Can LLMs be Good Graph Judge for Knowledge Graph Construction?

Haoyu Huang[1], **Chong Chen**[2], **Zeang Sheng**[3], **Yang Li**[3], **Wentao Zhang**[3]
[1]The Chinese University of Hong Kong,[2]Huawei Cloud BU
[3]Peking University
haoyuhuang@link.cuhk.edu.hk,chenchong55@huawei.com
{shengzeang18,liyang.cs,wentao.zhang}@pku.edu.cn

Abstract

In real-world scenarios, most of the data obtained from the information retrieval (IR) system is unstructured. Converting natural language sentences into structured Knowledge Graphs (KGs) remains a critical challenge. We identified three limitations with respect to existing KG construction methods: (1) There could be a large amount of noise in real-world documents, which could result in extracting messy information. (2) Naive LLMs usually extract inaccurate knowledge from some domain-specific documents. (3) Hallucination phenomenon cannot be overlooked when directly using LLMs to construct KGs. In this paper, we propose **GraphJudge**, a KG construction framework to address the aforementioned challenges. In this framework, we designed an entity-centric strategy to eliminate the noise information in the documents. And we fine-tuned a LLM as a graph judge to finally enhance the quality of generated KGs. Experiments conducted on two general and one domain-specific text-graph pair datasets demonstrate state-of-the-art performance against various baseline methods with strong generalization abilities. Our code is available at https://github.com/hhy-huang/GraphJudge.

1 Introduction

The transition from non-structured text to structured Knowledge Graphs (KGs) is a pivotal step in the evolution of data management and information retrieval systems. The task of automatic KG construction aims to develop a structured representation of knowledge from various data sources without the need for manual intervention. KGs usually serve as the backbone of numerous data science applications, including GraphRAG systems (Edge et al., 2024) (Peng et al., 2024) and recommendation systems (Wang et al., 2019) (Jiang et al., 2024). Exploring a way to construct high-quality KGs from unstructured data is crucial for different downstream applications based on KG (Ge et al., 2021) (Huang et al., 2024) (Wei et al., 2024) (Rabbani et al., 2023).

Recently, Large Language Models (LLMs) have demonstrated significant generalization capabilities in various Natural Language Processing (NLP) tasks (Pan et al., 2024) and KG related tasks, such as text generation (Li et al., 2024), KG Completion (KGC) (Yao et al., 2023) and Open Information Extraction (OpenIE) (Angeli et al., 2015) (Dagdelen et al., 2024). Consequently, there are many works that utilize LLMs to construct KGs from unstructured natural language documents. The incorporation of LLMs can address the issue of generalization in open-domain applications (Carta et al., 2023a). With its robust zero-shot generation capability, there is no need for us to gather a large volume of annotated data for tasks such as named entity recognition (NER), entity extraction, or relation extraction.

Although recent LLM-based methods (Mo et al., 2025) (Han et al., 2023) (Lairgi et al., 2024) have gained some success in the KG construction task, we find that they may still face three challenges:

(1) **Noise Information**. Real-world documents are not only voluminous but also rife with noise, which poses a significant challenge for LLMs extracting valuable structured information. The sheer volume of data can lead to the extraction of excessive and irrelevant information, overshadowing the critical insights that LLMs are meant to uncover (Liu et al., 2024b) (Shi et al., 2023a). For example, as shown in Figure 1, the triple <*Protein X*, *is on*, *a 50% Discount*>is incorrectly constructed due to the irrelevant advertisement with red lines in the document, which is the noise information that makes the LLM incorrectly believed that the *Protein X* is on sale with discounts.

(2) **Domain-Specific Knowledge**. Naive LLMs often generate inaccurate triples with domain-specific documents, which require a deep un-

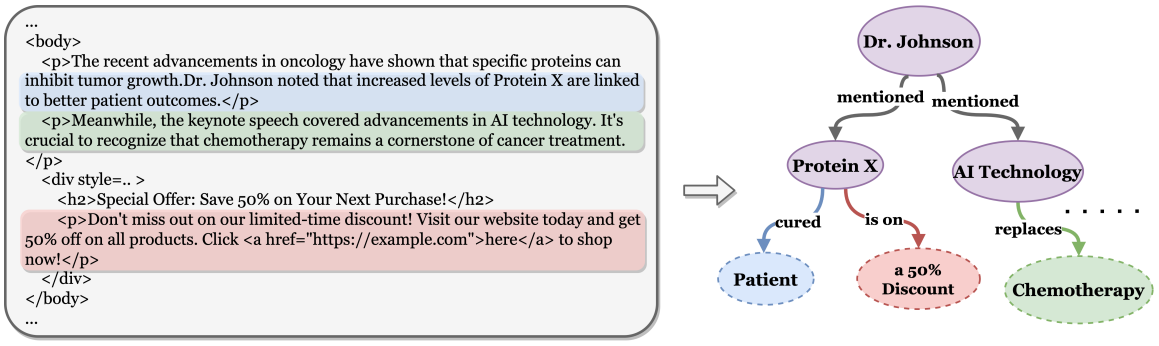


Figure 1: An demonstration of the challenges for constructing KGs with LLMs. The original document shown in the left part, while the constructed KG with some failure cases is displayed on the right side. The triple highlighted in red is wrongly formulated due to the presence of noisy information, the one in blue lacks domain knowledge, and the green-highlighted triple is a result of hallucinations by LLMs.

derstanding of specialized terminology and context (Zhong et al., 2023) (Zhu et al., 2024). And this kind of error is hard to be observed by naive LLMs. For example, in Figure 1, the triple <*Protein X*, *cured*, *Patient*>is inaccurately extracted due to a lack of medical domain-specific knowledge. While the document marks a reference with blue lines, note that the original text only suggests a link between *Protein X* and better patient results, not that it can cure patients in medical fields.

(3) **Hallucinations of LLMs**. When LLMs are directly used to build KGs, they are prone to generating false or distorted information, which is a phenomenon called hallucinations (Zhang et al., 2023) (Ji et al., 2023). This can lead to the incorporation of inaccurate or fabricated facts into the KG, undermining the reliability of the KG. For example, as shown in Figure 1, the triple marked in green <*AI Technology*, *replaces*, *Chemotherapy*>is incorrectly generated without any reference in the original document, even in the entity-related text highlighted with a green line.

To this end, we propose a new method called **GraphJudge**, which utilizes a fine-tuned open source LLM (e.g., LLaMA-2 (Touvron et al., 2023)) as an expert to judge the correctness of the triples generated by another closed-source LLM (e.g., GPT-4o-mini). To address the first challenge, we introduce the **Entity-Centric Text Denoising (ECTD)** module. We clean up the original documents by eliminating redundant words and irrelevant information not pertinent to the entities identified by the LLM. This module also leverages the robust zero-shot generation capabilities of LLMs to ensure the recall of a sufficient number of triple candidates (Wei et al., 2023) (Carta et al., 2023a). To overcome the second challenge, we suggest the module of **Knowledge Aware Supervised Fine-Tuning (KASFT)**. We introduce the graph judgement task from the triple classification task. To verify the accuracy of the triples generated by the closed-source LLM, we conduct supervised fine-tuning (SFT) on an open-source LLM, which can make it achieve over 90% accuracy on graph judgement tasks with strong generalization abilities. To settle the third challenge, the **Graph Judgement (GJ)** module is introduced. We utilize the fine-tuned open-source LLM to conduct judgement on the generated triples in the first module and filter out the wrong items to finally improve the quality of generated KGs.

In summary, the main contributions made in this work are as follows.

- Addressing challenges such as information noise, domain knowledge gaps and hallucinations in LLMs represents a critical step towards improving the quality of constructed KGs with real-world documents. To the best of our knowledge, we are the first to leverage both open- and closed-source LLMs to tackle these problems.
- We propose a new framework named **GraphJudge** to leverage their capability as a graph judge and enhance the performance of LLMs in KG construction tasks. We design an entity-centric strategy to eliminate the irrelevant and messy information in original documents. And we introduce graph judgment as the SFT task to enhance the quality of generated KGs.
- Experiments on two general and one domain-specific text-graph pair datasets demonstrate

that GraphJudge achieves state-of-the-art performance against various baseline methods with strong generalization abilities.

