SAND2019-13953C

Two Problems in Knowledge Graph Embedding: Non-Exclusive Relation Categories and Zero Gradients

Nasheen Nur
Dept. of Software Information Systems
University of North Carolina
Charlotte, North Carolina, USA
Email: nnur@uncc.edu

Noseong Park
Center for Secure Information systems
George Mason University
Fairfax, Virginia, USA
Email: npark9@gmu.edu

Kookjin Lee
Extreme-Scale Data Science Department
Sandia National Laboratories
Livermore, California, USA
Email: koolee@sandia.gov

Hyunjoong Kang, Soonhyeon Kwon
Electronics and Telecommunications Research Institute
Daejeon, South Korea
Email: {hjkang,kwonshzzang}@etri.re.kr

***Abstract*— Knowledge graph embedding (KGE) learns latent vector representations of named entities (i.e., vertices) and relations (i.e., edge labels) of knowledge graphs. Herein, we address two problems in KGE. First, relations may belong to one or multiple categories, such as functional, symmetric, transitive, reflexive, and so forth; thus, relation categories are not exclusive. Some relation categories cause non-trivial challenges for KGE. Second, we found that zero gradients happen frequently in many translation based embedding methods such as `TransE` and its variations. To solve these problems, we propose i) converting a knowledge graph into a bipartite graph, although we do not physically convert the graph but rather use an equivalent trick; ii) using multiple vector representations for a relation; and iii) using a new hinge loss based on *energy ratio* (rather than *energy gap*) that does not cause zero gradients. We show that our method significantly improves the quality of embedding.**

***Keywords*-Relational Machine Learning; Knowledge Graph; Knowledge Graph Embedding;**

1 Introduction

Knowledge graphs constructed after processing texts such as web pages are widely used in question answering and web search. They are considered as a way to give machines intelligence.

Knowledge graph embedding (KGE) maps entities (i.e., vertices) and relations (i.e., edge labels) of knowledge graphs into a d-dimensional vector space. Thus, a vertex v (resp. a relation r) is represented by a vector $\mathbf{v}$ (resp. $\mathbf{r}$) — we use boldface to denote vectors. Given a triple (v, r, u), where v and u are entities and r is a relation, $f(\mathbf{v}, \mathbf{r}) \approx \mathbf{u}$ if the triple is true. In other words, r translates v into u after a certain vector operation f. This type of KGE methods is called *translation based method* and it is yet the most popular one although many non-translation based methods [13], [14], [14], [19] were recently proposed. For instance, (v = Trump, r = isPresidentOf, u = USA) is one such triple and `TransE` [2] uses vector addition for the calculation of $f(\mathbf{v}, \mathbf{r})$ such that $\mathbf{v} + \mathbf{r} \approx \mathbf{u}$ (see Figure 1.a for an example of `TransE` embedding).

Many different KGE methods using different vector operations (sometimes neural network operations) have been proposed [1]–[4], [6], [8]–[13], [15]–[19], [21].

However, there are still a couple of problems to solve. First, there exist several *non-exclusive* relation categories, such as *functional*, *reflexive*, *transitive*, *symmetric*, and so forth, and previous approaches are not effective in embedding relations that belong to multiple relation categories. For instance, "is relative of" in Figure 1.b is both symmetric and transitive and hard to embed all the six triples correctly with existing methods. Second, zero gradients happen frequently. 25%∼75% of gradients were zero in our tests with many KGE methods.

a) Non-exclusive relation categories.: Authors of [21] considered symmetric and transitive relations separately. `HolE` [13] and `ComplEx` [15] are also able to consider some relation categories — for instance, the correlation operator used in `HolE` is not commutative and suitable to embed symmetric relations. However, relation categories are not exclusive and a relation can belong to multiple categories. Therefore, their methods that separately consider a subset of categories partially solve the problem, as a matter of fact. None of existing works discuss about this non-exclusive property of relations. We embed these relations by learning multiple vector representations for each relation after converting a knowledge graph into a bipartite graph. Our algorithm `TransB` is named after the bipartite conversion.

