely $r = 2$ and $\theta = 0.001$, allows us to obtain very competitive (in particular, suboptimal) values for Hit@1 (i.e., 0.28), Hit@3 (i.e., 0.55) and MRR (i.e., 0.47), and optimal values for Hit@5 (i.e., 0.68), Precision (i.e., 0.65), Recall (i.e., 0.49) and F1-Score (i.e., 0.56). This indicates that, for this pair of KG and LLM, a moderate neighborhood expansion ($r = 2$), combined with a small threshold of the Jaccard coefficient ($\theta = 0.001$), guarantees

[7] https://arxiv.org/abs/2310.06825.
[8] https://ai.meta.com/research/.
[9] https://www.alibabacloud.com/.

Table 4
Sensitivity analysis of EgoFine with WN18RR using Minerva-350M.

r	θ	Hit@1	Hit@3	Hit@5	MRR	Precision	Recall	F1-Score
1	0.0001	0.22	0.44	0.59	0.40	0.49	0.39	0.43
1	0.0005	0.23	0.46	0.60	0.41	0.49	0.38	0.43
1	0.001	0.25	0.48	0.61	0.42	0.50	0.38	0.43
1	0.005	0.26	0.49	0.60	0.43	0.52	0.39	0.45
1	0.01	0.24	0.50	0.58	0.42	0.46	0.32	0.38
1	0.05	0.22	0.47	0.56	0.40	0.44	0.31	0.36
1	0.1	0.23	0.50	0.58	0.47	0.41	0.32	0.36
2	0.0001	0.24	0.51	0.64	0.46	0.60	0.44	0.51
2	0.0005	0.27	0.53	0.67	0.46	0.61	0.46	0.52
2	0.001	0.28	0.55	**0.68**	0.47	**0.65**	**0.49**	**0.56**
2	0.005	0.29	0.55	0.67	0.46	0.61	0.46	0.52
2	0.01	**0.30**	**0.56**	**0.68**	**0.48**	0.54	0.43	0.48
2	0.05	0.27	0.52	0.64	0.45	0.58	0.40	0.47
2	0.1	0.28	0.53	0.62	0.45	0.60	0.41	0.49
3	0.0001	0.16	0.33	0.50	0.32	0.56	0.31	0.40
3	0.0005	0.18	0.38	0.52	0.34	0.59	0.32	0.41
3	0.001	0.19	0.40	0.55	0.36	0.62	0.34	0.44
3	0.005	0.20	0.41	0.53	0.37	0.60	0.35	0.44
3	0.01	0.18	0.32	0.43	0.33	0.59	0.36	0.45
3	0.05	0.17	0.30	0.41	0.32	0.55	0.33	0.41
3	0.1	0.26	0.50	0.62	0.44	0.64	0.48	0.55
4	0.0001	0.11	0.23	0.35	0.26	0.53	0.30	0.38
4	0.0005	0.12	0.26	0.37	0.28	0.55	0.33	0.41
4	0.001	0.14	0.28	0.38	0.30	0.57	0.34	0.43
4	0.005	0.16	0.28	0.37	0.31	0.53	0.32	0.40
4	0.01	0.17	0.30	0.40	0.34	0.54	0.33	0.41
4	0.05	0.15	0.28	0.37	0.32	0.50	0.29	0.37
4	0.1	0.14	0.27	0.35	0.28	0.49	0.28	0.36

the best trade-off between informativeness and noise, leading to the best overall performance for all the metrics considered. The results in Table 4 empirically confirm the validity of our heuristics presented in Sections 3.2 and 3.3, as the optimal performance for WN18RR is achieved when $r = 2$ and $\theta = 0.001$. We recall that the same heuristics, when applied on PrimeKG and YAGO3, yield $r = 2$ and $\theta = 0.0001$. We used these hyperparameter values for PrimeKG, WN18RR and YAGO3 in the experiments described in the next sections.

4.3. Comparison with existing approaches

After setting the EgoFine's hyperparameter values, we compare our system to the other related ones mentioned in Section 4.1 (AutoSF, BoxE, NodePiece, PairRE, TransE, GNN-RAG, KG-Adapter and Baseline-LLM), using three KGs (PrimeKG, WN18RR, and YAGO3), four LLMs (Minerva-350M, Llama3.2-1B, Qwen2-1.5B and Ministral-3B), and several metrics (Hit@1, Hit@3, Hit@5, MRR, Precision, Recall, F1-Score). For each metric, a higher value indicates a better model. First, we compare EgoFine with traditional embedding-based methods to assess the contribution of integrating LLMs into the link prediction task. Next, we present a detailed comparison among EgoFine, GNN-RAG, KG-Adapter and Baseline-LLM using different LLMs and KGs with the aim of isolating and quantifying the specific benefits introduced by our fine-tuning strategy.

4.3.1. Comparison with traditional embedding-based methods

To evaluate the contribution of EgoFine compared to conventional embedding-based approaches, we first compare it with widely adopted KG embedding models. Table 5 shows EgoFine's performance (using Minerva-350M as the underlying LLM) compared to highly representative embedding-based approaches. As seen in the table, EgoFine outperforms all of these traditional methods on every metric, confirming the effectiveness of integrating LLM reasoning with structural knowledge.

This performance can be explained by observing that, unlike classical embedding-based systems, which rely solely on representations of entities and relationships, EgoFine integrates contextualized semantic information derived from an LLM with structural cues extracted from ego network patterns. This hybrid mechanism allows the model to perform richer relational reasoning, effectively bridging the gap between symbolic and distributional representations. Overall, EgoFine's superiority to traditional embedding-based systems demonstrates the advantage of coupling LLMs with localized graph knowledge in link prediction tasks.

4.3.2. Comparison with LLM-based approaches

To evaluate the effectiveness of the EgoFine integration strategy when combined with different LLMs, we compared EgoFine to GNN-RAG [19], KG-Adapter [20] and the Baseline-LLM. We recall that the Baseline-LLM generates random paths by uniformly sampling KG arcs, regardless of node importance or structural properties.

Table 5
Results obtained by EgoFine and several embedding-based approaches previously proposed in the literature using Minerva-350M.

Dataset	Approach	Hit@1	Hit@3	Hit@5	MRR	Precision	Recall	F1-Score
PrimeKG	EgoFine	**0.84**	**0.93**	**0.95**	**0.89**	**0.90**	**0.85**	**0.87**
	AutoSF	0.06	0.18	0.31	0.20	0.24	0.06	0.10
	BoxE	0.05	0.11	0.20	0.17	0.11	0.05	0.07
	NodePiece	0.05	0.17	0.30	0.20	0.24	0.05	0.08
	PairRE	0.04	0.08	0.09	0.15	0.01	0.04	0.02
	TransE	0.07	0.21	0.34	0.22	0.21	0.07	0.11
WN18RR	EgoFine	**0.28**	**0.55**	**0.68**	**0.47**	**0.65**	**0.49**	**0.56**
	AutoSF	0.07	0.22	0.41	0.25	0.20	0.07	0.10
	BoxE	0.02	0.09	0.17	0.15	0.05	0.02	0.03
	NodePiece	0.07	0.20	0.37	0.24	0.21	0.07	0.11
	PairRE	0.04	0.08	0.15	0.16	0.19	0.04	0.07
	TransE	0.08	0.31	0.47	0.28	0.14	0.11	0.12
YAGO3	EgoFine	**0.42**	**0.63**	**0.72**	**0.55**	**0.83**	**0.55**	**0.66**
	AutoSF	0.04	0.16	0.26	0.18	0.13	0.04	0.06
	BoxE	0.01	0.08	0.14	0.13	0.16	0.01	0.02
	NodePiece	0.04	0.12	0.20	0.16	0.13	0.04	0.06
	PairRE	0.004	0.02	0.05	0.10	0.07	0.02	0.03
	TransE	0.09	0.23	0.37	0.24	0.23	0.09	0.13

Table 6 shows the results of this comparison across the four LLMs and the three KGs. The results indicate that EgoFine consistently outperforms both GNN-RAG and KG-Adapter across all considered configurations. This consistent advantage over recent approaches highlights the competitiveness of EgoFine with respect to current state-of-the-art approaches. In addition, EgoFine consistently matches or outperforms the Baseline-LLM in every setting, further confirming that its knowledge injection mechanism effectively enhances the reasoning capabilities of LLMs, regardless of their scale or architecture.

