Harnessing Diverse Perspectives: A Multi-Agent Framework for Enhanced Error Detection in Knowledge Graphs

Yu Li[1], Yi Huang[2] (✉), Guilin Qi[1] (✉), Junlan Feng[2], Nan Hu[1], Songlin Zhai[1], Haohan Xue[1], Yongrui Chen[1], Ruoyan Shen[1], and Tongtong Wu[3]

[1] Southeast University, Nanjing, China
{yuli_11, gqi, nanhu, songlin_zhai, thex1ay, yongruichen,
ry_shen}@seu.edu.cn
[2] China Mobile Research Institute, China
{huangyi, fengjunlan}@chinamobile.com
[3] Monash University, Australia
tongtong.wu@monash.edu

Abstract. Knowledge graphs are widely used in industrial applications, making error detection crucial for ensuring the reliability of downstream applications. Existing error detection methods often fail to effectively utilize fine-grained subgraph information and rely solely on fixed graph structures, while also lacking transparency in their decision-making processes, which results in suboptimal detection performance. In this paper, we propose a novel Multi-Agent framework for Knowledge Graph Error Detection (MAKGED) that utilizes multiple large language models (LLMs) in a collaborative setting. By concatenating fine-grained, bidirectional subgraph embeddings with LLM-based query embeddings during training, our framework integrates these representations to produce four specialized agents. These agents utilize subgraph information from different dimensions to engage in multi-round discussions, thereby improving error detection accuracy and ensuring a transparent decision-making process. Extensive experiments on FB15K and WN18RR demonstrate that MAKGED outperforms state-of-the-art methods, enhancing the accuracy and robustness of KG evaluation. For specific industrial scenarios, our framework can facilitate the training of specialized agents using domain-specific knowledge graphs for error detection, which highlights the potential industrial application value of our framework. Our code and datasets are available at `https://github.com/kse-ElEvEn/MAKGED`.[1]

Keywords: Large Language Models · Knowledge Graph · Multi-Agent

1 Introduction

Knowledge graphs (KGs) [1] represent facts in the real world as triples, such as *(Paris, capital_ of, France)*, facilitating the organization and scaling of informa-

---

tion [5], and have gained paramount importance in knowledge-based systems, such as retrieval-augmented generation and recommendation systems [20]. However, most large-scale KGs [1], built using rule-based methods and statistics-based methods to extract web data, often contain noisy or incorrect triples. For instance, the widely used knowledge graph NELL [15] contains around 600K incorrect triples, which account for 26% of the set of triples in NELL. Most knowledge graph-driven tasks assume all triples are correct [15], overlooking the impact of errors, which significantly degrades the performance of downstream tasks. This highlights the urgent need for effective KG error detection methods.

Existing methods can be broadly categorized into two types based on their evidence utilization. First, explicit evidence-based methods extract paths or sub-graphs from the knowledge graph that support the truth of triples and use these explicit evidences to predict the credibility of triples [16]. Second, embedding-based methods predict the credibility of triples by embedding entities and relations into vector space and calculating the embedded representation of paths or subgraphs [7, 9]. Moreover, when combined with contrastive learning and pre-trained models, their performance has shown significant improvement [13,14,24].

However, these methods have two main limitations: (1) They evaluate triples from a single, fixed perspective, either relying on static structural patterns (e.g., static graph embeddings) or a unidirectional semantic method (e.g., text embeddings). This means they assume predefined, unchanging connections between entities, without considering the dynamic context of the triples being evaluated. As a result, they struggle to adapt when the graph structure or textual representations change. (2) They lack transparency in the evaluation process, typically providing a single confidence score without sufficient context about how decisions are made. These two limitations hinder error identification and impede model improvement.

To address the limitations of existing methods, we introduce a multi-agent framework, as shown in Figure 1. For each triple in the KG, we assign two agents to the head and tail entities. The *Forward Agent* collects subgraphs with the entity as the head, and the *Backward Agent* collects subgraphs with it as the tail, integrating multiple perspectives for error detection. Then, we process the collected subgraphs using a Graph Convolutional Network (GCN) for structural features and an LLM for semantic features. By concatenating the GCN and LLM embeddings, we combine structural and semantic information, leveraging the LLM's text generation capabilities for error detection.

