Random Entity Quantization for Parameter-Efficient Compositional Knowledge Graph Representation

Jiaang Li¹, Quan Wang², Yi Liu, Licheng Zhang¹, Zhendong Mao¹
¹University of Science and Technology of China
²MOE Key Laboratory of Trustworthy Distributed Computing and Service, Beijing University of Posts and Telecommunications
³State Key Laboratory of Communication Content Cognition
{jali,zlczlc}@mail.ustc.edu.cn, wangquan@bupt.edu.cn, gavin1332@gmail.com, zdmao@ustc.edu.cn

Abstract

Representation Learning on Knowledge Graphs (KGs) is essential for downstream tasks. The dominant approach, KG Embedding (KGE), represents entities with independent vectors and faces the scalability challenge. Recent studies propose an alternative way for parameter efficiency, which represents entities by composing entity-corresponding codewords matched from predefined small-scale codebooks. We refer to the process of obtaining corresponding codewords of each entity as entity quantization, for which previous works have designed complicated strategies. Surprisingly, this paper shows that simple random entity quantization can achieve similar results to current strategies. We analyze this phenomenon and reveal that entity codes, the quantization outcomes for expressing entities, have higher entropy at the code level and Jaccard distance at the code-word level under random entity quantization. Therefore, different entities become more easily distinguished, facilitating effective KG representation. The above results show that current quantization strategies are not critical for KG representation, and there is still room for improvement in entity distinguishability beyond current strategies. The code to reproduce our results is available here.

1 Introduction

Knowledge Graphs (KGs) comprise (head entity, relation, tail entity) triplets. They are crucial external knowledge sources for various natural language processing tasks (Hu et al., 2022; Sun et al., 2022). Learning representations on KGs is necessary for expressing complex semantics and supporting downstream tasks. The most dominant paradigm, KG Embedding (KGE), maps entities and relations to a vector space (Dettmers et al., 2018; Sun et al., 2019; Zhang et al., 2020). Despite the popularity, KGE models need to represent each entity with an independent vector, which leads to a linear increase in the number of parameters with the number of entities. Consequently, scalability becomes a challenge for these models, posing difficulties in their implementation and deployment (Peng et al., 2021; Ji et al., 2022), especially for large-scale KGs (Safavi and Koutra, 2020; Mahdisoltani et al., 2014).

A recently proposed parameter-efficient KG representation method uses compositional entity representations to reduce parameters (Galkin et al., 2022; Chen et al., 2023). Instead of learning separate representations like KGE, it represents entities by composing their matched codewords from predefined codebooks through an encoder, and requires fewer parameters since codewords are much fewer than entities. See Fig 1 for an illustration of this method. We refer to the process of obtaining corresponding codewords to each entity as entity quantization due to its similarity to vector quantization (van den Oord et al., 2017). Specifically, existing methods construct two codebooks, in which codewords are the entire relation set and a selection (set to 1) with optional weights or not (set to 0).

Figure 1: The process of parameter-efficient compositional KG representation. Entities are quantized to entity codes, which are encoded to represent entities. Each dimension of an entity code stands for a codeword, and indicates whether current entity matches this codeword (set to 1) with optional weights or not (set to 0).

Corresponding author: Quan Wang
tive subset of entities, i.e., anchors, respectively. From these two codebooks, each entity matches two groups of codewords: connected relations and anchors nearby (Galkin et al., 2022) or with similar adjacent relations (Chen et al., 2023). Chen et al. (2023) also regards matching degrees as codeword weights for more expressiveness. Matched results are denoted as entity codes, including matched codewords and optional weights. A subsequent encoder uses entity codes to compose corresponding codewords and generate entity representations. This approach performs closely to KGE with fewer parameters, making KG training and deployment more efficient.

The key to entity quantization lies in two steps: (1) codebook construction and (2) codeword matching. Previous studies have dedicated their efforts to designing quantization strategies, which include selecting proper KG elements to construct codebooks and measuring the connectivity between codewords and entities to match them. We conduct experiments to randomize these strategies from shallow to deep. Surprisingly, we find that random entity quantization approaches can achieve similar or even better results.