As for PrimeKG, EgoFine outperforms Baseline-LLM across all four LLMs. Notably, with Minerva-350M, EgoFine improves Hit@1 (resp., Hit@3, Hit@5) by 23.53% (resp., 8.14%, 5.56%) while improving MRR by 12.66%. Precision has the same values in the two systems, while Recall increases by 23.19% in EgoFine and F1-Score increases by 11.54%. These results suggest that EgoFine recovers a higher proportion of correct relationships among the top-ranked results and achieves a better balance between Precision and Recall. With Llama3.2-1B, EgoFine improves Baseline-LLM by 5.62% in Hit@1, 3.26% in Hit@3, and 4.35% in MRR. Conversely, EgoFine has the same value as Baseline-LLM in Hit@5. Additionally, Precision, Recall, and F1-Score increase by 1.03%, 2.08% and 2.08% respectively. These results confirm that integrating localized knowledge enhances reasoning, even in more capable LLMs. As for Qwen2-1.5B, the improvements in Hit@1, Hit@3, and Hit@5 are 2.22%, 1.08% and 2.15% respectively, while the improvement in MRR is 1.06%. EgoFine also improves Precision by 4.21%, Recall by 2.08%, and F1-Score by 3.16% compared to Baseline-LLM, indicating greater consistency across metrics. Finally, with Ministral-3B, where Baseline-LLM already achieves near-saturated performance, EgoFine improves Hit@1 by 5.81%, Hit@3 by 2.11%, Hit@5 by 1.03%, and MRR by 3.30%. EgoFine's Precision is equal to that of Baseline-LLM, while Recall and F1-Score increase by 3.26% and 2.13%, respectively.

As for WN18RR, EgoFine outperforms Baseline-LLM in all metrics and models. With Minerva-350M, improvements are seen in Hit@1 (47.37%), Hit@3 (71.88%), Hit@5 (70.00%), and MRR (46.87%). Precision increases by 22.64%, Recall by 48.48%, and F1-Score by 36.59%. These results confirm that EgoFine substantially improves prediction accuracy and ranking quality, especially when the underlying LLM is small and benefits most from contextual knowledge. With Llama3.2-1B, EgoFine improves Hit@1 by 20.00%, Hit@3 by 12.28%, Hit@5 by 18.33%, and MRR by 30.43%. Compared to the corresponding values achieved by Baseline-LLM, Precision increases by 82.14%, Recall by 33.96%, and F1-Score by 59.46%. These results demonstrate that EgoFine effectively enhances the LLM's ability to identify the correct types of relationships, even when Baseline-LLM already exhibits moderate competence. As for Qwen2-1.5B, EgoFine increases Hit@1 by 12.20%, Hit@3 by 8.33%, Hit@5 by 15.87%, and MRR by 12.96%. Precision increases by 14.29%, Recall by 12.96%, and F1-Score by 14.89%. Finally, with Ministral-3B, EgoFine achieves steady, albeit minor, improvements. Specifically, Hit@1 increases by 11.90%, Hit@3 by 2.99%, Hit@5 by 1.32%, and MRR by 5.17%. Precision increases by 4.11%, Recall by 4.62%, and F1-Score by 4.35%.

As for YAGO3, EgoFine matches or outperforms Baseline-LLM on all models and metrics. With Minerva-350M, Hit@1 has the same value as Baseline-LLM, Hit@3 increases by 3.28%, and Hit@5 increases by 4.35%. The MRR value remains unchanged, while Precision improves by 5.06%, Recall by 17.02%, and F1-Score by 11.86%. These results suggest that even when top-ranked accuracy remains similar, EgoFine offers a better balance between Precision and Recall, resulting in more reliable predictions. With Llama3.2-1B, EgoFine improves Hit@1 by 12.28%, though Hit@3 and Hit@5 are the same as Baseline-LLM. MRR increases by 2.78%, Precision by 12.24%, Recall by 6.78%, and F1-Score by 9.26%. As for Qwen2-1.5B, the improvement is substantial and uniform across all metrics. Specifically, Hit@1 increases by 10.34%, Hit@3 by 23.19%, Hit@5 by 23.61%, and MRR by 18.18%. Precision, Recall and F1-Score improve by 22.22%, 21.31%, and 21.15%, respectively. Finally, with Ministral-3B, EgoFine improves Hit@1 by 8.20%, Hit@3 by 2.47%, Hit@5 by 1.15%, and MRR by 4.11%. Precision increases by 1.06%, Recall by 3.37%, and F1-Score by 3.30%.

The results of all the comparative experiments show that EgoFine is much more effective than traditional systems in the link prediction task. Regarding this, it should be recalled that traditional embedding-based systems (AutoSF, BoxE, NodePiece, PairRE,

Table 6
Comparison between EgoFine, GNN-RAG, KG-adapter and the baseline-LLM across different LLMs and KGs. The values in the table highlight that EgoFine outperforms both GNN-RAG and KG-adapter across all considered configurations. They also show that EgoFine consistently matches or outperforms the baseline LLM in every setting. This advantage over recent methods highlights the competitiveness of EgoFine with respect to current state-of-the-art approaches.