b) Zero gradient problem (ZGP).: Back-propagation is a popular method to train KGE methods. We observed many zero gradients during the stochastic gradient descent (SGD) updates in `TransE` and its many variations [1], [8], [9], [11], [17], [18], [21]. Once zero gradients happen, there are no updates of vectors. Those translation based methods are very

Noseong Park is the corresponding author.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

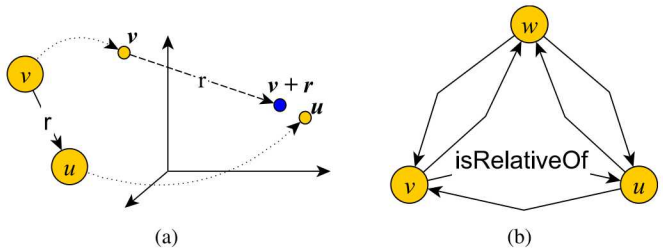

Fig. 1: (a) An example of TransE embedding. (b) An example of a symmetric and transitive relation. "isRelativeOf" is one of the most difficult relations to embed. Almost all existing KGE methods cannot smoothly embed this case (see Example 3.2 for more detailed descriptions).

popular currently and we improve their training method by proposing a new loss function that is free from the zero gradient problem (ZGP). In Theorems 3.3 and 3.4, we present the rigorous proofs of the cases where the ZGP occurs in TransE and its variations. Our proposed algorithm TransB uses the new loss function.

ComplEx [15] also showed that its log-likelihood based loss function tends to perform better than existing loss definitions. They showed that their loss definition works well in experiments without theoretical grounds. However, we theoretically prove why existing loss definition is sub-optimal and the proposed one is better. Our TransB equipped with the new loss definition outperforms many existing methods including ComplEx.

In our experiments with two standard knowledge graphs FB15K and WN18, TransB achieved the best accuracy (the filtered mean rank of 37 in FB15K and the filterd Hits@10 of 96.9% in WN18), compared to state-of-the-art KGE methods.

2 Related Work

A knowledge can be represented by a triple (v, r, u), where v and u are *named entities* and r is a *relation*, for example, (Trump, isPresidentOf, USA). A collection of such triples constitutes an extremely large connected graph, namely, *knowledge graph*, denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathsf{L})$, where $\mathcal{V}$ is a set of vertices (i.e., named entities), $\mathcal{E}$ is a set of edges (i.e., relationships), and each edge is labeled by a relation $r \in \mathsf{L}$.

KGE maps vertices and relations onto a d-dimensional vector space and infers missing triples. We use a bold character $\mathbf{v} \in \mathbb{R}^d$ (resp. $\mathbf{r} \in \mathbb{R}^d$) to denote a $d \times 1$ vector representation (column vector) of a vertex v (resp. a relation r). $\mathbf{x}_{[\mathsf{i}]}$ means the i-th scalar element of $\mathbf{x}$.

TransE, SE, and SME [1]–[3] are the most popular and pioneering works. These works proposed embedding frameworks based on the concept of *energy*. The *energy* of a triple means the uncertainty (or error) of the triple in a given graph structure. Thus, the higher the energy is, the greater the uncertainty (or error) is.

Definition 2.1 (Energy): Given a triple $t = (v, r, u)$, the energy of t is defined as follows:

1) $e(v, r, u) = \|\mathbf{v} + \mathbf{r} - \mathbf{u}\|_{1 \text{ or } 2}$ in TransE[1].
2) $e(v, r, u) = g_v(\mathbf{r}, \mathbf{v})^{\mathrm{T}} \cdot g_u(\mathbf{r}, \mathbf{u})$, where g_v and g_u are linear or bilinear functions in SME.
3) $e(v, r, u) = \|\mathbf{r_l}\mathbf{v} - \mathbf{r_r}\mathbf{u}\|_{1 \text{ or } 2}$, where $\mathbf{r_l}$ and $\mathbf{r_r}$ are left and right projection matrices[2] representing r in SE.

