GS-Quant: Granular Semantic and Generative Structural Quantization for Knowledge Graph Completion

Qizhuo Xie, Yunhui Liu, Yu Xing, Qianzi Hou, Xudong Jin, Tao Zheng, Tieke He*
State Key Laboratory for Novel Software Technology, Nanjing University, China
{mikumifa, hetieke}@gmail.com

Abstract

Large Language Models (LLMs) have shown immense potential in Knowledge Graph Completion (KGC), yet bridging the modality gap between continuous graph embeddings and discrete LLM tokens remains a critical challenge. While recent quantization-based approaches attempt to align these modalities, they typically treat quantization as flat numerical compression, resulting in semantically entangled codes that fail to mirror the hierarchical nature of human reasoning. In this paper, we propose GS-Quant, a novel framework that generates semantically coherent and structurally stratified discrete codes for KG entities. Unlike prior methods, GS-Quant is grounded in the insight that entity representations should follow a linguistic coarse-to-fine logic. We introduce a Granular Semantic Enhancement module that injects hierarchical knowledge into the codebook, ensuring that earlier codes capture global semantic categories while later codes refine specific attributes. Furthermore, a Generative Structural Reconstruction module imposes causal dependencies on the code sequence, transforming independent discrete units into structured semantic descriptors. By expanding the LLM vocabulary with these learned codes, we enable the model to reason over graph structures isomorphically to natural language generation. Experimental results demonstrate that GS-Quant significantly outperforms existing text-based and embedding-based baselines. Our code is publicly available at https://github.com/mikumifa/GS-Quant.

1 Introduction

Knowledge Graphs (KGs) serve as a cornerstone for symbolic knowledge representation, supporting explicit reasoning and driving progress in applications ranging from recommender systems (Guo et al., 2020; Wang et al., 2025) and question answering (Peng et al., 2024; Gutiérrez et al., 2024; Ma et al., 2025) to the mitigation of hallucination in Large Language Models (LLMs) (Agrawal et al., 2024; Sun et al., 2024). Despite their widespread adoption, KGs suffer from pervasive incompleteness: they encode observed facts while inevitably omitting numerous valid triples (Chen et al., 2020; Liu et al., 2025b). This limitation necessitates Knowledge Graph Completion (KGC), a task aiming to infer missing links.

Recently, LLMs have fundamentally reshaped the landscape of KGC (Pan et al., 2024) by introducing powerful generative reasoning capabilities. Prior approaches broadly fall into two paradigms: text-based and embedding-based. Text-based methods (Wei et al., 2023; Xu et al., 2024; Guo et al., 2024; Li et al., 2025; Liu et al., 2022) linearize KG triples into textual prompts. While this provides explicit reasoning paths, the linearization process shatters the intrinsic graph topology (Liu et al., 2025b) and incurs prohibitive computational costs due to excessive token length (Lin et al., 2025). Conversely, embedding-based methods (Zhang et al., 2024; Liu et al., 2024, 2025a; Yang et al., 2025) align rigid, continuous KG embeddings (Bordes et al., 2013; Sun et al., 2019) with the LLM's latent space. While computationally efficient, a fundamental *modality gap* persists: continuous embeddings are holistic and dense, whereas LLMs operate on discrete, sequential tokens. This mismatch hinders the LLM's ability to effectively interpret and reason over graph structures.

To bridge this gap, recent works like SSQR (Lin et al., 2025) and ReaLM (Guo et al., 2025) have explored representing entities as sequences of discrete codes via Vector Quantization (van den Oord et al., 2017; Guo et al., 2026). By converting embeddings into code tokens, these methods align the data format with linguistic sequences. However, current quantization approaches in KGC remain suboptimal. For example, SSQR derives codes of an entity based on multiple feed-forward pro-

*Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 16782–16797

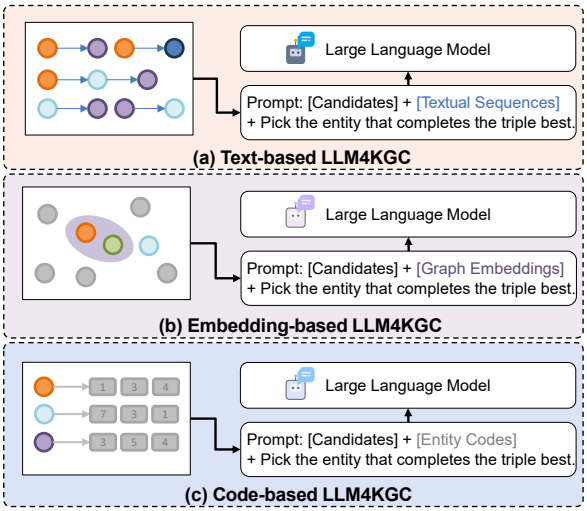


Figure 1: Illustration of LLM4KGC methods. (a) Text-based methods flatten the KG into textual sequences. (b) Embedding-based methods integrating graph embeddings into LLMs. (c) Our code-based method learns quantized codes for better LLM reasoning.

jections of its embeddings, resulting in flat and semantically entangled representations. The resulting code sequences lack intrinsic semantic logic; they function merely as compressed numerical approximations of a vector, rather than as structured, hierarchical semantic descriptors. This ignores a critical property of human language and LLM generation: reasoning is inherently hierarchical, proceeding from coarse-grained concepts to fine-grained details (Tenenbaum et al., 2011; He et al., 2024). Treating quantization as a flat compression problem fails to leverage the autoregressive nature of LLMs, which thrive on structured, causal dependencies.

In this paper, we propose GS-Quant, a framework that generates semantically coherent and structurally stratified quantized codes for KG entities. GS-Quant is grounded in the insight that an entity's discrete representation should mirror the hierarchical structure of language, i.e., moving from general semantic categories to specific instances. Built upon the Residual Quantized Variational Autoencoder (RQ-VAE) (Lee et al., 2022), our framework introduces two key innovations. First, the **Granular Semantic Enhancement (GSE)** module injects hierarchical knowledge (derived via clustering) into the codebook learning process. By forcing different quantization levels to align with a hierarchy tree, GSE ensures that earlier codes capture coarse-grained semantics (e.g., "Person") while later codes refine specific attributes (e.g., "Artist"), effectively creating a semantic coordinate system for the LLM. Second, the **Generative Structural Reconstruction (GSR)** module employs a lightweight GPT-style Transformer decoder to reconstruct the entity and its ancestors from the code sequence. This imposes a causal structure on the codes, ensuring that the discrete sequence is not just a set of independent units but a coherent "sentence" that encodes contextual and cross-level interactions. By treating KG entities as hierarchically structured code sequences, GS-Quant allows LLMs to reason over graph data in a manner isomorphic to natural language generation. We expand the LLM's vocabulary with the learned discrete codes and fine-tune the LLM to perform knowledge graph completion. Experimental results demonstrate that GS-Quant significantly outperforms existing baselines, setting a new standard for LLM-enhanced KGC.

2 Related Work

Traditional Embedding-based KGC methods. Embedding-based KGC methods learn low-dimensional representations of entities and relations under various geometric constraints. Classical translational and multiplicative models include TransE (Bordes et al., 2013), DistMult (Yang et al., 2015), ComplEx (Trouillon et al., 2016), and RotatE (Sun et al., 2019), while ATTH (Chami et al., 2020) and GIE (Cao et al., 2022) extend them to hyperbolic spaces to better capture hierarchical patterns in KGs. Beyond shallow embeddings, Graph Neural Networks (GNNs) (Zhou et al., 2020; Song et al., 2023b,a) have been applied to encode relational structures, with applications in various domains (Ma et al., 2023; Song et al., 2022, 2024). To handle relational heterogeneity, RGCN (Schlichtkrull et al., 2018), WGCN (Zhao et al., 2021), and CompGCN (Vashishth et al., 2020) introduce relation-aware message passing, and KBGAT (Nathani et al., 2019) incorporates attention mechanisms to weigh triple importance. Path-based models such as NBF-Net (Zhu et al., 2021), RED-GNN (Zhang and Yao, 2022), and A*Net (Zhu et al., 2023) exploit multi-hop relational paths to enhance inference and inductive generalization. Despite their efficiency and scalability, these approaches still lack the capacity to capture the rich semantic information of entities.