We design several model variants for experiments. First, to investigate the effectiveness of matching codewords with connectivity, we randomize the codeword matching step by randomly selecting codewords as matched results of entities. Moreover, we set codeword weights randomly or equally for (Chen et al., 2023) to verify whether designed weights from matching are critical. Finally, to explore whether mapping codewords to actual elements in the KG is critical, we randomly construct codebooks with codewords that have no actual meaning. We adopt random codeword matching to the randomly constructed codebook, to provide a fully random entity quantization. Counter-intuitively, empirical results show that the above operations achieve similar results compared to complicated quantization strategies and may even improve the model performance.

Moreover, we have verified that random entity quantization can better distinguish entities than current quantization strategies, which leads to more expressive KG representations (Zhang et al., 2022). Under the strategies designed by previous works, different entities could match the same codewords, making their code similar or identical. In contrast, random quantization leads to a lower possibility of matching same codewords and distributes entity codes more uniformly across a wide range. We prove this claim by analyzing the properties of entity codes. At the code level, we consider entity code as a whole and treat it as one sample of a random variable. The entropy of this variable can be derived from its distribution across all entity codes. We prove that random entity quantization has higher entropy and maximizes it with high probability, thus producing more diverse and unique entity codes. At the codeword level, each entity code indicates a set of matched codewords. We analyze the Jaccard distance between different sets and find that it is significantly increased by random entity quantization. As a result, different entities will have a more obvious dissimilarity when randomly quantized, making them easier to distinguish.

In summary, the contributions of our work are two-fold: (1) We demonstrate through comprehensive experiments that random entity quantization approaches perform similarly or even better than previously designed quantization strategies. (2) We analyze that this surprising performance is because random entity quantization has better entity distinguishability, by showing its produced entity codes have higher entropy and Jaccard distance. These results suggest that current complicated quantization strategies are not critical for the model performance, and there is potential for entity quantization approaches to increase entity distinguishability beyond current strategies.

2 Preliminaries

2.1 Knowledge Graph Representation

A knowledge graph $G \subseteq E \times R \times E$ is composed of entity-relation-entity triplets $(h, r, t)$, where $E$ is a set of entities, and $R$ is a set of relations. Each triplet indicates a relation $r \in R$ between two entities $h, t \in E$, where $h$ is the head entity, and $t$ is the tail entity. The goal of knowledge graph representation is to learn a vector representation $e_i$ for each entity $e_i \in E$, and $r_j$ for relation $r_j \in R$.

2.2 Compositional KG Representation

Compositional knowledge graph representation methods compose codewords from small-scale codebooks to represent entities. These methods obtain codewords for each entity by constructing codebooks and matching codewords. We refer to these two steps as entity quantization. The matched codewords are encoded to represent each entity.
This section presents two methods of this kind, i.e., NodePiece (Galkin et al., 2022) and EARL (Chen et al., 2023). We first introduce the definition of entity quantization and how these two methods represent entities with it. After that, we introduce how they are trained. Our subsequent experiments are based on these two methods.

2.2.1 Entity Quantization
We first formally define the entity quantization process. Existing entity quantization strategies construct a relation codebook \( B_r = \{r_1, \ldots, r_m\} \) and an anchor codebook \( B_a = \{a_1, \ldots, a_n\} \). The codewords \((r_1, \ldots, r_m)\) are all \( m \) relations in \( \mathcal{R} \) and \((a_1, \ldots, a_n)\) are \( n \) anchors selected from all entities in \( \mathcal{E} \) with certain strategies. After adding reverse edges to KG, each entity \( e_i \in \mathcal{E} \) matches \( s_i = \min(d_i, d)^1 \) unique relations from all its \( d_i \) connected relations in \( B_r \), and employs anchor-matching strategies to match \( k \) anchors from \( B_a \). Its matched codewords are denoted as a set \( W_i = \{r^i_1, \ldots, r^i_{s_i}, a^i_1, \ldots, a^i_{k_i}\} \). Each entity \( e_i \) will get its entity code \( c_i \) to represent \( W_i \), which is a \((m + n)\)-dimensional vector that is zero except for the \((s_i + k_i)\) dimensions representing matched codewords. Values in these dimensions are set to 1 or optional codeword weights if provided.