Moreover, to address the lack of transparency during evaluation, we introduce a structured discussion and voting mechanism. The multiple agents independently evaluate each triple and then engage in multiple rounds of discussion until they reach a final decision. This method enhances detection accuracy and increases transparency in the decision-making process. At the same time, our framework can utilize domain-specific knowledge graphs to train specialized agents for industrial scenarios, improving error detection accuracy. It addresses the limitations of existing LLMs, which struggle with domain-specific knowl-
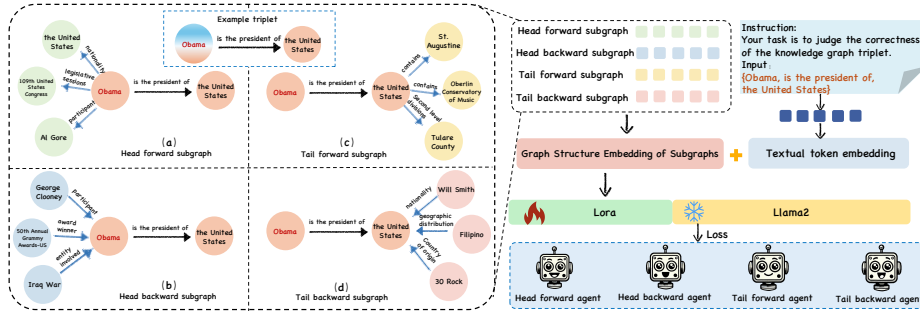
Fig. 1. Training method for bidirectional subgraph agents in MAKGED: We first construct bidirectional subgraphs for the head and tail entities of the triple, which are represented as embedding vectors using GCN. These embeddings are then concatenated with the query embeddings of the Llama2 model, resulting in four bidirectional subgraph agents to evaluate the correctness of the triples.

edge, thereby delivering significant industrial value. The main contribution are summarized as follows:

1. To our knowledge, we are the first to introduce a multi-agent framework, MAKGED, for knowledge graph error detection. Agents engage in several rounds of discussion and vote to determine triple correctness, thus enhancing the clarity and accuracy of decisions through collaborative evaluation.
2. We design four trainable agents, including the "*Forward-Agent*" and "*Backward-Agent*" for the head entity and tail entity, respectively. These agents are trained on bidirectional subgraph embeddings, which enhances the robustness and adaptability of error detection.
3. We build datasets with simulated graph noise, and experiments demonstrate that MAKGED improves accuracy by 0.73% on FB15K and 6.62% on WN18RR compared to the state-of-the-art methods.

2   Related Work

Knowledge Graph Error Detection: Knowledge graph error detection [15] includes methods based on explicit and implicit evidence.

Explicit Evidence : These methods evaluate the credibility of triples using direct evidence from the graph, such as paths and subgraphs. For example: Shi et al. [16] used graph traversal to extract meta-paths and calculate path weights. Lin et al. [12] applied graph pattern mining with metrics like support and confidence. However, these methods may struggle with incomplete knowledge graphs and underutilize contextual information.

Implicit Evidence : These methods use embedding models to map entities and relations into vector spaces for credibility evaluation. CKRL [21] optimized

triple credibility by considering local and global factors during embedding learning. KGTtm [9] combined random walks and embeddings, using MLP for scoring. Recent methods also use pre-trained language models and contrastive learning: KG-BERT [24] enriches semantic representations with pre-trained models. SeSICL [14], StAR [19], and CCA [13] align textual and structural representations via contrastive learning. These methods improve performance but rely on fixed graph structures, limiting adaptability to new or unseen structures.

Multiple LLM-based Agent System: Multi-Agent systems such as CAMEL [10], have gained significant attention for simulating complex interactions among intelligent agents. These systems coordinate multiple LLM agents to tackle complex tasks, particularly in NLP evaluation. For example, ChatEval [3] uses a multi-agent jury to explore the impact of communication strategies on evaluating open-ended questions and traditional NLG tasks, while MATEval [11] involves agents collaborating to evaluate story texts, enhancing the evaluation process through multi-agent interaction.

3 Problem Statement

3.1 Knowledge Graph Error

We define a knowledge graph $\mathcal{G}$ as a set of triples (h, r, t), where h is the head entity, r the relation, and t the tail entity. Error detection involves determining if a given triple is correct or incorrect, with the output being a binary label.

A triple is **incorrect** if the head or tail entity does not align with the relation [22]. For example, *(Harvard University, is_located_in, New York)*. Conversely, a triple is **correct** if all components align appropriately.

3.2 Subgraph Definitions

To analyze the context of a triple, we define two key concepts for each entity: *Out_Neighbor Subgraph* and *In_Neighbor Subgraph*.