LLM-based KGC methods. Early works like KG-BERT (Yao et al., 2019), KEPLER (Wang

et al., 2021), and CoLE (Liu et al., 2022) leverage PLMs to encode triples. Building upon these foundations, current LLM-based approaches primarily diverge into text-based and embedding-based paradigms. Text-based methods (Wei et al., 2023; Xu et al., 2024; Guo et al., 2024; Li et al., 2025) serialize triples into natural language prompts, occasionally augmenting them with generated descriptions (Li et al., 2024; Jiang et al., 2024), though this often incurs high token costs and structural information loss (Liu et al., 2025b; Lin et al., 2025). Conversely, embedding-based methods (Zhang et al., 2024; Liu et al., 2024, 2025a; Yang et al., 2025) inject structural priors by aligning pretrained KG embeddings with the LLM's latent space. To further bridge the modality gap between continuous embeddings and discrete text, recent studies explore discrete codes, such as SSQR (Lin et al., 2025) using vector quantization and ReaLM (Guo et al., 2025) applying residual quantization. Despite their progress, these methods remain limited: code-based approaches largely expand a single entangled embedding, offering weak semantic or structural disentanglement, and SFT-based (Ouyang et al., 2022) methods do not bridge the gap between KG embedding space and the LLM's semantic space. These challenges motivate the need for more structured and semantically coherent discrete representations, which GS-Quant framework aims to provide.

3 Preliminaries

Knowledge Graph. A Knowledge Graph (denoted as $\mathcal{G} = \{\mathcal{E}, \mathcal{R}, \mathcal{T}\}$) can be formally represented as a set of triples in the form of $(h, r, t) \in \mathcal{T}$, where $h, t \in \mathcal{E}$, $r \in \mathcal{R}$. The notations h and t denote the head and the tail entity of a triple.

LLM-based KGC. In the LLM-based formulation of KGC, we illustrate the task using the tail prediction query $(h, r, ?)$ as an example. For a given query, we construct an input to the LLM that combines (i) the query triple, (ii) structural relational context, and (iii) textual information from the KG. The LLM is then required to select the correct tail entity t from a predefined candidate set.

4 Methodology

In this section, we present GS-Quant (Figure 2). We first encode entities into dense embeddings and construct a semantic tree leveraging hierarchical clustering. The embeddings and tree are then utilized to train the residual quantization module to obtain the discrete tokens of the graph. This quantization process is jointly optimized using Generative Structural Reconstruction and Granular Semantic Enhancement. Subsequently, the learned discrete codes are utilized to expand the LLM's vocabulary, and the model with the augmented vocabulary is then fine-tuned for the KGC task.

4.1 Quantized Codes Learning for KG

4.1.1 Basic Codes Learning

KG Encoding. For the structural component, a knowledge graph embedding model (e.g., RotatE (Sun et al., 2019)) is employed as the backbone to generate the relational embedding $\mathbf{s}_x^{\mathcal{G}} \in \mathbb{R}^d$.

For the textual component, we utilize both the entity name and description. Specifically, we encode them independently using a pretrained PLM and concatenate their respective truncated feature representations to form the textual embedding $\mathbf{s}_x^T \in \mathbb{R}^d$, thereby ensuring dimensional alignment with $\mathbf{s}_x^{\mathcal{G}}$. The final representation $\mathbf{s}_x$ is obtained via a weighted fusion:

$$\mathbf{s}_x = \rho\, \mathbf{s}_x^{\mathcal{G}} + (1 - \rho)\, \mathbf{s}_x^T, \tag{1}$$

where $\rho \in [0, 1]$ controls the relative balance between relational and textual contributions.

Residual Quantization (RQ). We adopt the residual quantization mechanism (Lee et al., 2022) to encode the enriched entity embedding $\mathbf{s}$, which can be either $\mathbf{s}_h$ or $\mathbf{s}_t$, to learn a latent representation $\mathbf{z} = \text{MLP}(\mathbf{s})$, which serves as the initial residual at the 0-th level, denoted as $\mathbf{r}_0 = \mathbf{z}$. At each quantization level l, there is a codebook $\mathbf{C}^l = \{\mathbf{v}_k^l\}_{k=1}^K$, where $\mathbf{v}_k^l$ is the k-th vector in the codebook at level l and K is the codebook size.

The initial residual $\mathbf{r}_0$ is then used to find the index of the nearest embedding in $\mathbf{C}^0$, given by $\mathbf{v}_0 = \arg\min_k \|\mathbf{r}_0 - \mathbf{v}_k^0\|_2$. At each level l, the residual update follows the rule $c_l = \arg\min_k \|\mathbf{r}_l - \mathbf{v}_k^l\|_2$, and the residual is iteratively updated as $\mathbf{r}_{l+1} = \mathbf{r}_l - \mathbf{v}_{c_l}^l$. This recursive residual approximation ultimately generates a quantization code tuple $\mathcal{I} = \{c_i\}_{i=0}^{m-1}$, where m is the maximum level of RQ.

Residual Quantization Regularization. Following generating the code tuple $\mathcal{I}$, we apply a stop-gradient based commitment loss to stabilize codebook learning. The residual quantization loss is:

$$\mathcal{L}_{\text{Q}} = \sum_{l=0}^{m-1} \left(\|\,\text{sg}[\mathbf{r}_l] - \mathbf{v}_{c_l}^l\|_2^2 + \alpha \|\mathbf{r}_l - \text{sg}[\mathbf{v}_{c_l}^l]\|_2^2 \right) \tag{2}$$

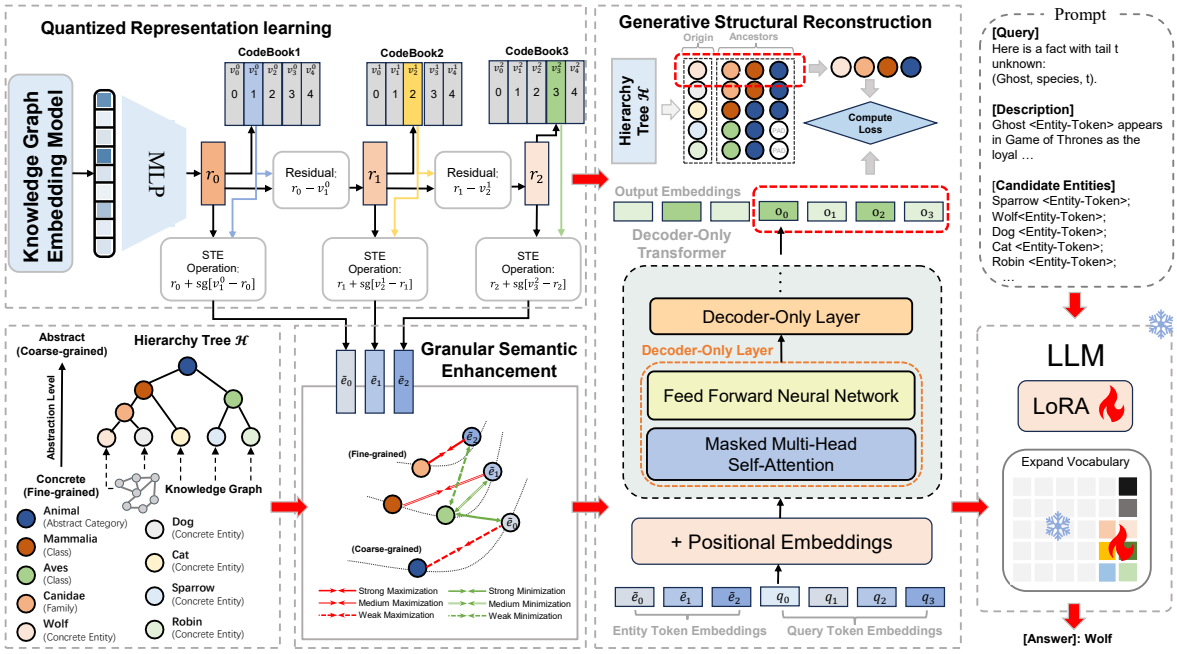


Figure 2: The overview of the GS-Quant framework. The pipeline first constructs a hierarchy tree. It then discretizes entities using Residual Quantization, optimized by **Granular Semantic Enhancement** for structural consistency. Subsequently, the **Generative Structural Reconstruction** module employs a GPT-style Transformer decoder to capture contextual and cross-level interactions. Finally, the learned codes are integrated into LLMs.

where $\mathrm{sg}[\cdot]$ denotes the stop-gradient operator, which prevents gradient updates for the quantized embeddings during backpropagation. The first term in $\mathcal{L}_{\mathrm{Q}}$ ensures that the trainable codebook vectors $\mathbf{v}_{c_l}^{l}$ are close to the corresponding residuals $\mathbf{r}_l$. Meanwhile, the second term, weighted by the hyperparameter α, constrains the residuals to remain close to the selected codebook entries.