Then, we provide the detailed quantization process of both models.

**NodePiece**  
NodePiece uses metrics such as Personalized PageRank (Page et al., 1999) to pick some entities as codewords in \( B_a \). Each \( e_i \in \mathcal{E} \) matches nearest anchors from \( B_a \) as \( \{a^i_1, \ldots, a^i_{k_i}\} \).

**EARL**  
EARL constructs \( B_a \) with 10% sampled entities. Each entity \( e_i \in \mathcal{E} \) matches anchors which have the most similar connected relations. Matched codewords are assigned to designed weights. Weights of \( r \in \{r^i_1, \ldots, r^i_{s_i}\} \) are based on its connection count with \( e_i \), and weights of each \( a \in \{a^i_1, \ldots, a^i_{k_i}\} \) are based on the similarity between connected relation sets of \( e_i \) and \( a \).

After quantization, codewords in \( W_i \) are composed by an encoder to output entity representation \( e_i \). The encoders of NodePiece and EARL are based on MLP and CompGCN (Vashishth et al., 2020), respectively.

2.2.2 Model Training
Here we introduce how to train both models. For each triplet \((h, r, t)\), representations of \( h \) and \( t \)

\[ 1 \]
methods we use are generally divided into three types: (1) Randomly match entities to relations or anchors in codebooks. (2) Randomly or equally set codeword weights. (3) Randomly construct codebooks, where codewords do not refer to natural elements in the KG. We will discuss the details of these model variants in Section 4.

3.2 Datasets
We use three knowledge graph completion datasets in total. We employ FB15k-237 (Toutanova et al., 2015) and WN18RR (Dettmers et al., 2018) to demonstrate the effectiveness of random entity quantization and extend our conclusions to a larger scale KG, CoDEx-L (Safavi and Koutra, 2020). FB15k-237 is based on Freebase (Bollacker et al., 2008), a knowledge base containing vast factual information. WN18RR has derived from Wordnet (Miller, 1995), specifically designed to capture semantic relations between words. CoDEx-L is the largest version of recently proposed CoDEx datasets, which improve upon existing knowledge graph completion benchmarks in content scope and difficulty level. For consistency with compared methods, we exclude test triples that involve entities not present in the corresponding training set. Table 1 presents the statistics of these datasets. We also experiment with inductive relation prediction datasets (Teru et al., 2020) (details in Appendix C).

<table>
<thead>
<tr>
<th>Datasets</th>
<th>FB15k-237</th>
<th>WN18RR</th>
<th>CoDEx-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Entity</td>
<td>14,505</td>
<td>40,559</td>
<td>77,951</td>
</tr>
<tr>
<td>#Relation</td>
<td>237</td>
<td>11</td>
<td>69</td>
</tr>
<tr>
<td>#Train</td>
<td>272,115</td>
<td>86,835</td>
<td>551,193</td>
</tr>
<tr>
<td>#Valid</td>
<td>17,526</td>
<td>2,824</td>
<td>30,622</td>
</tr>
<tr>
<td>#Test</td>
<td>20,438</td>
<td>2,924</td>
<td>30,622</td>
</tr>
</tbody>
</table>

Table 1: Statistics of the benchmark datasets, including the number of entities, relations, training triples, validation triples, and test triples.

3.3 Training and Evaluation Protocols

Training. Our experiments are based on the official implementation of NodePiece and EARL. We use the same loss functions and follow their hyperparameter setting for corresponding model variants. More details are provided in Appendix A.

Evaluation. We generate candidate triplets by substituting either \( h \) or \( t \) with candidate entities for each triplet \((h, r, t)\) in the test sets. The triples are then sorted in descending order based on their scores. We apply the filtered setting (Bordes et al., 2013) to exclude other valid candidate triplets from ranking. To assess the performance of the models, we report the mean reciprocal rank (MRR) and Hits@10. Higher MRR/Hits@10 indicates better performance. Additionally, we evaluate the efficiency of the models using \( \text{Eff} = \text{MRR}/\#P \), where \( \#P \) represents the number of parameters. The results of NodePiece and EARL are from their original papers.