Out_Neighbor Subgraph: The set of triples where the entity serves as the head. For an entity e, the *Out_Neighbor Subgraph* is $\{(e, r', t') \mid (e, r', t') \in \mathcal{G}\}$, where r' is outgoing relations from e, and t' is the corresponding tail entity.

In_Neighbor Subgraph: The set of triples where the entity serves as the tail. For an entity e, the *In_Neighbor Subgraph* is $\{(h', r', e) \mid (h', r', e) \in \mathcal{G}\}$, where h' is the corresponding head entity, and r' represents incoming relations to e.

Based on these concepts, for a given triple (h, r, t), we define the following subgraphs for both the head h and the tail t:

(a) *Head_Forward_Subgraph*: The *Out_Neighbor Subgraph* of the head entity h, excluding the current triple (h, r, t). Formally:

$$\{(h, r', t') \mid (h, r', t') \in \mathcal{G}, (r', t') \neq (r, t)\} \tag{1}$$

(b) *Head_Backward_Subgraph*: The *In_Neighbor Subgraph* of the head entity h, capturing all incoming relations to h. Formally:

$$\{(h', r', h) \mid (h', r', h) \in \mathcal{G}\} \tag{2}$$

(c) **_Tail_ _Forward_ _Subgraph_**: The _Out_ _Neighbor Subgraph_ of the tail entity t, capturing all outgoing relations from t. Formally:

$$\{(t, r', t') \mid (t, r', t') \in \mathcal{G}\} \tag{3}$$

(d) **_Tail_ _Backward_ _Subgraph_**: The _In_ _Neighbor Subgraph_ of the tail entity t, excluding the current triple (h, r, t). Formally:

$$\{(h', r', t) \mid (h', r', t) \in \mathcal{G}, (h', r') \neq (h, r)\} \tag{4}$$

3.3 Agent Construction

We construct four agents based on the above subgraphs: _Head_ _Forward_ _Agent_, _Head_ _Backward_ _Agent_, _Tail_ _Forward_ _Agent_ and _Tail_ _Backward_ _Agent_. Each agent analyzes the corresponding subgraph for the triple (h, r, t), enabling a multi-angle evaluation of the triple by considering both head and tail entities' forward and backward contexts.

4 Method

4.1 Design of the Framework

In our framework, we employ multiple LLM-based agents working collaboratively to detect errors in KGs. Using the structural information of the graph, we construct four bidirectional subgraph agents for both the head and tail entities. These agents analyze the contextual information of triples from different perspectives, and a final decision on the correctness of the triples is made through a voting mechanism. The detailed explanation of this process is provided below:

Bidirectional Subgraph Agents: In our MAKGED framework, we design four bidirectional subgraph agents to evaluate triples in the knowledge graph. Each of these agents is responsible for analyzing triples from a specific directional perspective, including the _Head_ _Forward_ _Agent_ and _Head_ _Backward_ _Agent_ for the head entity, and the _Tail_ _Forward_ _Agent_ and _Tail_ _Backward_ _Agent_ for the tail entity, as illustrated in Figure 1.

First, we construct bidirectional subgraphs for both the head and tail entities of each triple. For the head entity, the _Head_ _Forward_ _Agent_ extracts the _Out_ _Neighbor subgraph_, where the edges represent outgoing relations from the head entity; concurrently, the _Head_ _Backward_ _Agent_ extracts the _In_ _Neighbor subgraph_, where the edges represent incoming relations directed toward the head entity. Similarly, for the tail entity, the _Tail_ _Forward_ _Agent_ and _Tail_ _Backward-_ _Agent_ generate forward and backward subgraphs, representing the tail entity as either a head node or a tail node in related subgraphs.

Once the subgraphs are constructed, we process them using a Graph Convolutional Network to generate the corresponding subgraph embedding vectors. Let the subgraph embeddings be denoted as $\mathbf{z}_G$. These subgraph embeddings are then concatenated with the embedding vectors generated by Llama2, denoted
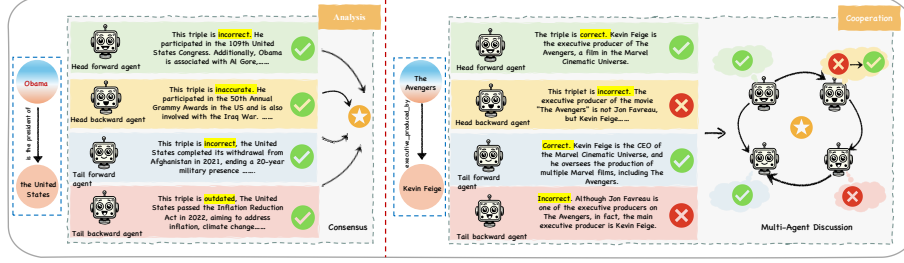
Fig. 2. This figure illustrates the collaborative decision-making process using multiple agents. In the "analysis" phase, the four agents independently evaluate the triple. If no consensus is reached, they proceed to the "cooperation" phase for discussion. The final decision is made either by majority rule after three rounds of discussion, or by a summarizer in case of a *2-vs-2* tie.