Eq. (2) alone is insufficient to ensure semantically meaningful codebooks. We introduce additional components in the following section.

4.1.2 Granular Semantic Enhancement

In KGC, accurate prediction often requires hierarchical inference. For instance, in the query (Christmas, month, ?), the model should ideally identify the coarse-grained semantic category of "month" before narrowing down to the specific entity "December". However, traditional residual vector quantization is insufficient for this purpose. While it naturally exhibits a numerical hierarchy due to the recursive residual subtraction $\mathbf{r}_{l+1} = \mathbf{r}_l - \mathbf{v}_{c_l}^{l}$, it lacks explicit alignment with distinct semantic granularity levels. Each quantization layer merely encodes the numerical residual of the previous one based on Euclidean distance. The resulting codebooks often entangle semantics across layers, failing to explicitly follow the coarse-to-fine semantic hierarchy required for effective knowledge graph completion.

To address this, we first perform hierarchical clustering (e.g., agglomerative clustering (Müllner, 2011)) on entity semantic representations to generate a hierarchy tree $\mathcal{H}$. Let E denote a batch of entities. For each $\mathbf{e} \in E$, let $\boldsymbol{\mu}_{\mathbf{e}}$ be its corresponding cluster centroid. We define a differentiable surrogate for each quantized output of $\mathbf{e}$ as $\mathcal{V}_e = \{\tilde{\mathbf{v}}_i\}_{i=0}^{m-1}$, formulated as $\tilde{\mathbf{v}}_i = \mathbf{r}_i + \mathrm{sg}[\mathbf{v}_{c_i}^{i} - \mathbf{r}_i]$, where $\mathbf{r}i$ denotes the residual representation at step i, and $\mathbf{v}_{c_i}^{i}$ is the selected codebook vector from the i-th codebook. The stop-gradient operator $\mathrm{sg}[\cdot]$ ensures stable optimization by allowing gradients to flow through the residual pathway while preventing direct updates to the discrete codebook selection.

Coarse-to-Fine Alignment. GSE calibrates the quantized output $\mathcal{V}_e$ to capture coarse-to-fine semantic granularity. The loss is defined as:

$$\mathcal{L}_1 = -\frac{1}{|E|}\sum_{\mathbf{e}\in E}\sum_{i=0}^{m-1}\frac{\lambda_1^{i+1}}{m} \times \log\frac{\exp(\tilde{\mathbf{v}}_i \cdot \boldsymbol{\mu}_{\mathbf{e}}/\tau)}{\sum_{\mathbf{e}'\in E}\exp(\tilde{\mathbf{v}}_i \cdot \boldsymbol{\mu}_{\mathbf{e}'}/\tau)}. \tag{3}$$

where τ is the temperature hyperparameter and $\lambda_1 \in (0, 1)$ controls an exponentially decaying

weight across layers. This design biases earlier layers toward coarse, global semantics, while allowing deeper layers to focus on finer-grained distinctions.

Hierarchical Separability. GSE distinguishes each vector in $\mathcal{V}_e$ from its neighbors in the hierarchy tree $\mathcal{H}$. Let $\mathcal{N}_e$ denote the set of neighbor centroids for entity e. This loss is formulated as:

$$\mathcal{L}_2 = \frac{1}{|E|} \sum_{\mathbf{e} \in E} \sum_{i=0}^{m-1} \frac{\lambda_2^{m-i}}{m \cdot |\mathcal{N}_\mathbf{e}|} \times \sum_{\mathbf{n} \in \mathcal{N}_\mathbf{e}} \log \frac{\exp(\tilde{\mathbf{v}}_i \cdot \mathbf{n}/\tau)}{\sum_{\mathbf{e}' \in E} \exp(\tilde{\mathbf{v}}_i \cdot \boldsymbol{\mu}_{\mathbf{e}'}/\tau)}. \tag{4}$$

where $\lambda_2 \in (0, 1)$ applies a reverse decay that progressively strengthens constraints in deeper layers. This encourages fine-grained separability while allowing higher layers to remain semantically compact. The total GSE objective is $\mathcal{L}_{\text{GSE}} = \mathcal{L}_1 + \mathcal{L}_2$.

4.1.3 Generative Structural Reconstruction

For the code tuple $\mathcal{I}$, its rich combinatorial semantic capacity is essential for downstream LLM generation tasks. Different code combinations ought to encode distinct semantic variations, enabling the model to support complex generative reasoning.

To achieve this, we introduce Generative Structural Reconstruction. We employ a standard Transformer decoder to treat the tuple $\mathcal{I}$ as a structured semantic sequence. Formally, we construct a sequence of learnable query embeddings $\mathcal{Q} = \{\mathbf{q}_i\}_{i=0}^{L}$ and concatenate them with the differentiable surrogate $\mathcal{V}_e$ as the input. Through causal self-attention, the decoder produces reconstruction outputs $\{\mathbf{o}_i\}_{i=0}^{L}$ aligned with multi-granular targets: the leading query $\mathbf{q}_0$ is tasked with recovering the intrinsic entity embedding $\mathbf{s}$, while subsequent queries $\{\mathbf{q}_i\}_{i=1}^{L}$ reconstruct the corresponding ancestors $\{\mathbf{h}_i\}$ in the semantic hierarchy $\mathcal{H}$ to enforce global consistency. The overall GSR objective is formulated as:

$$\mathcal{L}_{\text{GSR}} = \|\tilde{\mathbf{o}}_0 - \mathbf{s}\|_2^2 + \lambda_s \|\tilde{\mathbf{o}}_1 - \mathbf{h}_0\|_2^2 + \lambda_h \sum_{i=2}^{L} \|\tilde{\mathbf{o}}_i - \mathbf{h}_{i-1}\|_2^2. \tag{5}$$

where λ_s weights the reconstruction of the semantic target $\mathbf{h}_0$, and λ_h controls the strength of reconstructing the remaining hierarchical semantic targets $\{\mathbf{h}_i\}_{i=1}^{L}$. In practice, we assign a significantly smaller value to λ_s compared to λ_h, considering that $\mathbf{h}_0$ has been integrated into the GSE module.

Importantly, This design not only significantly enhances the combinatorial semantic capacity of the discrete codes, but also firmly anchors them to both fine-grained and coarse-grained semantic targets, thereby ensuring coarse-to-fine semantic alignment within the quantized representation.

4.1.4 Full Quantization Learning

The final training objective is the summation of the three loss components:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{Q}} + \mathcal{L}_{\text{GSE}} + \mathcal{L}_{\text{GSR}}. \tag{6}$$

To ensure efficient utilization of the discrete latent space across all quantization levels, We select the model checkpoint based on the highest Codebook Entropy. Let p_k^m denote the empirical activation frequency of the k-th code within the m-th codebook. The Codebook Entropy $\mathcal{Y}$ is calculated as:

$$\mathcal{Y} = -\frac{1}{M} \sum_{m=1}^{M} \sum_{k=1}^{K} p_k^m \log p_k^m. \tag{7}$$

Maximizing $\mathcal{Y}$ encourages the model to fully exploit the inherent expressive capacity of the dictionary, as the entropy attains its maximum value when all codes are utilized with equal probability.

4.2 Fine-tuning LLMs with Quantized Codes

To seamlessly integrate the learned quantized representations, we explicitly extend the LLM's vocabulary with tokens corresponding to the codebook entries. To facilitate efficient adaptation, we freeze the original model parameters and update only specific components: (1) the embeddings of the newly added code tokens, allowing them to align with the LLM's generation dynamics; and (2) the adapter matrices introduced via Low-Rank Adaptation (LoRA) (Hu et al., 2022) in the attention and feed-forward layers. This dual approach ensures the effective injection of rich domain knowledge while preserving the LLM's general capabilities.

5 Experiments

5.1 Baselines and Evaluation Metrics

We compare the GS-Quant framework against three distinct categories of baseline methods: (1) Embedding-based methods, including TransE (Bordes et al., 2013), ComplEx (Trouillon et al., 2016), RotatE (Sun et al., 2019), TuckER (Balazevic et al., 2019), CompGCN (Vashishth et al., 2020); (2) Text-based methods, including KG-BERT (Yao et al.,

2019), MEM-KGC (Choi et al., 2021), CoLE (Liu et al., 2022); and (3) LLM-based methods including, KICGPT (Wei et al., 2023), DIFT (Liu et al., 2024), KG-FIT (Jiang et al., 2024), MKGL (Guo et al., 2024), SSQR (Lin et al., 2025).