4 Random Entity Quantization
This section details the random variants we design and their experimental results. We design model variants by randomizing different steps of existing entity quantization strategies, including codeword matching and codebook construction. We find that these variants achieve similar performance to existing quantization strategies. These results suggest that the current entity quantization strategies are not critical for model performance.

4.1 Random Codeword Matching
We first randomize the codeword matching step, which includes strategies for (1) matching each entity to the corresponding codewords and (2) assigning weights to the matched codewords.

4.1.1 Matching Strategy
We randomize the codeword matching strategies to investigate whether current connectivity-based strategies are critical to the model performance. We design model variants by randomizing current methods’ relation-matching or anchor-matching strategies and keep other settings unchanged. Specifically, with the relation codebook \( B_r \) and the anchor codebook \( B_a \), we have the following model variants for both models.

- +RSR: Each entity \( e_i \in E \) Randomly Selects \( s_i \) Relations (RSR) from \( B_r \) and matches \( k \) anchors from \( B_a \) with the anchor-matching strategy designed by the current model, as matched codewords \( W_i \).

- +RSA: \( e_i \) Randomly Selects \( k \) Anchors (RSA) from \( B_a \), and matches \( s_i \) relations from \( B_r \) with current relation-matching strategy, as \( W_i \).

- +RSR+RSA: \( e_i \) randomly selects \( s_i \) relations from \( B_r \), and randomly selects \( k \) anchors from \( B_a \), as \( W_i \).
FB15k-237  WN18RR

<table>
<thead>
<tr>
<th></th>
<th>FB15k-237</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MRR Hits@10</td>
<td>MRR Hits@10</td>
<td></td>
</tr>
<tr>
<td>EARL</td>
<td>0.310</td>
<td>0.501</td>
<td>0.440</td>
</tr>
<tr>
<td>+RSR</td>
<td>0.306</td>
<td>0.500</td>
<td>0.439</td>
</tr>
<tr>
<td>+RSA</td>
<td>0.311</td>
<td>0.506</td>
<td>0.438</td>
</tr>
<tr>
<td>+RSR+RSA</td>
<td>0.308</td>
<td>0.502</td>
<td>0.442</td>
</tr>
<tr>
<td>NodePiece</td>
<td>0.256</td>
<td>0.420</td>
<td>0.403</td>
</tr>
<tr>
<td>+RSR</td>
<td>0.254</td>
<td>0.417</td>
<td>0.403</td>
</tr>
<tr>
<td>+RSA</td>
<td>0.258</td>
<td>0.423</td>
<td>0.419</td>
</tr>
<tr>
<td>+RSR+RSA</td>
<td>0.263</td>
<td>0.425</td>
<td>0.425</td>
</tr>
</tbody>
</table>

Table 2: Results for parameter-efficient compositional KG representation methods with randomly selected relations (RSR) or randomly selected anchors (RSA).

For all variants, we still assign codewords in $W_i = \{r_1, \ldots, r_s, a_1, \ldots, a_k\}$ with current connectivity-based weights, if required.

Table 2 shows the performance of original models and their respective variants. Surprisingly, randomizing codeword-matching and relation-matching does not affect the overall performance of existing methods on both datasets, whether used together or separately. The results suggest that current complicated codeword matching strategies are not critical to the model performance.

We further study the impact of randomizing the codeword matching strategy with only one codebook. We remove $B_r$ or $B_a$ respectively, and adopt different codeword matching approaches for the remaining codebook, forming following variants:

- w/o anc: Remove the anchor codebook $B_a$. Match $s_i$ relations from $B_r$ with the current relation-matching strategy as $W_i = \{r_1, \ldots, r_s\}$.
- w/o anc+RSR: Remove $B_a$. Randomly select $s_i$ relations from $B_r$ as $W_i = \{r_1, \ldots, r_s\}$.
- w/o rel: Remove the relation codebook $B_r$. Match $k$ anchors from $B_a$ with the current anchor-matching strategy as $W_i = \{a_1, \ldots, a_k\}$.
- w/o rel+RSA: Remove $B_a$. Randomly select $k$ anchors from $B_r$ as $W_i = \{a_1, \ldots, a_k\}$.