as $\mathbf{e}_{text}$, which provides textual information for assessing the correctness of the triples. Llama2 embeddings provide the textual representation of the triples, while the structural information from the subgraph embeddings adds complementary context. By concatenating both semantic and structural information, we create a richer, more expressive unified embedding representation:

$$\mathbf{e}_{concat} = [\mathbf{z}_G; \mathbf{e}_{text}] \tag{5}$$

where $\mathbf{z}_G$ denotes the graph-based embeddings generated by the GCN module. $\mathbf{e}_{text}$ represents the semantic embeddings derived from Llama2. Next, these concatenated embeddings are used as input to further fine-tune the Llama2 model. During fine-tuning, the model learns not only how to combine textual and structural embeddings to improve its accuracy but also how to optimize its decision-making based on the distinct features of the subgraphs in each direction. The input sequence to the model is defined as:

$$S_{it} = I_{it} \oplus \mathbf{e}_{concat} \oplus A_{it}, \tag{6}$$

where I_{it} is the instruction prompt, $\mathbf{e}_{concat}$ is the concatenated embedding from both the GCN and Llama2, and A_{it} is the predicted answer during training. The training objective is to minimize the following loss function:

$$\mathcal{L}_{it} = -\frac{1}{|S_{it}|} \sum_{i=1}^{|S_{it}|} \log P_{\mathcal{M}}(s_i \mid s_{<i}, \mathbf{e}_{concat}), \tag{7}$$

where $|S_{it}|$ represents the length of the input sequence S_{it}, and s_i is the token at position i in the input sequence. $P_{\mathcal{M}}(s_i \mid s_{<i}, \mathbf{e}_{concat})$ is the probability distribution predicted by the model for token s_i, conditioned on all previous tokens and the concatenated embedding. During the training process, we simultaneously train the Llama2 to evaluate the correctness of triples in scenarios where

reasoning is provided. In this part, subgraph information is incorporated into the input, allowing the model to fine-tune its ability to discuss the correctness of triples based on reasoning.

As a result, we train four specialized agents, each tailored to specific directional tasks for either the head or tail entities (forward or backward). This method allows us to comprehensively evaluate the correctness of triples from multiple directions, significantly enhancing the performance and accuracy.

Agent Decision: The agents trained in the previous process are used for the KG error detection task on the test set. This process is divided into two phases: the analysis phase and the cooperation phase. In the analysis phase, the four agents (*Head_Forward_Agent* and *Head_Backward_Agent* for the head entity, and *Tail_Forward_Agent* and *Tail_Backward_Agent* for the tail entity) evaluate the correctness of a target triple independently, making full use of the corresponding subgraph information they learned during training and minimizing mutual interference.

After collecting the results, a consistency check is performed. If all agents agree on the correctness of the triple (i.e., consensus), it is classified as correct or incorrect. If there is disagreement, the process moves to the cooperation phase.

In the cooperation phase, the four agents engage in a collective discussion, exchanging their viewpoints and background knowledge to resolve disagreements regarding the triple. This discussion process iterates for up to three rounds, stopping early if consensus is reached within these rounds. After each round of discussion, the agents update their judgments. At the end of the discussion, a "majority rule" strategy is employed to determine the final decision. If a *2-vs-2* tie still occurs after the three rounds, the final decision is made by a summarizer agent, which receives the full context of the three discussion rounds as a structured prompt. This prompt includes key arguments, evidence, and conclusions from all agents, enabling the summarizer to make an informed judgment that reflects the collective reasoning of the agents. On average, in our experience, agents reached a consensus within 1.8 rounds of discussion. In about 12% of cases, a *2-vs-2* tie occurred, which was resolved by the summarizer agent. The entire agent discussion process is illustrated in Figure 2.