2 Related Work

In this section, we will introduce recent LLM-based OpenIE and KG construction methods. Some work(Agrawal et al., 2022) (Wei et al., 2023) has demonstrated that LLMs have remarkable zero-shot and few-shot information extraction abilities. However, they face difficulties when it comes to more intricate tasks such as relation extraction and event extraction (Carta et al., 2023a). To address that, Kumar et al. (Kumar et al., 2020) propose a unified approach to construct KGs from unprocessed text. They initially fine-tuned a pre-trained language model (PLM) for NER. Subsequently, they introduced a "2-model BERT" architecture to extract relations. GPT-RE (Wan et al., 2023) introduces the in-context learning method and task-aware representations in demonstration retrieval and aims to enhance the connections between examples and triples. PiVe (Han et al., 2023) designs a paradigm that fine-tuning a PLM as the verifier to predict the missing triples. With iterative verifications, the graph-based generative capability of LLMs can be improved. VicunaNER (Ji, 2023) utilizes the open-source LLM Vicuna to do zero-shot or few-shot NER. Similarly, it also performs recognition to identify entities that were not recognized in the previous phase. Carta et al. (Carta et al., 2023a) develops an iterative LLM prompting-based pipeline to generate KGs without requiring predefined sets or external ontologies. iText2KG (Lairgi et al., 2024) proposes a zero-shot method to construct consistent KGs from documents with LLMs. It restructures the unprocessed documents using a preset template and identifies distinct entities and connections in a semantic manner. SAC-KG (Chen et al., 2024) exploits LLMs as skilled automatic constructors for domain KGs and employs a naive LLM to predict the correctness of constructed triples. KGGen (Mo et al., 2025) clusters related entities to reduce sparsity in the KGs constructed by LLMs.

3 Preliminary and Definition

In this section, we first formulate the task of KG construction and introduce the definitions we may use throughout the paper. Then we detail the definition of the graph judgement task.

Definition 1: (Knowledge Graph Construction Task) We define the KG construction task as a problem of how to extract entities $\mathcal{E}$ and relations $\mathcal{R}$ from a document $\mathcal{D}$, which is also called the text-to-graph generation (T2G) task. The constructed KG, is defined as $\mathcal{G} = \{(h, r, t)|h, t \in \mathcal{E}, r \in \mathcal{R}\}$, where $\mathcal{E}$ is the set of entities and $\mathcal{R}$ is the set of relations in the graph $\mathcal{G}$. In other words, each KG $\mathcal{G}$ has a corresponding original text $\mathcal{D}$. Our goal is to get a better KG $\mathcal{G}$ from a document $\mathcal{D}$.

We also define a set of KGs $\mathcal{S}_\mathcal{G} = \{\mathcal{G}_1, \mathcal{G}_2, .., \mathcal{G}_N\}$ and a set of documents $\mathcal{S}_\mathcal{D} = \{\mathcal{D}_1, \mathcal{D}_2, .., \mathcal{D}_N\}$. In our implementation, we have a set of graph-text pairs $\mathcal{S}_\mathcal{P} = \{\mathcal{P}_1, \mathcal{P}_2, .., \mathcal{P}_N\}$, where $\mathcal{P}_i = \{(\mathcal{G}_i, \mathcal{D}_i)|\mathcal{G}_i \in \mathcal{S}_\mathcal{G}, \mathcal{D}_i \in \mathcal{S}_\mathcal{D}\}$. And $N = |\mathcal{S}_\mathcal{P}|$ is the number of graph-text pairs.

Definition 2: (Graph Judgement Task) We introduce the task of graph judgement to classify each triple in generated graphs is correct or not.

Here we define the KG we constructed from a corresponding document as $\hat{\mathcal{G}}$ and $\mathcal{S}_{\hat{\mathcal{G}}}$ representing the set of graphs we constructed. And $\hat{\mathcal{T}}$ in Equation (1) represents the triples on which we need to make judgements. Our goal in the graph judgement task is to predict the label of each triple in $\hat{\mathcal{T}}$, represented as $\hat{y} \in \{0, 1\}^{|\hat{\mathcal{T}}|}$.

$$\hat{\mathcal{T}} = \bigcup_{\hat{\mathcal{G}} \in \mathcal{S}_{\hat{\mathcal{G}}}} \{(h, r, t)|(h, r, t) \in \hat{\mathcal{G}}\}. \quad (1)$$

4 Methodology

4.1 Overview

As shown in Figure 2, the proposed model **GraphJudge** consists of three modules. In the first module, which is **Entity-Centric Text Denoising**, we extract entities and relations separately following results described in (Carta et al., 2023b). In the phrase of entity extraction, we generate entities with the denoised document. In the phrase of relation extraction, we generate relations with the entities and the denoised document as many as possible. Then, in the module of **Knowledge Aware Supervised Fine-Tuning**, we perform SFT to let the LLM become an expert in graph judgement by enhancing their abilities to check facts from documents with the triple structure and deepening their comprehension of domain-specific knowledge contained in the text-graph pairs. After that, in the final module we conduct the **Graph Judgement**. With the denoised documents as contexts, we employ the fine-tuned LLM as the graph judge to ascertain

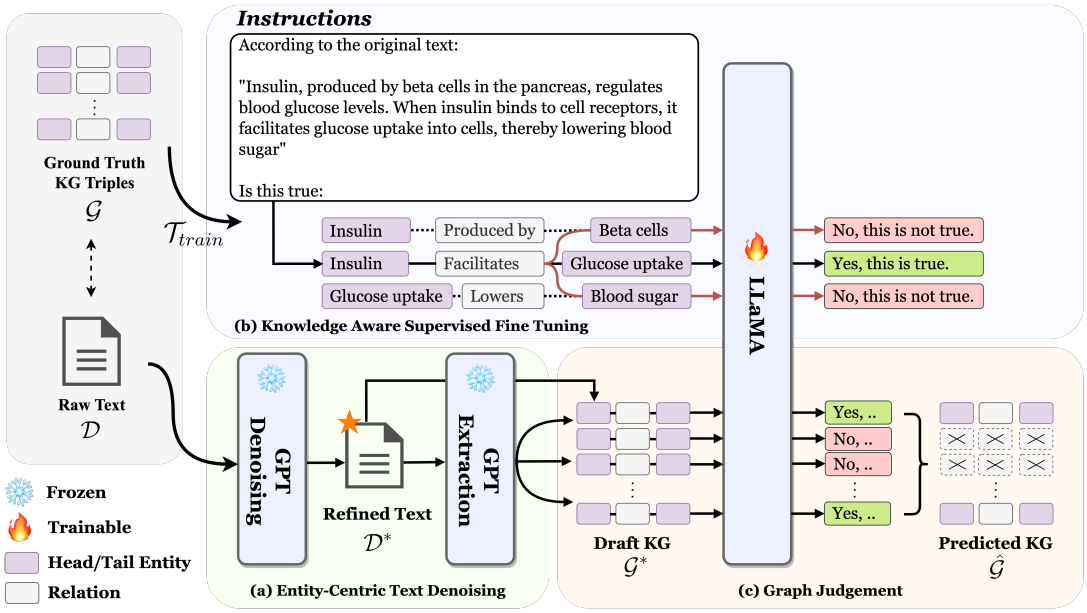


Figure 2: The overall architecture of our proposed GraphJudge framework for knowledge graph construction. It consists of three modules: (a) is the Entity-Centric Text Denoising module, (b) is the Knowledge Aware Supervised Fine Tuning module and (c) is the Graph Judgement module. The only component requiring training across the entire architecture is the open-source LLM utilized in the second module.

the accuracy of each triple within the graphs we generate. And then with the predicted results, we can filter out the triples that are judged as wrong. Finally, we can get high quality KGs.

4.2 Entity-Centric Text Denoising

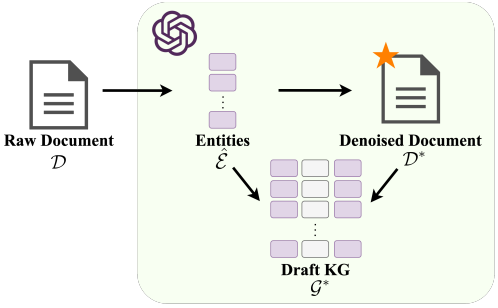


Figure 3: Illustrations of Entity-Centric Text Denoising.

In this module, a two-phrase extraction paradigm is designed to extract the entities and relations respectively. In phrase 1, we extract entities first and then denoise the original documents with extracted entities. In phrase 2, we conduct relation extraction and then we obtain the draft KGs. And Figure 3 is an overview of this module. In both of the two phrases we utilize a closed-source LLM to do the extraction and denoising.

4.2.1 Text denoising and entity extraction

In phrase 1, we consider that a substantial portion of real-world documents retrieved from information retrieval systems are consist of considerable noise information. And that may influence the quality of relations extracted by LLM (Shi et al., 2023b) (Liu et al., 2024c). So we design an iterative denoising method to remove messy information from the original text.

Specifically, we extract entities from the original document using LLM. Subsequently, we input these entities and the original document into LLM to generate the denoised document. In this way, we can achieve two goals: (1) The noise information that is not related to the topic of the document can be removed. (2) The content of the documents can be reorganized in an entity-centric way, which is friendly to the triple extraction in the next phrase. Finally, for each raw document $\mathcal{D}$ we will get the extracted entity set $\hat{\mathcal{E}}$ and the denoised document $\mathcal{D}^*$. Note that important information can be well preserved in $\mathcal{D}^*$ as verified in Appendix F.