We utilize the widely adopted evaluation metrics, namely Hits@k ($k = 1, 3, 10$) and MRR. Hits@k measures the proportion of query triples which the ground truth entities are ranked within the top-k position. MRR measures the mean reciprocal rank for each ground truth entities. In our framework, the finetuned LLM selects an entity as the answer from the ranking list of candidates. To assess its performance and make the results comparable to existing work, we move the selected entity to the top of the ranking list, and other candidates remain unchanged. Therefore, the gains in Hit@1 and MRR best highlight the practical value of our approach, as they reflect the model's ability to provide the correct answer immediately. Implementation details and the measures taken to ensure a fair comparison are described in Appendix A.

5.2 Main Results

Table 1 summarizes the performance of the GS-Quant framework. Specifically, compared to the strongest LLM-based baseline, GS-Quant achieves significant improvements across both datasets. On WN18RR, we observe a gain of approximately **1.7%** in MRR and **2.4%** in Hits@1, indicating more accurate top-ranked predictions. Similarly, on the FB15k-237 dataset, GS-Quant surpasses the strongest LLM-based baseline with improvements of **1.6%** in MRR and **2.2%** in Hits@1, demonstrating its robustness across datasets.

Notably, although we utilize the exact same candidate sets and instruction templates as DIFT in our experiments, GS-Quant consistently outperforms DIFT, particularly in Hits@1 on both datasets. This phenomenon strongly suggests that the quantized representations employed in GS-Quant are significantly more effective for LLM reasoning than the continuous embeddings used in DIFT.

Furthermore, GS-Quant shows clear advantages over SSQR, which also employs quantized representations. This performance boost is attributed to our codebook design, which not only ensures explicit alignment with distinct semantic granularity levels but also encourages the discrete units to capture contextual and cross-level interactions. These structured features make the representations inherently more aligned with the LLM's understanding and more effective for downstream tasks.

5.3 Ablation Studies

To assess the effectiveness of each component in our framework, we conducted a series of ablation studies, with results presented in Table 2. All ablation settings share the same training configuration, data splits, and optimization hyperparameters. Here, w/o $\mathcal{L}_1$ and w/o $\mathcal{L}_2$ each remove one of the two components that constitute the GSE module. w/o $\mathcal{L}_{\text{GSR}}$ disables the GSR while still retaining the use of quantized code tokens. Finally, w/o Code eliminates all quantized tokens from the vocabulary, preventing the LLM from leveraging any discrete semantic units produced by the quantization process. Several key observations can be seen: (1) Both $\mathcal{L}_1$ and $\mathcal{L}_2$ are essential for GSE. Removing either $\mathcal{L}_1$ or $\mathcal{L}_2$ leads to consistent performance drops on both datasets, indicating that the two objectives jointly support the formation of coherent multi-granular semantics (2) GSR further boosts the utility of multi-level quantization. Disabling GSR weakens the model's ability to preserve structurally consistent representations, resulting in degraded link prediction performance. This highlights the role of GSR in reinforcing the hierarchical organization learned during quantization. (3) Quantized entity tokens provide the largest performance gains. Removing entity tokens produces the most substantial degradation demonstrating that discrete semantic units are crucial for enabling LLMs to exploit KG-grounded representations.

5.4 Efficiency Analysis

To quantify the computational overhead of each key component, we measure the total training time on both datasets and summarize the results in a single figure. As shown in Figure 5, removing any individual component yields only a modest reduction in cost: eliminating $\mathcal{L}_1$ or $\mathcal{L}_2$ shortens training by 4%, while removing GSR results in a larger but still moderate 18% decrease. These findings show that the quantization-related objectives introduce only limited computational overhead. Detailed training time statistics are reported in Appendix C.

5.5 Hyperparameter Analysis

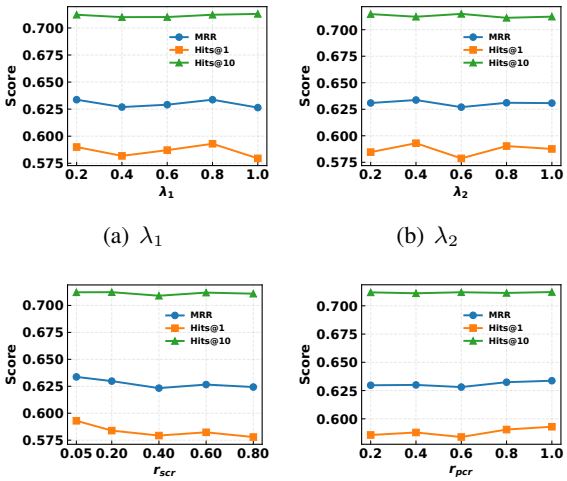
We analyze the sensitivity of our model to four key hyperparameters: λ_1, λ_2, λ_s, and λ_h. We vary each value within a designated range while keeping others fixed. The results show that a lower λ_s and a higher λ_h generally yield better performance. The

Model	WN18RR				FB15k237			
	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
Embedding-based Methods								
TransE (Bordes et al., 2013)	0.243	0.043	0.441	0.532	0.279	0.198	0.376	0.441
ComplEx (Trouillon et al., 2016)	0.440	0.410	0.460	0.510	0.247	0.158	0.275	0.428
RotatE (Sun et al., 2019)	0.476	0.428	0.492	0.571	0.338	0.241	0.375	0.533
TuckER (Balazevic et al., 2019)	0.470	0.443	0.482	0.526	0.358	0.266	0.394	0.544
CompGCN (Vashishth et al., 2020)	0.479	0.443	0.494	0.546	0.355	0.264	0.390	0.535
Text-based Methods								
KG-BERT (Yao et al., 2019)	0.216	0.041	0.302	0.524	-	-	-	0.420
MEM-KGC (Choi et al., 2021)	0.557	0.475	0.604	0.704	0.346	0.253	0.381	0.531
CoLE (Liu et al., 2022)	0.593	0.538	0.616	0.701	0.389	0.294	0.429	0.572
LLM-based Methods								
KICGPT (Wei et al., 2023)	0.564	0.478	0.612	0.677	0.412	0.327	0.448	0.581
DIFT (Liu et al., 2024)	<u>0.617</u>	<u>0.569</u>	<u>0.638</u>	<u>0.708</u>	<u>0.439</u>	<u>0.364</u>	<u>0.468</u>	0.586
KG-FIT (Jiang et al., 2024)	0.553	0.488	-	0.695	0.362	0.275	-	0.572
MKGL (Guo et al., 2024)	0.552	0.500	0.577	0.656	0.415	0.325	0.454	<u>0.591</u>
SSQR (Lin et al., 2025)	0.603	0.553	0.627	0.692	0.428	0.349	0.459	0.583
GS-Quant	**0.635**	**0.594**	**0.649**	**0.712**	**0.455**	**0.386**	**0.479**	**0.592**

Table 1: Comparison of KGC results on WN18RR and FB15k-237 datasets. The results are categorized into Embedding-based, Text-based, and LLM-based methods for fair comparison. The best results **boldfaced** and the second-best are <u>underlined</u>.

Settings	MRR	Hits@1	Hits@3	Hits@10
FB15k-237				
Ours	**0.455**	**0.386**	**0.479**	**0.592**
w/o $\mathcal{L}_1$	0.450 (-0.5%)	0.377 (-0.9%)	0.477 (-0.2%)	0.590 (-0.2%)
w/o $\mathcal{L}_2$	0.450 (-0.5%)	0.379 (-0.7%)	0.476 (-0.3%)	0.589 (-0.3%)
w/o $\mathcal{L}_{GSR}$	0.448 (-0.7%)	0.375 (-1.1%)	0.476 (-0.3%)	0.590 (-0.2%)
w/o Code	0.404 (-5.1%)	0.303 (-8.3%)	0.457 (-2.2%)	0.584 (-0.8%)
WN18RR				
Ours	**0.635**	**0.594**	**0.649**	**0.712**
w/o $\mathcal{L}_1$	0.629 (-0.5%)	0.587 (-0.6%)	0.646 (-0.4%)	0.711 (-0.1%)
w/o $\mathcal{L}_2$	0.625 (-0.9%)	0.577 (-1.6%)	0.646 (-0.4%)	0.710 (-0.2%)
w/o $\mathcal{L}_{GSR}$	0.627 (-0.7%)	0.585 (-0.8%)	0.642 (-0.8%)	0.710 (-0.2%)
w/o Code	0.607 (-2.7%)	0.541 (-5.2%)	0.640 (-1.0%)	0.708 (-0.4%)

Table 2: Ablation study on FB15k-237 and WN18RR.