5 Experiments

To validate the effectiveness of our proposed MAKGED framework, we conducted comprehensive experiments on two representative knowledge graph datasets in this section, as well as in industrial scenarios such as China Mobile. Specifically, we aim to answer the following research questions through experiments:
RQ1: How does MAKGED perform compared to state-of-the-art KG error detection methods?
RQ2: How does each component of the MAKGED framework contribute to its overall performance?
RQ3: Can the MAKGED framework successfully detect specific errors in knowledge graphs, especially in industrial applications such as those at China Mobile?

Table 1. Results on FB15K and WN18RR, comparing Accuracy, F1-Score, Precision, and Recall. We compared embedding-based methods, PLM-based methods, contrastive learning-based methods, and LLM-based methods.

Models	FB15K				WN18RR			
	Accuracy	F1-Score	Precision	Recall	Accuracy	F1-Score	Precision	Recall
Embedding-Based Methods								
TransE	0.6373	0.6312	0.6410	0.6531	0.3813	0.2927	0.6255	0.5083
DistMult	0.5938	0.5132	0.5261	0.5204	0.6401	0.5157	0.5965	0.5449
ComplEx	0.6268	0.4781	0.5413	0.5172	0.6414	0.4450	0.6464	0.5217
CAGED	0.6091	0.4574	0.5028	0.4552	0.6544	0.5064	0.5532	0.5013
KGTtm	0.6828	0.4078	0.6172	0.3045	0.6911	0.4487	0.6589	0.3402
PLM-based Methods								
KG-BERT	0.7675	0.6280	0.7371	0.5470	0.8162	0.7222	0.8177	0.6468
StAR	0.7350	0.6017	0.6900	0.5420	0.7012	0.6100	0.6572	0.5645
CSProm-KG	0.7078	0.5509	0.6139	0.4997	0.7116	0.6025	0.6138	0.4997
Contrastive Learning-based Methods								
SeSICL	0.5950	0.4600	0.5513	0.5172	0.5050	0.4073	0.4421	0.5711
CCA	0.7456	0.6810	0.7123	0.6537	0.7621	0.7134	0.7568	0.6912
LLM-based Methods								
Llama2	0.7420	0.6010	0.7250	0.6851	0.7100	0.6271	0.7021	0.6344
GPT-3.5	0.7445	0.6117	0.7185	0.6555	0.7603	0.7496	0.7120	0.6260
Llama3	0.7558	0.6264	0.7357	0.7148	0.7654	0.7522	0.7185	0.6327
Our Methods								
MAKGED	**0.7748**	**0.7367**	**0.7686**	**0.7252**	**0.8283**	**0.7909**	**0.8832**	**0.7704**

5.1 Experimental Settings

Datasets: We use two real-world knowledge graph datasets: FB15K [17] and WN18RR [6]. We chose these two datasets because they are highly representative in the field of knowledge graph error detection, encompassing most typical scenarios and possible graph structural representations found in knowledge graph data. In each dataset, we simulate realistic errors by replacing entities and relations with similar ones selected based on cosine similarity within the dataset, resulting in approximately 30% of the data being erroneous. We split each dataset into training, validation, and test sets with a ratio of 8:1:1. The fine-tuning process used only the training set, while the test set was reserved for final evaluations. FB15K is derived from Freebase and contains a rich set of entities and relations, while WN18RR is a subset of WordNet with corrected inverse relations, increasing the complexity. Additionally, we conducted experiments on a knowledge graph dataset from China Mobile's business scenarios, achieving the best results compared to other methods.

Baselines: We compare MAKGED against various baseline methods, including traditional knowledge graph embedding models such as TransE [2], DistMult [23], and ComplEx [18], which learn triple embeddings to compute confidence scores. Additionally, we compared recent embedding-based KG error detection

Table 2. The ablation study on FB15K and WN18RR evaluates Accuracy, F1-Score, Precision, and Recall. Results highlight the importance of bidirectional subgraph training and multi-agent discussions.