4.2.2 Relation extraction

In phrase 2, we aim to extract relationships (triples) as many as feasible with the denoised document $\mathcal{D}^*$ and the entity set $\hat{\mathcal{E}}$ obtained in phrase 1 utilizing LLMs as shown in Equation (2). We create numerous relationships between entities to ensure

a sufficient number of suitable candidate triples for filtering with LLM judgment in the Graph Judgement module. Then we can construct a draft KG $\mathcal{G}^*$ for each original document $\mathcal{D}$, as illustrated in Equation (3), where $\mathcal{R}^*$ is the draft relation set we generate.

$$\mathcal{R}^* = LLM(\hat{\mathcal{E}}, \mathcal{D}^*), \tag{2}$$

$$\mathcal{G}^* = \{(h, r, t) | h, t \in \hat{\mathcal{E}}, r \in \mathcal{R}^*\}. \tag{3}$$

4.3 Knowledge Aware Supervised Fine-Tuning

In this module, inspired by KG-LLaMA (Yao et al., 2023), we propose the method of treating triples in the draft KG $\mathcal{G}^*$ as textual sequences and model graph judgement task as a sequence-to-sequence problem. We construct instruction data from the training set and fine-tune an open-source LLM to achieve the goal of both excelling at checking facts from documents with the triple structure and acknowledgment of domain-specific knowledge. The LLM can also learn how to verify the consistency between the document and the extracted triples. Checking facts with the triple structure refers to the general structure of triples is often analogous to a grammatical subject, predicate, and object or a subject with a relational attribute. LLMs are anticipated to have the ability to identify their correctness from the give documents. Domain-specific knowledge refers to the knowledge in the documents could be a new domain (Zhong et al., 2023), which is typically not part of pre-training data of LLMs. By employing SFT, the domain-specific knowledge from the documents can be incorporated into the LLM, thus enhances its graph judgment performance. And only if LLMs are fine-tuned as graph judges, these types of knowledge can be well learned, as justified in Figure 5 and Appendix G.

Before we conduct SFT on the LLM, we construct instructions for the graph judgement task with text-graph pair data. Because we need to ensure that the LLM not only excels at verifying correct triples but also skilled at telling the incorrect triples with the paired documents as contexts, we employ negative sampling to construct instruction data for training. In detail, we first sample the positive triple set $\mathcal{T}^+$ from the KGs of training set as described in Equation (4), where $\mathcal{S}_{\mathcal{G}_{train}}$ is the set of all KGs in the training set.

$$\mathcal{T}^+ = \bigcup_{\mathcal{G} \in \mathcal{S}_{\mathcal{G}_{train}}} \{(h, r, t^+) | (h, r, t^+) \in \mathcal{G}\}. \tag{4}$$

Similarly, we sample negative triple set $\mathcal{T}^-$ from the KGs in training set as described in Equation (5), where $\mathcal{E}$ represents the entity set of the graph $\mathcal{G}$. (h, r, t^-) is a negative triple of the graph $\mathcal{G}$, where t^- is a negative entity. We replace the positive tail entity t^+ in each positive triple with a randomly selected negative tail entity t^-. Note that if the selected negative entity is the same as or similar to the original one, we will skip that because they may not construct a triple reflecting a false fact.

$$\mathcal{T}^- = \bigcup_{\mathcal{G} \in \mathcal{S}_{\mathcal{G}_{train}}} \{(h, r, t^-) | (h, r, t^+) \in \mathcal{G}, t^- \in \mathcal{E} \backslash \{t^+\}\}. \tag{5}$$

Then we merge the positive triple set $\mathcal{T}^+$ and negative triple set $\mathcal{T}^-$ constructed from KGs $\mathcal{S}_{\mathcal{G}_{train}}$. Then we can obtain all the triples $\mathcal{T}_{train}$ we need to construct instructions.

$$\mathcal{T}_{train} = \mathcal{T}^+ \cup \mathcal{T}^-. \tag{6}$$

Furthermore, we transfer the triples in $\mathcal{T}_{train}$ to natural language sentences to construct the instruction data with paired documents $\mathcal{D}$ as contexts following the prompt templates shown in Appendix H. The triple sentences either represent a real fact or a fake fact. Then let the LLM make judgements with these instructions. Mathematically, with tokenized sentences $\mathcal{X}_{\mathcal{T}_{train}}$ transferred from triples $\mathcal{T}_{train}$ and paired documents $\mathcal{D}$, and tokenized instruction $\mathcal{X}_{\mathcal{I}}$, for a sequence of length L, we compute the probability of generating the target output $\mathcal{X}_{\mathcal{O}}$ as follows:

$$p(\mathcal{X}_{\mathcal{O}} | \mathcal{X}_t, \mathcal{X}_{\mathcal{I}}) = \prod_{i=1}^{L} p_\theta(x_i | \mathcal{X}_t, \mathcal{X}_{\mathcal{I},<i}, \mathcal{X}_{\mathcal{O},<i}), \tag{7}$$

where $\mathcal{X}_t \in \mathcal{X}_{\mathcal{T}_{train}}$. And θ are the learnable parameters within the open-source LLM to be fine-tuned.

4.4 Graph Judgement

The KGs created in the first module are preliminary and that is also why we call that draft KGs. In this module, we will verify the triples in these draft KGs using our fine-tuned LLM in the second module.

In detail, we let LLM do the graph judgement task on the draft KGs $\mathcal{G}^*$. Here we define draft KG set as $\mathcal{S}_{\mathcal{G}^*}$, and the triples in all draft KGs can be symbolized as

$$\mathcal{T}^* = \bigcup_{\mathcal{G}^* \in \mathcal{S}_{\mathcal{G}^*}} \{(h, r, t) | (h, r, t) \in \mathcal{G}^*\}. \tag{8}$$

Then, the LLM needs to assess the correctness of each triple in $\mathcal{T}^*$ by considering whether it aligns with the knowledge in paired documents and avoids conflicting with both domain-specific knowledge as it learned. We obtain the predictions of all the triples in $\mathcal{T}^*$ with the learned parameters θ as shown in Equation (9). And $Pred(\cdot)$ is a function that transforms the outputs of LLM into the binary results $\hat{y} \in \{0,1\}^{|\mathcal{T}^*|}$. Based on the judgments made by LLM, we filter the triples $\mathcal{T}^*$ in draft KGs to obtain high-quality triples $\hat{\mathcal{T}}$ as described in Equation (10), which form the final KGs we seek. $\hat{y}_{(h,r,t)}$ is the predicted result of a triple (h,r,t).

$$\hat{y} = Pred(p_\theta(\mathcal{X}_{\mathcal{T}^*})), \tag{9}$$

$$\hat{\mathcal{T}} = \{(h,r,t) \in \mathcal{T}^* | \hat{y}_{(h,r,t)} = 1\}. \tag{10}$$

Similarly, the refined relation set $\hat{\mathcal{R}} = \{r|(h,r,t) \in \mathcal{G}^*, \hat{y}_{(h,r,t)} = 1\}$ can also be obtained. Lastly, for each draft KG $\mathcal{G}^* \in \mathcal{S}_{\mathcal{G}^*}$ we can get the refined KG $\hat{\mathcal{G}}$ that we desire as shown in Equation (11). The implementation details of the graph judgment procedure are demonstrated in Appendix D.

$$\hat{\mathcal{G}} = \{(h,r,t)|h,t \in \hat{\mathcal{E}}, r \in \hat{\mathcal{R}}, (h,r,t) \in \hat{\mathcal{T}}\}. \tag{11}$$

5 Experiments

In this section, we will conduct experiments to address the following key research questions: **RQ1:** How well does GraphJudge perform on both general knowledge data and domain-specific knowledge data? **RQ2:** How do the different key components in our proposed method GraphJudge contribute to its overall performance? **RQ3:** How about the generalization capability of GraphJudge when applied across different datasets?

5.1 Experimental Settings

5.1.1 Dataset

In our study, we conduct experiments on two general datasets (**REBEL-Sub** (Huguet Cabot and Navigli, 2021) and **GenWiki** (Jin et al., 2020)) and one domain-specific dataset (**SCIERC** (Luan et al., 2018)) with golden ground truth KGs. We demonstrate the detailed information and statistics of each dataset in Appendix A. For each dataset we randomly select a sample of 2000 data points from the training data for validation purposes during the fine-tuning of the LLM.

