Refining Noisy Knowledge Graph with Large Language Models

Dong Na[1] **Natthawut Kertkeidkachorn**[1] **Xin Liu**[2] **Kiyoaki Shirai**[1]

[1]Japan Advanced Institute of Science and Technology, Ishikawa, Japan
[2]National Institute of Advanced Industrial Science and Technology, Tokyo, Japan

{s2320036, natt, kshirai}@jaist.ac.jp, xin.liu@aist.go.jp

Abstract

Knowledge graphs (KGs) represent structured real-world information composed by triplets of head entity, relation, and tail entity. These graphs can be constructed automatically from text or manually curated. However, regardless of the construction method, KGs often suffer from misinformation, incompleteness, and noise, which hinder their reliability and utility. This study addresses the challenge of noisy KGs, where incorrect or misaligned entities and relations degrade graph quality. Leveraging recent advancements in large language models (LLMs) with strong capabilities across diverse tasks, we explore their potential to detect and refine noise in KGs. Specifically, we propose a novel method, *LLM_sim*, to enhance the detection and refinement of noisy triples. Our results confirm the effectiveness of this approach in elevating KG quality in noisy environments. Additionally, we apply our proposed method to Knowledge Graph Completion (KGC), a downstream KG task that aims to predict missing links and improve graph completeness. Traditional KGC methods assume that KGs are noise-free, which is unrealistic in practical scenarios. Our experiments analyze the impact of varying noise levels on KGC performance, revealing that LLMs can mitigate noise by identifying and refining incorrect entries, thus enhancing KG quality.

1 Introduction

Knowledge Graphs (KGs) provide a structured framework for representing interconnected data, widely used in research fields such as natural language processing and recommendation systems. However, automated KG construction often introduces noise, leading to inaccurate or misaligned triples that degrade the quality and reliability of downstream tasks like Knowledge Graph Completion (KGC) (Xie et al., 2018). Addressing noise in KGs is crucial for maintaining KGC performance, as this task relies on accurate triples to infer missing links and enhance KG completeness.

Large Language Models (LLMs), which demonstrate impressive capabilities across a variety of tasks like question answering (Lála et al., 2023), summarization (Jin et al., 2024), and translation (Huang et al., 2023), offer a promising solution for KG noise detection. By encoding extensive factual and contextual knowledge, LLMs can evaluate the coherence of entity-relationship pairs based on learned semantic patterns (Petroni et al., 2019). Leveraging LMs for noise detection presents a potential advancement over traditional noise detection methods, which typically depend on KG embedding models or rule-based techniques that may not effectively handle nuanced or context-specific noise.

In this study, we propose a novel approach, *LLM_sim*, which uses a LLM, Llama3[1], to detect and refine erroneous triples in noisy KGs. Our *LLM_sim*, generates candidate triples for detected noise and refines them using contextual similarity, matching them to existing KG triples. Our experiments show that *LLM_sim* is particularly effective under high noise conditions, underscoring the value of LLMs for KG refinement.

The contributions of this paper are as follows:

- We introduce *LLM_sim*, which leverages LLMs to detect and refine noise in KGs, improving downstream KG task performance.
- We validate our approach through experiments on WN18RR[2] and FB15k-237[3].
- We systematically evaluate the impact of various noise levels on KGC, showing our

[1]https://huggingface.co/meta-llama/Meta-Llama-3-8B
[2]https://huggingface.co/datasets/VLyb/WN18RR
[3]https://huggingface.co/datasets/VLyb/FB15k-237

Proceedings of the Generative AI and Knowledge Graph Workshop (GenAIK), pages 78–86
January 19, 2025.

method's robustness under different noise conditions.

The remainder of this paper is organized as follows. Section 2 reviews related work on noise detection and KGC methods. Section 3 explains our proposed method *LLM_sim* for detecting and refining noisy KGs. Section 4 describes the experimental setup, including datasets and model configurations. Section 5 presents experimental results, highlighting the effectiveness of our approach. Finally, Section 6 concludes the paper and outlines future research directions.

2 Related Work

In noise detection research, various approaches have been proposed to demonstrate the effectiveness of their models in detecting noise. These methods can be broadly categorized into three types: traditional KG embedding models, noise detection based on pre-trained language models(PLMs), and unsupervised rule-based noise detection models.