Dataset	Model	Method	Hit@1	Hit@3	Hit@5	MRR	Precision	Recall	F1-Score
PrimeKG	Minerva-350M	GNN-RAG	0.76	0.85	0.89	0.81	0.84	0.76	0.80
		KG-Adapter	0.79	0.88	0.91	0.83	0.86	0.79	0.82
		Baseline-LLM	0.68	0.86	0.90	0.79	**0.90**	0.69	0.78
		EgoFine	**0.84**	**0.93**	**0.95**	**0.89**	**0.90**	**0.85**	**0.87**
	Llama3.2-1B	GNN-RAG	0.88	0.91	0.94	0.89	0.94	0.91	0.92
		KG-Adapter	0.90	0.93	0.95	0.91	0.95	0.93	0.94
		Baseline-LLM	0.89	0.92	**0.97**	0.92	0.97	0.96	0.96
		EgoFine	**0.94**	**0.95**	**0.97**	**0.96**	**0.98**	**0.98**	**0.98**
	Qwen2-1.5B	GNN-RAG	0.90	0.91	0.92	0.92	0.92	0.90	0.91
		KG-Adapter	0.91	0.92	0.93	0.93	0.94	0.93	0.93
		Baseline-LLM	0.90	0.93	0.93	0.94	0.95	0.96	0.95
		EgoFine	**0.92**	**0.94**	**0.95**	**0.95**	**0.99**	**0.98**	**0.98**
	Ministral-3B	GNN-RAG	0.89	0.91	0.93	0.90	0.92	0.89	0.90
		KG-Adapter	0.90	0.93	0.95	0.91	0.94	0.91	0.92
		Baseline-LLM	0.86	0.95	0.97	0.91	**0.97**	0.92	0.94
		EgoFine	**0.91**	**0.97**	**0.98**	**0.94**	**0.97**	**0.95**	**0.96**
WN18RR	Minerva-350M	GNN-RAG	0.25	0.50	0.62	0.41	0.61	0.44	0.51
		KG-Adapter	0.22	0.44	0.56	0.37	0.58	0.40	0.47
		Baseline-LLM	0.19	0.32	0.40	0.32	0.53	0.33	0.41
		EgoFine	**0.28**	**0.55**	**0.68**	**0.47**	**0.65**	**0.49**	**0.56**
	Llama3.2-1B	GNN-RAG	0.39	0.60	0.67	0.51	0.46	0.65	0.54
		KG-Adapter	0.37	0.58	0.64	0.49	0.44	0.62	0.51
		Baseline-LLM	0.35	0.57	0.60	0.46	0.28	0.53	0.37
		EgoFine	**0.42**	**0.64**	**0.71**	**0.60**	**0.51**	**0.71**	**0.59**
	Qwen2-1.5B	GNN-RAG	0.43	0.62	0.69	0.55	0.44	0.57	0.50
		KG-Adapter	0.41	0.59	0.66	0.53	0.42	0.55	0.48
		Baseline-LLM	0.41	0.60	0.63	0.54	0.42	0.54	0.47
		EgoFine	**0.46**	**0.65**	**0.73**	**0.61**	**0.48**	**0.61**	**0.54**
	Ministral-3B	GNN-RAG	0.45	0.65	0.74	0.57	0.72	0.64	0.68
		KG-Adapter	0.43	0.63	0.71	0.54	0.70	0.62	0.66
		Baseline-LLM	0.42	0.67	0.76	0.58	0.73	0.65	0.69
		EgoFine	**0.47**	**0.69**	**0.77**	**0.61**	**0.76**	**0.68**	**0.72**
YAGO3	Minerva-350M	GNN-RAG	0.36	0.56	0.64	0.48	0.74	0.45	0.56
		KG-Adapter	0.38	0.58	0.66	0.50	0.76	0.46	0.57
		Baseline-LLM	**0.42**	0.61	0.69	**0.55**	0.79	0.47	0.59
		EgoFine	**0.42**	**0.63**	**0.72**	**0.55**	**0.83**	**0.55**	**0.66**
	Llama3.2-1B	GNN-RAG	0.58	0.79	0.83	0.69	0.50	0.59	0.54
		KG-Adapter	0.60	0.81	0.84	0.71	0.52	0.61	0.56
		Baseline-LLM	0.57	**0.84**	**0.87**	0.72	0.49	0.59	0.54
		EgoFine	**0.64**	**0.84**	**0.87**	**0.74**	**0.55**	**0.63**	**0.59**
	Qwen2-1.5B	GNN-RAG	0.59	0.78	0.82	0.71	0.49	0.68	0.57
		KG-Adapter	0.61	0.81	0.85	0.74	0.51	0.70	0.59
		Baseline-LLM	0.58	0.69	0.72	0.66	0.45	0.61	0.52
		EgoFine	**0.64**	**0.85**	**0.89**	**0.78**	**0.55**	**0.74**	**0.63**
	Ministral-3B	GNN-RAG	0.61	0.79	0.84	0.71	0.91	0.88	0.89
		KG-Adapter	0.63	0.81	0.86	0.73	0.93	0.90	0.91
		Baseline-LLM	0.61	0.81	0.87	0.73	0.94	0.89	0.91
		EgoFine	**0.66**	**0.83**	**0.88**	**0.76**	**0.95**	**0.92**	**0.94**

and TransE) do not use an LLM for this purpose. Actually, using an LLM provides an extremely high increase in link prediction capability. As a proof of this, we observe that the suboptimal system is always Baseline-LLM, which, like EgoFine and unlike the other systems, uses an LLM. EgoFine proved superior to Baseline-LLM in nearly all cases. This demonstrates that EgoFine's approach of selecting portions of the KG to inject into the LLM, in addition to using the LLM itself, improves its performance compared to Baseline-LLM which uses the LLM but lacks additional refinements.

4.4. Scalability and computational efficiency

In this section, we evaluate the scalability and computational efficiency of EgoFine by analyzing the resources it requires in terms of dataset building time, fine-tuning time, and memory consumption. In this evaluation, we consider EgoFine, Baseline-LLM, and

Table 7
Resource consumption by EgoFine, baseline-LLM, and full when applied to the three KGs using Minerva-350M as LLM.

KG	Approach	Dataset building time	Fine-tuning time	Memory consumption
PrimeKG	Full	582.6 s	14,927 min	200.61 GB
	Baseline-LLM	12.92 s	174 min	11.72 GB
	EgoFine	97.33 s	174 min	11.72 GB
WN18RR	Full	74.48 s	1590 min	3.13 GB
	Baseline-LLM	2.89 s	42 min	0.83 GB
	EgoFine	7.52 s	42 min	0.83 GB
YAGO3	Full	193.85 s	3597 min	38.09 GB
	Baseline-LLM	10.30 s	87 min	2.66 GB
	EgoFine	41.77 s	87 min	2.66 GB

a third configuration, called “Full”, which involves fine-tuning the LLM on the entire KG without any structural reduction. Table 7 shows the values obtained for these three approaches when applied to the three KGs (i.e., PrimeKG, WN18RR, and YAGO3) using Minerva-350M as the underlying LLM.

The results highlight clear trends regarding the scalability and efficiency of EgoFine. The Full configuration involves a significant computational burden in terms of both fine-tuning time and memory consumption. Therefore, operating with Full is not feasible for large-scale applications. Compared to Full, Baseline-LLM and EgoFine drastically reduce the amount of required resources. For instance, as for PrimeKG, the dataset building time of Baseline-LLM (resp., EgoFine) is 2.22% (resp., 16.71%) of that required by Full. Similarly, the fine-tuning time (resp., memory consumption) of Baseline-LLM and EgoFine is 1.17% (resp., 5.84%) of that required by Full. Similar trends can be observed for WN18RR and YAGO3.

The fine-tuning time and memory consumption for Baseline-LLM and EgoFine are the same while the dataset building time of EgoFine is greater than that of Baseline-LLM. However, it is important to note that the values for dataset building time are marginal compared to those for fine-tuning time. This confirms that most of the computational effort lies in model fine-tuning rather than in data preparation. EgoFine only slightly increases dataset building time due to the additional computation of node centrality and ego network filtering. Nevertheless, this overhead remains negligible compared to fine-tuning. Another consistent trend is the stability of memory consumption across various KGs in both Baseline-LLM and EgoFine. In conclusion, we can say that EgoFine strikes a balance between expressiveness and computational efficiency. Its design allows for the efficient management of large and complex KGs, supporting large-scale fine-tuning even in environments with limited hardware resources. The scalability trends observed across different datasets indicate that EgoFine maintains consistent performance while keeping computational requirements feasible.

4.5. Evaluation of EgoFine for multiple choice question answering

In addition to the link prediction experiments, we evaluated the effectiveness of EgoFine when adapted to a Multiple Choice Question–Answering (MCQA) task [49]. The training dataset was constructed using the same procedure as for link prediction. In this experimental setting, the function f_{ft} presented in Section 3.5 was modified to generate structured question–answer pairs consistent with the MCQA format. Rather than generating conversational prompts that ask about the relationship type between two entities, f_{ft} now formulates each instance as a multiple choice question in which the LLM must identify the correct target node from several alternatives. Each question was obtained from the PrimeKG graph by selecting a starting node and a relationship connected to it. Specifically, given a triple (n_i, ρ, n_j), the function generates a question in the following format *“Which node is linked to $<n_i>$ through the relationship $<\rho>$?”*.

Each question includes four randomly ordered answer options, where one of them corresponds to the correct node n_j, two of them correspond to randomly selected nodes that are not connected to the starting node in the graph, and the fourth option, called “None”, reflects the negative sampling mechanism used in the link prediction experiments. This latter option represents the case where the starting node has no connection through the specified relationship. This formulation allows us to evaluate the model’s ability to identify correct connections and recognize non-existent relationships, promoting more discriminative and robust learning. We conducted all experiments using the same LLMs described in Section 4.1, with PrimeKG as the reference KG. We based the evaluation on the accuracy of the given answers. Table 8 shows the accuracy values obtained by EgoFine and Baseline-LLM when operating on PrimeKG with different LLMs.