5.1.2 Baselines

In our performance comparison, we consider two methods for comprehensive evaluation: **GPT-4o-mini**: We conduct experiments on GPT-4o with one-shot learning method. The instructions we have developed are identical to those outlined in our method. **GPT-4o** (Hurst et al., 2024): The same settings as GPT-4o-mini. **PiVe** (Han et al., 2023): We follow the default parameter settings of PiVe. We use the largest verifier module in PiVe, Flan-T5-XXL (Chung et al., 2024). We employ the LoRA adapter checkpoint[1], which has been well trained. And the LLM we use in this model is GPT-4o-mini. We implement an iterative prompting approach with three rounds, which represents the optimal number of iteration rounds as outlined in their study. **KGGen** (Mo et al., 2025): We also employ GPT-4o-mini as the LLM used in this baseline.

5.1.3 Implementation Details

Large Language Model: The LLMs we employed in this research are various in different modules. In the ECTD module, we utilize the closed-source LLM GPT-4o-mini to denoise the original documents and extract triples from documents. In the KASFT module, an open-source LLM LLaMA-2-7B (Touvron et al., 2023) is used as our base model.

Supervised Fine-Tuning: We employ LLaMA-2-7B as the base model to carry out SFT with LoRA (Hu et al., 2021). The instructions are constructed with the documents, query sentences, and the triple sentences. We perform SFT on autoregression generation tasks, which is a common approach to fine-tune LLMs (Black et al., 2022). The expected responses (labels) are either "Yes, that is true." or "No, that is not true.". Training settings are illustrated in Appendix C. The training was done using a single L20 GPU with 48GB of RAM.

5.1.4 Evaluation Metrics

We acknowledge that conventional evaluation techniques are rule-based. They assess the resemblance between predictions and ground-truth KGs through strict string matching, potentially overlooking semantic similarities. Therefore, to better evaluate the quality of the produced KGs against the ground-truth KGs, similar to PiVe (Han et al., 2023), we utilize one semantic level and two soft string matching evaluation metrics to calculate the **Accuracy**,

[1] https://huggingface.co/Jiuzhouh/flan-t5-xxl-lora-verifier

Dataset	Method	G-BS-Acc↑	G-BS-Recall↑	G-BS-F1↑	G-BL-Acc↑	G-BL-Recall↑	G-BL-F1↑	G-RO-Acc↑	G-RO-Recall↑	G-RO-F1↑
REBEL-Sub	*GPT-4o-mini*	0.3571	0.9024	0.4289	0.2343	0.6687	0.3018	0.2095	0.6266	0.2779
	GPT-4o	0.3131	0.9432	0.4163	0.2345	0.7284	0.3158	0.2201	0.6851	0.2966
	PiVe	0.3082	0.9378	0.4090	0.2217	0.7089	0.3010	0.2068	0.6693	0.2823
	KGGen	0.4190	0.8937	0.4995	0.2587	0.5794	0.3146	0.2233	0.5037	0.2719
	GraphJudge	0.4868	0.9144	**0.5796**	0.3391	0.6490	**0.4057**	0.3032	0.5878	**0.3571**
GenWiki	*GPT-4o-mini*	0.7825	0.9334	0.8368	0.6136	0.7353	0.6577	0.5451	0.6568	0.5857
	GPT-4o	0.7871	0.9393	0.8428	0.6318	0.7561	0.6774	0.5614	0.6742	0.6028
	PiVe	0.7463	0.9485	0.8230	0.5884	0.7516	0.6503	0.5251	0.6746	0.5817
	KGGen	0.8578	0.8230	0.8169	0.5799	0.4700	0.5542	0.4845	0.5598	0.4641
	GraphJudge	0.7936	0.9375	**0.8457**	0.6407	0.7591	**0.6836**	0.5714	0.6796	**0.6106**
SCIERC	*GPT-4o-mini*	0.5974	0.9183	0.6882	0.4368	0.6725	0.5040	0.3876	0.6065	0.4490
	GPT-4o	0.6272	0.9079	0.7035	0.4469	0.6530	0.5032	0.3914	0.5807	0.4425
	PiVe	0.5738	0.9225	0.6725	0.4192	0.6757	0.4924	0.3719	0.6092	0.4385
	KGGen	0.8394	0.6500	0.6635	0.6045	0.4594	0.4725	0.5426	0.4100	0.4211
	GraphJudge	0.6847	0.8775	**0.7283**	0.4898	0.6273	**0.5216**	0.4321	0.5591	**0.4603**

Table 1: Comparisons of GraphJudge with other baseline methods across three datasets. The cells marked with red color hold the worst performance in each column of Acc and Recall. The **best** and second-best results are also highlighted in each column of F1 scores.

Recall, and **F1** scores: **G-BERTScore (G-BS)**, **G-BLEU (G-BL)** and **G-ROUGE (G-RO)**. We elaborate on them in Appendix B.

5.2 Overall Performance Comparison (RQ1)

We demonstrate the evaluation results of our method GraphJudge with GPT-4o-mini and other baseline methods across three datasets in Table 1. We have the following insights:

GraphJudge's superior performance. GraphJudge outperforms other baselines in most of the cases. The superiority of GraphJudge's F1 scores (marked with gray color) demonstrates that, while maintaining a reasonable level of recall for triples, it also achieves improvement in accuracy. For example, as the results marked with red color show, although PiVe exhibits stronger recall ability, it overlooks triple accuracy. KGGen excels in accuracy but fails at recall. In contrast, GraphJudge leverages the ECTD module based on a closed-source LLM to ensure recall ability, while the KASFT and GJ modules with a fine-tuned open-source LLM guarantee accuracy, enabling its F1 score to surpass those of other baseline models. We can also observe that GraphJudge excels not only with domain-specific documents, but also demonstrates superior performance with general documents.

GraphJudge is cost-effective. Remarkably, GraphJudge achieves state-of-the-art performance by fine-tuning only a 7B LLM, which is significantly more efficient and cost-effective compared to the 70B LLM employed in PiVe. In addition, GraphJudge can even outperform GPT-4o with GPT-4o-mini, which is a small model with lower token cost. However, other baseline methods fail to achieve that.

5.3 Module Ablation Study (RQ2)

Dataset	Method	G-BS-F1↑	G-BL-F1↑	G-RO-F1↑
REBEL-Sub	*GraphJudge*	**0.5796**	**0.4057**	**0.3571**
	w/o ECTD	0.4548	0.3343	0.3094
	w/o GJ	0.4203	0.3052	0.2820
	w/o KASFT	0.4506	0.3219	0.2935
SCIERC	*GraphJudge*	**0.7283**	**0.5216**	**0.4603**
	w/o ECTD	0.6818	0.5029	0.4509
	w/o GJ	0.7172	0.5146	0.4552
	w/o KASFT	0.6700	0.4644	0.4084

Table 2: The results of ablation study on REBEL-Sub dataset and SCIERC dataset.

We perform an ablation study to explore the specific impacts of various modules within GraphJudge, and the results are reported in Table 2. The insights are outlined below:

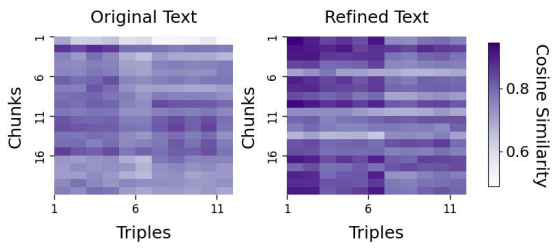


Figure 4: (a) The left map is the semantic similarity between the original document and paired KG triples. (b) The right map is the semantic similarity between the denoised document and paired KG triples.

Effect of Entity-Centric Text Denoising. We investigate the benefit of introducing entity-centric denoising paradigm using the variant "w/o ECTD", where we do not conduct document denoising and

directly extract entities and relations from original documents. The results show that our full model performs significantly better than this ablated version. It suggests that ECTD module can avoid LLMs extract wrong structured information from irrelevance or not well-formatted corpus.

Furthermore, to showcase the noise reduction capability of ECTD, we visualize the semantic correlation of the triples in a known KG with the denoised and original document, respectively. As shown in Figure 4, deeper color in the heat maps suggests a stronger relevance. The refined document exhibits greater relevance to the triples, demonstrating the effectiveness of ECTD. Implementation details are described in Appendix E.

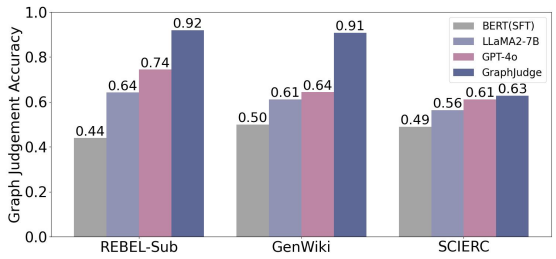


Figure 5: A comparison of the capabilities of bert-base-uncased (SFT) (Devlin et al., 2019), LLaMA-2-7B, GPT-4o, and our GraphJudge in graph judgment tasks.