Knowledge Graph Embedding (KGE) models assess the validity of triples by estimating confidence scores for embedded representations, based on the principle that correct triples approximate the vector equation $h + r \approx t$, where h and t represent the head and tail entities, and r represents the relationship. Notable models in this category include TransE (Bordes et al., 2013), RotatE (Sun et al., 2019), DistMult (Yang et al., 2014), and ComplEx (Trouillon et al., 2016). However, noise in training data can degrade embedding quality, as KG embeddings rely on clean data for optimal performance. While KGE models can determine a triple's validity, their performance remains limited due to their sensitivity to noisy data.

PLMs, such as *GPT-2 XL*[4], approach noise detection by assessing the semantic relationship between text and entities within a triple. They evaluate correctness by measuring the model's confidence or probability score for a given triple. However, these models often struggle with domain-specific or temporal knowledge, as they depend on the contexts present in their training data. Additionally, these models primarily rely on associative reasoning rather than causal inference, making it challenging to detect implicit noise in complex knowledge reasoning scenarios. Consequently, such models often have limited capacity for inferring non-explicit relationships and may not effectively detect noise in superficially similar triples.

Unsupervised rule-based noise detection models use predefined rules or constraints to detect anomalies or noise in triples. For example, (Hong et al., 2021) introduced a rule-based triple confidence framework for noise detection in KGE, assigning confidence scores to improve noise filtering and enhance the robustness and accuracy of embedding models. Probabilistic models have also been applied to noisy data (Yi and Wu, 2019; Garg et al., 2021), using statistical methods to quantify uncertainty and model noise, facilitating robust error correction and data refinement, thereby enhancing the quality, usability, and reliability of KGs. However, these models often struggle to intuitively grasp the semantic information of triples and lack a solid foundation of real facts and logical coherence. Although they are not dependent on labeled data, rule-based approaches lose efficacy in dynamic KGs or frequently updated datasets, as fixed rules may become outdated, leading to inefficiency and reduced scalability.

3 Methodology

To effectively detect noise in KGs, which refers to erroneous triples, we propose a novel framework to detect and refine noisy triples in large-scale KGs using LLMs. Our approach consists of two key components: noise detection and noise refinement. Figure 1 illustrates the overall process of noise detection and refinement in KGs. The LLM detection model first identifies noise within the *KG*, which is detected as noise data, *filtered KG* in the figure, which is then passed to the LLM refinement for correction. The *refined KG* along with detected correct triplets, *filtered KG*, forms a new dataset, *renewed KG*, which is subsequently utilized for KGC tasks. The refinement process begins with the LLM generating five candidate triples for a given noisy triple. The candidates are divided into head-relation and relation-tail pairs, which are then matched against the noisy KG. The most suitable candidate is selected based on similarity calculations and used as the final refinement triple.

3.1 Noise Detection

Noise detection plays a crucial role in our proposed method. In this paper, we propose a novel approach that leverages the generative capabilities of LLMs to detect noisy triples. We call this pro-