Upon analyzing this table, we can see that EgoFine has a higher Accuracy value than Baseline-LLM for all LLMs considered. Specifically, EgoFine’s Accuracy exceeds the Baseline-LLM one by 30.91% for Minerva-350M, 16.42% for Llama3.2-1B, 12.33% for Qwen2-1.5B, and 7.59% for Mistral-3B. These results suggest that the proposed fine-tuning strategy based on ego networks enhances the reasoning and relational inference capabilities of LLMs in MCQA tasks involving structured knowledge. The observed trend shows a gradual reduction in improvement as the model grows.

4.6. Ablation study

To further investigate the contribution of each technical choice within EgoFine, we conducted an ablation study. For this study, we used PrimeKG as the reference KG and Minerva-350M as the reference LLM. The only exception was the analysis of centrality

Table 8
Comparison between EgoFine and baseline-LLM on PrimeKG with different LLMs. The results highlight the significant impact of ego network-based knowledge injection on performance in MCQA settings.

Model	Method	Accuracy
Minerva-350M	Baseline-LLM	0.55
	EgoFine	**0.72**
Llama3.2-1B	Baseline-LLM	0.67
	EgoFine	**0.78**
Qwen2-1.5B	Baseline-LLM	0.73
	EgoFine	**0.82**
Ministral-3B	Baseline-LLM	0.79
	EgoFine	**0.85**

Table 9
Comparison of different centrality measures on PrimeKG, WN18RR and YAGO3 using Minerva-350M.

KG	Centrality measure	Hit@1	Hit@3	Hit@5	MRR	Precision	Recall	F1-Score
PrimeKG	Degree	**0.84**	**0.93**	**0.95**	**0.89**	**0.90**	**0.85**	**0.87**
	Betweenness	0.78	0.88	0.91	0.84	0.87	0.83	0.85
	PageRank	0.75	0.89	0.91	0.83	0.84	0.82	0.83
WN18RR	Degree	**0.28**	**0.55**	**0.68**	**0.47**	**0.65**	**0.49**	**0.56**
	Betweenness	0.26	0.52	0.65	0.44	0.63	0.48	0.55
	PageRank	0.25	0.53	0.65	0.44	0.61	0.47	0.53
YAGO3	Degree	**0.42**	**0.63**	**0.72**	**0.55**	**0.83**	**0.55**	**0.66**
	Betweenness	0.39	0.60	0.69	0.52	0.80	0.54	0.64
	PageRank	0.38	0.60	0.69	0.51	0.77	0.53	0.62

measures, which was conducted on all KGs. This analysis aimed to isolate and quantify the roles of the main components and design decisions that characterize EgoFine's approach. In particular, Section 4.6.1 examines the influence of the centrality measure used to select ego networks. Section 4.6.2 evaluates the impact of different strategies for generating the fine-tuning dataset. Section 4.6.3 examines the trade-off between accuracy and computational cost associated with modules for ego network extraction, negative sample generation, and redundancy filtering.

4.6.1. Impact of the centrality measure

In this section, we analyze the effect of the node centrality criterion used to weigh the entities from which the ego networks are extracted. The default configuration of EgoFine uses degree centrality, which prioritizes the most connected nodes. However, other measures, such as betweenness centrality and PageRank, can highlight nodes with different structural roles within the KG. Therefore, we compared these alternatives to see if they would lead to more informative and consistent subgraphs for fine-tuning. The corresponding results for all KGs using Minerva-350M are shown in Table 9.

The analysis of this table shows that, despite the different sizes, domains, and structural properties of evaluated KGs, degree centrality provides the best overall performance in terms of Hit@K, MRR, Precision, Recall, and F1-Score consistently across all KGs. This consistency suggests that the effectiveness of degree centrality is not tied to a specific domain but rather reflects a robust property across heterogeneous graph settings. These results underscore the importance of local connectivity in identifying the most informative ego networks. Indeed, betweenness centrality and PageRank achieve lower values for all metrics. In particular, betweenness centrality tends to favor nodes that act as bridges between heterogeneous communities, generating more fragmented and semantically less coherent subgraphs. PageRank emphasizes globally influential nodes, which may be located in contexts with few direct relationships. This reduces the structural compactness of the resulting ego networks. These results suggest that nodes with strong local connections provide richer and more coherent contexts for LLM adaptation because they encapsulate dense semantic relationships and recurring knowledge patterns in the graph. Conversely, measures that reflect global influence or intermediation capability disperse context across different domains, reducing the specificity of the information learned. For these reasons, we selected degree centrality as the reference one in EgoFine because it strikes an effective balance between simplicity, interpretability, and the ability to identify the most representative and semantically consistent regions of the KG for fine-tuning.

4.6.2. Impact of the ego network sampling strategy

In this section, we analyze how the traversal mode of each ego network influences the diversity and representativeness of the generated training data. In addition to the random path sampling strategy adopted by EgoFine, we consider two other exploration modes, i.e., Breadth-First Search (BFS) and Depth-First Search (DFS). Our goal is to evaluate the impact of depth and breadth exploration on the model's performance and generalization capacity. Table 10 shows the results obtained on PrimeKG using Minerva-350M as the LLM.

Table 10
Comparison of results with different ego network traversal techniques on PrimeKG with Minerva-350M.

Traversal strategy	Hit@1	Hit@3	Hit@5	MRR	Precision	Recall	F1-Score
Random path	**0.84**	**0.93**	**0.95**	**0.89**	**0.90**	**0.85**	**0.87**
DFS	0.77	0.81	0.83	0.79	0.87	0.82	0.84
BFS	0.70	0.79	0.82	0.77	0.83	0.79	0.81

Table 11
Evaluation of several design variants of EgoFine on PrimeKG with Minerva-350M.

Setting	Hit@1	Hit@3	Hit@5	MRR	Precision	Recall	F1-Score
Full EgoFine	**0.84**	**0.93**	**0.95**	**0.89**	**0.90**	**0.85**	**0.87**
Without Jaccard Filtering	0.79	0.82	0.87	0.84	0.82	0.83	0.82
Without Degree Centrality	0.75	0.79	0.84	0.78	0.83	0.77	0.80
Without Ego Networks	0.69	0.84	0.88	0.78	0.89	0.70	0.78

As shown in this table, the traversal strategy significantly influences the quality of learning. Among the methods considered, random path is the most effective, achieving the best performance in all metrics. This strategy favors generating more varied paths consistent with the graphs' semantic structure, allowing the model to learn representations that reflect a more natural and continuous flow of knowledge. In contrast, the BFS strategy yields the poorest results because it tends to favor nodes that are directly connected to the starting node, which may not have meaningful connections to each other. This type of exploration produces fragmented and semantically inconsistent paths that limit the model's ability to capture higher-order relationships between concepts. In other words, BFS explores breadth, but it does not capture the sequentiality and logical dependency that characterize a coherent flow of knowledge. EgoFine's strength lies in its ability to provide the LLM, during fine-tuning, with paths representing complete and interconnected chains of information, in which each node contributes to the overall meaning.

4.6.3. Contribution of key components

This section presents an ablation study evaluating the contribution of each EgoFine component to the system's overall performance and computational cost. Specifically, we focus on four key modules, i.e., degree centrality computation, ego network extraction, redundant subgraph filtering through Jaccard similarity, and negative sample generation. To identify any issues with these modules individually, we start with the full version of EgoFine and remove one component at a time, measuring the impact on prediction accuracy and processing time. Table 11 summarizes the results obtained on the PrimeKG dataset using Minerva-350M as the underlying LLM. The first row, "Full EgoFine", corresponds to the complete configuration of EgoFine, which includes all modules. The next rows depict simplified variants obtained by removing one component. Specifically, the "Without Jaccard Filtering" configuration omits redundancy control based on Jaccard similarity. This results in the processing of multiple ego networks that may share a large portion of nodes in the training set. The "Without Degree Centrality" configuration creates ego networks from randomly selected nodes, instead of prioritizing the most central ones, thus removing structural information related to node importance. Finally, the "Without Ego Networks" configuration replaces the extraction of ego networks and the subsequent generation of random paths within them; the random paths generated are sampled uniformly across the entire KG, which omits local structural information. The "Without Negative Sample Generation" configuration is not included in the table because removing this mechanism causes a drastic degradation of results. In fact, if the model never encounters entity pairs without a relationship, it assumes that there is always a connection between any two nodes. Consequently, it overfits on positive triples, producing incorrect outputs during inference.

