Exploiting Relation-aware Attribute Representation Learning in Knowledge Graph Embedding for Numerical Reasoning

Gayeong Kim*
Sungkyunkwan University
Suwon, Republic of Korea
gayeongkim@o365.skku.edu

Sookyung Kim*
Sungkyunkwan University
Suwon, Republic of Korea
tnrud929@skku.edu

Ko Keun Kim
LG Electronics
Seoul, Republic of Korea
kokeun.kim@gmail.com

Suchan Park
LG Electronics
Seoul, Republic of Korea
sucpark0317@gmail.com

Heesoo Jung
Sungkyunkwan University
Suwon, Republic of Korea
steve305@skku.edu

Hogun Park†
Sungkyunkwan University
Suwon, Republic of Korea
hogunpark@skku.edu

ABSTRACT

Numerical reasoning is an essential task for supporting machine learning applications, such as recommendation and information retrieval. The reasoning task aims to compare two items and infer new facts (e.g., *is taller than*) by leveraging existing relational information and numerical attributes (e.g., *the height of an entity*) in knowledge graphs. However, most existing methods rely on leveraging attribute encoders or additional loss functions to predict numerical relations. Therefore, the prediction performance is often not robust in cases when attributes are sparsely observed. In this paper, we propose a **R**elation-**A**ware attribute representation learning-based **K**nowledge **G**raph **E**mbedding method for numerical reasoning tasks, which we call **RAKGE**. RAKGE incorporates a newly proposed *attribute representation learning* mechanism, which can leverage the association between relations and their corresponding numerical attributes. In addition, we introduce a robust self-supervised learning method to generate unseen positive and negative examples, thereby making our approach more reliable when numerical attributes are sparsely available. In the evaluation of three real-world datasets, our proposed model outperformed state-of-the-art methods, achieving an improvement of up to 65.1% in *Hits*@1 and up to 52.6% in *MRR* compared to the best competitor. Our implementation code is available at https://github.com/learndatalab/RAKGE.

CCS CONCEPTS

• **Computing methodologies** → **Knowledge representation and reasoning**.

KEYWORDS

Numerical Reasoning; Knowledge Graph; Contrastive Learning

*Equal contribution
†Corresponding author

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599338

ACM Reference Format:
Gayeong Kim, Sookyung Kim, Ko Keun Kim, Suchan Park, Heesoo Jung, and Hogun Park. 2023. Exploiting Relation-aware Attribute Representation Learning in Knowledge Graph Embedding for Numerical Reasoning. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3580305.3599338

1 INTRODUCTION

Numerical reasoning is a form of simulated thinking that involves numerical attributes (e.g., *height*) and operations such as sorting and comparison. While it has been addressed in many Natural Language Processing (NLP) applications [1], it has received relatively less attention in the context of Knowledge Graphs (KGs). Numerical reasoning over KGs can uncover missing numerical relationships or orders among entities, but the sparsity of numerical attributes in KGs makes this task challenging. For instance, knowledge graphs about customers in a company often lack information on some customers' ages, revenues, and expenses. However, the lack of effective methods for processing KGs with sparsely observed numerical attributes hinders the improvement of information services, such as recommendation systems and question-answering tasks.

Knowledge Graph Embedding (KGE) has been proposed as a method for encoding entities and relationships of knowledge graphs (KGs) into low-dimensional vectors while preserving the inherent relational properties between the entities. Traditional KGE models, such as TransE [5] and TransR [36], learn vector representations of head entities, relations, and tail entities with the aim of maximizing the performance of link prediction, i.e., a single-hop reasoning task. However, these models do not exploit numerical attributes, which limits the performance of numerical reasoning tasks. Recent efforts have been made to incorporate attributes into KGE models, such as LiteralE [15], TransEA [30], and MT-KGNN [20], to solve this issue. For instance, in LiteralE [15], attribute values are represented as a literal vector, which is combined with the entity embedding through a gating function to form a single embedding vector. The combined entity representation is then fed into the score functions of the corresponding KGE models, such as TransE and TransR, allowing for the inclusion of richer information in the entity embeddings for numerical reasoning tasks. Other existing methods [20, 30] are also limited to introducing simple attribute encoders or additional regression loss functions for the reasoning tasks.

Meanwhile, Graph Neural Networks (GNNs) have been leveraged in KGE problems [17, 18, 24] in order to learn higher-order connectivity. They aggregate information from neighboring nodes to learn entity representations, and the score functions of the existing KGE models [5, 11, 12] are adopted to model the relations among entities. Although GNN-based KGE models have demonstrated promising results for graph completion tasks, recent studies (e.g., LTE [37]) have suggested that many of these models are overly complex and redundant. The study found that a simple linear transformation applied to existing KGE models could outperform or produce results equivalent to GNN-based models.

To overcome the problems of previous approaches, we propose a new **R**elation-**A**ware attribute representation learning-based **K**nowledge **G**raph **E**mbedding method for numerical reasoning tasks, namely **RAKGE**. Our proposed **RAKGE** method can leverage the association between entity relations and numerical attributes by obtaining the relevance (or importance) of each attribute using a multi-head attention mechanism. Therefore, our model can obtain more robust attribute representations, even when the attributes are sparsely available. For example, if the *height* attribute is partially or completely missing for head/tail entities in a height comparison task, our algorithm can assign more importance to the existing *weight* and *age* attributes to obtain representations. The attribute representations are then combined with the entity embeddings using a gating function.

Furthermore, we propose a novel contrastive learning method to generate more appropriate positive/negative examples for numerical reasoning. Contrastive learning, a representative method of self-supervised learning, aims to maximize the agreement between input data and its augmentation while simultaneously repelling other data samples, thereby providing effective supervision. Although contrastive learning has been successful in various domains [9, 16, 32, 33], it remains challenging to directly apply the augmentation-based contrastive learning in the numerical reasoning tasks. Because numeric attributes are continuous, randomly perturbing numeric values can lead to severe information loss. To capture the continuous characteristics of numeric attributes, we devised a new mixup-based augmentation method that produces true hard positive/negative samples for numerical reasoning.

The contributions of this paper can be summarized as follows:

- We propose a *relation-aware attribute representation learning* approach that captures the relevance of numeric attributes with respect to each relation.
- We propose a novel contrastive learning method that generates true but hard positive/negative samples for numerical reasoning.
- Our proposed model demonstrates consistently and significantly better performance than recent KGE models, including LiteralE and message passing-based models, on three real-world datasets in all evaluation measures.

2 RELATED WORK

2.1 Knowledge Graph Embedding

In recent years, various Knowledge Graph Embedding (KGE) methods have been proposed to capture the latent representations of entities and relations in knowledge graphs (KGs). TransE [5] is a translation-based method that optimizes the equation $head + relation \approx tail$, leading to several extensions such as TransH [29] and TransR [36]. DistMult [12] proposes a tri-linear dot product of head, relation, and tail as a score function, and ComplEx [22] extends DistMult by using complex-valued embeddings to cover asymmetric relations. ConvE [11] employs convolutional layers to extract features from the head entity and relation. Additionally, there have been various efforts to embed KGs into hyperbolic space to model hierarchical relations, such as ConE [2], MuRP [3], and AttH [8]. However, these methods often suffer from the incompleteness of KGs as they depend solely on structural information.

2.2 Knowledge Graph Embedding with Numerical Attributes

To address the incompleteness issue in KGs, several KGE models aim to incorporate side information of entities, such as numerical attributes (e.g., *height, age, and weight*). For example, MT-KGNN [20] and TransEA [30] are multi-task learning methods to predict numeric attribute values and entity/relation embeddings. KBLRN [13] uses numerical values to learn a KGE model by considering the differences between the numerical values of entity pairs. Recently, LiteralE [15] introduces a learnable gating function to integrate entity embedding and numeric attributes. However, these methods are often not robust when attributes are sparsely observed, and they do not consider relational information when incorporating attributes in KGE. To effectively exploit the relation information, we propose an encoding method that considers relation-aware attribute relevance.

2.3 GNN-based Knowledge Graph Embedding

To model the high-order connectivity in knowledge graphs, various extensions of graph neural networks (GNNs) have been proposed for KGE problems. R-GCN [17] includes a relation-specific transformation matrix in the neighborhood aggregation step, and WGCN [18] has relations with learnable scalar weights that are multiplied by the incoming neighborhood message. CompGCN [24] jointly embeds nodes and relations and provides various composition operations between entities and relations. However, these models do not consider numerical values, making it difficult to apply them to numerical reasoning tasks.