Effect of Knowledge Aware Supervised Fine-Tuning. We conduct graph judgement on the triples without fine-tuning the open-source LLM, which is denoted as "w/o KASFT". The result in Table 2 indicates that without SFT, the naive LLM has weak graph judgement abilities. And with a fine-tuned LLM as a graph judge, the performance can be improved a lot. Because KASFT enables the LLM to acquire both fact-checking capabilities and domain-specific knowledge within the triples in our instruction training data.

Furthermore, we apply negative sampling to construct instructions on the **test set** like what we did on the training set. We randomly select 500 samples and perform graph judgement to compare the capabilities of different models. As shown in Figure 5 and Appendix G, both fine-tuned small models like BERT and naive powerful LLMs like GPT-4o show poor performance on the graph judgement task even with documents as contexts. However, **GraphJudge can achieve over 90% judgement accuracy on REBEL-Sub and GenWiki**, which demonstrates the KASFT module can indeed enhance the effectiveness of LLMs as a graph judge.

Effect of Graph Judgement. We compare the performances of our full model and the model without GJ module denoted as "w/o GJ". The result suggests that GJ module plays a very important role in GraphJudge. It can significantly enhance the quality of KGs generated by the closed-source LLM and reduce the effects of the inaccuracies or hallucinations that may arise from LLMs. The closed-source LLM excels in zero-shot generation, boosting recall but suffering accuracy due to hallucinations or knowledge inadequacy. The GJ module relieves this by filtering inaccurate triples, enhancing the quality of constructed KGs.

Method	GenWiki @ REBEL-Sub		
	G-BS-F1↑	**G-BL-F1↑**	**G-RO-F1↑**
GPT-4o	0.4163	0.3158	0.2966
PiVe	0.4090	0.3010	0.2823
KGGen	0.4995	0.3146	0.2719
GraphJudge	**0.5814**	**0.4055**	**0.3649**

Table 3: Results of generalization study on REBEL-Sub with the LLM fine-tuned on GenWiki.

5.4 Generalization Capabilities of GraphJudge (RQ3)

To demonstrate the generalization abilities of GraphJudge, we conduct experiments in cross-dataset scenarios. We train it on GenWiki and then evaluate it on REBEL-Sub. As shown in Table 3, our method can still outperform baseline methods, which indicates GraphJudge has great capabilities of generalization across various corpus. This is because the ability to check facts with triple structure learned from graph judgement tasks can be generalized. It also suggests that GraphJudge once well trained on a general dataset, can be readily applied to diverse datasets with common knowledge.

6 Conclusions

In this paper, we introduce a new method called GraphJudge for automatically constructing KGs, which leverages the potential of LLMs to act as graph judges. In GraphJudge, we propose ECTD, KASFT and GJ modules to mitigate the impact of irrelevant information from documents and exploit the benefits of trainable open-source LLMs and harnessing the strong zero-shot generation capabilities of closed-source LLMs. The experiments conducted on two general and one domain-specific datasets demonstrate GraphJudge's consistent superiority against various baseline methods.

7 Limitations

GraphJudge has the following limitations. First, even though we employ the LLMs act as both the extractor and the judge to improve the quality of constructed KGs, we still use the entity-level triples to construct KGs and there could be better knowledge units to form a better KG. Second, a more reasonable benchmark to evaluate the quality of constructed KGs should be proposed in the future. Currently, most of the work just utilize the "ground truth" KGs to calculate the correctness and comprehensiveness of constructed KGs, However, the quality of "ground truth" KGs may still deserve suspicion. So using a self-supervised approach to evaluate the KGs is in demands. We will research for more KG constructing and evaluating method to improve the performance of knowledge extraction.

References

Monica Agrawal, Stefan Hegselmann, Hunter Lang, Yoon Kim, and David Sontag. 2022. Large language models are few-shot clinical information extractors. *arXiv preprint arXiv:2205.12689*.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, Beijing, China. Association for Computational Linguistics.

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. SemEval 2017 task 10: ScienceIE - extracting keyphrases and relations from scientific publications. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 546–555, Vancouver, Canada. Association for Computational Linguistics.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. 2022. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*.

Salvatore Carta, Alessandro Giuliani, Leonardo Piano, Alessandro Sebastian Podda, Livio Pompianu, and Sandro Gabriele Tiddia. 2023a. Iterative zero-shot llm prompting for knowledge graph construction. *arXiv preprint arXiv:2307.01128*.

Salvatore M. Carta, Alessandro Giuliani, Lee Cecilia piano, Alessandro Sebastian Podda, Livio Pompianu, and Sandro Gabriele Tiddia. 2023b. Iterative zero-shot llm prompting for knowledge graph construction. *ArXiv*, abs/2307.01128.

Hanzhu Chen, Xu Shen, Qitan Lv, Jie Wang, Xiaoqi Ni, and Jieping Ye. 2024. Sac-kg: Exploiting large language models as skilled automatic constructors for domain knowledge graphs. *arXiv preprint arXiv:2410.02811*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

John Dagdelen, Alexander Dunn, Sanghoon Lee, Nicholas Walker, Andrew S Rosen, Gerbrand Ceder, Kristin A Persson, and Anubhav Jain. 2024. Structured information extraction from scientific text with large language models. *Nature Communications*, 15(1):1418.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Kata Gábor, Davide Buscaldi, Anne-Kathrin Schumann, Behrang QasemiZadeh, Haïfa Zargayouna, and Thierry Charnois. 2018. SemEval-2018 task 7: Semantic relation extraction and classification in scientific papers. In *Proceedings of the 12th International Workshop on Semantic Evaluation*, pages 679–688, New Orleans, Louisiana. Association for Computational Linguistics.

Congcong Ge, Xiaoze Liu, Lu Chen, Baihua Zheng, and Yunjun Gao. 2021. Largeea: Aligning entities for large-scale knowledge graphs. *arXiv preprint arXiv:2108.05211*.

Jiuzhou Han, Nigel Collier, Wray Buntine, and Ehsan Shareghi. 2023. Pive: Prompting with iterative verification improving graph-based generative capability of llms. *arXiv preprint arXiv:2305.12392*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Peng Huang, Meihui Zhang, Ziyue Zhong, Chengliang Chai, and Ju Fan. 2024. Representation learning for entity alignment in knowledge graph: A design space

exploration. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 3462–3475. IEEE.

Pere-Lluís Huguet Cabot and Roberto Navigli. 2021. REBEL: Relation extraction by end-to-end language generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2370–2381, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Bin Ji. 2023. Vicunaner: Zero/few-shot named entity recognition using vicuna. *arXiv preprint arXiv:2305.03253*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Yangqin Jiang, Yuhao Yang, Lianghao Xia, and Chao Huang. 2024. Diffkg: Knowledge graph diffusion model for recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 313–321.

Zhijing Jin, Qipeng Guo, Xipeng Qiu, and Zheng Zhang. 2020. GenWiki: A dataset of 1.3 million content-sharing text and graphs for unsupervised graph-to-text generation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2398–2409, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Abhijeet Kumar, Abhishek Pandey, Rohit Gadia, and Mridul Mishra. 2020. Building knowledge graph using pre-trained language model for learning entity-aware relationships. In *2020 IEEE International Conference on Computing, Power and Communication Technologies (GUCON)*, pages 310–315. IEEE.

Yassir Lairgi, Ludovic Moncla, Rémy Cazabet, Khalid Benabdeslem, and Pierre Cléau. 2024. itext2kg: Incremental knowledge graphs construction using large language models. *arXiv preprint arXiv:2409.03284*.

Junyi Li, Tianyi Tang, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2024. Pre-trained language models for text generation: A survey. *ACM Computing Surveys*, 56(9):1–39.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024a. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Jingyu Liu, Jiaen Lin, and Yong Liu. 2024b. How much can rag help the reasoning of llm? *arXiv preprint arXiv:2410.02338*.

Jingyu Liu, Jiaen Lin, and Yong Liu. 2024c. How much can rag help the reasoning of llm? *Preprint*, arXiv:2410.02338.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreferencefor scientific knowledge graph construction. In *Proc. Conf. Empirical Methods Natural Language Process. (EMNLP)*.

Belinda Mo, Kyssen Yu, Joshua Kazdan, Proud Mpala, Lisa Yu, Chris Cundy, Charilaos Kanatsoulis, and Sanmi Koyejo. 2025. Kggen: Extracting knowledge graphs from plain text with language models. *arXiv preprint arXiv:2502.09956*.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2024. Graph retrieval-augmented generation: A survey. *arXiv preprint arXiv:2408.08921*.

Kashif Rabbani, Matteo Lissandrini, and Katja Hose. 2023. Extraction of validating shapes from very large knowledge graphs. *Proceedings of the VLDB Endowment*, 16(5):1023–1032.

Swarnadeep Saha, Prateek Yadav, Lisa Bauer, and Mohit Bansal. 2021. Explagraphs: An explanation graph generation task for structured commonsense reasoning. *arXiv preprint arXiv:2104.07644*.