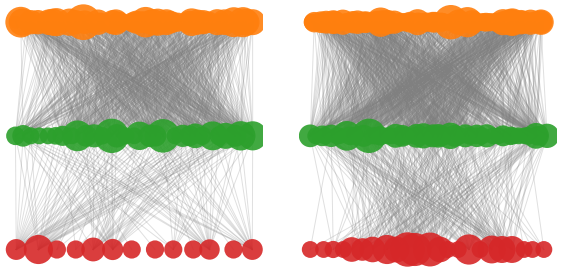
(a) λ_1 (b) λ_2 (c) λ_s (d) λ_h

Figure 3: Parameter sensitivity analysis on WN18RR.

best results appear when $\lambda_1 = 0.8$ and $\lambda_2 = 0.4$, indicating that the model benefits from a balanced configuration of these regularization terms. Overall, our model shows robustness to λ_1 and λ_2: by maintaining these parameters within the appropriate range, it achieves competitive performance.

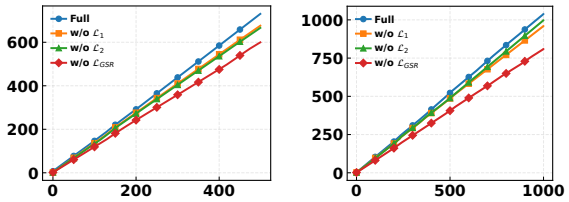
5.6 Codebook Analysis

Figure 4 illustrates the semantic hierarchy, comparing our method against a vanilla baseline implemented as a standard RQ-VAE(Lee et al., 2022). Specifically, in this visualization, nodes colored red, green, and orange correspond to discrete codes within the coarse-, medium-, and fine-grained quantization layers. The size of each node indicates its connectivity degree, reflecting the usage frequency of the code. The edges linking nodes across layers



(a) FB15k237 (Ours) (b) FB15k237 (Vanilla)

Figure 4: Visualization of the layer distribution of entity codes.



(a) FB15k237 (b) WN18RR

Figure 5: Visualization of cumulative training time.

demonstrate how distinct codes are hierarchically composed to form specific entity representations.

As shown in the figure, the coarse-grained codes exhibit a sparse yet globally uniform distribution, while the fine-grained layers display a significantly more discriminative distribution. This observation indicates that our model successfully disentangles semantic granularity, effectively correcting the misalignment observed in the baseline where broad concepts and specific entities are often conflated.

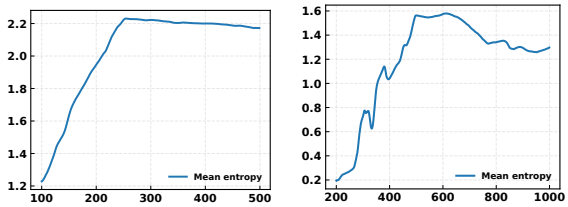
5.7 Codebook Entropy Analysis

Correlation between Entropy and Performance. To validate our selection strategy, we analyze the relationship between $\mathcal{Y}$ and downstream performance metrics on the FB15k-237 and WN18RR dataset. In Table 3, a higher codebook entropy $\mathcal{Y}$ consistently correlates with better performance across MRR and Hits@K metrics. This positive correlation confirms that $\mathcal{Y}$ serves as a robust indicator for identifying the most effective model checkpoint.

Entropy Evolution During Training. We further visualize the evolution of codebook entropy throughout the training process. Figure 6 illustrates the changes in entropy over training steps on both

Entropy ($\mathcal{Y}$)	MRR	Hits@1	Hits@3	Hits@10
		FB15k-237		
1.2285	0.447 (-0.8%)	0.376 (-1.0%)	0.473 (-0.6%)	0.585 (-0.7%)
1.7339	0.451 (-0.4%)	0.381 (-0.5%)	0.477 (-0.2%)	0.589 (-0.3%)
1.9421	0.453 (-0.2%)	0.384 (-0.3%)	0.477 (-0.2%)	0.590 (-0.2%)
2.2156	**0.455**	**0.386**	**0.479**	**0.592**
		WN18RR		
0.1909	0.629 (-0.6%)	0.583 (-1.2%)	0.646 (-0.3%)	0.711 (-0.0%)
0.6229	0.633 (-0.2%)	0.588 (-0.7%)	0.646 (-0.3%)	0.712 (-0.0%)
1.2897	0.633 (-0.2%)	0.590 (-0.4%)	0.648 (-0.1%)	0.712 (-0.0%)
1.3852	**0.635**	**0.594**	**0.649**	**0.712**

Table 3: Impact of codebook entropy on ranking performance. For each dataset, scores are annotated with the absolute gap to the best configuration.



(a) FB15k237 (b) WN18RR

Figure 6: Evolution of $\mathcal{Y}$ over training steps.

FB15k-237 and WN18RR datasets. Ideally, the entropy should grow or stabilize at a high value, indicating that the model is continuously exploring and utilizing the codebook space effectively.

6 Additional Analysis

We provide comprehensive supplementary analyses of GS-Quant in the Appendix, covering: (i) training dynamics (Appendix B); (ii) the layout of KG token embeddings (Appendix D); (iii) qualitative case studies (Appendix F); (iv) sensitivity to KG embedding initializations (Appendix G); (v) robustness across varying LLM backbones (Appendix H).

7 Conclusion

In this paper, we presented GS-Quant, a framework bridging KGs and LLMs via quantization. By integrating Granular Semantic Enhancement for coarse-to-fine consistency and Generative Structural Reconstruction for alignment with LLM generation dynamics, GS-Quant produces semantically rich discrete codes. Extensive experiments confirm that our approach consistently outperforms state-of-the-art methods on standard benchmarks, validating the effectiveness of hierarchical semantic quantization for knowledge–LLM integration.

Limitations

Despite the promising results, this work has several limitations that are worth noting.

First, our framework relies on a pre-trained large language model as the backbone. Although experiments across multiple 7B-scale models demonstrate robustness, the overall performance and efficiency are still bounded by the capacity and inference cost of the underlying LLM. Scaling the approach to larger models or deploying it in low-resource settings may introduce additional computational challenges.

Second, the proposed quantization and codebook construction are learned on specific knowledge graph benchmarks. While we observe consistent improvements on WN18RR and FB15k-237, the generalization of the learned code semantics to other knowledge graphs or domains with substantially different structural properties remains to be systematically evaluated.

Finally, our evaluation primarily focuses on link prediction tasks. While the design of GS-Quant is motivated by enhancing generative reasoning in LLMs, further validation on downstream generative or reasoning-intensive tasks would be necessary to fully assess its broader applicability.

We leave these directions for future work.

Ethical Considerations

We adhere to the ACL Code of Ethics and have taken every measure to ensure that our research complies with the ethical guidelines set forth. Our work does not involve human subjects, nor does it raise any ethical concerns related to privacy or data usage. All datasets used in our experiments are publicly available, and we ensure that their release and use adhere to proper data-sharing policies. We have carefully selected the datasets and evaluation metrics to ensure fairness and transparency in our findings. No potential conflicts of interest, sponsorship biases, or ethical violations have been identified in our study. We commit to maintaining the highest standards of research integrity, and we are open to addressing any ethical concerns that may arise during the review process.

Acknowledgments

This work is partially supported by National Science and Technology Major Project (2026ZD1611200), and Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (No. JYB2025XDXM118).

References

Garima Agrawal, Tharindu Kumarage, Zeyad Alghamdi, and Huan Liu. 2024. Can knowledge graphs reduce hallucinations in llms?: A survey. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3947–3960. Association for Computational Linguistics.

Ivana Balazevic, Carl Allen, and Timothy Hospedales. 2019. TuckER: Tensor factorization for knowledge graph completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5185–5194, Hong Kong, China. Association for Computational Linguistics.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 2787–2795.

Zongsheng Cao, Qianqian Xu, Zhiyong Yang, Xiaochun Cao, and Qingming Huang. 2022. Geometry interaction knowledge graph embeddings. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(5):5521–5529.

Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. 2020. Low-dimensional hyperbolic knowledge graph embeddings. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6901–6914, Online. Association for Computational Linguistics.