Furthermore, several self-supervised GNN-based models have employed contrastive learning (CL) to learn entity/graph representation. HeCO [28] introduced co-contrastive learning by suggesting two views: the network-schema and meta-path views to capture both local and global features of the entity. SLiCE [27] learns contextual subgraph representations by allowing each close node to attract each other. SimGCL [33] is an augmentation-free CL method that adds uniform noises to the entity representation. However, none of the above methods considers continuous values in augmentation; therefore, they cannot guarantee the effectiveness of their proposed method for numerical reasoning tasks. In contrast, our newly proposed CL method can generate guaranteed but hard positive/negative samples using a mixup-based augmentation technique, which compensates for the sparse observations in the numerical value space.

3 PRELIMINARIES

3.1 Problem Definition

Let us denote the set of entities as $\mathcal{E}$ and the set of relations as $\mathcal{R}$. A knowledge graph $\mathcal{G}$ can be defined as $\{(h,r,t)|\, h,t \in \mathcal{E}, r \in \mathcal{R}\}$. Let $\mathbf{X} \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{M}|}$ be an entity-numeric value matrix for all entities in $\mathcal{G}$, where $\mathcal{M}$ is the set of numeric attribute fields (e.g., *age*, *height*, and *debut_year*). X_{im} indicates the m-th numeric value of entity i.

Numerical reasoning task. We formalize the numerical reasoning [15] as a link prediction task in an attributed KG. Link prediction maps each triple (h,r,t) to a score by leveraging the KGE models, where a higher score indicates that the triple is more likely to be true. Our goal is to learn a function $\psi : \mathcal{E}\times\mathcal{R}\times\mathcal{E} \rightarrow \mathbb{R}$ that predicts the plausibility of possible triples. In this paper, we define a ***numerical reasoning task*** as comparing two entities concerning a single aspect, such as *height* or *weight*. The comparison of entities based on numerical attributes has been less explored, and we constructed and evaluated new large-scale datasets with varying sparsity levels for this purpose.

3.2 Knowledge Graph Embedding Methods

As in previous studies, our model employs existing KGE methods to predict the plausibility of a triple. In this paper, we introduce TransE [5] and Order-embedding [26]. Let $\mathbf{e_h}, \mathbf{e_t},$, and $\mathbf{e_r}$ denote the embeddings of entities h and t and relation r, respectively. The plausibility score function for a triple $(\mathbf{e_h}, \mathbf{e_r}, \mathbf{e_t})$ is calculated as follows.

TransE. is a translation-based method that aims to optimize $\mathbf{e_h} + \mathbf{e_r} \approx \mathbf{e_t}$.

$$score(\mathbf{e_h}, \mathbf{e_r}, \mathbf{e_t}) = \epsilon - \|\mathbf{e_h} + \mathbf{e_r} - \mathbf{e_t}\|_{1/2}\,, \tag{1}$$

where ϵ is a hyperparameter and $\|\cdot\|$ can be L1 or L2 distance.

Order-embedding. learns ordered representations to model the partial order structure in the hierarchy. The main idea of Order-embedding is to embed elements in the higher position has smaller coordinates than those in the lower position. However, in the numerical reasoning task, it is more intuitive that the lower position (e.g., 160 cm) has smaller coordinates than the higher position (e.g., 170cm). Therefore, we reformulate this idea in terms of the score function of KGE as follows:

$$score(\mathbf{e_h}, \mathbf{e_t}) = \epsilon - \|max(0, \mathbf{e_t} - \mathbf{e_h})\|^2, \tag{2}$$

where ϵ is a hyperparameter.

4 METHODOLOGY

In this section, we first introduce the overall structure of RAKGE, which enhances numerical reasoning through *relation-aware attribute representation learning*. As shown in Figure 1(a), RAKGE consists of three main components: a Relation-Aware Encoder (RA Encoder), a positive/negative sample generator, and a score function. The first component, the RA Encoder, adaptively encodes the numeric attributes of entities based on the relation they are associated with. The second component, the positive/negative sample generator, then produces synthetic but true positive/negative samples based on the encoded entity embeddings. Finally, our score function, designed specifically for numerical reasoning, calculates the plausibility score of the target triple.

4.1 Relation-Aware (RA) Encoder

The components of the RA Encoder are illustrated in Figure 1(b). The RA Encoder is composed of three layers: numeric value embedding, relation-aware attention, and gating layer. The numeric value embedding layer converts each numeric value of the entity into a vector representation. The relation-aware attention layer generates an attribute vector that encodes numeric information by considering a specific relation. Finally, the gating layer combines the relation-aware attribute vector and entity embedding.

Numeric Value Embedding Layer. As described in our problem definition, one entity has multiple numeric values for different attributes (e.g., 38 years old, 5.87 ft/in, and 2008 year), and each attribute field (e.g., *age, height, debut_year*) has its own unique distribution. Hence, it is crucial to develop a method that incorporates the context of the attribute field while preserving the properties of the scalars (e.g., magnitude and quantity). To achieve this, we transform each attribute field into an embedding space using the field embedding method [14, 19].

For observable scalars, we map them into a low-dimensional embedding space using a learnable embedding matrix. For missing scalars, instead of using a fixed value, such as zero, we use a learnable special missing value embedding to preserve the magnitude information and prevent the task from being threatened. The representation of both missing and observable scalars is given by:

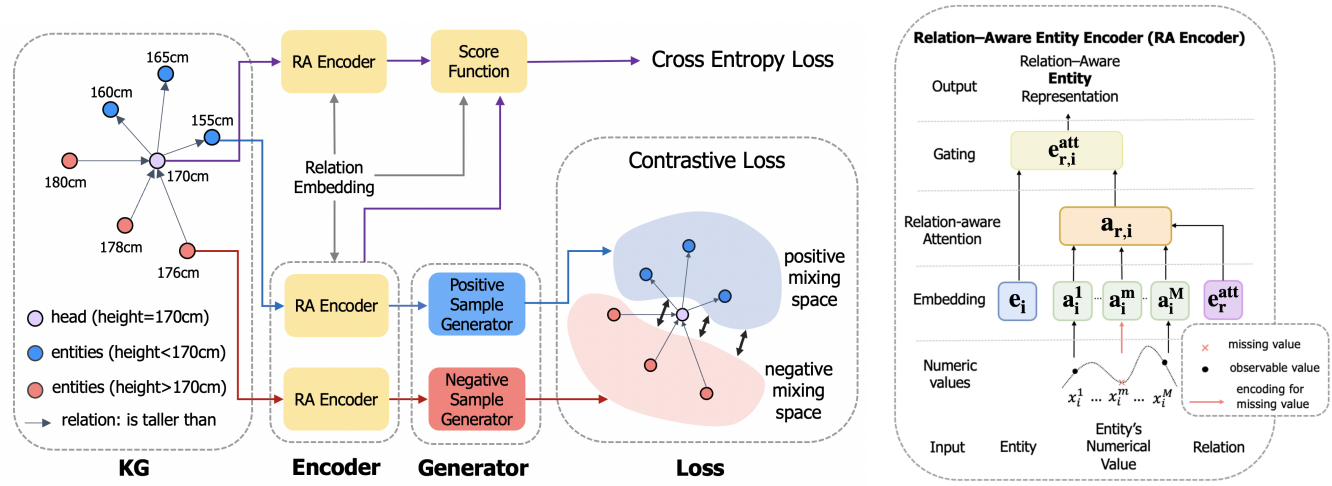
$$\mathbf{a_i^m} = \begin{cases} \mathbf{W_x^m v_m} & \text{If } X_{im} \text{ is missing,} \\ (\mathbf{w_m} X_{im}) \odot \mathbf{v_m} & \text{Otherwise,} \end{cases} \tag{3}$$

where $\odot$ represents point-wise multiplication. $\mathbf{v_m} \in \mathbb{R}^{d_{att}}$ is the embedding vector of the numeric attribute field m, where d_{att} represents the attribute embedding dimension. X_{im} is the corresponding numeric value of the entity i. $\mathbf{w_m} \in \mathbb{R}^{d_{att}}$ and $\mathbf{W_x^m} \in \mathbb{R}^{d_{att}\times d_{att}}$ are linear transformations that learn the context of each value and attribute. We denote $\mathbf{a_i^m} \in \mathbb{R}^{d_{att}}$ as the m-th field attribute embedding of the entity i.