Christoph Schuhmann, Gollam Rabby, Ameya Prabhu, Tawsif Ahmed, Andreas Hochlehnert, Huu Nguyen, Nick Akinci Heidrich, Ludwig Schmidt, Robert Kaczmarczyk, Sören Auer, et al. 2025. Project alexandria: Towards freeing scientific knowledge from copyright burdens via llms. *arXiv preprint arXiv:2502.19413*.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. 2023a. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR.

Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. 2023b. Large language models can be easily distracted by irrelevant context. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 31210–31227. PMLR.

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

Zhen Wan, Fei Cheng, Zhuoyuan Mao, Qianying Liu, Haiyue Song, Jiwei Li, and Sadao Kurohashi. 2023. Gpt-re: In-context learning for relation extraction using large language models. *arXiv preprint arXiv:2305.02105*.

Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 950–958.

Xiang Wei, Xingyu Cui, Ning Cheng, Xiaobin Wang, Xin Zhang, Shen Huang, Pengjun Xie, Jinan Xu, Yufeng Chen, Meishan Zhang, et al. 2023. 'chatie: Zero-shot information extraction via chatting with chatgpt. *arXiv preprint arXiv:2302.10205*.

Yuyang Wei, Wei Chen, Xiaofang Zhang, Pengpeng Zhao, Jianfeng Qu, and Lei Zhao. 2024. Multi-modal siamese network for few-shot knowledge graph completion. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 719–732. IEEE.

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

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2023. Siren's song in the ai ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*.

Lingfeng Zhong, Jia Wu, Qian Li, Hao Peng, and Xindong Wu. 2023. A comprehensive survey on automatic knowledge graph construction. *ACM Computing Surveys*, 56(4):1–62.

Yuqi Zhu, Xiaohan Wang, Jing Chen, Shuofei Qiao, Yixin Ou, Yunzhi Yao, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2024. Llms for knowledge graph construction and reasoning: Recent capabilities and future opportunities. *World Wide Web*, 27(5):58.

Appendix

A Datasets

Dataset	REBEL-Sub	GenWiki	SCIERC
# of Train KGs	45,791	69,788	350
# of Test KGs	1,799	1,000	100
# of Train Triples	268,864	588,642	6,429
# of Test Triples	5,595	3,915	974

Table 4: Statistics of datasets.

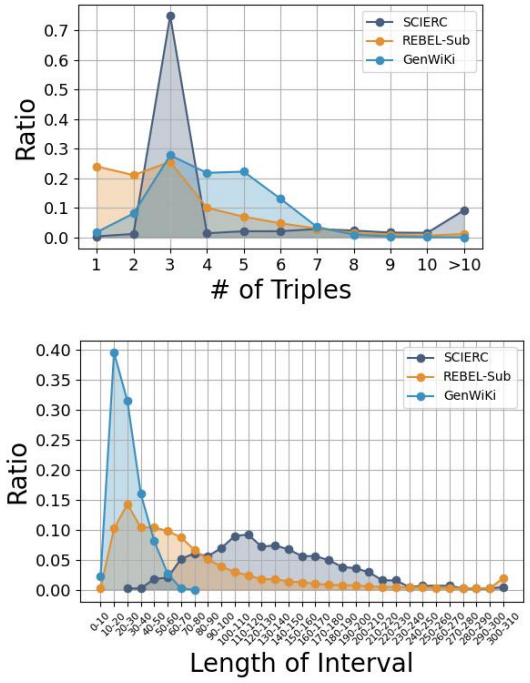


Figure 6: The figure above is the normalized distribution of the number of triplets in each dataset and the figure below is the normalized distribution of the length of documents in each dataset.

We conduct experiments on the following three datasets, And we also calculate the percentages of KGs with different numbers of triples and different lengths of original documents in each dataset in Figure 6. The statistics of each dataset are shown in Table 4.

REBEL-Sub. REBEL (Huguet Cabot and Navigli, 2021) dataset comes from Wikipedia text before the table of contents, as well as Wikidata for the triplets annotation. The dataset is collected by the extraction pipeline cRocoDiLe (Huguet Cabot and Navigli, 2021). The original REBEL dataset is a large-scale corpus. We utilize a subset of REBEL referred to as REBEL-sub, consisting of

50,000/2,000/2,000 samples for the training, validation, and test set respectively, randomly chosen from the original dataset. Moreover, we filter out the samples with empty ground truth KG.

GenWiki. GenWiki (Jin et al., 2020) is an extensive dataset sourced from general Wikipedia, comprising 1.3 million non-parallel texts and graphics that share content. In our study, for efficient validation, we use a subset of $GenWiki_{FINE}$ as our training set and employ another subset of $GenWiki_{FINE}$ for testing. And the documents in original testing data are too short for us to validate our method. To enhance the quality of training data lacking human annotations, we also exclude the triples with incorrect formats.

SCIERC. SCIERC (Luan et al., 2018) is a scientific domain-specific dataset comprises annotations for scientific entities, their relations, and coreference clusters within 500 scientific abstracts. It expands upon the datasets from SemEval 2017 Task 10 (Augenstein et al., 2017) and SemEval 2018 Task 7 (Gábor et al., 2018) by introducing additional entity types, relation types, broader relation coverage, and incorporating cross-sentence relations through coreference links. In addition, we filter out the samples with empty ground truth KG.

B Experimental Metrics

In this section, we explain the details of our evaluation metrics.

G-BERTScore (G-BS): Here we use a matching metric that evaluate the degree of similarity between the ground-truth and predicted graphs, which is called G-BERTScore(Saha et al., 2021). And it is designed as an extension of the text generation metric BERTScore(Zhang et al., 2019). In G-BERTScore, each triple within knowledge graphs is treated as a sentence, and subsequently, the similarity score between sentences of triples in the ground-truth and predicted knowledge graphs is computed. And we compute the accuracy, recall, and F1 score of each constructed KG against the ground-truth using G-BERTScore, denoted as **G-BS-Acc**, **G-BS-Recall**, and **G-BS-F1**, respectively.

G-BLEU (G-BL): BLEU (Bilingual Evaluation Understudy)(Papineni et al., 2002) is a metric for evaluating the quality of text which has been machine-translated from one natural language to another. Here we use this approach to determine the resemblance between the triple sentences in the ground-truth and predicted KGs, which is called G-BLEU. The formulas are shown in (16), (15), (12), (13), (14). N is the maximum order of n-grams considered in the evaluation and we set $N = 4$, which is a default number in the Python package[2]. BP is the brevity penalty, which is used to avoid giving too much credit to short translations. And we compute the accuracy, recall, and F1 score of each constructed KG against the ground-truth using G-BLEU, denoted as **G-BL-Acc**. **G-BL-Recall**, and **G-BL-F1**, respectively.

$$Match(n) = \sum_{\mathcal{X}_t \in \mathcal{X}_\mathcal{T}} \sum_{gram(n) \in \mathcal{X}_t} Count(gram(n), \mathcal{X}_{\hat{t}}) \quad (12)$$

$$Total(n) = \sum_{\mathcal{X}_t \in \mathcal{X}_\mathcal{T}} \sum_{gram(n) \in \mathcal{X}_{\hat{t}}} Count(gram(n), \mathcal{X}_{\hat{t}}) \quad (13)$$

$$BP = \begin{cases} 1 & \text{if } |\mathcal{X}_{\hat{t}}| > |\mathcal{X}_t| \\ e^{(1-\frac{|\mathcal{X}_t|}{|\mathcal{X}_{\hat{t}}|})} & \text{if } |\mathcal{X}_{\hat{t}}| \leq |\mathcal{X}_t| \end{cases} \quad (14)$$

$$w_n = \frac{Match(n)}{Total(n)} \quad (15)$$

$$\text{G-BLEU} = BP \times (\prod_{n=1}^{N} w_n)^{\frac{1}{N}} \quad (16)$$

G-ROUGE (G-RO): ROUGE (Recall-Oriented Understudy for Gisting Evaluation)(Lin, 2004) is a set of metrics for evaluating automatic summarization and machine translation systems. And here we utilize ROUGE to compare the similarities between the triple sentences in the ground-truth and predicted KGs, which is G-ROUGE. Here our G-ROUGE score is based on the notion of n-gram co-occurrence statistics and we set $n = 2$. For G-ROUGE-N, which focuses on the overlap of n-grams between the ground-truth triple sentence and the predicted triple sentence, the formulas are shown in (17), (18), (19). Unlike G-BLEU, G-ROUGE is computed using recall as a metric. And $Count(gram(n), \mathcal{X}_{\hat{t}})$ is the number of times the n-gram appears in the predicted triple sentence $\mathcal{X}_{\hat{t}}$. And we compute the accuracy, recall, and F1 score of each constructed KG against the ground-truth using G-ROUGE, denoted as **G-RO-Acc**, **G-RO-Recall**, and **G-RO-F1**, respectively.

$$Match(n) = \sum_{\mathcal{X}_t \in \mathcal{X}_\mathcal{T}} \sum_{gram(n) \in \mathcal{X}_t} Count(gram(n), \mathcal{X}_{\hat{t}}) \quad (17)$$

[2]https://pypi.org/project/bert-score/

$$Total(n) = \sum_{\mathcal{X}_t \in \mathcal{X}_\mathcal{T}} \sum_{gram(n) \in \mathcal{X}_t} Count(gram(n), \mathcal{X}_t) \quad (18)$$

$$\text{G-ROUGE} = \frac{Match(n)}{Total(n)} \quad (19)$$

C Experimental Settings

Hyper-parameter	Experimental Setting
Micro Batch Size	8
Batch Size	128
Gradient Accumulation Steps	16
Training Steps	500
Learning Rate	$3e^{-4}$
Lora Attention Dimension	8
Alpha Parameter	16
Target Modules	q_proj, v_proj
Warmup Steps	100
Optimizer	AdamW

Table 5: Implementation detail of SFT in GraphJudge.