As shown in Table 11, all modules play a significant role in the overall effectiveness of EgoFine. The full configuration achieves the best performance across all metrics, confirming that the combination of degree-based node selection, ego network extraction, redundancy filtering, and negative sampling guarantees semantic richness and structural diversity in the training data. Removing the Jaccard filtering step results in a slight decrease in performance, with reductions of 5.95% in Hit@1, 11.83% in Hit@3, 8.42% in Hit@5, 5.62% in MRR, 8.89% in Precision, 2.35% in Recall, and 5.75% in F1-Score. When the degree centrality computation is excluded, the decrease in performance becomes more pronounced. Hit@1 decreases by 10.71%, Hit@3 by 15.05%, Hit@5 by 11.58%, MRR by 12.36%, Precision by 7.78%, Recall by 9.41%, and F1-Score by 8.05%. These results confirm that degree-based ranking is crucial to guide the extraction process toward the most informative regions of the graph. Without this prioritization, the extracted subgraphs contain less relevant information, which reduces the model's ability to learn meaningful structural and semantic dependencies. Removing ego networks results in the most substantial decline in performance. Specifically, Hit@1 decreases by 17.86%, Hit@3 by 9.68%, Hit@5 by 7.37%, and MRR by 12.36%. Precision decreases by 1.11%, Recall decreases by 17.65% and F1-Score decreases by 10.34%. These results clearly demonstrate that the extraction of ego networks is central to the EgoFine approach. By capturing local structural and relational patterns, ego networks enable the model to internalize coherent knowledge representations that cannot be obtained through random sampling.

Our ablation study confirms that EgoFine's performance stems from the components' synergistic interaction. The combination of degree-based node selection and ego network extraction provides the structural foundation that ensures informativeness and contextual coherence. Meanwhile, Jaccard filtering enhances data diversity. Negative samples are essential to preserve the model's discriminative power and to prevent systematic bias toward positive relationships.

Table 12
Structural comparison between $\mathcal{G}_{init}$ and $\mathcal{G}_{update}$.

Statistic	$\mathcal{G}_{init}$	$\mathcal{G}_{update}$
Number of nodes	125,564	129,375
Number of arcs	3,828,511	4,049,642
Number of relation types	16	18

Table 13
Performance comparison between full fine-tuning on $\mathcal{G}_{init}$ and incremental fine-tuning on $\mathcal{G}_{update}$, using either a global LoRA-based updating strategy or a localized one. Training time refers only to the fine-tuning phase. The results were computed using two PrimeKG snapshots with Minerva-350M. These performance values highlight EgoFine's ability to handle new triples and relationships.

KG	Fine-tuning strategy	Hit@1	Hit@3	Hit@5	MRR	Precision	Recall	F1-Score	Time
$\mathcal{G}_{init}$	Full fine-tuning	0.81	0.88	0.90	0.86	0.87	0.83	0.85	162 min
$\mathcal{G}_{update}$	LoRA fine-tuning	0.84	0.93	0.94	0.89	0.92	0.86	0.89	22 min
$\mathcal{G}_{update}$	Localized fine-tuning	0.82	0.88	0.91	0.83	0.91	0.80	0.85	35 min

4.7. Evaluation of EgoFine on dynamic knowledge graphs

The experiments discussed above considered static snapshots of KGs. However, in some real-world settings, KGs evolve over time due to the continuous introduction of new facts and structural refinements. To analyze EgoFine's behavior under these conditions, we performed an additional experiment to evaluate its ability to incorporate new knowledge incrementally while preserving the previously acquired knowledge.

We performed this analysis on PrimeKG, using Minerva-350M as the underlying LLM. To simulate a temporally evolving KG, we constructed two successive snapshots of the same KG. The first snapshot, denoted $\mathcal{G}_{init}$, represents an earlier and incomplete version of the KG. We obtained it by first removing two relationship types from the KG, thereby reducing its relationship schema. Then, we randomly removed 3% of the nodes and arcs from the KG. As a result, some entities became structurally disconnected. For this reason, we removed all nodes that became isolated after this pruning process. Thus, the resulting KG, $\mathcal{G}_{init}$, has a reduced schema and partial structural sparsification compared to the schema of the original KG.

We applied EgoFine to $\mathcal{G}_{init}$ and fine-tuned Minerva-350M using the training dataset generated from it. To ensure that the model learned only the relational patterns available in $\mathcal{G}_{init}$, we constructed training, validation, and test splits from the corresponding triples.

Next, we created a second snapshot, $\mathcal{G}_{update}$, restoring the KG to its original configuration. Specifically, we reintroduced the two relationship types that we had removed, reinserted all the deleted triples, and reinstated the pruned nodes and arcs. Thus, $\mathcal{G}_{update}$ represents the KG at a later stage, characterized by a higher number of triples and relationship types. Table 12 reports the structural statistics of $\mathcal{G}_{init}$ and $\mathcal{G}_{update}$.

Rather than retraining the model from scratch using $\mathcal{G}_{update}$, we adopted a parameter-efficient, incremental fine-tuning strategy based on Low-Rank Adaptation (LoRA) [29]. Specifically, we froze the model parameters that we had previously fine-tuned on $\mathcal{G}_{init}$, and inserted trainable low-rank adapters into its layers. During the second training process, we only updated the adapter parameters using the newly introduced triples.

In addition to this global fine-tuning strategy, we evaluated a localized variant, restricting the update to the portion of the KG directly affected by the newly introduced structural elements. In this case, we applied EgoFine only to the subgraph consisting of the reintroduced triples and their neighboring nodes. This localized strategy confines the fine-tuning process to the updated region of the KG, limiting potential interference with unrelated parts of it while still exploiting the structural context of the affected nodes.

We performed the evaluation using the same metrics as the previous experiments. Table 13 shows the results of full fine-tuning on $\mathcal{G}_{init}$, global LoRA-based fine-tuning on $\mathcal{G}_{update}$, and localized fine-tuning on $\mathcal{G}_{update}$.

From the analysis of this table we can see that the LoRA-based incremental fine-tuning maintains stable performance on previously learned relationship types while effectively incorporating new ones. The localized strategy achieves comparable predictive performance, suggesting that EgoFine can effectively handle incremental KG growth and the evolution of the relationships stored therein.

4.8. EgoFine's robustness to noisy knowledge graphs

KGs built from heterogeneous sources or automatic extraction pipelines may contain inaccurate or inconsistent triples. Since these imperfections could affect the quality of the fine-tuning process, we analyzed the behavior of EgoFine after the introduction of different levels of noise into the KG. In particular, we aimed to evaluate EgoFine's stability under progressively corrupted relationship structures and quantify how noisy triples impact link prediction performance.

Starting from PrimeKG, we generated noisy variants by corrupting a fixed percentage of the triples. We considered two complementary noise mechanisms. The first introduced semantic noise by randomly changing the relationship type for an existing triple while keeping the triple's entities unchanged. The second introduced structural noise by generating spurious triples by randomly pairing entities that were not connected in the original KG and assigning a randomly chosen relationship type to the entity pair.

Table 14
Performance of EgoFine under increasing levels of synthetic noise injected into the KG. The values in the table highlight EgoFine's robustness against noise.