Relation-aware Attention Layer. The main contribution of our model is the effective learning of numerical attributes $\mathbf{v_m}$ by considering the relations. For example, when predicting (*Ryan Reynolds, is_taller_than, Blake Lively*), the *height* and *weight* attributes should be given more consideration than the *salary* attribute. We propose a *relation-aware attention mechanism* using the multi-head self-attention [25] to capture the importance of numeric attributes according to each relation.

First, the relation embedding $\mathbf{e_r} \in \mathbb{R}^{d_{emb}}$ is projected onto the attribute embedding space using a single layer perceptron f_{att}. Thereafter, the *relation-aware attention layer* takes the projected relation embedding $\mathbf{e_r^{att}} \in \mathbb{R}^{d_{att}}$ as the query and attribute embeddings $\mathbf{a_i^1}, \cdots, \mathbf{a_i^{|\mathcal{M}|}} \in \mathbb{R}^{d_{att}}$ as the key and value. Under a specific attention head k, we map $\mathbf{e_r^{att}}$ and $\mathbf{a_i^1}, \cdots, \mathbf{a_i^{|\mathcal{M}|}}$ onto smaller subspaces and capture the attention score. The attention weight $\alpha_{r,i,m}^{(k)}$ of the attribute embedding $\mathbf{a_i^m}$ is expressed as follows:

$$\mathbf{e_r^{att}} = f_{att}(\mathbf{e_r}), \tag{4}$$



(a) RAKGE (b) RA Encoder

Figure 1: (a) Overview of our proposed model (RAKGE). For relation-aware embedding, all entities are encoded by the RA Encoder, and then all of these are utilized to obtain the triplet score. Generators exist for the numerical reasoning task. By using generators, we can generate both hard positive and hard negative samples. (b) Structure of the Relation-Aware (RA) Encoder. In the RA attention layer, we calculate the importance of numerical values concerning each relation. The output is a newly generated entity's representation.

$$s^{(k)}(\mathbf{e}_\mathbf{r}^{\mathbf{att}}, \mathbf{a}_\mathbf{i}^{\mathbf{m}}) =< \mathbf{W}_\mathbf{Q}^{(\mathbf{k})}\mathbf{e}_\mathbf{r}^{\mathbf{att}}, \mathbf{W}_\mathbf{K}^{(\mathbf{k})}\mathbf{a}_\mathbf{i}^{\mathbf{m}} >, \tag{5}$$

$$\alpha_{r,i,m}^{(k)} = \frac{\exp(s^{(k)}(\mathbf{e}_\mathbf{r}^{\mathbf{att}}, \mathbf{a}_\mathbf{i}^{\mathbf{m}}))}{\sum_{n=1}^{|\mathcal{M}|}\exp(s^{(k)}(\mathbf{e}_\mathbf{r}^{\mathbf{att}}, \mathbf{a}_\mathbf{i}^{\mathbf{n}}))}, \tag{6}$$

where $\mathbf{W}_\mathbf{Q}^{(\mathbf{k})}, \mathbf{W}_\mathbf{K}^{(\mathbf{k})} \in \mathbb{R}^{d_{sub} \times d_{att}}$ are linear transformations that project relation embedding $\mathbf{e}_\mathbf{r}^{\mathbf{att}}$ and attribute embedding $\mathbf{a}_\mathbf{i}^{\mathbf{m}}$ into low-dimensional subspaces, respectively. Next, $s^{(k)}$ outputs the attention score between a relation and an attribute by the inner product $< \cdot, \cdot >$ for each attention head k.

We repeat this mechanism for all attribute fields $m = 1, 2, \cdots, |\mathcal{M}|$. The attention weight is then normalized using equation Eq. (6) and multiplied by the corresponding attribute embedding as follows:

$$\mathbf{a}_{\mathbf{r,i}}^{(\mathbf{k})} = \sum_{m=1}^{|\mathcal{M}|} \alpha_{r,i,m}^{(k)}(\mathbf{W}_\mathbf{V}^{(\mathbf{k})}\mathbf{a}_\mathbf{i}^{\mathbf{m}}), \tag{7}$$

where $\mathbf{W}_\mathbf{V}^{(\mathbf{k})} \in \mathbb{R}^{d_{sub} \times d_{att}}$ is a linear transformation matrix that projects attribute embedding $\mathbf{a}_\mathbf{i}^{\mathbf{m}}$ into low-dimensional subspaces. Normalization captures the importance of each numeric attribute from the given relation. We repeat this step for all attention heads $k = 1, 2, \cdots, K$, where K is the number of multi-head attentions. The final attribute embedding for entity i related to r is as follows:

$$\mathbf{a}_{\mathbf{r,i}} = \mathbf{a}_{\mathbf{r,i}}^{(\mathbf{1})} \oplus \mathbf{a}_{\mathbf{r,i}}^{(\mathbf{2})} \oplus \cdots \oplus \mathbf{a}_{\mathbf{r,i}}^{(\mathbf{K})}, \tag{8}$$

where $\oplus$ is a concatenation operator. We consider $\mathbf{a}_{\mathbf{r,i}} \in \mathbb{R}^{d_{att}}$ as the attribute vector of entity i, which is aware of relation r.

Gating Layer. To balance the relation-aware attribute vector $\mathbf{a}_{\mathbf{r,i}}$ and an entity embedding $\mathbf{e}_\mathbf{i} \in \mathbb{R}^{d_{emb}}$, a gating layer is introduced to merge them as follows:

$$\mathbf{g} = \sigma(\mathbf{W}_{\mathbf{ge}}\mathbf{e}_\mathbf{i} + \mathbf{W}_{\mathbf{ga}}\mathbf{a}_{\mathbf{r,i}} + \mathbf{b}), \tag{9}$$

$$\mathbf{u} = \tau(\mathbf{W}_\mathbf{u}(\mathbf{e}_\mathbf{i} \oplus \mathbf{a}_{\mathbf{r,i}})), \tag{10}$$

$$\mathbf{e}_{\mathbf{r,i}}^{\mathbf{att}} = \mathbf{g} \odot \mathbf{u} + (1 - \mathbf{g}) \odot \mathbf{e}_\mathbf{i}, \tag{11}$$

where $\mathbf{W}_\mathbf{u} \in \mathbb{R}^{d_{emb} \times (d_{att} + d_{emb})}$, $\mathbf{W}_{\mathbf{ge}} \in \mathbb{R}^{d_{emb} \times d_{emb}}$, and $\mathbf{W}_{\mathbf{ga}} \in \mathbb{R}^{d_{emb} \times d_{att}}$ are linear transformations and $\mathbf{b}$ is a bias. σ is a sigmoid function, and τ is hyperbolic tangent function. We name $\mathbf{e}_{\mathbf{r,i}}^{\mathbf{att}} \in \mathbb{R}^{d_{emb}}$ as the attribute-enriched vector of entity i. To adaptively filter numeric information, we adopt the gating layer proposed in [15], which is a variation of the GRU [10]. The reset gate is repurposed to determine the manner in which past relation-aware information is used.

4.2 Contrastive Learning

Generator. For numerical reasoning, generating high-quality positive and negative samples is crucial, as numerical data consist of continuous scalar values that are sensitive to even small differences. For example, consider the head entity h with a *height* of 170 cm and the relation r, *is_taller_than*. The ideal learning scenario involves generating all possible positive tail entities with heights less than 170 cm. It is particularly important to clearly differentiate between the tails near the decision boundary, such as those ranging from 169 to 171 cm. To address this problem, first, given the head entity h and relation r, we generate unseen positive/negative samples as follows:

$$\mathbf{e}^{+} = \frac{\sum_{i=1}^{|\mathcal{P}[h,r]|} w_i \mathbf{e}_{\mathbf{r,i}}^{\mathbf{att}}}{\sum_{i=1}^{|\mathcal{P}[h,r]|} w_i}, i \in \mathcal{P}[h,r], \tag{12}$$

$$\mathbf{e}^{-} = \frac{\sum_{j=1}^{|\mathcal{N}[h,r]|} w_j \mathbf{e}_{\mathbf{r,j}}^{\mathbf{att}}}{\sum_{j=1}^{|\mathcal{N}[h,r]|} w_j}, j \in \mathcal{N}[h,r], \tag{13}$$

where $\mathcal{P}[h,r] = \{i|(h,r,i) \in \mathcal{G}\}$, $\mathcal{N}[h,r] = \{j|(j,r,h) \in \mathcal{G}\}$. To promote the diversity of the representations, w_i and w_j are randomly selected from a uniform distribution [0,1] for every epoch. $\mathcal{P}[h,r]$ represents the set of positive tails containing entities whose height is less than 170 cm (e.g., 155, 160, and 165 cm). $\mathcal{N}[h,r]$ represents the set of negative tails containing entities whose *height* is greater than 170 cm (e.g., 176, 178, and 180 cm). Unlike traditional link prediction, where the negative set consists of unrelated entities, our negative set is composed of entities that are headed in opposite directions.