Zhe Chen, Yuehan Wang, Bin Zhao, Jing Cheng, Xin Zhao, and Zongtao Duan. 2020. Knowledge graph completion: A review. *Ieee Access*, 8:192435–192456.

Bonggeun Choi, Daesik Jang, and Youngjoong Ko. 2021. Mem-kgc: Masked entity model for knowledge graph completion with pre-trained language model. *IEEE Access*, 9:132025–132032.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Lingbing Guo, Zhongpu Bo, Zhuo Chen, Yichi Zhang, Jiaoyan Chen, Lan Yarong, Mengshu Sun, Zhiqiang Zhang, Yangyifei Luo, Qian Li, and 1 others. 2024. Mkgl: mastery of a three-word language. *Advances in Neural Information Processing Systems*, 37:140509–140534.

Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3549–3568.

Wenbin Guo, Xin Wang, Jiaoyan Chen, Lingbing Guo, Zhao Li, and Zirui Chen. 2025. Realm: Residual quantization bridging knowledge graph embeddings and large language models. *Preprint*, arXiv:2510.09711.

Zeyuan Guo, Enmao Diao, Cheng Yang, and Chuan Shi. 2026. Graph tokenization for bridging graphs and transformers. In *The Fourteenth International Conference on Learning Representations*.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. In *Advances in Neural Information Processing Systems*, volume 37, pages 59532–59569. Curran Associates, Inc.

Yuan He, Moy Yuan, Jiaoyan Chen, and Ian Horrocks. 2024. Language models as hierarchy encoders. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Pengcheng Jiang, Lang Cao, Cao Xiao, Parminder Bhatia, Jimeng Sun, and Jiawei Han. 2024. Kg-fit: Knowledge graph fine-tuning upon open-world knowledge. In *Advances in Neural Information Processing Systems*, volume 37, pages 136220–136258. Curran Associates, Inc.

Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. 2022. Autoregressive image generation using residual quantization. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11513–11522.

Dawei Li, Zhen Tan, Tianlong Chen, and Huan Liu. 2024. Contextualization distillation from large language model for knowledge graph completion. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 458–477. Association for Computational Linguistics.

Muzhi Li, Cehao Yang, Chengjin Xu, Xuhui Jiang, Yiyan Qi, Jian Guo, Ho-fung Leung, and Irwin King. 2025. Retrieval, reasoning, re-ranking: A context-enriched framework for knowledge graph completion. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4349–4363.

Qika Lin, Tianzhe Zhao, Kai He, Zhen Peng, Fangzhi Xu, Ling Huang, Jingying Ma, and Mengling Feng. 2025. Self-supervised quantized representation for seamlessly integrating knowledge graphs with large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13587–13602, Vienna, Austria. Association for Computational Linguistics.

Ben Liu, Jihai Zhang, Fangquan Lin, Cheng Yang, and Min Peng. 2025a. Filter-then-generate: Large language models with structure-text adapter for knowledge graph completion. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 11181–11195.

Yang Liu, Zequn Sun, Guangyao Li, and Wei Hu. 2022. I know what you do not know: Knowledge graph embedding via co-distillation learning. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM '22, page 1329–1338, New York, NY, USA. Association for Computing Machinery.

Yang Liu, Xiaobin Tian, Zequn Sun, and Wei Hu. 2024. Finetuning generative large language models with discrimination instructions for knowledge graph completion. In *The Semantic Web – ISWC 2024: 23rd International Semantic Web Conference, Baltimore, MD, USA, November 11–15, 2024, Proceedings, Part I*, page 199–217, Berlin, Heidelberg. Springer-Verlag.

Yu Liu, Yanan Cao, Xixun Lin, Yanmin Shang, Shi Wang, and Shirui Pan. 2025b. Enhancing large language model for knowledge graph completion via structure-aware alignment-tuning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20981–20995.

Chuangtao Ma, Yongrui Chen, Tianxing Wu, Arijit Khan, and Haofen Wang. 2025. Large language models meet knowledge graphs for question answering: Synthesis and opportunities. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 24589–24608, Suzhou, China. Association for Computational Linguistics.

Yueen Ma, Zixing Song, Xuming Hu, Jingjing Li, Yifei Zhang, and Irwin King. 2023. Graph component contrastive learning for concept relatedness estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 13362–13370.

Daniel Müllner. 2011. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*.

Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. 2019. Learning attention-based embeddings for relation prediction in knowledge graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4710–4723, Florence, Italy. Association for Computational Linguistics.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

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*, pages 1–20.

Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2024. Graph retrieval-augmented generation: A survey. *ACM Transactions on Information Systems*.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer.

Zixing Song, Yueen Ma, and Irwin King. 2022. Individual fairness in dynamic financial networks. In *NeurIPS 2022 Workshop: New Frontiers in Graph Learning*.

Zixing Song, Ziqiao Meng, and Irwin King. 2024. A diffusion-based pre-training framework for crystal property prediction. In *AAAI*, pages 8993–9001. AAAI Press.

Zixing Song, Yifei Zhang, and Irwin King. 2023a. No change, no gain: Empowering graph neural networks with expected model change maximization for active learning. In *NeurIPS*.

Zixing Song, Yifei Zhang, and Irwin King. 2023b. Optimal block-wise asymmetric graph construction for graph-based semi-supervised learning. In *NeurIPS*.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. 2024. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations*.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.

Joshua B Tenenbaum, Charles Kemp, Thomas L Griffiths, and Noah D Goodman. 2011. How to grow a mind: Statistics, structure, and abstraction. *science*, 331(6022):1279–1285.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Eric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2071–2080, New York, New York, USA. PMLR.

Aaron van den Oord, Oriol Vinyals, and koray kavukcuoglu. 2017. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2020. Composition-based multi-relational graph convolutional networks. In *International Conference on Learning Representations*.

Shijie Wang, Wenqi Fan, Yue Feng, Lin Shanru, Xinyu Ma, Shuaiqiang Wang, and Dawei Yin. 2025. Knowledge graph retrieval-augmented generation for LLM-based recommendation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27152–27168, Vienna, Austria. Association for Computational Linguistics.

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.

Yanbin Wei, Qiushi Huang, Yu Zhang, and James Kwok. 2023. KICGPT: Large language model with knowledge in context for knowledge graph completion. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 8667–8683, Singapore. Association for Computational Linguistics.

Derong Xu, Ziheng Zhang, Zhenxi Lin, Xian Wu, Zhihong Zhu, Tong Xu, Xiangyu Zhao, Yefeng Zheng,

and Enhong Chen. 2024. Multi-perspective improvement of knowledge graph completion with large language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 11956–11968.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 43 others. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Mengxue Yang, Chun Yang, Jiaqi Zhu, Jiafan Li, Jingqi Zhang, Yuyang Li, and Ying Li. 2025. SLiNT: Structure-aware language model with injection and contrastive training for knowledge graph completion. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 13658–13671, Suzhou, China.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Kg-bert: Bert for knowledge graph completion.

Yichi Zhang, Zhuo Chen, Lingbing Guo, yajing Xu, Wen Zhang, and Huajun Chen. 2024. Making large language models perform better in knowledge graph completion. In *ACM Multimedia 2024*.

Yongqi Zhang and Quanming Yao. 2022. Knowledge graph reasoning with relational digraph. In *Proceedings of the ACM Web Conference 2022*, WWW '22, page 912–924, New York, NY, USA. Association for Computing Machinery.

Yunxiang Zhao, Jianzhong Qi, Qingwei Liu, and Rui Zhang. 2021. Wgcn: Graph convolutional networks with weighted structural features. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 624–633, New York, NY, USA. Association for Computing Machinery.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81.

Zhaocheng Zhu, Xinyu Yuan, Michael Galkin, Louis-Pascal Xhonneux, Ming Zhang, Maxime Gazeau, and Jian Tang. 2023. Aast net: A scalable path-based reasoning approach for knowledge graphs. In *Advances in Neural Information Processing Systems*, volume 36, pages 59323–59336. Curran Associates, Inc.

Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. 2021. Neural bellman-ford networks: A general graph neural network framework for link prediction. In *Advances in Neural Information Processing Systems*, volume 34.

A Implementation Details

The statistics of utilized datasets are shown in Table 4. We conduct all our experiments on a Linux server equipped with two Intel Xeon Gold 6248R processors and eight A100-PCIE-40GB GPUs.[1]

To ensure a fair comparison among LLM-based methods, we standardize the experimental configurations for DIFT, SSQR, and our approach. Specifically, we employ Llama3-8B as the unified backbone to eliminate architectural variance and adopt the CoLE candidate sets and prompts released by DIFT. Under this aligned protocol, we reproduce the SSQR results while keeping its other hyperparameters consistent with the official implementation.