Table 3 shows that when removing one codebook, random matching codewords from the remaining codebook performs better than using current designed matching strategies. It even performs similarly to the original methods in most cases. NodePiece performs poorly on WN18RR without $B_a$, because the number of relations in this dataset is small, and only using $B_r$ sharply decreases model parameters. The above results further validate the effectiveness of random codeword matching and demonstrate its robustness with only one codebook.

4.1.2 Codeword Weights from Matching

During the matching process, EARL will further assign weights to the matched codewords based on the connectivity between entities and codewords. We conduct experiments to explore whether such weights are critical. Specifically, we design following model variants for EARL using random or equal codeword weights (RW) or equal codeword weights (UW) cases.

<table>
<thead>
<tr>
<th></th>
<th>FB15k-237</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MRR Hits@10</td>
<td>MRR Hits@10</td>
<td></td>
</tr>
<tr>
<td>EARL</td>
<td>0.310</td>
<td>0.501</td>
<td>0.440</td>
</tr>
<tr>
<td>+RW</td>
<td>0.308</td>
<td>0.498</td>
<td>0.442</td>
</tr>
<tr>
<td>+EW</td>
<td>0.308</td>
<td>0.500</td>
<td>0.437</td>
</tr>
</tbody>
</table>

Table 4: Results for parameter-efficient compositional KG representation methods with random codeword weights (RW) or equal codeword weights (UW)
Table 4 shows that either using random weights or equal weights does not significantly impact the performance. Thus, we can conclude that the connectivity-based weights are not critical for model performance. For subsequent experiments, we use equal codeword weights for simplicity.

4.2 Random Codebook Construction

We construct codebooks randomly to investigate whether it is critical to construct codebooks with specific relations or entities as codewords, like current strategies. We design a model variant for each model, which uses a randomly constructed codebook B instead of B_r and B_a. Codewords in B do not have any real meanings. Moreover, we adopt random codeword matching and equal codeword weights to such variants. We call this type of variant the fully random entity quantization (RQ) since it randomizes all parts of the entity quantization.

• +RQ: Fully Random entity QuanTization (RQ). Randomly construct a codebook whose size equals the sum of B_r and B_a: \( B = \{z_1, \ldots, z_{m+n}\} \). Each entity \( e_i \in \mathcal{E} \) randomly selects \( (s + k) \) codewords as its matched codewords \( W_i = \{z_{i1}, \ldots, z_{i(s+k)}\} \) with equal weights, where \( s = \frac{1}{|\mathcal{E}|} \sum_{i=1}^{|\mathcal{E}|} s_i \) and \(|\mathcal{E}|\) is the number of entities.

Table 5 shows variants of both models lead to similar or even better performance across all datasets with varying sizes. We will analyze this surprising observation in Section 5. Through \#P(M) and Effi, we further show that random entity quantization requires equal or even fewer parameters with higher efficiency. It does not require composing each codebook’s codewords separately, saving parameters for EALR. The above results show that constructing codebooks by KG elements is not critical to the model performance. From all variants and results above, we can conclude that random entity quantization is as effective as current complicated quantization strategies.

5 Why Random Quantization Works

This section analyzes the reasons for the surprising performance of random entity quantization. The entity codes directly affect entity representation and model performance. We analyze the entity codes produced by different entity quantization approaches to compare their ability to distinguish different entities. We find random entity quantization obtains entity codes with greater entropy at the code level and Jaccard distance at the code-word level. Thus, it can better distinguish different entities and represent KGs effectively.