To synthesize harder positive/negative samples using the mixup method [34], we introduce a variant called head mixing. The head mixing technique is proposed as follows:

$$\mathbf{e}^{+}_{\mathbf{mix}} = \alpha \cdot \mathbf{e}^{+} + (1-\alpha) \cdot \mathbf{e}^{\mathbf{att}}_{\mathbf{r,h}}, \tag{14}$$

$$\mathbf{e}^{-}_{\mathbf{mix}} = \beta \cdot \mathbf{e}^{-} + (1-\beta) \cdot \mathbf{e}^{\mathbf{att}}_{\mathbf{r,h}}, \tag{15}$$

where α, β are mixing coefficients sampled from the uniform distribution [0,1] for every epoch. As a result of the RA Encoder, the head h has an attribute-enriched vector that emphasizes the numerical value *170* and field *height*. Because our final goal is to distinguish whether each tail has a *height* of less than 170 cm, head mixing embedding with each weighted summation of positive and negative samples makes the samples closer to a decision boundary and more difficult to distinguish. Note that $\mathbf{e}^{+}_{\mathbf{mix}}$ and $\mathbf{e}^{-}_{\mathbf{mix}}$ do not cross the decision boundary, which means that they are always true. Our contrastive learning loss is defined as follows:

$$L_{CL} = -\frac{1}{|\mathcal{G}|} \sum_{l \in \mathcal{G}} \log \sigma(score(\mathbf{e}^{\mathbf{att}}_{\mathbf{r,h}}, \mathbf{e}_{\mathbf{r}}, \mathbf{e}^{+}_{\mathbf{mix}}) - score(\mathbf{e}^{\mathbf{att}}_{\mathbf{r,h}}, \mathbf{e}_{\mathbf{r}}, \mathbf{e}^{-}_{\mathbf{mix}})), \tag{16}$$

where *score* can be any score function, such as TransE (Eq. (1)). By randomly mixing the positive tails (Eq. (12)), and head mixing (Eqs. (14) - (15)), it is possible to obtain positive tails that do not exist in training data. Therefore, we expect our model to better understand the inaccessible attribute space by exploiting only training data.

4.3 Training

Score Function. A naive assumption for typical KGE methods is calculating the distance between head, relation, and tail. However, these methods have severe issues in that they do not work well for complex relation patterns. For example, given the head entity h (*height* = 170 cm) and relation r (*is_taller_than*) and leveraging the TransE score function, two tail entities t_1 (*height* = 160 cm) and t_2 (*height* = 150 cm) can be mapped to the same position. To alleviate this problem, we designed a plausibility score function specific to numerical reasoning tasks as follows:

$$\begin{aligned} score(\mathbf{e}^{\mathbf{att}}_{\mathbf{r,h}}, \mathbf{e}_{\mathbf{r}}, \mathbf{e}^{\mathbf{att}}_{\mathbf{r,i}}) = \epsilon - \left\| \mathbf{e}^{\mathbf{att}}_{\mathbf{r,h}} + \mathbf{e}_{\mathbf{r}} - \mathbf{e}^{\mathbf{att}}_{\mathbf{r,i}} \right\|_{1/2} \\ + \delta \left\| max(0, \mathbf{W}_{\mathbf{r}} \mathbf{e}^{\mathbf{att}}_{\mathbf{r,h}} - \mathbf{W}_{\mathbf{r}} \mathbf{e}^{\mathbf{att}}_{\mathbf{r,i}}) \right\|^2, \end{aligned} \tag{17}$$

where δ is a weight for the numerical reasoning score function. The first term is derived from TransE (Eq. (1)), and the second term is the variation of the Order-embedding (Eq. (2)). The projection matrix $\mathbf{W}_{\mathbf{r}} \in \mathbb{R}^{d_{emb} \times d_{emb}}$ plays a pivotal role in mapping entities to a relational-specific space. This implies that each entity has a different order depending on the type of relations.

Loss Function. We used binary cross-entropy loss as presented in ConvE [11] and LiteralE [15]. We applied the sigmoid function to the resulting *score* obtained using Eq. (17) so that the *score* of each triple can be treated as a probability. Let $\mathcal{T} = \mathcal{G} \cup \mathcal{G}^{-}$ denote the training dataset, where $\mathcal{G}^{-}$ denotes the set of negative knowledge triples $\{(h,r,t')|\, h,t' \in \mathcal{E}, r \in \mathcal{R}, (h,r,t') \notin \mathcal{G}\}$. The binary-cross entropy loss is then defined as follows:

$$L_{BCE} = -\frac{1}{|\mathcal{T}|} \sum_{l \in \mathcal{T}} (y_l \log(p_l) + (1-y_l) \log(1-p_l)), \tag{18}$$

where $p_l \in [0,1]$ is the probability of each triple $l \in \mathcal{T}$. $y_l \in \{0,1\}$ is the ground truth label. Combining Eq. (16) and Eq. (18), the final loss can be expressed as follows:

$$L_{total} = L_{BCE} + \lambda L_{CL}, \tag{19}$$

where λ is the coefficient of contrastive loss. The details of learning the losses are described in Algorithm 1 in the Appendix.

4.4 Time Complexity Analysis

RAKGE consists of the RA Encoder, positive/negative sample generators, and a score function. The RA Encoder is further divided into a numeric value embedding layer, a relation-aware attention layer, and a gating layer. In the numeric value embedding layer, we transform the attribute field into an embedding vector with a time complexity of $O(|\mathcal{M}|)$. The relation-aware attention layer also has a time complexity of $O(|\mathcal{M}|)$ because the number of keys and values corresponds to the number of attribute fields, and the query is the target relation. The gating layer combines the entity embedding with the attribute embedding without changing the time complexity. Consequently, the time complexity of the RA Encoder for processing a single triple is $O(|\mathcal{M}|)$. Our contrastive learning method generates positive and negative samples and is associated with Eqs. (12) - (13). The time complexity of this process is proportional to the numbers of positive samples $|\mathcal{P}[h,r]|$ and negative samples $|\mathcal{N}[h,r]|$ for a given triple (h,r,t). This results in the time complexity of $O(|\mathcal{M}| \cdot max\_deg)$, where max_deg denotes the maximum relation-specific node degree. Using the TransE score function in calculating the score of the model does not change the time complexity, as it still involves only summation or subtraction operations. Therefore, the time complexity remains unchanged. In summary, the overall time complexity of RAKGE is $O(\mathcal{T} \cdot (|\mathcal{M}| + |\mathcal{M}| \cdot max\_deg)) \approx O(|\mathcal{T}| \cdot max\_deg)$, where $\mathcal{T}$ is the number of positive and negative triples used for training.

5 EXPERIMENT

5.1 Dataset

Three real-world KG datasets were used to evaluate the proposed model on a numerical reasoning task.

US-Cities is a dataset from the United States Cities Database [1]. It contains 75 numeric attribute fields of cities in the USA (e.g., *latitude and percentage of income*). We generated comparison relations between cities by selecting some of these fields (e.g., *has_more_incomes*).

[1] https://simplemaps.com/data/us-cities

Moreover, structural relations are created between cities and other components (e.g., *is_located_in*).

Spotify is a knowledge graph from Spotify for Developers [2,3]. We set each song as an entity, and each entity has 10 attributes containing numerical features, such as *loudness* and *energy*. The comparison relations include which song is *louder than* or *has more energy than* another song.

Credit is a numerical comparison graph constructed from default payments in Taiwan [31]. Credit card holders correspond to entities, and the numerical comparison relations (e.g., *has_higher_credit*) are generated by comparing numerical attributes (e.g., *credit*) between each entity pair. Moreover, we connected the cardholder entities to other entities (e.g., *high_school, university*) through structural relations (e.g., *education*), which can potentially aid in numerical comparison.