During the Knowledge Aware Supervised Fine-Tuning module, we follow the parameter settings in Table 5 referring to the tuning process used for triple classification tasks in the KG-LLaMA (Yao et al., 2023).

D Graph Judgement Algorithm

The detailed procedure of the graph judgement algorithm is demonstrated in Algorithm 1.

Dataset	Context	Model	
		LLaMA-3-8B	LLaMA-3-70B
REBEL-Sub	[Lower-Upper]	47.33-99.33	76.67-100.0
	$\mathcal{D}^*$	94.67	96.00
	$\hat{\mathcal{G}}$	85.33	90.67
GenWiki-Hard	[Lower-Upper]	53.33-100.0	68.00-100.0
	$\mathcal{D}^*$	98.00	96.00
	$\hat{\mathcal{G}}$	93.00	93.00
SCIERC	[Lower-Upper]	65.00-99.67	77.00-99.67
	$\mathcal{D}^*$	95.67	96.33
	$\hat{\mathcal{G}}$	90.33	93.67

Table 6: MCQ performance across datasets. Each row displays the lower-upper bound performance (no context vs. original document), denoised document performance, and our KG performance for different models. Using $\mathcal{D}^*$ and $\hat{\mathcal{G}}$ preserves most information for answering MCQs, perform close to the using the original document (upper bound) across datasets and models.

E Effect of ECTD Module

To validate that ECTD module can lead to a cleaner refined text, we sample a text-graph pair from the REBEL-Sub dataset. Then we split the original and refined document into the same number of chunks, which we set 20 here. And we use a PLM BERT (bert-base-uncased[3]) (Devlin et al., 2019) to process these chunks and get the embedding of each chunk. And we calculate the cosine similarities between these document chunks and triple sentences, as shown in Figure 4. Deeper color in the heat maps suggests a stronger relevance between the specific triple and document chunk.

Algorithm 1 The Graph Judgement procedure of GraphJudge

Input: The fine-tuned expert LLM p_θ ; Candidate triples $\mathcal{T}^*$ in the draft KGs $\mathcal{S}_{\mathcal{G}^*}$; Paired refined text $\mathcal{D}^*$ of the candidate triples;

Output: The predicted KGs $\mathcal{S}_{\hat{\mathcal{G}}}$ with refined triples $\hat{\mathcal{T}}$;

$\mathcal{S}_{\hat{\mathcal{G}}} \leftarrow \{\}$;
$\hat{\mathcal{T}} \leftarrow \{\}$;
for *each* $\mathcal{G}^*$ *in* $\mathcal{S}_{\mathcal{G}^*}$, *each denoised document* d^* *in* $\mathcal{D}^*$ **do**
  $\hat{\mathcal{R}} \leftarrow \{\}$;
  $\hat{\mathcal{E}} \leftarrow \{\}$;
  for *each triple* $t^* =< h, r, t > \in \mathcal{G}^*$ **do**
    */*Transform the triple and refined text into a sentence*/*
    $\mathcal{X}_{t^*} = Sentence(< h, r, t >, d^*)$;
    */*Verify the correctness of the current triple with fine-tuned LLM*/*
    $\hat{y}_{t^*} = Pred(p_\theta(\mathcal{X}_{t^*}))$;
    if $\hat{y}_{t^*}$ *is not 'False'* **then**
      $\hat{\mathcal{T}} \leftarrow t^*$;
      $\hat{\mathcal{R}} \leftarrow \hat{\mathcal{R}} \cup \{r\}$;
      $\hat{\mathcal{E}} \leftarrow \hat{\mathcal{E}} \cup \{h, t\}$;
    end if
  end for
  $\hat{\mathcal{G}} = \{< h, r, t > | h, t \in \hat{\mathcal{E}}, r \in \hat{\mathcal{R}}, < h, r, t > \in \hat{\mathcal{T}}\}$;
  $\mathcal{S}_{\hat{\mathcal{G}}} \leftarrow \mathcal{S}_{\hat{\mathcal{G}}} \cup \{\hat{\mathcal{G}}\}$;
end for

F Knowledge Retention of ECTD Module and KG

While the ECTD module has the ability to remove irelevant information contained in the original documents, it is also necessary to verify that the important knowledge is well preserved in the documents denoised by ECTD. We test how well **multiple-**

[3]https://huggingface.co/google-bert/bert-base-uncased

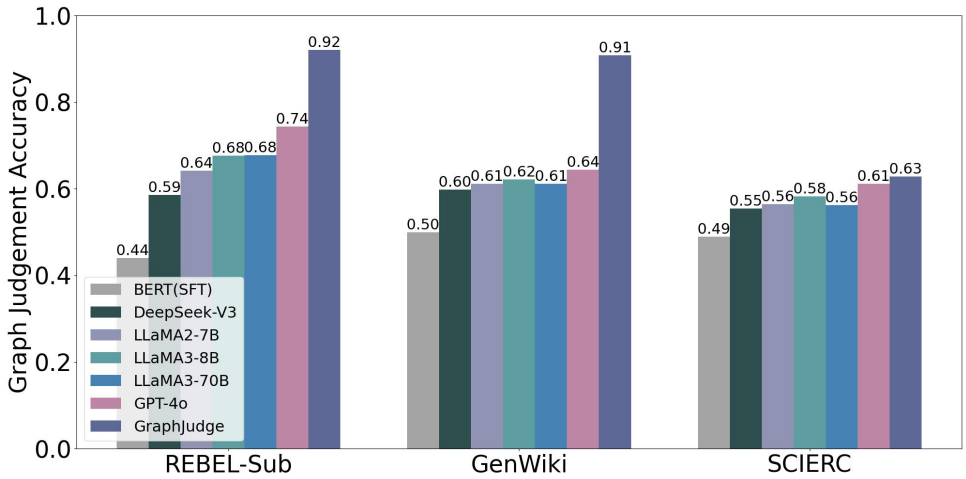


Figure 7: A comparison of the capabilities of bert-base-uncased (SFT) (Devlin et al., 2019), DeepSeek-V3 (Liu et al., 2024a), LLaMA-2-7B, LLaMA-3-8B, LLaMA-3-70B, GPT-4o, and our GraphJudge in graph judgment tasks.

choice question (MCQ) performance is preserved after we refined the original documents.

In detail, similar to the existing work (Schuhmann et al., 2025), we generate various MCQs with LLaMA-3-70B for each original document. For REBEL-Sub, we randomly sample 500 documents and generate 3 MCQs for each document. For SCIERC, because the test set of that is very small, we used the full test set of SCIERC with 3 MCQs for each document. For GenWiki, because the average lengths of the documents are very short, we generate only 1 MCQ for each document. Then we ask LLaMA-3-8B to answer them with no context (denoted as lower bound), then ask them again with the original passage (denoted as upper bound) for sanity check. Finally, we conduct tests using denoised documents (denoted as $\mathcal{D}^*$) and KG triples constructed by GraphJudge (denoted as $\hat{\mathcal{G}}$). The results in Table 6 demonstrates that MCQs performance with $\mathcal{D}^*$ or $\hat{\mathcal{G}}$ remains far above the lower bound baseline and approaches the original-document upper bound. It proofs that **important information is well preserved in both our denoised documents and constructed KG.**

G Effect of KASFT Module

In this section, we extend the baseline models to explore their abilities to be a graph judge with the same experimental settings in Section 5.3. We extend baseline models to fine-tuned BERT, DeepSeek-V3, LLaMA-2-7B, LLaMA-3-8B, LLaMA-3-70B, and GPT-4o. Compared with them, our proposed GraphJudge demonstrates consistent superiority in graph judgement tasks, which further proofs that the KASFT module can improve the capabilities of open-source LLMs as a graph judge. And neither fine-tuning a PLM with a smaller parameter size nor directly employing a powerful closed-source LLM can achieve a high accuracy on graph judgement tasks, which suggests the necessity to introduce our proposed GraphJudge.