5.1 Code Level Distinguishability

We analyze the ability to distinguish entities at the code level of different entity quantization approaches. Specifically, we treat the entity code \( c_i \) of each entity \( e_i \in \mathcal{E} \) as a sampling of a random variable \( X \). The entity codes of all entities represent \( |\mathcal{E}| \) samplings of \( X \). From these samplings, we get \( v \) different entity codes and their frequencies. We denote these codes as \( \{x_1, \ldots, x_v\} \) and their frequencies as \( \{f_1, \ldots, f_v\} \). We denote \( l = m + n \) as the total number of codewords, where \( m \) and \( n \) are numbers of codewords in \( B_r \) and \( B_a \). The number of all possible entity codes is \( 2^l \). We estimate the probability distribution of \( X \) on all codes based on the relative frequency of different entity codes in the sampling results, and then derive its entropy:

$$ H(X) = - \sum_{i=1}^{2^l} P(x_i) \cdot \log_2 P(x_i), \quad (1) $$

where \( P(x_i) \) is the relative frequency of \( x_i \):

$$ P(x_i) = \begin{cases} \frac{f_i}{|\mathcal{E}|} & \text{if } i = 1, \ldots, v, \\ 0 & \text{if } i = v + 1, \ldots, 2^l. \end{cases} $$
We use the entropy $H(X)$ in eq. 1 to measure the diversity of entity codes. A higher entropy means more diverse entity codes, thus indicating the better entity-distinguish ability of the quantization approach. In this sense, this entropy can reflect entity distinguishability at the code level.

We use equation 1 to derive the entity code entropy produced by random and previously designed quantization strategies. The results are listed in Table 6. The entropy of random quantization is averaged in 100 runs with different seeds, while the entropy of previous strategies is deterministic and does not change in different runs. Empirically, we demonstrate that random quantization achieves higher entropy, producing more diverse entity codes and enabling easier distinguishability between different entities.

We confirm that higher entity code entropy brings better model performance through extensive experiments. Specifically, after random entity quantization, we randomly select a subset of entity codes and set them to be identical, to obtain entity codes with different entropy values. Figure 2 shows the comparison experiments across these entity codes with EARL and NodePiece. As the entropy rises, the model performance gradually increases and eventually saturates. EARL performs better than NodePiece across varying entropy, as its GNN encoder involves the entity neighborhood and can better distinguish entities. From above, more diverse entity codes with higher entropy benefit the model performance, which aligns with our claims.

In addition, the entity code entropy is maximized when all entity codes are unique. The probability of random entity quantization to produce $|\mathcal{E}|$ unique entity codes is close to 1, as shown in Theorem 1. The detailed proof is in Appendix B. This theorem shows that random entity quantization expresses entities uniquely with high probability, thus distinguishing entities well.

**Theorem 1.** The probability of random entity quantization to produce $|\mathcal{E}|$ unique entity codes is $P = \prod_{i=0}^{2^{|E|}-1} \frac{2^{|E|}}{2^{|E|}}$, which approaches 1 when $2^{|E|} \gg |\mathcal{E}|$.

From above, we demonstrate random entity quantization expresses diverse entity codes and clearly distinguishes entities at the code level.

### 5.2 Codeword Level Distinguishability

We further analyze the ability to distinguish entities at the codeword level of different entity quantization approaches. Specifically, for entities $e_i, e_j \in \mathcal{E}$ with entity codes $c_i$ and $c_j$, their corresponding sets of matched codewords are $W_i = \{r_1^i, \ldots, r_k^i, a_1^i, \ldots, a_k^i\}$ and $W_j = \{r_1^j, \ldots, r_k^j, a_1^j, \ldots, a_k^j\}$. The Jaccard distance between $c_i$ and $c_j$ is:

$$d_{j}(c_i, c_j) = \frac{|W_i \cup W_j| - |W_i \cap W_j|}{|W_i \cup W_j|},$$

where $|\cdot|$ denotes the number of elements in a set.

We use the Jaccard distance $d_{j}(c_i, c_j)$ to measure the distinctiveness between entity codes $c_i$ and $c_j$. A larger distance means their indicated codewords are more distinct and makes entities $e_i$ and $e_j$ more easily to be distinguished. In this sense, this distance can reflect the entity distinguishability at the codeword level.