Furthermore, to make the numerical comparison task more realistic, we performed two modifications on all of these datasets. First, we masked 20% of numerical values as zeros (missing values). Second, we employed a minimal number of triples as training sets. The remaining triples were used for evaluation. The statistics of the overall datasets are given in Appendix A.1.

5.2 Baseline

We classified various existing representation learning methods into five groups: Euclidean KGE, Hyperbolic KGE, GNN-based KGE, Attributed KGE, and Self-supervised model for graph.

- **Euclidean KGE** is a representation learning method that models entity and relation embeddings in Euclidean space. While TransE [5] employs euclidean distance, ConvE [11], ComplEx [22], TuckER [4], and DistMult [12] use inner product as a distance measure. Order-embedding [26], HAKE [35], and MuRE [3] capture hierarchical relations. The performance improvement strategy utilized in LTE-KGE [37] was implemented in all Euclidean KGE baselines we mentioned above.
- **Hyperbolic KGE** operates well in circumstances where relations in KGs are hierarchical. MuRP [3], ConE [2], and AttH [8] model hierarchical relations in latent space, and GIE [7] considers both hierarchical and ring-like structures.
- **GNN-based KGE** makes use of GCN layers as entity encoders and predicts the plausibility score of each triple using the score functions of other KGE models. In this group, we adopt two baselines: R-GCN [17] and WGCN [18].
- **Attributed KGE** utilizes side information to obtain more robust embeddings in sparsely observed KGs. We compared our proposed RAKGE to numerical attribute-based KGE models: KBLRN [13], MT-KGNN [20], and LiteralE [15].
- **Self-supervised model for graph** enriches node representation through self-supervised learning. BiGI [6] and SLiCE [27] capture both global and local structures to enhance node representation. SimGCL [33] adjusts the uniformity of the learned representation by adding random noise vectors to node representations.

[2]https://www.kaggle.com/datasets/geomack/spotifyclassification
[3]https://developer.spotify.com/documentation/web-api

Table 1: Comparison to other models. 'Numeric-aware Learning' indicates the availability of numeric information, while 'Relation-aware Learning' indicates the presence of relation-specific operations. 'Missing value encoding' refers to Eq. (3).

	TransE	BiGI	R-GCN	LiteralE	RAKGE
Numeric-aware Learning	×	×	×	○	○
Relation-aware Learning	×	×	○	×	○
Self-Supervised Learning	×	○	×	×	○
Missing Value Encoding	×	×	×	×	○

To clarify how our model differs from these models, we distinguish between several models in Table 1.

6 RESULT

This section presents two types of experiments. The first experiment compares the performance of the baseline models with that of our RAKGE for link prediction. The second experiment is an ablation study that analyzes the effectiveness of each module in RAKGE.

6.1 Performance Comparison

This section presents the results of the primary purpose of our model, which involves comparing the numerical values between two entities and predicting their connections. Our model demonstrated strong performance across all three datasets and proved its superiority in numerical reasoning tasks.

6.1.1 Numerical Reasoning. This experiment compared the performance of the proposed RAKGE model with that of existing KGE models in numerical reasoning tasks. The comparison results are summarized in Table 2. The asymmetric nature of the datasets' relations is well captured by models such as TransE, ConvE, and TuckER, whereas symmetric models, such as DistMult, lag behind. This paper further highlights that while hierarchy-aware models, such as HAKE and hyperbolic embedding-based methods, were expected to outperform others because of the hierarchical nature of their numerical reasoning relations, their performance was limited by their inability to leverage attributes. Attribute-based models, such as LiteralE, performed well compared to the other models. However, KBLRN and MT-KGNN suffered performance degradation due to the sparsity of numerical attributes. The Order-embedding model (denoted as *Order* in the model name column of the table) considers the rank of the entities but not the type of relationships, leading to suboptimal performance. The experimental results demonstrate the effectiveness of the proposed RAKGE model, which outperformed the best competitor, with an improvement of 65.1% in $Hits@1$ and 52.6% in *MRR* on the Spotify dataset.

6.1.2 Self-supervised Learning. To verify the effectiveness of our contrastive learning method with generated positive/negative samples, we compared its performance with that of similar self-supervised techniques applied to graph-based approaches, including BiGI, SLiCE, and SimGCL. In particular, although BiGI and SLiCE learn richer representations by considering both the entire graph (global) and the corresponding subgraph (local), they are not designed to model numerical relations. In contrast, RAKGE focuses on generating hard negative and positive samples to distinguish numerical magnitudes more explicitly. This renders contrastive learning more effective for numerical reasoning tasks, as shown in Table 2.

Table 2: Hits@1, Hits@3, Hits@10, MR, and MRR results for numerical reasoning. Bold scores indicate the best results, while underlined scores represent the second-best results. The % of Improvement (Imp.) column shows the relative improvements of RAKGE compared to the second-best scores.

Model		US-Cities					Spotify					Credit				
		H@1	H@3	H@10	MR	MRR	H@1	H@3	H@10	MR	MRR	H@1	H@3	H@10	MR	MRR
Euclidean	TransE	0.192	0.249	0.322	368	0.238	0.259	0.354	0.464	116	0.330	0.420	0.510	0.629	38	0.489
	DistMult	0.014	0.030	0.063	842	0.033	0.024	0.049	0.129	225	0.063	0.129	0.196	0.298	120	0.189
	ConvE	0.159	0.204	0.271	383	0.200	0.233	0.305	0.412	109	0.294	0.170	0.278	0.428	60	0.258
	ComplEx	0.085	0.125	0.185	586	0.121	0.125	0.187	0.283	175	0.180	0.291	0.386	0.511	66	0.365
	TuckER	0.159	0.214	0.305	317	0.209	0.210	0.294	0.406	101	0.277	0.406	0.514	0.641	34	0.485
	Order	0.000	0.295	0.395	249	0.168	0.000	0.350	0.492	91	0.202	0.000	0.474	0.603	44	0.260
	HAKE	0.003	0.026	0.063	971	0.026	0.008	0.087	0.118	199	0.073	0.050	0.153	0.274	125	0.132
	MuRE	0.032	0.064	0.163	425	0.077	0.000	0.167	0.339	125	0.102	0.068	0.148	0.445	74	0.165
Hyperbolic	MuRP	0.070	0.104	0.172	518	0.107	0.025	0.182	0.313	133	0.122	0.144	0.239	0.424	81	0.229
	ConE	0.008	0.040	0.109	246	0.045	0.000	0.006	0.061	216	0.026	0.006	0.239	0.394	84	0.154
	AttH	0.052	0.071	0.106	1315	0.073	0.049	0.080	0.141	360	0.084	0.175	0.263	0.391	103	0.248
	GIE	0.096	0.136	0.203	577	0.134	0.113	0.175	0.272	191	0.169	0.287	0.386	0.504	73	0.362
GNN-based	R-GCN	0.221	0.282	0.362	310	0.270	0.295	0.388	0.501	88	0.367	0.490	0.570	0.669	34	0.551
	WGCN	0.029	0.056	0.104	968	0.057	0.095	0.162	0.261	331	0.153	0.170	0.256	0.369	100	0.241
Attributed	LiteralE	0.250	0.313	0.398	237	0.301	0.266	0.377	0.494	74	0.347	0.478	0.564	0.673	34	0.543
	KBLRN	0.005	0.015	0.042	2283	0.018	0.017	0.038	0.084	353	0.043	0.007	0.019	0.079	267	0.062
	MT-KGNN	0.068	0.102	0.153	695	0.099	0.109	0.183	0.300	141	0.173	0.211	0.302	0.434	75	0.286
Self-supervised	BiGI	0.185	0.249	0.331	359	0.236	0.260	0.354	0.468	118	0.331	0.418	0.507	0.622	39	0.487
	SLiCE	0.185	0.250	0.331	359	0.237	0.261	0.354	0.469	117	0.332	0.420	0.510	0.622	38	0.490
	SimGCL	0.344	0.415	0.502	**162**	0.399	0.000	0.255	0.467	59	0.167	0.000	0.399	0.645	22	0.239
RAKGE		**0.388**	**0.468**	**0.556**	170	**0.447**	**0.487**	**0.602**	**0.692**	**37**	**0.560**	**0.658**	**0.731**	**0.823**	**12**	**0.712**
% Imp.		12.8	12.8	10.8	-4.9	12.0	65.1	55.2	38.1	37.3	52.6	34.3	28.2	22.3	45.5	29.2

6.2 Model Analysis

6.2.1 Discussions of RAKGE Variants. An ablation study was conducted to compare the performances of RAKGE variants, each with an individual component of the model removed. First, we treated missing values as zeros and computed the attribute embedding $\mathbf{a}_i^m$ without missing the encoding from Eq. (3). We also trained the model without the contrastive loss from Eq. (16) and the second term of the score function from Eq. (17) (a variant of the Order-embedding model). The results in Table 3 indicate that removing any single component leads to decreased performance, highlighting the importance of considering missing numerical values, incorporating contrastive loss, and regularizing the score function in numerical reasoning tasks. In particular, the significant decline in performance when only CL was excluded is due to its association with Order-embedding. Although Order-embedding has the representation power to present the order of each numerical attribute, our model does not learn well when Order-embedding is solely applied because of the sparse observation of attributes and numerical relations. It can be inferred that CL helps alleviate the problem of Order-embedding under high sparsity.