Example 2.2 (TransE): In Figure 1.a, v, r, and u (on the left side) are mapped into the 3-dimensional space (on the right side). $\mathbf{v} + \mathbf{r}$, represented by the blue vertex, is the predicted position of v's neighbor connected through r. Thus, the energy $\|\mathbf{v}+\mathbf{r}-\mathbf{u}\|$ refers to the distance between the blue vertex and $\mathbf{u}$ in the feature space. Given (v = Trump, r = isPresidentOf, u = USA), for instance, the sum of two vectors representing 'Trump' and 'isPresidentOf' should be close to the vector representing 'US' because the triple is true. □

TransE has the most preferred energy definition for its simplicity and good accuracy, and many variations of TransE exist [6], [8], [9], [11], [17], [18], [20], [21]. Thus, our descriptions and examples will also use TransE as the default energy definition, but we will compare with SE, SME, and many others in our experiments.

KGE learns vertex and relation vector representations by minimizing the following hinge loss function. Note that we can use any TransE, SE, and SME energy definitions in the loss function. In fact, all existing works' overall algorithms are more or less identical except for the energy definitions.

$$\mathcal{L} = \sum_{t^+ \in \mathcal{E}} \sum_{t^- \in \mathcal{N}(t^+)} \max\left(0, \gamma + e(t^+) - e(t^-)\right), \qquad (1)$$

where γ is a margin set by user, $\mathcal{E}$ is a set of triples (i.e., edges in $\mathcal{E}$) and $\mathcal{N}(t^+)$ is a set of negative triples derived from a true triple $t^+ \in \mathcal{E}$. $t^- \in \mathcal{N}(t^+)$ differs from t^+ only in one vertex. Please see the following true and derived negative triple examples.

Example 2.3 (Negative triples): We present two negative triples derived from one true triple. (Trump, isPresidentOf, UK) and (Bill Gates, isPresidentOf, USA) are negative triples derived from (Trump, isPresidentOf, USA). □

There also exist many TransE's variations such as TransR [11], TransD [8], TransH [17], TransG [18], TransF [6], etc.

3 Problem Statement

We introduce two problems associated with existing KGE methods. In the next section, we describe our proposed methods to address the two problems.

3.1 Non-exclusive relation categories

We consider the following non-exclusive categorization system of relations: *functional*, *symmetric*, *transitive*, *reflexive*, and so forth. This classification is widely used in the field of logic programming and the Semantic Web where the concept of knowledge graph (or base) was created.

Definition 3.1 (Relation categories): A relation r is classified as follows:

[1] $\|\cdot\|_1$ (resp. $\|\cdot\|_2$) refers to the ℓ_1-norm (resp. ℓ_2-norm) of a vector.
[2] After vectorization, a matrix can still be represented by a vector.

1) Symmetric: if (v, r, u) implies (u, r, v). For example, $isFriendOf$ is a symmetric relation because $(v, isFriendOf, u)$ implies $(u, isFriendOf, v)$.
2) Reflexive: if $v = u$ is the case given (v, r, u). For example, $knows$ can be a reflexive relation because one knows oneself.
3) Transitive: if (v, r, u) is implied by two triples (v, r, w) and (w, r, u). For example, "isRelativeOf" is a transitive relation. $(v, isRelativeOf, u)$ is implied by $(v, isRelativeOf, w)$ and $(w, isRelativeOf, u)$.
4) Functional: if u is unique for a given v of (v, r, u). For example, $isPresidentOf$ is a functional relation because one person can serve as president of only one country.
5) There also exist some other types such as *inverse functional*, *irreflexive*, *antisymmetric*, etc.