Prompt-Graph Judgement

Goal:
You need to do the graph judgement task, which means you need to clarify the correctness of the given triple with the give original text.

Here is the question:
According to the original text: **{Text}**
Is this ture: {head_entity} {relation} {tail_entity} ?

No, it is not true. / Yes, it is true.

Figure 8: The prompt template for the open-source LLM LLaMA to construct graph judgement instructions.

H Prompt Templates

As shown in Figure 8 and Figure 9, we demonstrate the prompt templates for the closed-source LLM to conduct relation extraction and for the open-source LLM to perform graph judgements on the results generated from the closed-source LLM. We also provide the prompt templates used to generate and answer MCQs in Appendix F.

I Case Study

In this section, we present an instance of constructing a KG from a document, achieved through the integration of a naive LLM (GPT-4o-mini) and our GraphJudge. We select a text-graph pair from the SCIERC dataset and contrast the results yielded by our approach with that of GPT-4o-mini. As shown

Prompt-Relation Extraction

Goal:
Transform the text into a semantic graph(a list of triples) with the given text and entities. Extract subject-predicate-object triples from the assistant message. A predicate (1-3 words) defines the relationship between the subject and object. Relationship may be fact or sentiment based on assistant's message. Subject and object are entities. Entities provided are from the assistant message and prior conversation history, though you may not need all of them. This is for an extraction task, please be thorough, accurate, and faithful to the reference text.

Note:
1.Generate triples as many as possible.
2.Make sure each item in the list is a triple with strictly three items.

Here are two examples:
Example#1:
Text: \"Shotgate Thickets is a nature reserve in the United Kingdom operated by the Essex Wildlife Trust.\"
Entity List: [\"Shotgate Thickets\", ... , \"Essex Wildlife Trust\"]
Semantic Graph: [[\"Shotgate Thickets\", \"instance of\", \"Nature reserve\"], ...]
Example#2:
Text: ..
Semantic Graph: ...

Refer to the examples and here is the question:
Text: **{Text}**
Entity List:**{entities}**
Semantic graph:

[**{triple_0}**, **{triple_1}**, ..., **{triple_i}**]

Figure 9: The prompt template for the closed-source LLM GPT-4o-mini to construct KGs.

in Table 7, the KG constructed by GPT-4o-mini with the given original document includes lots of meaningless triples. For example, <*We*, *suggest*, *goal*>, <*We*, *suggest*, *evaluation criterion*>, <*We*, *present*, *measure*>, <*We*, *present*, *selection function*>, etc. It is obvious that these triples do not convey any beneficial information that could be applied to subsequent tasks. And the triple <*evaluation criterion*, *new*, *goal*>does not even follow the general structure of triples, which means that the adjective word "new" is generally not employed as a relational term within triples. The naive LLM have strong zero-shot ability to generate them but it does not have the capability to determine whether they are useful. However, there are no such triples in the KG constructed by our GraphJudge. On the one hand, this is because the triples without any useful information will be clarified as wrong triples by our fine-tuned LLM in graph judgement module. On the other hand, as demonstrated in the case, the document refined by ECTD module exhibits enhanced standardization and a reduction in irrelevant terms, for instance, terms such as "-LRB-" and "-RRB-" have been excluded as they are irrelevant to the document's subject matter.

Prompt-MCQ Generation

You are an expert in generating multiple-choice questions (MCQs) from scientific texts.
Your task is to generate **{i}** multiple-choice questions based on the following passage.

Each question should:
- Focus on factual claims, numerical data, definitions, or relational knowledge from the passage.
- Have 4 options (one correct answer and three plausible distractors).
- Clearly indicate the correct answer.

The output should be in JSON format, with each question as a dictionary containing:
- "question": The MCQ question.
- "options": A list of 4 options (e.g., ["A: ..", "B: ..", "C: ..", "D: .."]).
- "answer": The correct answer (e.g., "A").

Output Example:
```
[
 {
  "question": "What is the primary role of a catalyst in a chemical reaction?",
  "options": [
   "A: To make a thermodynamically unfavorable reaction proceed",
   "B: To provide a lower energy pathway between reactants and products",
   "C: To decrease the rate of a chemical reaction",
   "D: To change the overall reaction itself"
  ],
  "answer": "B"
 },
 ...
]
```

Passage:
{passage}

Output:

[**{MCQ_0}**, **{MCQ_1}**, ..., **{MCQ_i}**]

Figure 10: The prompt template for the MCQ generations.

Prompt-MCQ Answering

Given the contexts or evidences:
{contexts}

Here is a multiple-choice question:

Question: **{question}**

Options:
A. {options_0}
B. {options_1}
C. {options_2}
D. {options_3}

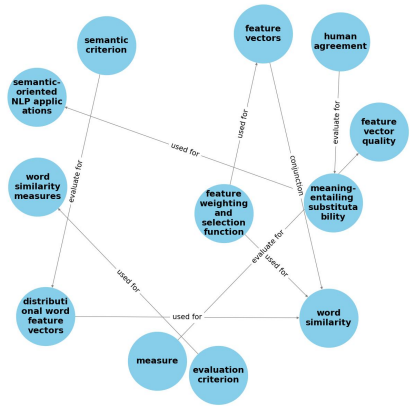
Please select the correct answer by choosing A, B, C, or D. Respond with only the letter of your choice.

A/B/C/D

Figure 11: The prompt template for the MCQ answering.

Original Document: We suggest a new goal and evaluation criterion for word similarity measures .The new criterion – meaning entailing substitutability – fits the needs of semantic-oriented NLP applications and can be evaluated directly -LRB-independent of an application -RRB- at a good level of human agreement. Motivated by this semantic criterion we analyze the empirical quality of distributional word feature vectors and its impact on word similarity results, proposing an objective measure for evaluating feature vector quality. Finally, a novel feature weighting and selection function is presented , which yields superior feature vectors and better word similarity performance.

Ground-Truth Knowledge Graph:



[[”feature weighting and selection function”, ”used for”, ”word similarity”], [”measure”, ”evaluate for”, ”feature vector quality”], [”feature vectors”, ”conjunction”, ”word similarity”], [”evaluation criterion”, ”used for”, ”word similarity measures”], [”meaning-entailing substitutability”, ”used for”, ”semantic-oriented NLP applications”], [”human agreement”, ”evaluate for”, ”meaning-entailing substitutability”], [”semantic criterion”, ”evaluate for”, ”distributional word feature vectors”], [”distributional word feature vectors”, ”used for”, ”word similarity”], [”feature weighting and selection function”, ”used for”, ”feature vectors”]]

GPT-4o-mini:

[[”We”, ”suggest”, ”goal”], [”We”, ”suggest”, ”evaluation criterion”], [”We”, ”propose”, ”measure”], [”evaluation criterion”, ”new”, ”goal”], [”evaluation criterion”, ”fits”, ”needs”], [”evaluation criterion”, ”evaluated directly”, ”good level of human agreement”], [”We”, ”analyze”, ”quality”], [”quality”, ”impact on”, ”word similarity results”], [”measure”, ”evaluating”, ”feature vector quality”], [”feature vector quality”, ”yield”, ”superior feature vectors”], [”feature vector quality”, ”yield”, ”better word similarity performance”], [”We”, ”present”, ”feature weighting function”], [”feature weighting function”, ”yields”, ”superior feature vectors”], [”We”, ”present”, ”selection function”], [”selection function”, ”yields”, ”superior feature vectors”]]

GraphJudge:

[[’evaluation criterion’, ’related to’, ’word similarity measures’], [’evaluation criterion’, ’assessed by’, ’human agreement’], [’evaluation criterion’, ’related to’, ’semantic-oriented NLP applications’], [’evaluation criterion’, ’related to’, ’meaning-entailing substitutability’], [’word similarity measures’, ’quality assessed by’, ’distributional word feature vectors’], [’distributional word feature vectors’, ’impact on’, ’word similarity results’], [’distributional word feature vectors’, ’quality evaluated by’, ’objective measure’], [’feature vectors’, ’generated by’, ’feature weighting and selection function’], [’feature vectors’, ’enhances’, ’word similarity performance’]]

Denoised Document: We propose a new evaluation criterion for word similarity measures, called meaning-entailing substitutability, which is suitable for semantic-oriented NLP applications. This criterion can be assessed independently of any specific application with a high level of human agreement. We examine the quality of distributional word feature vectors and their impact on word similarity results, introducing an objective measure for evaluating the quality of these feature vectors. Additionally, we present a novel feature weighting and selection function that generates superior feature vectors and enhances word similarity performance.

Table 7: Comparison of Construction Results between our GraphJudge and GPT-4o-mini.