To capture the overall distinguishability among all entity codes, we propose a $k$-nearest neighbor evaluation metric based on the Jaccard distance. This evaluation assesses the average distance between each entity code and its $k$ nearest codes, denoted as $\mathcal{J}_k$. A higher $\mathcal{J}_k$ means the entity codes are more distinct. We use different values of $k$ to observe the distance among entity codes in different neighborhood ranges. The metric $\mathcal{J}_k$ is derived as:

$$\mathcal{J}_k = \frac{1}{|\mathcal{E}|} \sum_{e_i \in \mathcal{E}} \sum_{e_j \in kNN(e_i)} d_{j}(c_i, c_j),$$

where $|\mathcal{E}|$ is the number of entities. $kNN(e_i)$ is a set of $k$ entities whose codes are nearest to the
Figure 3 shows the average Jaccard distance $J_k$ between entity codes w.r.t. different numbers $k$ of nearest codes. We can see that random entity quantization achieves higher $J_k$ than current quantization strategies across the varying $k$. Thus, entity codes produced by random entity quantization are more distinct within different neighborhood ranges. Based on the above observation, random entity quantization makes different entities easier to distinguish at the codeword level.}

5.3 Discussion

We can derive from the above that, entity distinguishability is the reason why current quantization strategies based on attribute similarity don’t work better than the random approach. From Table 6 and Figure 3, it’s proved that random entity quantization distinguishes entities better in both code and codeword levels. Furthermore, in Figure 2, we show that entity quantization strategies with higher entity distinguishability tend to perform better. Therefore, it’s reasonable that random quantization works comparable to current strategies or even better.

6 Related Work

Knowledge Graph Embedding KG embedding aims at learning independent representations for entities and relations of a given KG, which benefits downstream tasks such as question answering (Hu et al., 2022), reading comprehension (Sun et al., 2022), and pre-trained language representation Wang et al., 2021c. Recent years have witnessed a growing number of KG embedding techniques being devised, including distance-based models (Bordes et al., 2013; Sun et al., 2019; Zhang et al., 2020), semantic matching models (Trouillon et al., 2016; Balažević et al., 2019), neural encoding models (Dettmers et al., 2018; Schlichtkrull et al., 2018; Wang et al., 2019), and text augmented models (Yao et al., 2019; Wang et al.). We refer readers to (Wang et al., 2017; Ji et al., 2022) for a comprehensive overview of the literature.

Parameter-Efficient KG Representation KG embedding methods face the scalability challenge. The number of parameters scales up linearly to the entity number. Several studies compress learned parameters from KG embedding models, trying to solve this issue. Incorporating knowledge distillation techniques, MulDE (Wang et al., 2021b) transfers knowledge from multiple teacher models to a student model. Expanding on this, DuADE (Zhu et al., 2022) considers the dual influence between the teacher and student and adapts the teacher to better align with the student, thus enhancing the performance of the distilled model. To directly compress existing models, Sachan (2020) discretizes the learned representation vectors for less parameter storage space, then maps the discrete vectors back to the continuous space for testing. LightKG (Wang et al., 2021a) introduces dynamic negative sampling and discretizes learned representations through vector quantization.

However, the above methods firstly need to train KG embedding models with full parameter size. Recently proposed compositional parameter-efficient KG representation models (Galkin et al., 2022; Chen et al., 2023), which are illustrated in this paper, enable a more efficient training process.

Random Features in Graph Representation In homogeneous graph learning, researchers prove that message-passing neural networks are more powerful when having randomly initialized node features (Sato et al., 2021; Abboud et al., 2021). In KGs, Zhang et al. (2022) finds that random perturbation of relations does not hurt the performance of graph convolutional networks (GCN) in KG completion. Degraeve et al. (2022) further implies
that an R-GCN with random parameters may be comparable to an original one. To the best of our knowledge, there is no existing study on representing entities with randomness in parameter-efficient compositional KG representation.