[4] https://huggingface.co/openai-community/gpt2-xl

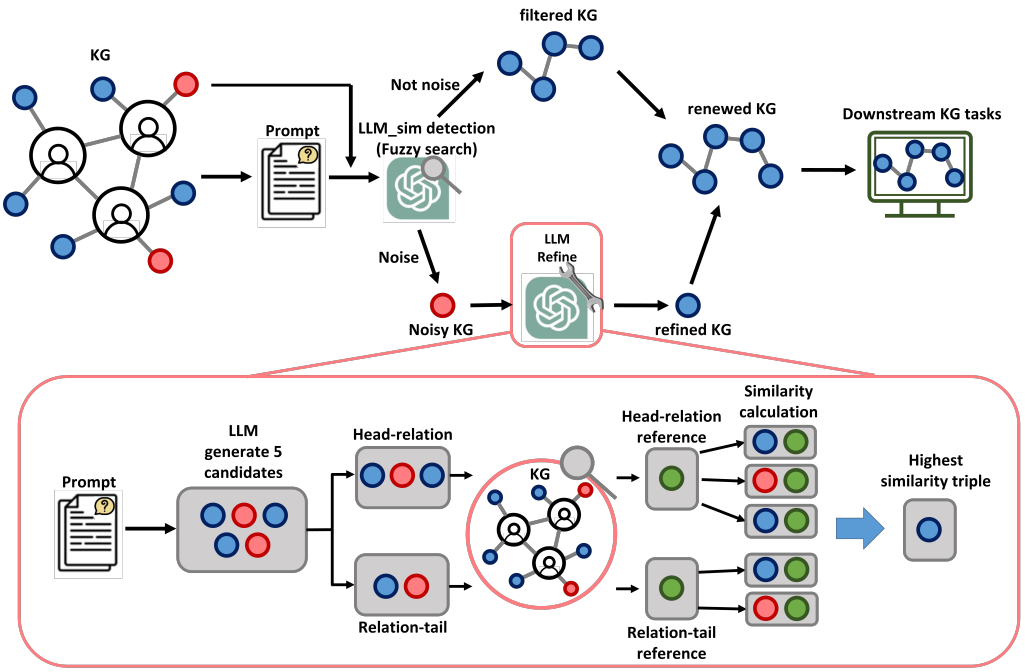

Figure 1: Overview of *LLM_sim*. The red circles represent noisy triples, the blue circles indicate correct triples, and the green circles represent stochastic triples, which may be either correct or noisy.

cess *LLM_sim* detection. To enhance the LLM's evaluation with relevant contextual information, we employ a fuzzy search (Fu et al., 2016) to search similar triplets within the original KG, which is provided as *ADDITIONAL_CONTEXT*. This context includes triples that are structurally and semantically similar to the target triple $\langle E1, R, E2\rangle$, aiding the LLM in evaluating factual accuracy. Our method involves the following five steps.

Query Vector Construction: We encode the target triple as a query vector $\mathbf{q}$ that captures its semantic information.

$$\mathbf{q} = f(realization(E1, R, E2)), \quad (1)$$

where $f(\cdot)$ is a function based on the Sentence Transformer model *all-MiniLM-L6-v2*[5], mapping each entity or relation to its corresponding embedding. The function $realization()$ is used to create a simple sentence from a triple into a statement like "$E1\ R\ E2$".

Fuzzy Search in KG: We use the query vector $\mathbf{q}$ to perform a fuzzy search over the KG and identify triples with high semantic similarity, calculated using cosine similarity:

$$\text{cosine}(\mathbf{q}, \mathbf{t}) = \frac{\mathbf{q} \cdot \mathbf{t}}{||\mathbf{q}|| \cdot ||\mathbf{t}||}, \quad (2)$$

where $\mathbf{t}$ represents embedding vectors of other triples in the KG, as computed by Equation 1.

Selection of Similar Triples: To avoid confusion from multiple contexts, we select the single triple $\langle E1', R', E2'\rangle$ most similar to $\mathbf{q}$ based on similarity scores computed by Equation 2. This selected triple serves as *ADDITIONAL CONTEXT* to support the LLM in assessing the validity of the target triple.

Prompt Design: A structured prompt leverages the *ADDITIONAL CONTEXT* to assist the LLM in accurately evaluating the relationship between entities. The prompt is crafted to ensure the LLM can process and reason through the query effectively. The prompt format is as follows:

> Based on all your knowledge and **the given context** ⟨***ADDITIONAL_CONTEXT***⟩**.** Determine if the ⟨*E1*⟩ has a ⟨*R*⟩ with the ⟨*E2*⟩. Answer the question **by reasoning step-by-step**, and provide your final answer within 'yes' or 'no'.
> **Answer in this format:**
> Final Answer: [yes/no]

[5] https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

Filtering of Erroneous Triplets : By systematically applying the prompt and interpreting the LLMs' responses, we effectively filter out false triples, thereby enhancing the overall quality and reliability of the data.

3.2 Noise Refinement

The bottom half of Figure 1 illustrates the workflow of refining noisy triples in KGs using LLMs. Starting with a set of noisy triples, despite providing context, directly correcting erroneous triples remains challenging for LLMs. In order to balance model efficiency and prediction accuracy, the LLM generates five candidate triples for each noisy instance. These candidate triples are grouped based on two distinct pairings: Head-relation and relation-Tail. The *KG* is then utilized to compute similarity scores between the candidate triples and the reference triples in the KG. Finally, the candidate triple with the highest similarity score is selected as the optimal refinement for the noisy triple, ensuring improved data quality in the KG. This approach achieves higher accuracy compared to directly using the LLM to correct noisy triples (as demonstrated in our experiments). In summary, this methodology comprises three key steps: candidate generation, grouping strategy, and similarity calculation.