For parameter-efficient fine-tuning, we adopt LoRA (Hu et al., 2022), with the detailed prompts provided in Appendix I. For hierarchical clustering, we use the LLM-enhanced agglomerative clustering following KG-FIT (Jiang et al., 2024). All related hyperparameters are reported in Table 5. The discrete codebooks of GS-Quant are learned with a dedicated decoder-only module, whose training configurations are summarized in Table 6. The computational cost of LoRA-based re-ranking is presented in Table 10.

Dataset	FB15k237	WN18RR
Entities	14,541	40,943
Relations	237	11
Train	272,115	86,835
Valid	17,535	3,034
Test	20,466	3,134

Table 4: Statistics of Datasets

B Training Process of GS-Quant

Figure 7 presents the training process of GS-Quant on FB15K-237 and WN18RR, reporting the evolution of MRR and Hits@{1,3,10} across training

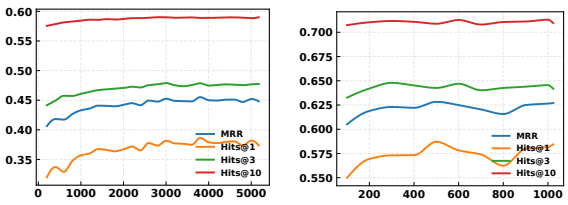
[1]Only a subset of the GPUs are utilized in our experiments.

Hyperparameters	Settings
Precision	bf16
Random seed	42
Learning rate	2e-4
Per-device train batch size	16
Per-device eval batch size	8
Gradient accumulation steps	1
Optimizer	AdamW
Scheduler	Constant LR
Warmup ratio	0.03
Weight decay	0.0
Max grad norm	1.0
DeepSpeed config	Zero-3
LoRA rank	32
LoRA α	64
Max new tokens	64
Min new tokens	1
Max training steps	3800
Dataset-Specific Hyperparameters	
FB15K-237	Max training epoch: 8
WN18RR	Max training epoch: 4

Table 5: Hyperparameters for LoRA and LLM

Hyperparameters	Settings
Embedding dim	512
Codebook layers	4
Codebook size	1024
Encoder layers	[512, 512, 512]
Reconstruction layers	2
Reconstruction heads	4
Parent recon count	5
Dropout probability	0.0
λ_1	0.8
λ_2	0.4
Learning rate	1e-4
Commit loss weight	0.25
Dataset-Specific Hyperparameters	
FB15K-237	Max steps: 500 Batch size: 14,541 λ_s: 1 λ_h: 1
WN18RR	Max steps: 1000 Batch size: 16,348 λ_s: 0.05 λ_h: 1

Table 6: Hyperparameters for training the quantized codebooks.



(a) FB15k237 (b) WN18RR

Figure 7: Visualization of the evolution of MRR and Hits@{1,3,10} across training checkpoints.

checkpoints. Overall, GS-Quant shows stable convergence behavior on both datasets, with consistent improvements across all evaluation metrics.

On WN18RR, the model converges rapidly in the early stage, where MRR and Hits@1 increase noticeably within the first few hundred checkpoints, and then stabilize with only minor fluctuations. The best performance is reached around checkpoint 1024, indicating that GS-Quant can efficiently capture the core semantic structure of this dataset.

In contrast, FB15K-237 exhibits a longer optimization process, with performance improving steadily over a larger number of training steps. The model achieves its peak MRR at checkpoint 3800, while maintaining stable performance in later stages. This difference in training steps is mainly due to the substantially larger training set of FB15K-237 compared to WN18RR.

Notably, among all metrics, Hits@1 shows the most significant relative improvement during training on both datasets. The gains in Hits@1 and MRR best highlight the practical value of our approach, as they directly reflect the model's ability to rank the correct entity at the top position and provide the correct answer immediately, rather than merely placing it within a broader candidate set.

C Detailed Training Time Statistics

This section reports detailed wall-clock training time statistics for GS-Quant and its ablated variants. All experiments are conducted under identical hardware configurations and training settings, with only the corresponding objective terms removed for each variant.

Table 7 summarizes the total training time on FB15k-237 and WN18RR, as well as the absolute and relative time savings compared to the full model. These statistics provide a detailed reference for the efficiency trends discussed in the main text.

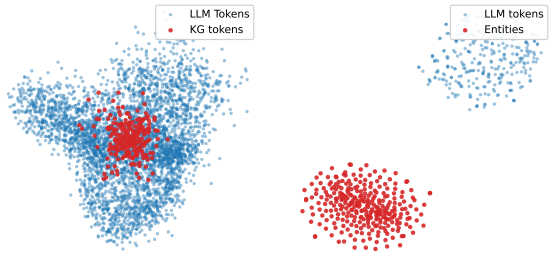
D Token Embeddings in LLMs

To analyze how GS-Quant represents KG tokens in comparison with base language tokens, we visualize all KG tokens together with LLM tokens on the FB15k-237 dataset. All token embeddings are projected to two dimensions using t-SNE (van der Maaten and Hinton, 2008), as shown in Figure 8.

As illustrated in Figure 8(a), KG tokens in GS-

Settings	FB15K-237		WN18RR	
	Time (s)	Saved (%)	Time (s)	Saved (%)
Full	730.48	—	1037.34	—
w/o $\mathcal{L}_1$	676.27	7.42%	959.00	7.55%
w/o $\mathcal{L}_2$	666.18	8.80%	997.03	3.89%
w/o $\mathcal{L}_{\text{GSR}}$	599.62	17.91%	808.86	22.03%

Table 7: Training cost comparison across ablated variants.



(a) Ours (b) DIFT

Figure 8: Embedding virtualization in LLMs (FB15k-237 dataset): KG tokens versus base word tokens for ours, and entity embeddings versus base tokens for DIFT.

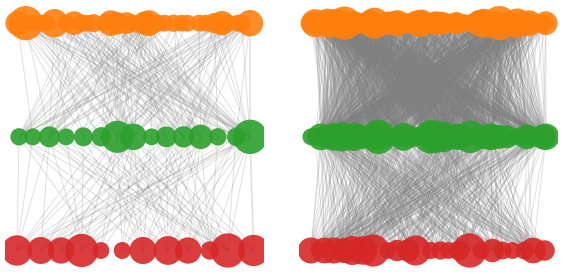
Quant form a compact and distinguishable region that remains well integrated with the overall LLM token space. In contrast, DIFT exhibits a more isolated separation between entity embeddings and language tokens (Figure 8(b)), suggesting weaker embedding-space integration. Overall, these results indicate that GS-Quant achieves token-level differentiation while preserving compatibility with the native LLM embedding geometry.

E The Layer Distribution on WN18RR

As shown in Figure 9, the coarse-grained code layers maintain a sparse and globally uniform distribution, while deeper layers gradually evolve into highly discriminative and structured patterns. The evidence also suggests that our model enforces a clearer separation of semantic granularity on WN18RR, where entities with subtle relational differences are more cleanly differentiated in fine-grained layers.

F Case Studies

To demonstrate the robustness of our codebook-based representation, we present a qualitative analysis of four distinct cases in Table 8. These examples highlight how the learned discrete codes capture multi-granular semantics to correct various types



(a) WN18RR (Ours) (b) WN18RR (Vanilla)

Figure 9: Visualization of the layer distribution of entity codes.

of baseline errors. In this comparison, the baseline refers to the model's configuration where the discrete codes are removed from the input prompt, forcing the model to rely solely on textual descriptions.

Geographical Reasoning (Case 1 & 2). The first two cases illustrate improvements in geographical reasoning. In Case 1, the baseline suffers from a "self-loop" error, failing to understand that *Beverly Hills* is contained within *Los Angeles*. Our model resolves this by capturing the hierarchical relationship encoded in the suffix codes. Similarly, in Case 2, the baseline predicts a "Congressional District" when the relation requires a "State" level entity. Our approach successfully corrects this granularity mismatch, identifying *Virginia* as the correct administrative level.

Fine-grained & Type Consistency (Case 3 & 4). The benefits extend beyond geography. Case 3 shows that our model can distinguish between fine-grained academic degrees (Bachelor vs. Master), whereas the baseline retrieves a generic related concept. Finally, Case 4 demonstrates strict type enforcement: while the baseline erroneously predicts an event (*2010 Winter Olympics*) for a query about medals, our model correctly retrieves the object *Bronze medal*, ensuring semantic consistency with the relation.