6.2.2 Analysis of RA Encoder. As previously stated, the central concept of our RA Encoder is to extract relevant numeric information for each relation. The attention matrix, obtained using Eq. (6), is visualized in Figure 2. The heatmap highlights two important observations. Firstly, the numeric comparison relations (bottom three rows) correspond to the relevant numeric attributes. For instance, the *commute_time_comp* relation has the highest relevance score for the *commute_time* attribute. Secondly, structural relations (top three rows) also benefit from numeric attributes. For example, when predicting the *located in state* relation, the RA Encoder places more importance on the *latitude*, *population*, and *density* attributes than on the others.

Table 3: Ablation study of RAKGE on US-cities dataset. 'ME' stands for Missing Encoding, 'CL' stands for Contrastive Learning, and 'OE' stands for Order-embedding.

ME	CL	OE	H@1	H@3	H@10	MR	MRR
◯	◯	◯	**0.388**	**0.468**	0.556	170	**0.447**
×	◯	◯	0.288	0.388	0.488	219	0.359
◯	×	◯	0.036	0.074	0.132	522	0.071
◯	◯	×	0.324	0.453	**0.578**	**135**	0.413
◯	×	×	0.365	0.440	0.530	152	0.422

6.2.3 Analysis of Contrastive Learning. We proposed a new contrastive learning method for handling the continuous nature of numeric attributes. To demonstrate how our generator produces truly hard positive and negative samples, we replaced our attribute-enriched vector $\mathbf{e}_{\mathbf{r,i}}^{\mathbf{att}}$ in Eq. (11) with standard entity embeddings learned by Order-embedding in Eq. (2). To accumulate a large number of tail samples, Eqs. (12) - (15) were repeated 200 times using a given head entity with *height* = 170 cm and the relation *taller_than*. These samples were visualized in 2D space using T-SNE [23], as shown in Figure 3. It can be seen that the samples generated by head mixing are well distributed in the space between the positive/negative entities and the head without mixing with each other. The detailed procedure for the sample generation is described in Appendices B and C.2.

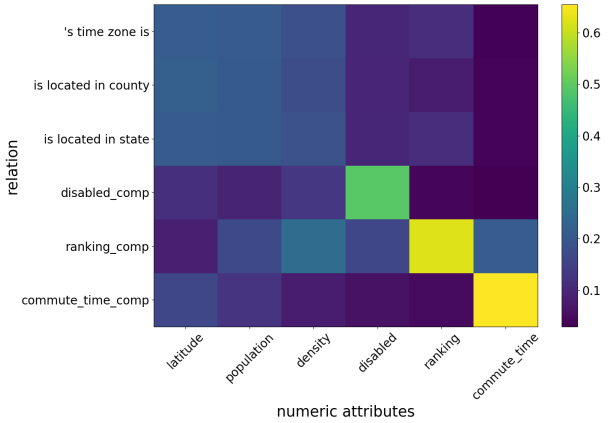


Figure 2: Relevance score of each numeric attribute for each relation on the US-Cities dataset. The lighter the color, the higher the relevance score. 'comp' stands for comparison.

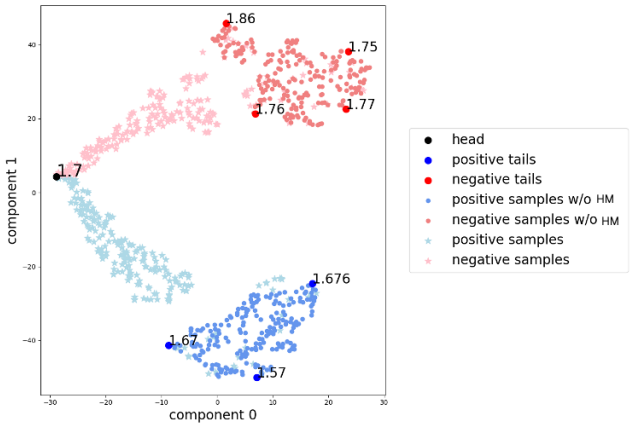


Figure 3: Visualization of positive/negative samples generated by our contrastive learning. The notation 'HM' stands for head mixing, corresponding to Eqs. (14) - (15).

Table 4: Ablation study of RAKGE on Credit dataset

Model	H@1	H@3	H@10	MR	MRR
RAKGE+TransE	0.658	0.731	0.823	12	0.712
RAKGE+ConvE	0.182	0.264	0.361	104	0.247
RAKGE+ComplEx	0.000	0.225	0.397	79	0.146
RAKGE+TuckER	0.000	0.347	0.548	35	0.209

6.2.4 Analysis of Score Function. We utilized TransE in our score function owing to its compatibility with Order-embedding and its ability to handle asymmetric relations. By contrast, ConvE uses convolutional layers, which can disrupt the preservation of the order scheme in entity embeddings. Similarly, ComplEx and TuckER, both of which depend on cosine similarity as their distance metric, struggle to retain the ordering information.

Table 4 supports the effectiveness of TransE compared with the other score functions on the Credit dataset. Our RAKGE+TransE model significantly outperforms the RAKGE+ConvE model, exhibiting enhancements of 261.5 % in *Hits*@1 and 188.3 % in *MRR*. The choice of TransE is grounded in its proficiency in managing asymmetric relations and its affinity with the Order-embedding loss term. The proposed contrastive learning method was applied equally to all models.

6.2.5 Influence of Sparsity of Attributes. Through our RA Encoder and contrastive learning, we expect RAKGE to be robust even when numerical attributes are sparsely observed. Figure 4 shows the performance trend of KGE models with numerical attributes at varying densities. The performance of our RAKGE model slightly decreases with a decrease in density; however, it maintains its leading position in terms of performance. In contrast, LiteralE and MT-KGNN demonstrate insignificant change, even in the absence of numerical attributes. This observation suggests that LiteralE relies mainly on structural information and is less effective for numerical comparisons, whereas our RAKGE method effectively handles numerical values.

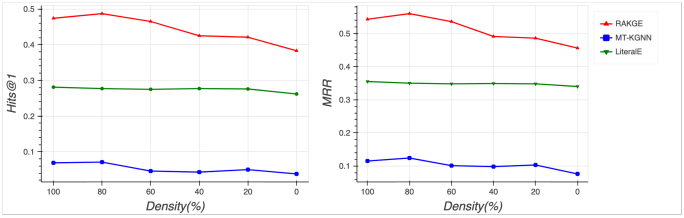


Figure 4: Performance change in Hits@1 and MRR with respect to the density of numeric attributes is shown for the Spotify dataset. A density of 100% means that all entities have numeric values for every attribute field, while a density of 0% means that none of the entities have numeric values.

7 CONCLUSION

In this paper, we proposed RAKGE, which enhances numerical reasoning through *relation-aware attribute representation learning*, and a new contrastive learning method. Our key insight is that by exploiting relational information in both *attribute representation learning* and the scoring function, numerical reasoning is significantly enhanced across various datasets. The proposed method outperformed the best competitor (R-GCN on Spotify) by up to 65.1% in *Hits*@1 and 52.6% in *MRR*. Furthermore, RAKGE is more robust than other models under various attribute sparsity conditions.

8 ACKNOWLEDGEMENT

This work was supported by LG Electronics; in part by the National Research Foundation of Korea (NRF) (2021R1C1C1005407); and by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT): (No. 2019-0-00421, Artificial Intelligence Graduate School Program (Sungkyunkwan University)) and (No. RS-2023-00225441, Knowledge Information Structure Technology for the Multiple Variations of Digital Assets).