3.2.1 Candidate Generation

To address the noisy triples, we designed a prompt that allows the LLM to automatically refine the mismatches. We designed the prompt as follows:

> The entities in the given triple **do not correctly correspond to each other**. Based on your knowledge, please rectify the triple and generate five correct triples, ensuring that **the original ⟨R⟩ remains unchanged**. Each refined triple should **include either ⟨*E1*⟩ or ⟨*E2*⟩ from the given triple**.
> Please output the refined triples in the following format:
> 1. (entity, relation, entity)
> 2. (entity, relation, entity)
> 3. (entity, relation, entity)
> 4. (entity, relation, entity)
> 5. (entity, relation, entity)

In this prompt, we first need to clearly indicate that the given triples are not correct. In generating refined triples, the task requires the model to retain the original relation while modifying one of the entities. This constraint significantly reduces the likelihood of the LLM producing fabricated or irrelevant information. By grounding the refinement process in the given structure—fixing either the head or tail entity and preserving the original relation, we aim to enhance the factual accuracy of the output and maintain consistency with the KGs.

Each triple consists of a head entity (E_1), a relation (r), and a tail entity (E_2). In cases where the entities and relations within a triple are misaligned, it is often unclear whether the mismatch originates from the E_1 or the E_2. To tackle this uncertainty, our prompt instructs the model to fix one entity and the relation while predicting the other entity. This approach mitigates the risk of LLM hallucination and improves both the prediction accuracy and the efficiency of the model by systematically narrowing down the sources of error within the triples.

3.2.2 Grouping Strategy

To address the uncertainty about whether noise originates from the head or tail entity in a triple, we employ a grouping strategy that enhances the accuracy of LLM predictions for both entities. Specifically, we organize generated candidate triples into two grouping criteria: (1) the "Head-relation" pair (E_1, r), where triples that share the same head entity and relation and (2) the "relation-Tail" pair (r, E_2), where those that share the same relation and tail entity. This grouping process helps to mitigate noise introduced by entity permutations, ensuring that semantically similar triples are compared effectively.

After grouping, we search for similar combinations in the reference dataset, which is the original *KG* in this study. Instead of searching for the complete triple (E_1, r, E_2), we search separately for either the (E_1, r) or the (r, E_2) in the reference dataset.

Without grouping, directly matching entire triples (E_1, r, E_2) may lead to incorrect alignments with unrelated KG entries. Grouping enables a focused comparison on each entity's role, ensuring accuracy and reducing noise.

3.2.3 Similarity Calculation

Both the original noisy triple and its generated candidates are embedded into a dense vector space by converting them into textual representations. For

any given triple (E_1, r, E_2), we construct its text representation by concatenating the head entity, relation, and tail entity into a single sequence:

$$T = \text{“}E_1\ r\ E_2\text{”}. \tag{3}$$

To represent these triples in a vector space, we employ the Sentence Transformer model, specifically the `all-MiniLM-L6-v2` variant. The model encodes each triple's textual representation into a dense vector:

$$\mathbf{v}_T = \text{SentenceTransformer}(T). \tag{4}$$

Thus, for the sequence T of the given triple, the corresponding embedding vector $\mathbf{v}_T$ is obtained by encoding its textual representation.

After generating embeddings for candidate and reference triples, we measure their similarity using cosine similarity. For a pair of triples (T_1, T_2), the cosine similarity between their embeddings $\mathbf{v}_{T_1}$ and $\mathbf{v}_{T_2}$ is calculated as:

$$\text{cos\_sim}(\mathbf{v}_{T_1}, \mathbf{v}_{T_2}) = \frac{\mathbf{v}_{T_1} \cdot \mathbf{v}_{T_2}}{\|\mathbf{v}_{T_1}\| \|\mathbf{v}_{T_2}\|}. \tag{5}$$

For each group, we select the candidate triple with the highest cosine similarity to any reference triple, a triple in *KG*. A higher cosine similarity indicates that the textual representations of the two triples are more semantically consistent, suggesting that the entity relationships they express are more closely aligned. Let G_j represent a candidate triple and F_i represent a reference triple from the dataset. We aim to find the candidate triple G_j^* that maximizes the cosine similarity to any reference triple within the group:

$$G_j^* = \arg\max_{G_j \in \text{gen}} \left(\max_{F_i \in \text{ref}} \text{cos\_sim}(\mathbf{v}_j, \mathbf{v}_i) \right), \tag{6}$$

- gen represents the set for all generated candidate triples.
- ref represents the set for all reference triples in the dataset.

Despite the presence of noise in the dataset, the majority of triples remain accurate. Leveraging this fact, our method addresses noise effectively by calculating cosine similarity between generated triples and reference triples. This similarity-based approach allows us to isolate noise from correct data with high accuracy, thereby enhancing the dataset's reliability.