7 Conclusion

In conclusion, this paper demonstrates the effectiveness of random entity quantization in parameter-efficient compositional knowledge graph representation. We explain this surprising result by illustrating that random quantization could distinguish different entities better than current entity quantization strategies. Thus, we suggest that existing complicated entity quantization strategies are not critical for model performance, and there is still room for entity quantization approaches to improve entity distinguishability beyond these strategies.

Limitations

This paper only studies entity quantization with encoders proposed in early works, while designing more expressive encoders is also important and can improve the performance of parameter-efficient compositional knowledge graph representation. We leave this part as future work.

Acknowledgments

We would like to thank Ruichen Zheng for the discussion on measuring distinguishability, Zhengbo Wang, Haotian Zhang, Chengchao Xu for the discussion on entropy computation, and Jiahao Li for help to improve the technical writing of this paper.

This work is supported by the National Key Research and Development Program of China under Grant 2021YFF0901600, the National Science Fund for Excellent Young Scholars under Grant 62222212, and the National Natural Science Foundation of China under Grant 62376033.

References


A Training Details

We use the exact same hyperparameter settings as in the original papers of NodePiece and EARL. The results of model variants with random approaches are averaged in three runs with different seeds. We do not further tune hyperparameters for our proposed model variants. Thus, their performance may be underestimated. But so long as the model variants perform similarly to the original models, we can still make our conclusions. For NodePiece, we obtain matched anchors with its ‘shortest-path’ mode to be consistent with the original paper. For EARL, we use its released anchor set. We run our experiments with a single RTX 3090 GPU with 24GB RAM.

We use uniformly distributed random numbers in our model variants. It means that in model variants that randomly match entities to codewords, each codeword has an equal probability of being matched.

B Proof of Theorem 1

Proof. Since random entity quantization matches entities to codewords with equal probabilities, it produces all entity codes uniformly. Thus, the probability of random entity quantization to get $|\mathcal{E}|$ different entity codes is:

$$
Prob = \frac{P(2l, |\mathcal{E}|)}{(2^l)^{|\mathcal{E}|}} = \frac{2^l \cdot (2^l - 1)!}{(2^l)^{|\mathcal{E}|}} = \prod_{i=0}^{|\mathcal{E}|-1} \frac{2^l - i}{2^l}
$$

where $P(n, r)$ is the number of permutations of selecting $r$ elements from a set of $n$ elements. $l$ is the total number of codewords, and the total number of entity codes that can be produced is $2^l \gg |\mathcal{E}|$.

C Inductive Relation Prediction Results

Besides FB15K-237 and WN18RR datasets used in the main text, we further test the effectiveness of random entity quantization on inductive relation prediction, where NodePiece has shown superiority. This task requires learning from one KG, and generalizes to another KG with no shared entities for inference. We follow previous works (Zhu et al., 2021; Galkin et al., 2022; Li et al., 2023) and use the datasets proposed by Teru et al. (2020), including four versions of subsets generated from FB15k-237. We test the performance of NodePiece with the fully random entity quantization (RQ) as described in Section 4.2, on these subsets. We set the learning rate to 1e-3 and train the variant with 300 epochs on v1/v4, and 120 epochs on v2/v3. The other hyperparameters remain the same as the original method. We use the exact same evaluation protocol as in previous works (Teru et al., 2020, Zhu et al., 2021; Galkin et al., 2022; Li et al., 2023). The results are shown in Table 7.

We can see that NodePiece’s variants with random entity quantization perform as well as the original model in inductive link prediction. The results support and strengthen our claim that random entity quantization is effective, both in the transductive and inductive settings.

<table>
<thead>
<tr>
<th></th>
<th>v1</th>
<th>v2</th>
<th>v3</th>
<th>v4</th>
</tr>
</thead>
<tbody>
<tr>
<td>NodePiece</td>
<td>0.873</td>
<td>0.939</td>
<td>0.944</td>
<td>0.949</td>
</tr>
<tr>
<td>NodePiece+RQ</td>
<td>0.867</td>
<td>0.942</td>
<td>0.945</td>
<td>0.944</td>
</tr>
</tbody>
</table>

Table 7: Inductive relation prediction results (Hits@10) of NodePiece and its variant with fully random entity quantization (RQ).