Models	FB15K				WN18RR			
	Accuracy	F1-Score	Precision	Recall	Accuracy	F1-Score	Precision	Recall
Only use a specific sub-graph for discussion								
MAKGED (Head_as_Head)	0.6920	0.6129	0.6496	0.6098	0.6940	0.5467	0.7826	0.5799
MAKGED (Head_as_Tail)	0.7220	0.6283	0.7115	0.6254	0.6920	0.5422	0.7802	0.5771
MAKGED (Tail_as_Head)	0.7100	0.6144	0.6875	0.6135	0.7000	0.5571	0.7999	0.5870
MAKGED (Tail_as_Tail)	0.7080	0.6147	0.6820	0.6133	0.6940	0.5498	0.7726	0.5811
Discuss without combining sub-graph information								
Llama2	0.7020	0.6432	0.6627	0.6373	0.6800	0.5368	0.7064	0.5689
Make direct judgments without discussion								
MAKGED (Analysis)	0.7300	0.6916	0.7410	0.6750	0.7100	0.5733	0.7967	0.5955
Use the complete framework structure								
MAKGED	**0.7748**	**0.7367**	**0.7686**	**0.7252**	**0.8283**	**0.7909**	**0.8832**	**0.7704**

methods including CAGED [25] and KGTtm [9]. We also compared KG-BERT [24] and CSProm-KG [4] models that combine pre-trained language models for error detection, as well as models that use text structure and graph structure for comparative learning, such as StAR [19], SeSICL [14], and CCA [13] models. All comparison experiments are conducted under the same experimental settings. **Implementation Details:** We use the Llama2 for fine-tuning and employ LoRA [8] for instruction tuning. Experiments are conducted on V100 GPU servers. **GCN Component:** A three-layer GCN with 128 hidden dimensions and 64-dimensional embeddings is trained separately using the Adam optimizer (learning rate: 0.001), a batch size of 64, for 100 epochs. The resulting GCN embeddings are concatenated with Llama2's text embeddings to create a unified representation for each agent. **Fine-Tuning Llama2:** During fine-tuning, the combined embeddings (GCN + Llama2) serve as model inputs. We utilize mixed precision training and gradient checkpointing to accelerate training and reduce memory usage. We use Accuracy, F1-Score, Precision, and Recall as evaluation metrics. These metrics use macro averaging for both classes. Each round of agent discussion took an average of 2.3 seconds. This demonstrates the framework's practical feasibility for industrial applications.

5.2 Effectiveness Analysis

Experiment Setup: To study RQ1, we conducted comprehensive experiments on two KG datasets, comparing it with the previously mentioned baseline models across four key metrics. The experimental results are presented in Table 1.

Comparison to Embedding-based Methods: MAKGED combines subgraph structural information with semantic insights from LLMs, leading to a 10-20% improvement in accuracy and a significant increase in the F1-Score.

			Knowledge Graph triple: (Huawei Honor 10, network support, 5G) Error: <5G->4G>		
Methods	Agent	initial responses	First round of discussion	Second round of discussion	Third round of discussion
MAKGED	Head as Head	incorrect	Huawei Honor 10 supports 4G network, released in 2018, operating system is Android 8.1, processor is Kirin 970, manufacturer is Huawei. According to this information, Huawei Honor 10 supports 4G network, and does not mention support for 5G network. Therefore, I think the triplet is incorrect.	I noticed that other agents mentioned that Huawei Honor 10 supports 5G, but I thought it only supports 4G. In addition, Huawei Honor 10 was released in 2018, when 5G was not yet commercially available. I also found some new sub-graph information: Huawei Honor 10's network standard is LTE (4G), with a maximum downlink rate of 600 Mbps. Therefore, I think the triplet is incorrect.	Thanks to all the participants for their clarification. Now we can be sure that this triple is incorrect.
	Head as Tail	correct	The Honor series includes the Huawei Honor 10, which is a high-end smartphone, a flagship model in 2018. Although there is no direct network support information, considering that the Huawei Honor 10 is a high-end smartphone and an important model in the Honor series, it may have 5G capabilities. So I think the triplet may be correct.	It sounds like your evidence is very convincing. In fact, 5G commercial use started in 2019, and Huawei Honor 10 is a 2018 model. Mobile phones released in 2018 include Huawei Honor 10, and mobile phones that support 5G do not include Huawei Honor 10. Based on this new information, I think the triplet may be incorrect.	Yes, based on the evidence provided by everyone, the triplet (Huawei Honor 10, network support, 5G) is incorrect.
	Tail as Head	correct	5G will be commercially available in 2019, and supported devices include Huawei Honor 10. It features high speed and low latency, and its application areas include the Internet of Things and autonomous driving. According to my data, the list of devices supported by 5G includes Huawei Honor 10. Therefore, I think the triplet is correct.	I understand your concerns, but the information I have obtained shows that 5G-supported devices include Huawei Honor 10. It is correct. Perhaps Huawei will provide 5G support for Honor 10 in subsequent system updates?	After further verification, I found that my initial data was indeed wrong. The Huawei Honor 10 does not support 5G, but was only used in testing. 5G test equipment is not equal to commercial equipment, and the release date of mobile phones that support 5G is after 2019. Therefore, So this triplet is incorrect.
	Tail as Tail	incorrect	Huawei Mate30 supports network 5G, Huawei P40 supports network 5G, and Xiaomi 10 supports network 5G. I did not find any relevant information about Huawei Honor 10 supporting 5G. Phones that support 5G are mainly models released after 2019. Therefore, I think the triplet is incorrect.	According to my data, 5G requires hardware support and cannot be achieved through software updates. Moreover, processors that support 5G include Kirin 990, and the processor of Huawei Honor 10 is Kirin 970. This shows that the processor of Huawei Honor 10 does not support 5G. It is incorrect.	Agreed. We had a very productive discussion. The triplet (Huawei Honor 10, network support, 5G) is incorrect.
Llama2	/	correct	I think the triplet is correct. Huawei Honor 10 is equipped with Kirin 970 processor, which has the ability to support 5G network. In addition, the release date of Huawei Honor 10 is 2018, and 5G technology began trial commercial use in the same year. 5G network services may have been provided in some areas. Therefore, Huawei Honor 10 may be able to connect to the 5G network through software updates or hardware support.		