This three-step refinement process aims to effectively enhance the quality of KGs by correcting noisy triples.

4 Experiment

This section describes our experimental setup, including the dataset description, noise construction methods, and evaluation metrics.

4.1 Dataset

We conduct experiments on two datasets, WN18RR and FB15k-237, derived from WordNet (Miller, 1995) and Freebase (Bollacker et al., 2008), respectively, that are widely used for KGC benchmarks. Both datasets contain structured triples representing relationships between entities, making them suitable for evaluating KGC. Table 1 summarizes key statistics for each dataset.

Dataset	WN18RR	FB15k-237
Entities	40,943	14,541
Relationships	11	237
Train Triples	86,835	272,115
Validation Triples	3,034	17,535
Test Triples	3,134	20,466

Table 1: WN18RR and FB15k-237 Datasets

4.2 Noisy KG Dataset Construction

To simulate real-world conditions where KGs are often noisy, we introduce controlled levels of noise into WN18RR and FB15k-237 by injecting erroneous triples. Specifically, we vary the noise ratio at 10%, 20%, and 30%, based on reported noise levels in real-world datasets (Hasan and Chu, 2022; Song et al., 2022). Noise is introduced by replacing one of the entities in a triple as:

$$G' = (h', r, t)\ or\ (h, r, t'), \tag{7}$$

where h' and t' represent randomly chosen entities that do not relate to t or h in the context of the original relation r.

4.3 Baseline Methods

We evaluate our proposed method, *LLM_sim*, against several baselines categorized as follows:

Pre-trained Language Models: GPT-2 XL, which detects noise based on general language understanding. A prompt used for noise detection:

> Is ⟨Entity1⟩ a ⟨relationship⟩ of ⟨Entity2⟩? Answer the question within yes and no:

Note that this prompt differs from the one used for Llama3, as described in subsection 3.1. Initially, we used identical prompts for both GPT-2 XL and Llama3. When Llama3 received prompts designed for GPT-2 XL, the lack of specificity led to irrelevant responses. Similarly, GPT-2 XL struggled with prompts tailored for Llama3, resulting in inadequate answers. Consequently, we developed distinct prompts for each model to better suit their capabilities and improve output quality.

KGE Models: We include TransE, RotatE, and ExpressivE (Pavlović and Sallinger, 2022) as baseline models, where the validity of triples is assessed using embeddings from these models. For TransE, for instance, noise detection is performed by verifying if the norm of $||(\mathbf{h} + \mathbf{r} - \mathbf{t})||$ is less than a threshold γ. After testing various values from 0 to 1, we chose $\gamma = 0.1$ for TransE and RotatE, and $\gamma = 0.2$ for ExpressivE for each score function, respectively.

Rule-based Methods: For WN18RR, we identify noisy triples by assessing the consistency of the part-of-speech tags for the head and tail entities in each relation. In FB15k-237, however, the larger variety of relations makes it impractical to design rules for each one. Instead, we first group triples by relation, apply Named Entity Recognition (NER) within each group, and filter out entities whose types deviate from the dominant entity types, identifying them as noise.

Noise Detection Methods: For robust noise detection, we adopt CAGED (Zhang et al., 2022) as our baseline, a state-of-the-art approach renowned for its effectiveness in identifying and filtering noisy relations. CAGED leverages advanced entity and relation embedding techniques to detect inconsistencies within KGs, providing high precision in distinguishing authentic triples from erroneous data.

4.4 Evaluation Metric

4.4.1 Noise Detection

The performance of noise detection is measured using accuracy, precision, recall, and F1 score.

4.4.2 Noise Refinement

We evaluated the refinement process using two primary methods due to the lack of gold-standard labels, which made direct evaluation challenging. First, we assessed whether the generated triples were present in the original noise-free dataset, referring to this metric as "correctness." This approach offers an initial indication of refinement accuracy by checking alignment with verified data. Second, we performed a manual evaluation, where 100 randomly selected samples were inspected to qualitatively assess refinement accuracy.

4.4.3 KGC task

We use KGC metrics to evaluate the impact of noise detection and noise refinement on the downstream task, selecting the ExpressivE model to perform KGC on the datasets. Evaluation metrics include Mean Reciprocal Rank (MRR) and Hit@k (Hit@1, Hit@3, and Hit@10), measuring model effectiveness in predicting missing links. Specifically, MRR represents the average of the reciprocal ranks of the correct entities, highlighting how close the predictions are to the top rank. Hit@k calculates the proportion of correct entities ranked within the top k predictions, reflecting the model's ability to rank true triples highly.