G Comparison of Different KG Embeddings

To demonstrate the robustness of our framework, we employ four representative models as initial embeddings: ComplEx (Trouillon et al., 2016), DistMult (Yang et al., 2015), pRotatE (Sun et al., 2019),

Case 1: Geographical Containment (Hierarchy)

Query: (?, `/location/.../contains`, Beverly Hills)

Gold: Los Angeles

✗ **Base:** Beverly Hills *(Self-loop Error)*

✓ **Ours: Los Angeles**

Analysis: *Baseline fails to model the containment hierarchy (Part vs. Whole); Ours identifies the super-region.*

Case 2: Administrative Granularity (Level)

Query: (?, `/location/.../religion`, Presbyterianism)

Gold: Virginia (State)

✗ **Base:** Pennsylvania's 12th congressional district *(Too Specific)*

✓ **Ours: Virginia**

Analysis: *Baseline confuses a District with a State; Ours fixes the granularity to the correct admin level.*

Case 3: Educational Degree (Fine-grained)

Query: (?, `/.../institution`, Royal Holloway)

Gold: Bachelor of Arts

✗ **Base:** Master of Arts *(Wrong Degree Level)*

✓ **Ours: Bachelor of Arts**

Analysis: *Baseline retrieves a related but incorrect degree; Ours distinguishes fine-grained academic ranks.*

Case 4: Entity Type Consistency (Semantic)

Query: (Bohemia, `/olympics/.../medal`, ?)

Gold: Bronze medal

✗ **Base:** 2010 Winter Olympics *(Event vs. Object)*

✓ **Ours: Bronze medal**

Analysis: *Baseline predicts an event instead of a medal object; Ours enforces strict type consistency.*

Table 8: Qualitative examples comparing our codebook-based approach against the baseline across four distinct scenarios. The codebook effectively corrects errors related to hierarchical containment, administrative level alignment, fine-grained semantic discrimination, and entity type consistency.

and TransE (Bordes et al., 2013). The quantitative comparison is reported in Table 9. It is worth noting that the results for the baselines (Base) and the competitor KG-FIT (Jiang et al., 2024) are collected from their original literature and our reproduction experiments to ensure a fair comparison.

As shown in the table, our framework achieves consistent and substantial improvements over the original embedding models across all metrics. Furthermore, when compared to KG-FIT, our method maintains a clear advantage, particularly in terms of MRR and Hits@1. While KG-FIT shows competitive performance in broader retrieval metrics, our framework demonstrates superior ranking precision, effectively placing the correct entities at the top ranks. These results highlight the strong generalization capacity of our framework, regardless of the specific KG embedding model used.

Model	Setting	MRR	Hits@1	Hits@3	Hits@10
ComplEx	Base	0.440	0.410	0.460	0.510
	KG-FIT	0.585	0.520	-	0.705
	Ours	**0.630**	**0.588**	**0.646**	**0.710**
	Δ vs Base	*+0.190*	*+0.178*	*+0.186*	*+0.200*
	Δ vs FIT	*+0.045*	*+0.068*	-	*+0.005*
DistMult	Base	0.430	0.390	0.440	0.490
	KG-FIT	0.590	0.530	-	0.710
	Ours	**0.636**	**0.595**	**0.651**	**0.713**
	Δ vs Base	*+0.206*	*+0.205*	*+0.211*	*+0.223*
	Δ vs FIT	*+0.046*	*+0.065*	-	*+0.003*
pRotatE	Base	0.460	0.416	0.479	0.552
	KG-FIT	0.590	0.511	-	0.722
	Ours	**0.626**	**0.581**	**0.646**	**0.709**
	Δ vs Base	*+0.166*	*+0.165*	*+0.167*	*+0.157*
	Δ vs FIT	*+0.036*	*+0.070*	-	*-0.013*
TransE	Base	0.243	0.043	0.441	0.532
	KG-FIT	0.550	0.450	-	0.695
	Ours	**0.621**	**0.570**	**0.647**	**0.713**
	Δ vs Base	*+0.378*	*+0.527*	*+0.206*	*+0.181*
	Δ vs FIT	*+0.071*	*+0.120*	-	*+0.018*

Table 9: Detailed performance comparison on WN18RR. "Δ" denotes the absolute performance gain of our method compared to the baselines and KG-FIT. "-" indicates the metric was not reported in the baseline paper.

H Performance with different LLMs

This appendix experiment evaluates the robustness of our framework under different LLM backbones, including Mistral (Jiang et al., 2023), Qwen2.5 (Yang et al., 2024), Vicuna (Zheng et al., 2023), and DeepSeek-R1-Distill-Qwen (DeepSeek-AI et al., 2025). All models are trained and evaluated using the same data splits, optimization settings, and hyperparameters. As shown in Table 11, the performance remains stable across different LLM choices on both WN18RR and FB15k-237, indicating that GS-Quant does not rely on a specific LLM implementation and exhibits strong robustness to backbone variations.

Dataset	SFT Time	Re-rank Time
FB15K-237	1h 56min	0.180s / query
WN18RR	17min	0.152s / query

Table 10: Training and inference time costs for SFT and re-ranking.

I Prompt Templates

Table 12 presents the prompt templates employed in our method for knowledge graph completion.

Model	WN18RR				FB15k-237			
	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
Qwen2.5-14B	**0.6292**	**0.5842**	0.6496	0.7117	0.4506	0.3799	0.4761	0.5888
Qwen2.5-7B	0.6249	0.5747	**0.6517**	**0.7128**	0.4385	0.3640	0.4654	0.5860
Mistral-7B	0.6205	0.5723	0.6431	0.7123	**0.4515**	**0.3810**	**0.4774**	0.5891
Llama-2-13B	0.6082	0.5585	0.6335	0.7068	0.4465	0.3722	0.4750	0.5885
Llama-2-7B	0.6105	0.5628	0.6295	0.7058	0.4378	0.3615	0.4672	0.5855
Vicuna-13B	0.6065	0.5547	0.6305	0.7056	0.4484	0.3750	0.4771	**0.5900**
Vicuna-7B	0.6121	0.5645	0.6310	0.7055	0.4391	0.3635	0.4687	0.5866
DeepSeek-R1-Distill-Qwen-7B	0.6010	0.5455	0.6286	0.7042	0.4189	0.3362	0.4512	0.5797

Table 11: Comparison of LLM-based KGC performance on WN18RR and FB15k-237. The best results are **boldfaced** and the second-best are underlined.

Query triple (telephone, *verb group*, < ??? >)

Prompts

User: Here is a triplet with tail entity t unknown: (telephone, *verb group*, t).

Following are some details about t:
telephone, get or try to get into communication (with someone) by telephone; “I tried to call you all night”; “Take two aspirin and call me in the morning”

Quantized representation:
<#bau><#ya><#bcq><#rm>

Following are some triplets about t:
(telephone, derivationally related form, tintinnabulation); (telephone, derivationally related form, telephony); (telephone, synset domain topic of, telephony); (telephone, derivationally related form, telephoner); (telephone, hypernym, telecommunicate); (telephone, derivationally related form, telephone call); (tintinnabulation, derivationally related form, telephone); (telephoner, derivationally related form, telephone); (telephony, derivationally related form, telephone); (telephone call, derivationally related form, telephone)

What is the entity name of t? Select one from the list:

call <#bau><#ri><#ci><#ux>
telephone <#bau><#ya><#bcq><#rm>
cell phone <#bau><#ri><#yz><#bbp>
call in <#bau><#ya><#bcq><#cg>
telephonist <#bau><#ri><#bfy><#nx>
telephony <#bau><#bn><#yz><#s>
telephone call <#bau><#bdp><#nu><#tn>
telephone set <#bau><#bn><#ci><#to>
dial <#bau><#ya><#gx><#bma>
telecommunicate <#bau><#ya><#tp><#cs>
telephoner <#bau><#vq><#va><#dd>
netmail <#bau><#ri><#ft><#rt>
hold the line <#bau><#ya><#bw><#bcy>
telex <#bau><#ya><#gx><#wp>
telefax <#bau><#ya><#gx><#cs>
wire <#bau><#ya><#gx><#mc>
radiotelephonic <#bau><#ya><#bcq><#ff>
trunk call <#bau><#bn><#nu><#kk>
sampling frequency <#bau><#jp><#gx><#vw>
sampling <#bau><#jp><#bcq><#vw>

Results Assistant: call

Table 12: Prompt Template for Knowledge Graph Completion.