Fig. 3. The figure shows an example of using our framework on the China Mobile business scenario knowledge graph. We compare the evaluation results of the original Llama2 model without subgraph fine-tuning, a method without multi-agent discussion, and our complete MAKGED framework.

Comparison to PLM-based Methods: MAKGED further demonstrates its strong error detection capabilities. Although models like KG-BERT show advantages in semantic understanding, their utilization of knowledge graph structure is limited. By integrating subgraph embeddings with LLMs embeddings, MAKGED improves recall by around 10% on the WN18RR compared to KG-BERT, demonstrating better coverage and precision in detecting complex errors.
Comparison to Contrastive Learning Methods: MAKGED shows stronger performance by leveraging multi-agent collaboration and integrating multiple perspectives. On the FB15K dataset, MAKGED achieved an F1-Score improvement of approximately 5-8%.
Comparison to LLM-based Methods: While LLM-based methods, such as Llama2 and GPT-3.5, perform well in knowledge graph tasks, their limited understanding of graph structures reduces their effectiveness in error detection. MAKGED overcomes this limitation by combining the semantic power of LLMs with the structural insights of GCN, allowing for more accurate error detection. Notably, its performance rivals that of GPT-3.5 and Llama3. Overall, MAKGED consistently surpasses state-of-the-art models in accuracy, F1-Score, and recall, demonstrating outstanding error detection capabilities.

5.3   Ablation Study

Experiment Setup: To address RQ2, we conducted an ablation study evaluating several model variants under the same experimental setup as the full framework. The **_Head_as_Head_** variant refers to the results where all four agents

are trained using the **_Head_ _Forward Subgraph_** constructed from the triple's head entity, with other ablation versions following a similar naming convention. The Analysis variant performs only initial judgments without multi-agent discussions, while the Llama2 baseline excludes subgraph information during fine-tuning, relying solely on Llama2's original outputs.

Full Framework Outperforms Variants: As shown in Table 2, when only specific subgraphs are used or subgraph information is excluded, performance drops notably compared to the complete MAKGED, especially in F1-Score and Recall, indicating that a single perspective or lack of structural information limits error detection. While using multiple agents without discussion offers slight improvements, it still lags behind the full framework. The complete framework achieves the best results on two datasets, maximizing four metrics.

5.4 Case Study

Running Example: To study RQ3, we select an incorrect triple *(Huawei Honor 10, network support, 5G)* from the industrial KG of China Mobile to demonstrate the effectiveness of our framework in industrial applications. Fig 3 shows the evaluation and discussion paths for this triple under our framework.

Subgraph-Aided Error Correction: The framework effectively uses subgraph information for in-depth analysis, enabling agents to correct initial errors and reach the correct conclusion. In contrast, using the original Llama2 model without subgraph fine-tuning leads to a lower accuracy performance.

Value of Multi-Agent Collaboration: If we had relied only on the initial model output, the result would have been "correct", conflicting with the ground truth. However, after three rounds of discussion, the agents reached the correct conclusion, demonstrating the effectiveness of multi-agent collaboration.