We compare the following KGC models using different KGs to systematically compare different noise detection and refinement methods:

KGC: Original KG with injected noise at varying ratios, without any filtering or refinement.

KGC + CAGED: KG filtered using the CAGED model, which removes noisy triples based on domain-specific criteria.

KGC + GPT2: KG filtered using GPT-2 XL, which removes detected noisy triples.

KGC + *LLM_sim:* without context KG filtered using Llama3. Noisy triples are not refined but just removed. The additional context is not used for noise detection.

KGC + *LLM_sim:* detection Similar to KGC + LLM_sim detection without context, but it includes context-based filtering, with neighboring triples providing additional information for noise detection.

KGC + *LLM_sim*: Final refined KG consisting of both filtered and refined triples.

Noise Level	Model	WN18RR				FB15k-237			
		Accuracy	Precision	Recall	F1 score	Accuracy	Precision	Recall	F1 score
10% Noise	TransE	0.473	0.896	0.468	0.615	0.411	0.884	0.398	0.549
	RotatE	0.511	0.900	0.509	0.650	0.464	0.895	0.459	0.606
	ExpressivE	0.603	0.902	0.585	0.709	0.527	0.890	0.521	0.660
	GPT-2 XL	0.823	0.562	0.468	0.511	0.787	0.899	0.860	0.879
	Rule-base	0.672	0.900	0.715	0.797	0.681	**1.000**	0.645	0.784
	CAGED	0.853	0.963	0.856	0.906	**0.839**	0.850	**0.857**	**0.900**
	LLM_sim without context	0.868	**0.980**	0.871	0.922	0.806	0.969	0.810	0.883
	LLM_sim detection	**0.911**	0.934	**0.969**	**0.951**	0.806	0.962	0.816	0.883
20% Noise	TransE	0.335	0.708	0.305	0.426	0.353	0.724	0.325	0.449
	RotatE	0.423	0.774	0.407	0.533	0.438	0.782	0.425	0.551
	ExpressivE	0.593	0.810	0.558	0.661	0.501	0.811	0.501	0.619
	GPT-2 XL	0.786	0.811	0.960	0.829	0.775	0.809	**0.945**	**0.872**
	Rule-base	0.634	0.811	0.715	0.760	0.719	**1.000**	0.653	0.790
	CAGED	0.804	0.933	0.814	0.869	0.712	0.694	0.927	0.794
	LLM_sim without context	0.869	**0.960**	0.875	0.915	**0.798**	0.935	0.807	0.866
	LLM_sim detection	**0.883**	0.894	**0.971**	**0.930**	0.784	0.918	0.806	0.858
30% Noise	TransE	0.310	0.553	0.278	0.370	0.322	0.568	0.291	0.385
	RotatE	0.377	0.630	0.352	0.452	0.403	0.655	0.383	0.483
	ExpressivE	0.504	0.728	0.511	0.600	0.486	0.720	0.483	0.578
	GPT-2 XL	0.712	0.730	0.960	0.829	0.717	0.737	**0.953**	0.831
	Rule-base	0.600	0.730	0.716	0.723	0.749	**1.000**	0.656	0.792
	CAGED	0.734	0.895	0.703	0.788	0.732	0.702	0.892	0.785
	LLM_sim without context	**0.865**	**0.934**	0.875	0.904	**0.798**	0.903	0.809	**0.854**
	LLM_sim detection	0.853	0.850	**0.970**	**0.906**	0.777	0.842	0.855	0.848

Table 2: Comparison of various noise detection models on WN18RR and FB15k-237 with different levels of noise (10%, 20%, 30%).

	WN18RR		FB15k-237	
	correctness	Human evaluation (randomly select 100 refined triple)	correctness	Human evaluation (randomly select 100 refined triple)
10% noise	82.37%	89.00%	87.24%	92.00%
20% noise	80.75%	87.00%	85.27%	88.00%
30% noise	78.46%	82.00%	83.46%	83.00%

Table 3: Results of noise refinement

5 Results and Analysis

In this section, we present the results of our experiments and conduct a thorough analysis to gain insight into the outcomes.