REFERENCES

[1] Hadeel Al-Negheimish, Pranava Madhyastha, and Alessandra Russo. 2021. Numerical reasoning in machine reading comprehension tasks: are we there yet?. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 9643–9649.

[2] Yushi Bai, Zhitao Ying, Hongyu Ren, and Jure Leskovec. 2021. Modeling heterogeneous hierarchies with relation-specific hyperbolic cones. *Advances in Neural Information Processing Systems* 34 (2021), 12316–12327.

[3] Ivana Balazevic, Carl Allen, and Timothy Hospedales. 2019. Multi-relational poincaré graph embeddings. *Advances in Neural Information Processing Systems* 32 (2019).

[4] Ivana Balazevic, Carl Allen, and Timothy Hospedales. 2019. TuckER: Tensor Factorization for Knowledge Graph Completion. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 5185–5194.

[5] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in Neural Information Processing Systems* 26 (2013), 2787–2795.

[6] Jiangxia Cao, Xixun Lin, Shu Guo, Luchen Liu, Tingwen Liu, and Bin Wang. 2021. Bipartite graph embedding via mutual information maximization. In *Proceedings of the International Conference on Web Search and Data Mining*. 635–643.

[7] Zongsheng Cao, Qianqian Xu, Zhiyong Yang, Xiaochun Cao, and Qingming Huang. 2022. Geometry interaction knowledge graph embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 5521–5529.

[8] Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. 2020. Low-dimensional hyperbolic knowledge graph embeddings. *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the International Conference on Machine Learning*. 1597–1607.

[10] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 1724–1734.

[11] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 1811–1818.

[12] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data mining*. 601–610.

[13] Alberto García-Durán and Mathias Niepert. 2018. KBLRN: End-to-End Learning of Knowledge Base Representations with Latent, Relational, and Numerical Features. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. 372–381.

[14] Huifeng Guo, Bo Chen, Ruiming Tang, Weinan Zhang, Zhenguo Li, and Xiuqiang He. 2021. An embedding learning framework for numerical features in ctr prediction. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data mining*. 2910–2918.

[15] Agustinus Kristiadi, Mohammad Asif Khan, Denis Lukovnikov, Jens Lehmann, and Asja Fischer. 2019. Incorporating literals into knowledge graph embeddings. In *Proceedings of the International Semantic Web Conference*. 347–363.

[16] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems* 26 (2013), 3111–3119.

[17] 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 *Proceedings of the European Semantic Web Conference*. 593–607.

[18] Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. 2019. End-to-end structure-aware convolutional networks for knowledge base completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 3060–3067.

[19] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. 2019. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the ACM International Conference on Information and Knowledge Management*. 1161–1170.

[20] Yi Tay, Luu Anh Tuan, Minh C Phan, and Siu Cheung Hui. 2017. Multi-task neural network for non-discrete attribute prediction in knowledge graphs. In *Proceedings of the ACM International Conference on Information and Knowledge Management*. 1029–1038.

[21] Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*. 57–66.

[22] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of the International Conference on Machine Learning*. 2071–2080.

[23] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, 11 (2008), 2579–2605.

[24] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2019. Composition-based multi-relational graph convolutional networks. In *Proceedings of the International Conference on Learning Representations*.

[25] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* (2017), 6000–6010.

[26] Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-embeddings of images and language. In *Proceedings of the International Conference on Learning Representations*.

[27] Ping Wang, Khushbu Agarwal, Colby Ham, Sutanay Choudhury, and Chandan K Reddy. 2021. Self-supervised learning of contextual embeddings for link prediction in heterogeneous networks. In *Proceedings of the ACM Web Conference*. 2946–2957.

[28] Xiao Wang, Nian Liu, Hui Han, and Chuan Shi. 2021. Self-supervised heterogeneous graph neural network with co-contrastive learning. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1726–1736.

[29] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 1112–1119.

[30] Yanrong Wu and Zhichun Wang. 2018. Knowledge graph embedding with numeric attributes of entities. In *Proceedings of the Workshop on Representation Learning for NLP*. 132–136.

[31] I-Cheng Yeh and Che-hui Lien. 2009. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications* 36, 2 (2009), 2473–2480.

[32] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems* 33 (2020), 5812–5823.

[33] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. 2022. Are graph augmentations necessary? simple graph contrastive learning for recommendation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1294–1303.

[34] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. 2018. mixup: Beyond Empirical Risk Minimization. In *Proceedings of the International Conference on Learning Representations*.

[35] Zhanqiu Zhang, Jianyu Cai, Yongdong Zhang, and Jie Wang. 2020. Learning hierarchy-aware knowledge graph embeddings for link prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 3065–3072.

[36] Zhenghang Zhang, Jinlu Jia, Yalin Wan, Yang Zhou, Yuting Kong, Yurong Qian, and Jun Long. 2021. TransR*: Representation learning model by flexible translation and relation matrix projection. *Journal of Intelligent & Fuzzy Systems* (2021), 1–9.

[37] Zhanqiu Zhang, Jie Wang, Jieping Ye, and Feng Wu. 2022. Rethinking Graph Convolutional Networks in Knowledge Graph Completion. In *Proceedings of the ACM Web Conference*. 798–807.

A EXPERIMENTAL SETTINGS

The following subsections describe the data statistics, hyperparameters, experimental setup, and additional experimental results.

A.1 Dataset Statistics

To evaluate the performance of RAKGE in this paper, we used the US-Cities, Spotify, and Credit datasets, and detailed information about the datasets is described in Table 5 below.

Table 5: Data statistics

Dataset		US-Cities	Spotify	Credit
# of Triplets	All	60,231,493	6,098,734	4,712,388
	Train	86,124	30,255	52,419
	Valid	30,072,684	3,034,245	2,329,984
	Test	30,072,685	3,034,245	2,329,985
# of Entities		9,296	3,374	1,790
# of Non-numeric Relations		3	3	3
# of Numeric Relations		3	3	3
# of Attributes		65	10	5
# of Numerical Triplets		465,505	20,170	8,915

A.2 Hyperparameters

To ensure the reproducibility of our model, we list the hyperparameters used in RAKGE in Table 6. These parameters were selected based on the validation splits across all three datasets: US-Cities, Spotify, and Credit.

Table 6: Hyperparameters to reproduce our results.

Hyperparameter	Search range	Selected value
Entity/Relation Dimension d_{emb}	[50, 100, 200]	200
Attribute Dimension d_{att}	[50, 100, 200]	200
# of Attention Heads K	[1,2,4,5]	5
Score Function Bias ϵ	[1.0, 3.0, 9.0]	9.0
Numerical Reasoning Weight δ	[0, 0.25, 0.5, 0.75]	0.25
CL Loss Coefficient λ	[0, 0.25, 0.5, 0.75]	0.25
Dropout Rate	[0.1,0.3,0.5, 0.7, 0.9]	0.7
Learning Rate	[0.0001, 0.005, 0.001]	0.001
Batch Size	[128, 256, 512, 1024]	256

A.3 Experimental Setup

For all Euclidean KGEs, including RAKGE, we used the improved learning methods based on dropout, batch normalization, and linear transformation proposed alongside LTE [37][4] for evaluation. The hyperparameters of the other models followed the settings of the proposed models. The evaluation metrics were $Hits@1(H@1)$, $Hits@3(H@3)$, $Hits@10(H@10)$, Mean Rank (*MR*), and Mean Reciprocal Rank (*MRR*). Higher *Hits@K* and *MRR* values indicate a better performance, whereas lower *MR* values imply a better performance.

[4]https://github.com/MIRALab-USTC/GCN4KGC

B LEARNING PROCEDURE OF RAKGE

Algorithm 1 describes the overall learning procedure of RAKGE. For the number of epochs, we first initialized the loss. For each triple in the training set, as described in Line 5, we obtained the head embedding $\mathbf{e}^{\mathbf{att}}_{\mathbf{r,h}}$ using the RA Encoder (Section 4.1). Additional steps for contrastive learning (Section 4.2) are required if r is a comparison relation and (h, r, t) is in the knowledge graph triple set. Through Lines 7-8, we acquire positive and negative embeddings of the head entity and relation via the RA Encoder, $\mathcal{H}_r^{\mathcal{P}}$ and $\mathcal{H}_r^{\mathcal{N}}$, respectively. Lines 9-10 illustrate the mixing process used to generate hard positive and negative embeddings. The contrastive loss is then calculated using Eq. (16), and our loss is updated (Line 11). Finally, the cross-entropy loss is added according to Eq. (18), and θ is updated.