Noise level	Hit@1	Hit@3	Hit@5	MRR	Precision	Recall	F1-Score
0%	0.84	0.93	0.95	0.89	0.90	0.85	0.87
1%	0.84	0.92	0.94	0.89	0.89	0.85	0.87
3%	0.80	0.91	0.93	0.86	0.87	0.84	0.85
5%	0.73	0.88	0.91	0.81	0.82	0.80	0.81
10%	0.67	0.86	0.90	0.78	0.78	0.81	0.79

Table 15
A schematic summary of the main advantages of EgoFine.

Main advantages of EgoFine
i) It constructs compact, ego-centric subgraphs around key KG entities, reducing traversal costs while preserving essential relational structures.
ii) Unlike RAG-based approaches, it embeds structured domain knowledge directly into model parameters, eliminating runtime retrieval and accelerating inference.
iii) Through ego network-based knowledge injection, it strengthens relational reasoning and improves generalization to MCQA tasks.
iv) It supports incremental updates, enabling the integration of new triples and relationships without requiring the entire model to be retrained.

We constructed noisy KG variants by introducing four levels of noise, each involving a different percentage of triples, i.e., 1%, 3%, 5%, and 10%. We applied EgoFine independently to each noisy KG obtained in this way to generate the corresponding training set. We used the same hyperparameters as in the clean KG setting. To isolate the effect of noise in the training phase, we employed the same test set used in the previous experiments, where noise was not considered. Table 14 reports the results obtained.

These results show a gradual decrease in performance as the noise level increases. However, degradation remains limited under moderate corruption, indicating EgoFine's robustness against inaccurate triples. Significant performance degradation only occurs when the noise reaches 10% of the KG. This suggests that EgoFine does not rely heavily on individual triples, but rather learns patterns that are resilient to partial inconsistencies in the KG. We did not consider noise levels higher than 10% because KGs are typically curated to maintain consistent domain knowledge representation. Excessive corruption would compromise the KG's semantic coherence and undermine its role as a structured knowledge representation.

5. Discussion

In the previous section, we showed that EgoFine outperforms related and baseline systems on all metrics for both the link prediction and MCQA tasks. Moreover, we achieved this result on all the four tested LLMs, demonstrating the effectiveness of our framework in fine-tuning LLMs of different sizes and families.

The experiments conducted allow us to summarize the advantages of EgoFine into four main points, as shown in the schematic summary of Table 15.

First, it leverages ego-centric subgraphs (in particular, ego networks) to achieve a cohesive semantic structure for fine-tuning the LLM. By focusing on ego networks centered around high-degree nodes, EgoFine identifies the most influential entities in the KG structure and extracts semantically cohesive KG regions. It then explores these subgraphs along random paths to construct fine-tuning datasets that capture the most relevant parts of the domain knowledge without requiring an exhaustive traversal of the entire KG. This focused sampling strategy allows EgoFine to inject domain-specific knowledge into the LLM preserving important relationship patterns modeled in the KG. As a result, the fine-tuned LLM can achieve superior performance in the knowledge completion task.

The second key point concerns the comparison between EgoFine and Retrieval-Augmented Generation (RAG) techniques. While RAG-based systems maintain a separation between LLMs and external knowledge sources, thus requiring runtime retrieval operations to access relevant information, EgoFine integrates structured knowledge into the LLM through fine-tuning. By embedding domain knowledge directly into LLM parameters, EgoFine enables faster response times and eliminates dependency on external retrieval components during inference. This not only improves efficiency, but also simplifies the system architecture by eliminating the need for dynamic interaction with external KGs at query time.

The third key point concerns EgoFine's performance on the MCQA task. In this context, EgoFine shows that knowledge injected through ego network-based fine-tuning not only enhances relational reasoning, but also improves the model's ability to generalize to higher-level reasoning tasks that require integrating multiple pieces of structured knowledge. This result highlights EgoFine's ability to transfer domain knowledge to LLMs beyond link prediction, allowing for more accurate and semantically grounded responses in question–answering scenarios.

The last key point concerns EgoFine's adaptability to dynamic or noisy KG settings. Indeed, EgoFine can be incrementally fine-tuned when new entities or relationships are introduced in the context, without requiring the underlying model to be fully retrained. Our framework supports KG updates involving the addition of nodes, the insertion of new relationship instances between nodes, and even modifications to the KG schema through the introduction of previously unseen relationship types. In Section 4.7, we validated two complementary incremental strategies, namely: *(i)* a parameter-efficient global update based on LoRA [29], which updates only a fraction of the model parameters, and *(ii)* a localized fine-tuning procedure restricted to the subgraph affected by structural changes. Both strategies reduce computational costs while limiting interference with previously acquired knowledge. Moreover, the robustness

analysis in Section 4.8 shows that EgoFine remains stable under moderate levels of structural and semantic noise. Another strategy could involve continual learning techniques [50], which could further mitigate catastrophic forgetting when sequential updates occur over time. Together, these results suggest that EgoFine is not limited to static KG configurations, but can accommodate dynamic scenarios involving structural updates and noise robustness while preserving previously acquired knowledge.

We conclude this part by discussing the degree centrality-based sampling strategy of EgoFine. Indeed, our framework assumes that nodes with the highest degree centrality are also among the most informative for fine-tuning the LLM. Our ablation study in Section 4.6.1 supports this choice and shows its consistent effectiveness across heterogeneous KGs. However, EgoFine is not restricted to degree-based sampling. Alongside this default strategy, our framework can incorporate alternative or hybrid sampling criteria to better satisfy domain-specific requirements and ensure the coverage of less represented yet relevant knowledge regions.

6. Conclusion

In this paper, we have proposed EgoFine, a new approach to fine-tune Large Language Models (LLMs) that makes use of structured subgraphs extracted from a Knowledge Graph (KG). EgoFine represents an advance over previous approaches, as it introduces the practice of fine-tuning an LLM through a KG by exploiting ego-centric subgraphs (in particular, ego networks). This way of proceeding focuses the learning process on localized areas of the KG, rather than treating the entire KG as a monolithic entity. Specifically, EgoFine identifies high-degree nodes, extracts their ego networks, generates random paths within them, and filters overlapping structures to ensure diversity, thus creating a rich, domain-focused dataset for fine-tuning the LLM. Furthermore, it includes negative samples to enhance the LLM's ability to discriminate between existing and non-existing relationships, addressing the hallucination problem commonly encountered in LLMs.

We evaluated EgoFine on four different LLMs (i.e., Minerva-350M, Llama3.2-1B, Qwen2-1.5B and Ministral-3B) and three KGs (i.e., PrimeKG, WN18RR, and YAGO3), focusing on the link prediction task. Experimental results show that EgoFine consistently outperforms traditional embedding-based methods (i.e., AutoSF, BoxE, NodePiece, PaiRE, and TransE) that do not use an LLM to support their activity. It also outperforms two state-of-the-art approaches combining LLMs and KGs, i.e., GNN-RAG and KG-Adapter, as well as a baseline approach that, like EgoFine, uses paths but, differently from EgoFine, does not use ego-networks centered on nodes with high degree centrality. The improvements observed on all evaluation metrics confirm the effectiveness of EgoFine in improving the understanding of the relationships between the entities of a domain of interest. Furthermore, the results obtained for the MCQA task highlight EgoFine's adaptability. In fact, EgoFine enables LLMs to reason through complex and multi-relational contexts, thereby extending its applicability beyond link prediction. The ablation study showed that design components significantly contribute to EgoFine's performance, confirming their essential role in achieving optimal results. Finally, studies on evolving or noisy KGs showed the EgoFine's ability to handle such scenarios.

EgoFine has important theoretical and practical implications. From a theoretical perspective, it introduces a way to inject structured and contextually relevant knowledge into LLMs using ego network structures. This method allows for a more precise and semantically meaningful transfer of domain knowledge into the LLM. In addition, path-based random sampling, coupled with diversity-enhancing filtering strategies, ensures that the fine-tuning process captures a wide range of relationship patterns while maintaining a focus on semantically heterogeneous areas. From a practical perspective, EgoFine offers a novel and efficient paradigm for developing domain-specific expert systems, where low hallucination rates are critical. EgoFine's ability to preserve meaningful knowledge structures stored in a KG and inject them into LLMs could significantly improve the reliability and interpretability of the output returned by an LLM.