5.1 Result of Noise Detection

We analyze the performance of *LLM_sim* in noise detection for both WN18RR and FB15k-237 datasets, using prior work as a baseline for comparison.

Tables 2 shows the results for WN18RR and FB15k-237 with 10%, 20%, and 30% noise, respectively.

Our experimental results reveal several key insights. First, both the PLMs and the noise detection models outperform KGE models. KGE models perform notably worse in noise detection, likely due to their inability to account for noise during training.

Second, the PLM, represented by GPT-2 XL, performs well in recall but underperforms in other metrics. This could be due to the high proportion of positive samples in the datasets, making it difficult for the model to distinguish between noisy and non-noisy samples.

Third, the performance of *LLM_sim* differs between WN18RR and FB15k-237. In WN18RR, *LLM_sim* detection surpasses *LLM_sim* without context, at 10% and 20% noise. However, at 30% noise, *LLM_sim* without context performs better, possibly due to degraded quality from increased noise. In contrast, *LLM_sim* detection consistently underperforms in FB15k-237 due to low inter-triple correlation, reducing the value of contextual information.

		WN18RR				FB15k-237			
		MRR	Hit@1	Hit@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10
	KGC in clean data	0.506	0.459	0.519	0.597	0.212	0.148	0.235	0.339
10% noise	KGC	0.423	0.375	0.448	0.508	**0.191**	**0.128**	**0.214**	**0.319**
	KGC + CAGED	0.400	0.343	0.429	0.505	0.176	0.117	0.195	0.298
	KGC + GPT2	0.412	0.359	0.433	0.467	0.143	0.106	0.156	0.216
	KGC + *LLM_sim* without context	0.402	0.343	0.430	0.509	0.174	0.116	0.193	0.293
	KGC + *LLM_sim* detection	**0.434**	**0.380**	**0.460**	**0.533**	0.173	0.116	0.190	0.292
	KGC + *LLM_sim*	0.404	0.346	0.431	0.510	0.183	0.122	0.206	0.302
20% noise	KGC	0.363	0.317	0.396	0.435	0.174	0.116	0.192	0.293
	KGC + CAGED	0.351	0.295	0.381	0.450	0.168	0.115	0.185	0.276
	KGC + GPT2	0.338	0.396	0.370	0.404	0.171	0.115	0.188	0.286
	KGC + *LLM_sim* without context	0.370	0.312	0.401	0.473	0.175	0.117	0.196	0.294
	KGC + *LLM_sim* detection	**0.390**	**0.333**	**0.421**	**0.487**	0.170	0.115	0.187	0.284
	KGC + *LLM_sim*	0.376	0.318	0.407	0.481	**0.178**	**0.120**	**0.201**	**0.306**
30% noise	KGC	0.290	0.246	0.324	0.358	0.141	0.109	0.152	0.201
	KGC + CAGED	0.294	0.241	0.327	0.382	0.167	0.116	0.185	0.273
	KGC + GPT2	0.321	0.271	0.356	0.401	0.167	0.114	0.184	0.278
	KGC + *LLM_sim* without context	0.335	0.280	0.367	0.429	0.169	0.115	0.188	0.281
	KGC + *LLM_sim* detection	0.343	0.289	0.375	0.434	0.166	0.113	0.185	0.277
	KGC + *LLM_sim*	**0.370**	**0.312**	**0.391**	**0.463**	**0.177**	**0.119**	**0.199**	**0.295**

Table 4: Performance KGC task for WN18RR and FB15k-237 datasets with different noise conditions

Finally, as the noise ratio increases, the overall model performance declines. Nevertheless, the performance of LLMs remains relatively stable, further demonstrating their robustness in handling noisy datasets.

5.2 Result for Noise Refinement

Table 3 presents the results of both evaluations. The results suggest that our refinement method effectively refines triples to a certain extent. Notably, the manual inspection scores are slightly higher than those of correctness, likely due to dataset incompleteness. This implies that the model may predict correct triples that do not appear in the original dataset, resulting in some cases being marked as incorrect even if they are accurate.

5.3 Result and Analysis for KGC Task

The experimental results offer key insights into the impact of dataset noise on downstream KGC tasks. As expected, increased noise ratios correlate with greater performance degradation in KGC. However, an intriguing exception occurred in the FB15k-237 dataset with 10% noise: the dataset with noise (i.e., KGC) performed better than the data where the noise is detected and corrected. This outcome may be attributed to the relatively small proportion of noise within a large dataset, suggesting that the KGC model can still perform better due to the abundance of correct data.