Algorithm 1 *Training RAKGE*

1: **Input:** Training set $\mathcal{T} = \mathcal{G} \cup \mathcal{G}^-$; Positive sample set for $\mathcal{P}[e, r] = \{i | (e, r, i) \in \mathcal{G}\}$; Negative sample set $\mathcal{N}[e, r] = \{j | (j, r, e) \in \mathcal{G}\}$; Initialized RA Encoder *RA_Encoder*; Comparison relation set $\mathcal{R}_{Comparison}$; Coefficient for contrastive loss λ; # of epochs for training *epochs*
2: **for** $1, 2, \cdots,$ *epochs* **do**
3: Initialize loss $L = 0$
4: **for** each $(h, r, t) \in \mathcal{T}$ **do**
5: Acquire the head embedding $\mathbf{e}^{\mathbf{att}}_{\mathbf{r,h}} = RA\_Encoder(h, r)$
6: **if** $r \in \mathcal{R}_{comparison}$ and $(h, r, t) \in \mathcal{G}$ **then**
7: Get $\mathcal{H}_r^{\mathcal{P}} = \{RA\_Encoder(p, r) \mid p \in \mathcal{P}[h, r]\}$
8: Get $\mathcal{H}_r^{\mathcal{N}} = \{RA\_Encoder(n, r) \mid n \in \mathcal{N}[h, r]\}$
9: Generate a hard positive $\mathbf{e}^{+}_{\mathbf{mix}}$ with $\mathcal{H}_r^{\mathcal{P}}$ and $\mathbf{e}^{\mathbf{att}}_{\mathbf{r,h}}$ via Eqs. (12), (14)
10: Generate a hard negative $\mathbf{e}^{-}_{\mathbf{mix}}$ with $\mathcal{H}_r^{\mathcal{N}}$ and $\mathbf{e}^{\mathbf{att}}_{\mathbf{r,h}}$ via Eqs. (13), (15)
11: $L = L + \lambda L_{CL}$, (L_{CL} via Eq. (16))
12: **end if**
13: $L = L + L_{BCE}$, (L_{BCE} via Eq. (18))
14: **end for**
15: Update θ via $\nabla_\theta L$
16: **end for**

C ADDITIONAL EXPERIMENTAL RESULTS

C.1 Feasibility Test

To verify our conclusion further, we conducted a feasibility test to investigate whether RAKGE learns numerical values based on their magnitude or existence. The numeric values in the original dataset were transformed into binary values, indicating the presence of numerical attributes. If the value existed, it was assigned a value of 1; otherwise, a value of 0 was assigned. The results of the feasibility test on the Spotify dataset are reported in Table 7. The best competitor of our model, LiteralE [15], showed no significant change, regardless of whether binary or numerical values were used. However, the performance of RAKGE decreased by 17.6% and 16.8% in *Hits@1* and *MRR*, respectively. This result suggests that our method performs numerical comparisons, whereas LiteralE uses only numerical values as indicators of their existence.

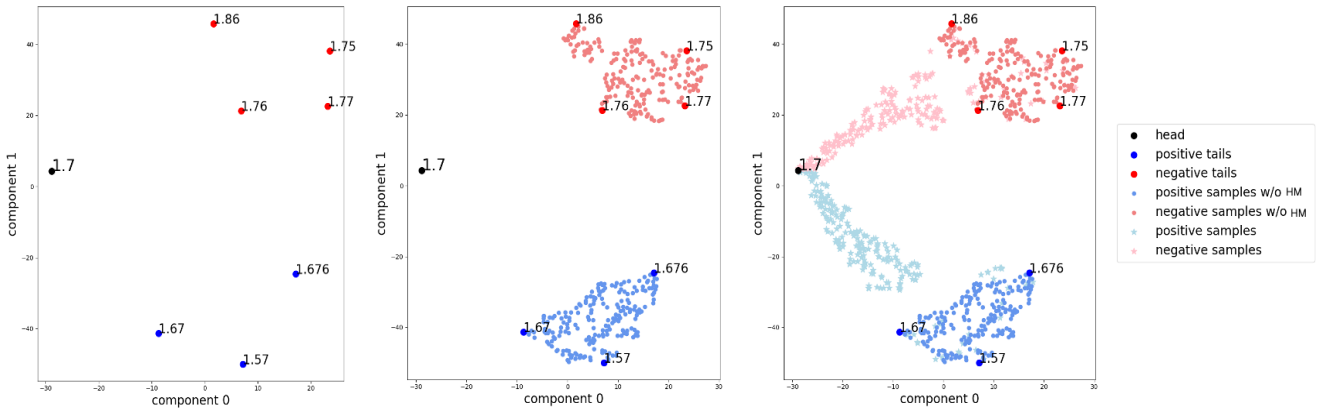


Figure 5: Visualization of how true hard positive/hard samples are generated through our contrastive learning. 'HM' stands for head mixing, corresponding to Eqs. (14) and (15). The leftmost figure shows the head and positive/negative tails given by training data. The middle figure presents samples generated only by Eqs. (12) - (13), and the rightmost figure shows the final positive/negative samples.

Table 8: Case study on US-Cities dataset

Relation	Important Attributes (Attention Scores)
commute_time_comp	**income (0.106)**, age (0.038), male population (0.020)
ranking_comp	**hispanic (0.168)**, density (0.096), longitude (0.095)
disabled_comp	**income (0.201)**, divorced (0.067), widowed (0.043)

Table 7: Feasiblity test of LiteralE and RAKGE on Spotify dataset

Model	Setting	H@1	H@3	H@10	MR	MRR
LiteralE	**Binary**	0.263	0.362	0.480	79	0.339
	Numeric	0.266	0.377	0.494	74	0.347
RAKGE	**Binary**	0.394	0.479	0.571	62	0.459
	Numeric	0.478	0.592	0.687	44	0.552

C.2 Visualization

To demonstrate how our generator produces hard positive and negative samples, we replaced our attribute-enriched vector $\mathbf{e}^{\mathbf{att}}_{\mathbf{r,i}}$ in Eq. (11) with the standard entity embeddings learned by the order-embedding in Eq. (2). To obtain numerous tail samples, Eqs. (12) - (15) were repeated 200 times using a given head entity with *height* = 170 cm and the relation *taller_than*. The step-by-step generation of samples through our contrastive learning is illustrated in Figure 5. All the generated positive and negative samples show a tendency to be close to the head entity; however, they do not cross the decision boundary.

C.3 Qualitative Analysis

The RA Encoder in our model was designed to obtain more robust attribute representations when the attributes were sparsely observed. Therefore, we investigated how well other significant attributes contributed to the inference of a derived relation when the corresponding numeric attribute is missing. Table 8 presents the attention scores of the RA Encoder for the top three most significant numeric attributes used to infer the comparison relations in the US-Cities dataset. We observed that the comparison relation places importance on other appropriate attributes, such as *income*, instead of *commute_time*.

C.4 Evaluation on FB15k-237 Dataset

In standard link prediction tasks for knowledge graph completion, the FB15k-237 dataset [21] is often used. As the datasets used in our study were newly produced for large-scale analysis, we also conducted an additional experiment using FB15k-237. As shown in Table 9, RAKGE outperformed the other models.

Table 9: Link prediction results on FB15k-237 [21]. The results marked with ⋆ are taken from [37], and those with † are taken from [15]. The results of other baseline models were obtained from their original paper. Bold scores represent the best results, and underlined scores represent the second-best results.

Model	H@1	H@3	H@10	MR	MRR
TransE ⋆ [5]	0.240	0.368	0.516	**182**	0.332
DistMult ⋆ [12]	0.202	0.306	0.433	392	0.279
ConvE ⋆ [11]	0.232	0.351	0.492	<u>276</u>	0.319
TuckER [4]	<u>0.266</u>	<u>0.394</u>	**0.544**	-	<u>0.358</u>
MuRP [3]	0.245	0.370	0.521	-	0.336
R-GCN [17]	0.153	0.258	0.414	-	0.248
LiteralE [15]	0.232	0.348	0.483	280	0.317
KBLRN † [13]	0.215	0.333	0.468	358	0.301
MT-KGNN † [20]	0.204	0.312	0.445	532	0.285
RAKGE	**0.282**	**0.402**	<u>0.531</u>	321	**0.366**