In the future, we plan to extend our work in several directions. The first extension could be the ability to construct ego networks from a KG based on semantic embeddings rather than degree centrality. By computing entity embeddings and measuring their similarity, we aim to build more semantically cohesive subgraphs, further improving the relevance and coherence of the knowledge injected into the LLM. A second direction for future work is the development of dynamic fine-tuning strategies driven by the ego networks' properties. In particular, instead of treating all ego networks equally, we plan to adapt the fine-tuning process taking into account the structural properties of ego networks, such as density and clustering coefficient, optimizing the learning effort according to the complexity and possible contribution of each ego network. Finally, a further line of research concerns the exploration of sampling techniques other than random paths, BFS, and DFS. Techniques like biased paths, motif-based sampling, or reinforcement learning-based traversals could capture more complex relationship patterns within ego networks. Such strategies could increase the diversity and informativeness of the training data, leading to more effective knowledge injection in the LLM.

CRediT authorship contribution statement

Alessia Amelio: Writing – original draft, Project administration, Formal analysis, Conceptualization. **Christopher Buratti:** Validation, Software, Resources, Investigation, Data curation. **Michele Marchetti:** Writing – original draft, Visualization, Supervision, Methodology, Investigation, Formal analysis. **Davide Traini:** Writing – original draft, Visualization, Methodology, Investigation, Formal analysis, Data curation. **Domenico Ursino:** Writing – review & editing, Supervision, Methodology, Funding acquisition. **Luca Virgili:** Writing – review & editing, Supervision, Methodology, Investigation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We acknowledge the support of the PNRR project FAIR - Future AI Research (PE00000013), Spoke 9 - AI, under the NRRP MUR program funded by the NextGenerationEU.

Data availability

The dataset used in our study is publicly available at the link https://github.com/christopherburatti/EgoFine.

References

[1] A. Ferrera, G. Mezzotero, D. Ursino, A linguistics-based approach to refining automatic intent detection in conversational agent design, in: Information Sciences, vol. 689(121493), Elsevier, 2025.
[2] S. Feng, Z. Lang, J. He, H. Zhang, W. Chen, J. Cao, A group recommendation method based on automatically integrating members' preferences via taking advantages of LLM, in: Information Sciences, vol. 709(122067), Elsevier, 2025.
[3] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, A survey on hallucination in large language models: principles, taxonomy, challenges, and open questions, in: ACM Transactions on Information Systems, vol. 43(42), ACM, 2023.
[4] Z. Li, L. Chen, Y. Jian, H. Wang, Y. Zhao, M. Zhang, K. Xiao, Y. Zhang, H. Deng, X. Hou, Aggregation or separation? Adaptive embedding message passing for knowledge graph completion, in: Information Sciences, vol. 691(121639), Elsevier, 2025.
[5] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, X. Wu, Unifying large language models and knowledge graphs: a roadmap, IEEE Trans. Knowl. Data Eng. (2024) 3580–3599. IEEE.
[6] Z. Zhang, X. Liu, Y. Zhang, Q. Su, X. Sun, B. He, Pretrain-KGE: learning knowledge representation from pretrained language models, in: Findings of the Association for Computational Linguistics: EMNLP 2020, ACL, 2020, pp. 259–266, Virtual Event.
[7] B.Y. Lin, X. Chen, J. Chen, X. Ren, KagNet: knowledge-aware graph networks for commonsense reasoning, in: Proc. of the International Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP'19), ACL, Hong Kong, China, 2019, pp. 2829–2839.
[8] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, D. Kiela, Retrieval-augmented generation for knowledge-intensive NLP tasks, in: Proc. of the International Conference on Neural Information Processing Systems (NeurIPS'20), Curran Associates Inc, 2020, pp. 9459–9474, Virtual Event.
[9] X. Wang, T. Gao, Z. Zhu, Z. Zhang, Z. Liu, J. Li, J. Tang, KEPLER: a unified model for knowledge embedding and pre-trained language representation, in: Transactions of the Association for Computational Linguistics, vol. 9, MIT Press, 2021, pp. 176–194.
[10] P. Ke, H. Ji, Y. Ran, X. Cui, L. Wang, L. Song, X. Zhu, M. Huang, JointGT: graph-text joint representation learning for text generation from knowledge graphs, Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, ACL, Bangkok, Thailand, 2021.
[11] M. Calamo, J. Rossi, Programming large language models, in: Engineering Information Systems with Large Language Models, Springer, 2025, pp. 111–137.
[12] P. Chandak, K. Huang, M. Zitnik, Building a knowledge graph to enable precision medicine, in: Scientific Data, vol. 10(1), Nature, 2023, pp. 67.
[13] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: Proc. of the Advances in Neural Information Processing Systems (NIPS'13), vol. 26, ACM, Lake Tahoe, NV, USA, 2013.
[14] T. Rebele, F. Suchanek, J. Hoffart, J. Biega, E. Kuzey, G. Weikum, YAGO: a multilingual knowledge base from wikipedia, wordnet, and geonames, in: Proc. of the International Semantic Web Conference (ISWC'16), Springer, Kobe, Japan, 2016, pp. 177–185.
[15] Y. Zhang, Q. Yao, W. Dai, L. Chen, AutoSF: searching scoring functions for knowledge graph embedding, in: Proc. of the IEEE International Conference on Data Engineering (ICDE'20), IEEE, 2020, pp. 433–444, Virtual event.
[16] R. Abboud, I. Ceylan, T. Lukasiewicz, T. Salvatori, BoxE: a box embedding model for knowledge base completion, in: Proc. of the Advances in Neural Information Processing Systems (NIPS'20), vol. 33, ACM, 2020, pp. 9649–9661, Virtual event.
[17] M. Galkin, E. Denis, J. Wu, W.L. Hamilton, NodePiece: compositional and parameter-efficient representations of large knowledge graphs, in: Proc. of the International Conference on Learning Representations (ICLR'22), Virtual event, ICLR, 2022.
[18] L. Chao, J. He, T. Wang, W. Chu, PairRE: knowledge graph embeddings via paired relation vectors, in: Proc. of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing (ACL-IJCNLP'21), ACL, 2021, pp. 4360–4369, Virtual Event.
[19] C. Mavromatis, G. Karypis, GNN-RAG: graph neural retrieval for efficient large language model reasoning on knowledge graphs, in: Findings of the Association for Computational Linguistics: ACL'25, ACL, Vienna, Austria, 2025, pp. 16682–16699.
[20] S. Tian, Y. Luo, T. Xu, C. Yuan, H. Jiang, C. Wei, X. Wang, KG-adapter: enabling knowledge graph integration in large language models through parameter-efficient fine-tuning, in: Findings of the Association for Computational Linguistics: ACL 2024, ACL, Bangkok, Thailand, 2024, pp. 3813–3828.
[21] H. Tian, C. Gao, X. Xiao, H. Liu, B. He, H. Wu, H. Wang, F. Wu, SKEP: sentiment knowledge enhanced pre-training for sentiment analysis, in: Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL'20), ACL, 2020, pp. 4067–4076, Virtual Event.
[22] T. Sun, Y. Shao, X. Qiu, Q. Guo, Y. Hu, X.J. Huang, Z. Zhang, CoLAKE: contextualized language and knowledge embedding, in: Proc. of the International Conference on Computational Linguistics (COLING'20), International Committee on Computational Linguistics, 2020, pp. 3660–3670, Virtual Event.
[23] H. Luo, E. Haihong, Z. Tang, S. Peng, Y. Guo, W. Zhang, C. Ma, G. Dong, M. Song, W. Lin, Y. Zhu, A.T. Luu, ChatKBQA: a generate-then-retrieve framework for knowledge base question answering with fine-tuned large language models, Findings of the Association for Computational Linguistics: ACL 2024, ACL, Bangkok, Thailand, pp. 2039–2056, 2024.
[24] Y. Wu, Y. Zhao, B. Hu, P. Minervini, P. Stenetorp, S. Riedel, An efficient memory-augmented transformer for knowledge-intensive NLP tasks, in: Proc. of the International Conference on Empirical Methods in Natural Language Processing (EMNLP'22), ACL, Abu Dhabi, United Arab Emirates, 2022, pp. 5184–5196.
[25] J. Wang, Q. Sun, X. Li, M. Gao, Boosting language models reasoning with chain-of-knowledge prompting, in: Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL'24), ACL, Bangkok, Thailand, 2024, pp. 4958–4981.
[26] Y. Wen, Z. Wang, J. Sun, MindMap: knowledge graph prompting sparks graph of thoughts in large language models, in: Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL'24), ACL, Bangkok, Thailand, 2024, pp. 10370–10388.
[27] L. Luo, T. Vu, D. Phung, R. Haf, Systematic assessment of factual knowledge in large language models, in: Findings of the Association for Computational Linguistics: EMNLP 2023, ACL, Singapore, 2023, pp. 13272–13286.
[28] M. Yasunaga, H. Ren, A. Bosselut, P. Liang, J. Leskovec, QA-GNN: reasoning with language models and knowledge graphs for question answering, in: Proc. of the International Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT'21), ACL, 2021, pp. 535–546, Virtual Event.
[29] E.J. Hu, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, LoRA: low-rank adaptation of large language models, in: Proc. of the 10th International Conference on Learning Representations (ICLR'22), Virtual Event, 2022.
[30] T. Dettmers, A. Pagnoni, A. Holtzman, L. Zettlemoyer, QLoRA: efficient finetuning of quantized LLMs, in: Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems (NeurIPS'23), New Orleans, LA, USA, 2023.
[31] P.F. Christiano, J. Leike, T.B. Brown, M. Martic, S. Legg, D. Amodei, Deep reinforcement learning from human preferences, in: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems (NeurIPS'17), Long Beach, CA, USA, 2017, pp. 4299–4307.
[32] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon, C. Chen, C. Olsson, C. Olah, D. Hernandez, D. Drain, D. Ganguli, D. Li, E. Tran-Johnson, E. Perez, J. Kerr, J. Mueller, J. Ladish, J. Landau, K. Ndousse, K. Lukosiute, L. Lovitt, M. Sellitto, N. Elhage, N. Schiefer, N. Mercado, N. DasSarma, R. Lasenby, R. Larson, S. Ringer, S. Johnston, S. Kravec, S.E. Showk, S. Fort, T. Lanham, T. Telleen-Lawton, T. Conerly, T.