Additionally, Tables 2 and 4 show a clear relationship that higher noise detection accuracy leads to better KGC performance. This highlights the importance of effective noise detection in improving downstream task accuracy. When the noise detection method is imperfect, filtering out noisy data results in a reduction in the number of good data samples, reducing the effectiveness of the model.

Our *LLM_sim* method also demonstrated distinct effects under varying noise levels. In the low-noise WN18RR dataset, while it achieved slightly better results than the *LLM_sim* without context but did not surpass *LLM_sim* detection. However, in high-noise conditions, *LLM_sim* significantly improved performance, demonstrating its value in noise-heavy scenarios.

Finally, our refinement method showed a substantial positive effect on the FB15k-237 dataset, likely due to the LLM's reliance on factual concepts (in FB15k-237) rather than purely semantic content (in WN18RR). This preference for fact-based knowledge enables LLMs to perform particularly well on datasets that prioritize factual correctness.

6 Conclusion

The results confirmed that our proposed *LLM_sim* method significantly enhanced KG reliability, benefiting downstream tasks such as KGC. These findings underscored the broader potential of LLMs for KG-specific tasks by detecting and refining noise in dynamic, evolving KGs. The demonstrated robustness of LLMs in high-noise settings highlighted

their applicability to real-world scenarios where KGs are frequently updated.

Moving forward, we plan to refine our methods to enhance noise detection and refinement capabilities, aiming for improved robustness and adaptability across diverse datasets.

7 Limitations

While our method achieved promising results on the WN18RR and FB15k-237 datasets, we have not yet tested it on real-world datasets. Additionally, our approach is limited by the difficulty of rigorously evaluating refined triples, a common challenge in practical KG applications. As a result, some limitations remain in fully identifying and removing all noise.

Acknowledgments

This work is supported by JSPS Grant-in-Aid for Early-Career Scientists (Grant Number 24K20834). This work was based on results obtained from a project, JPNP20006, commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

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

Zhangjie Fu, Xinle Wu, Chaowen Guan, Xingming Sun, and Kui Ren. 2016. Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement. *IEEE Transactions on Information Forensics and Security*, 11(12):2706–2716.

Siddhant Garg, Goutham Ramakrishnan, and Varun Thumbe. 2021. Towards robustness to label noise in text classification via noise modeling. In *CIKM 2021*, pages 3024–3028.

Rashida Hasan and Cheehung Chu. 2022. Noise in datasets: What are the impacts on classification performance? In *ICPRAM*.

Yan Hong, Chenyang Bu, and Xindong Wu. 2021. High-quality noise detection for knowledge graph embedding with rule-based triple confidence. In *PRICAI 2021*. Springer.

Hui Huang, Shuangzhi Wu, Xinnian Liang, Bing Wang, Yanrui Shi, Peihao Wu, Muyun Yang, and Tiejun Zhao. 2023. Towards making the most of llm for translation quality estimation. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 375–386. Springer.

Hanlei Jin, Yang Zhang, Dan Meng, Jun Wang, and Jinghua Tan. 2024. A comprehensive survey on process-oriented automatic text summarization with exploration of llm-based methods. *arXiv preprint arXiv:2403.02901*.

Jakub Lála, Odhran O'Donoghue, Aleksandar Shtedritski, Sam Cox, Samuel G Rodriques, and Andrew D White. 2023. Paperqa: Retrieval-augmented generative agent for scientific research. *arXiv preprint arXiv:2312.07559*.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Aleksandar Pavlović and Emanuel Sallinger. 2022. Expressive: A spatio-functional embedding for knowledge graph completion. *arXiv preprint arXiv:2206.04192*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *EMNLP-IJCNLP*, pages 2463–2473.

Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. 2022. Learning from noisy labels with deep neural networks: A survey. *IEEE Transactions*.

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

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International conference on ML*. PMLR.

Ruobing Xie, Zhiyuan Liu, Fen Lin, and Leyu Lin. 2018. Does william shakespeare really write hamlet? knowledge representation learning with confidence. In *AAAI 2018*, volume 32.

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

Kun Yi and Jianxin Wu. 2019. Probabilistic end-to-end noise correction for learning with noisy labels. In *CVPR 2019*, pages 7017–7025.

Qinggang Zhang, Junnan Dong, Keyu Duan, Xiao Huang, Yezi Liu, and Linchuan Xu. 2022. Contrastive knowledge graph error detection. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 2590–2599.