6 Conclusion

In this paper, we propose MAKGED, a novel framework for knowledge graph error detection. By combining subgraph embeddings from a GCN with LLM embeddings, we train four agents to evaluate triples through multi-agent discussions, enabling multi-perspective analysis. Experiments demonstrate that MAKGED significantly outperforms traditional and LLM-based methods, improving accuracy, F1-Score, precision, and recall across two datasets. Moreover, our framework has also shown excellent performance in industrial scenarios, validating the industrial application value of our method.

References

1. Bollacker, K.D., et al.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of SIGMOD. pp. 1247–1250. ACM (2008)
2. Bordes, A., Usunier, N., García-Durán, A., et al.: Translating embeddings for modeling multi-relational data. In: Proceedings of NeurIPS. pp. 2787–2795 (2013)

3. Chan, C., et al.: Chateval: Towards better llm-based evaluators through multi-agent debate. In: The Twelfth International Conference on Learning Representations, ICLR 2024 (2024)
4. Chen, C., et al.: Dipping PLMs sauce: Bridging structure and text for effective knowledge graph completion via conditional soft prompting. In: ACL (2023)
5. Chen, X., Jia, S., Xiang, Y.: A review: Knowledge reasoning over knowledge graph. Expert Syst. Appl. **141** (2020)
6. Dettmers, T., Minervini, P., Stenetorp, P., et al.: Convolutional 2d knowledge graph embeddings. In: Proceedings of AAAI. vol. 32 (2018)
7. Ge, C., Gao, Y., Weng, H., et al.: Kgclean: An embedding powered knowledge graph cleaning framework. arXiv preprint arXiv:2004.14478 (2020)
8. Hu, E.J., Shen, Y., Wallis, P., et al.: Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685 (2021)
9. Jia, S., Xiang, Y., Chen, X., et al.: Triple trustworthiness measurement for knowledge graph. In: Proceedings of WWW. pp. 2865–2871 (2019)
10. Li, G., et al.: Camel: Communicative agents for" mind" exploration of large language model society. NeurIPS **36**, 51991–52008 (2023)
11. Li, Y., et al.: Mateval: A multi-agent discussion framework for advancing open-ended text evaluation. In: International Conference on Database Systems for Advanced Applications. pp. 415–426. Springer (2024)
12. Lin, P., Song, Q., et al.: Discovering graph patterns for fact checking in knowledge graphs. In: International Conference on Database Systems for Advanced Applications. pp. 783–801. Springer (2018)
13. Liu, X., Liu, Y., Hu, W.: Knowledge graph error detection with contrastive confidence adaption. In: Proceedings of AAAI. pp. 8824–8831 (2024)
14. Liu, X., et al.: Sesicl: Semantic and structural integrated contrastive learning for knowledge graph error detection. IEEE Access **12**, 56088–56096 (2024)
15. Paulheim, H.: Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic Web **8**(3), 489–508 (2017)
16. Shi, B., Weninger, T.: Discriminative predicate path mining for fact checking in knowledge graphs. Knowledge-based systems **104**, 123–133 (2016)
17. Toutanova, K., Chen, D., Pantel, P., et al.: Representing text for joint embedding of text and knowledge bases. In: Proceedings of EMNLP. pp. 1499–1509 (2015)
18. Trouillon, T., Welbl, J., Riedel, S., et al.: Complex embeddings for simple link prediction. In: Proceedings of ICML. pp. 2071–2080 (2016)
19. Wang, B., et al.: Structure-augmented text representation learning for efficient kg completion. In: Proceedings of TheWebConf. pp. 1737–1748 (2021)
20. Wang, H., Zhao, M., Xie, X., et al.: Knowledge graph convolutional networks for recommender systems. In: Proceedings of WWW. pp. 3307–3313 (2019)
21. Xie, R., et al.: Does william shakespeare really write hamlet? knowledge representation learning with confidence. In: AAAI. vol. 32 (2018)
22. Xue, B., Zou, L.: Knowledge graph quality management: a comprehensive survey. IEEE Transactions on Knowledge and Data Engineering **35**(5), 4969–4988 (2022)
23. Yang, B., Yih, W., He, X., et al.: Embedding entities and relations for learning and inference in knowledge bases. In: ICLR (2015)
24. Yao, L., Mao, C., Luo, Y.: Kg-bert: Bert for knowledge graph completion. arXiv preprint arXiv:1909.03193 (2019)
25. Zhang, Q., Dong, J., Duan, K., et al.: Contrastive knowledge graph error detection. In: Proceedings of CIKM. pp. 2590–2599 (2022)