Henighan, T. Hume, S.R. Bowman, Z. Hatfield-Dodds, B. Mann, D. Amodei, N. Joseph, S. McCandlish, T. Brown, J. Kaplan, Constitutional AI, arXiv preprint arXiv:2212.08073, 2022.
[33] R. Rafailov, A. Sharma, E. Mitchell, C.D. Manning, S. Ermon, C. Finn, Direct preference optimization: your language model is secretly a reward model, in: Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems (NeurIPS'23), New Orleans, LA, USA, 2023.
[34] H. Ye, N. Zhang, S. Deng, X. Chen, H. Chen, F. Xiong, X. Chen, H. Chen, Ontology-enhanced prompt-tuning for few-shot learning, in: Proc. of the ACM International Web Conference (WWW'22), ACM, Lyon, France, 2022, pp. 778–787, Virtual Event.
[35] J. Wang, W. Huang, M. Qiu, Q. Shi, H. Wang, X. Li, M. Gao, Knowledge prompting in pre-trained language model for natural language understanding, in: Proc. of the International Conference on Empirical Methods in Natural Language Processing (EMNLP'22), ACL, Abu Dhabi, United Arab Emirates, 2022, pp. 3164–3177.
[36] J. Sun, Z. Du, Y. Chen, Knowledge graph tuning: real-time large language model personalization based on human feedback, arXiv preprint arXiv:2405.19686, 2024.
[37] Y. Luo, Z. Zhou, H. Chen, K. Qiu, M. Savvides, Y. Li, J. Wang, KnowledgeSmith: uncovering knowledge updating in LLMs with model editing and unlearning, arXiv preprint arXiv:2510.02392, 2025.
[38] L. Luo, Y.F. Li, R. Haf, S. Pan, Reasoning on graphs: faithful and interpretable large language model reasoning, in: Proc. of the International Conference on Learning Representations (ICLR'24), Vienna, Austria, 2024.
[39] T. Feng, L. He, RGR-KBQA: generating logical forms for question answering using knowledge-graph-enhanced large language model, in: Proc. of the International Conference on Computational Linguistics (COLING'25), ACL, Abu Dhabi, United Arab Emirates, Jan 2025, pp. 3057–3070.
[40] D. Banerjee, Semantic parsing for knowledge graph question answering with large language models, in: Proc. of the European Semantic Web Conference (ESWC'23), Springer, Hersonissos, Greece, 2023, pp. 234–243.
[41] L. Ding, N. Ding, Q. Tao, P. Shi, Enhancing graph multi-hop reasoning for question answering with LLMs: an approach based on adaptive path generation, J. Intell. Inf. Syst. 63 (2025) 1455–1485.
[42] R. Liao, X. Jia, Y. Li, Y. Ma, V. Tresp, GenTKG: generative forecasting on temporal knowledge graph with large language models, in: Findings of the Association for Computational Linguistics: NAACL'24, ACL, Mexico City, Mexico, 2024, pp. 4303–4317.
[43] Y. Ji, K. Wu, J. Li, W. Chen, M. Zhong, J. Xu, M. Zhang, Retrieval and reasoning on KGs: integrate knowledge graphs into large language models for complex question answering, in: Findings of the Association for Computational Linguistics: EMNLP 2024, ACL, Miami, FL, USA, 2024, pp. 7598–7610.
[44] F. Remy, K. Demuynck, T. Demeester, BioLORD-2023: semantic textual representations fusing large language models and clinical knowledge graph insights, J. Am. Med. Inform. Assoc. 31 (9) (2024) 1844–1855.
[45] A. Cadeddu, A. Chessa, V. De Leo, G. Fenu, E. Motta, F. Osborne, D.R. Recupero, A. Salatino, L. Secchi, Optimizing tourism accommodation offers by integrating language models and knowledge graph technologies, in: Information, vol. 15(7), MDPI, 2024, pp. 398.
[46] P. Liu, L. Qian, X. Zhao, B. Tao, Joint knowledge graph and large language model for fault diagnosis and its application in aviation assembly, IEEE Trans. Ind. Inform. 20 (6) (2024) 8160–8169. IEEE.
[47] H. Chen, X. Shen, J. Wang, Z. Wang, Q. Lv, J. He, R. Wu, F. Wu, J. Ye, Knowledge graph finetuning enhances knowledge manipulation in large language models, in: Proc. of the International Conference on Learning Representations, ICLR'25), Singapore, 2025. OpenReview.net.
[48] S. Dernbach, K. Agarwal, A. Zuniga, M. Henry, S. Choudhury, GLaM: fine-tuning large language models for domain knowledge graph alignment via neighborhood partitioning and generative subgraph encoding, in: Proc. of the AAAI Symposium Series (AAAI'24), vol. 3, Vancouver, British Columbia, Canada, 2024, pp. 82–89.
[49] Z. Zhang, Z. Jiang, L.X. Hongkun, H. Hao, R. Wang, Multiple-choice questions are efficient and robust LLM evaluators, arXiv preprint arXiv:2405.11966, 2024.
[50] L. Wang, X. Zhang, H. Su, J. Zhu, A comprehensive survey of continual learning: theory, method and application, IEEE Trans. Pattern Anal. Mach. Intell. 46 (8) (2024) 5362–5383. IEEE.