The first three categories create non-trivial challenges for KGE. In particular, the reflexive category is very difficult to embed because its source and target are the same. This requires a zero relation vector, which is not desired. In 1.8K triples of FB15K, their two end vertices are the same, i.e., reflexive.

To further complicate the situation, the relation categories are not exclusive, i.e., a relation may belong to multiple categories. For example, "isRelativeOf" is both symmetric and transitive and cannot be correctly embedded in any of the existing methods. In [21], the authors proposed a method for exclusive symmetric or transitive cases. However, they did not consider the case that a relation has multiple categories.

This overlapping relation category problem frequently occurs. For instance, 19K triples of FB15K fall into the joint category of symmetric and transitive.

Example 3.2 (isRelativeOf relation): In Figure 1.b, there are three completely connected vertices. KGE requires low energy for all the triples in this example. However, it is impossible in existing works. For example, $e(v, r, u)$ and $e(u, r, v)$ cannot be minimized simultaneously in almost all methods introduced in Section 2. To embed (v, r, u) and (u, r, v) simultaneously in `TransE`, for example, $\|\mathbf{v}+\mathbf{r}-\mathbf{u}\| = \|\mathbf{u}+\mathbf{r}-\mathbf{v}\| \approx 0$. However, this is possible only if $\mathbf{v} \approx \mathbf{u}$ and $\mathbf{r} \approx 0$ that are not correct representations. This is the same in many other methods. Even with the enhancements of [13], [15], [21], smoothly embedding all six triples is impossible. For instance, in `HolE` [13] it is still required that $\mathbf{v} \approx \mathbf{u}$ to embed triples between v and w and between w and u. However, this is not the ideal to embed triples between v and u. □

We propose two solutions: converting into a bipartite graph and learning multiple vector representations for each relation.

3.2 Zero gradient problem (ZGP)

We found that the gradients of the energy margin loss in Equation (1) can be zero if the ℓ_1-norm is used. In fact, the ℓ_2-norm is not used in many KGE methods. For instance, check the implementation code[3] of `TransG`, one of the most advanced translation methods. Its line number 216 in the link is to update the vector representation of a source vertex (called "head" in their codes). They use only the gradients of the ℓ_1-norm and didn't implement the ℓ_2-norm at all. Likewise, almost all `TransE`'s variations use only the ℓ_1-norm because it gives better performance in the majority of cases in their experiments. Thus, our theorem raises a critical issue in the state-of-the-art training method.

Theorem 3.3: Suppose the energy margin loss $\mathcal{L}$ in Equation (1). Given $t^+ = (v, r, u)$, $t^- = (v, r, u')$ and the `TransE`'s energy function based on the ℓ_1-norm, $\nabla_{\mathbf{v}_{[i]}} e(t^+) - e(t^-)$, i.e., the gradient w.r.t. i-th element of $\mathbf{v}$, will be 0 if $sgn(|\mathbf{v}_{[i]}+\mathbf{r}_{[i]}-\mathbf{u}_{[i]}|) = sgn(|\mathbf{v}_{[i]}+\mathbf{r}_{[i]}-\mathbf{u}'_{[i]}|)$, where $sgn(x)$ returns the sign of x. Note that this theorem states a sufficient condition of $\nabla_{\mathbf{v}_{[i]}} \mathcal{L} = 0$.

Proof: We first prove for `TransE`, and then we prove for it variations. Let $\mathbf{f} = e(t^+) - e(t^-)$, where $t^+ = (v, r, u)$ and $t^- = (v, r, u')$. Thus, $\mathbf{f} = \|\mathbf{v}+\mathbf{r}-\mathbf{u}\|_1 - \|\mathbf{v}+\mathbf{r}-\mathbf{u}'\|_1$.

Using the chain rule and $\frac{\partial |x|}{\partial x} = \frac{x}{|x|} = sgn(x)$, the gradient w.r.t. i-th element of $\mathbf{v}$ can be calculated as follows[4]:

$$\nabla_{\mathbf{v}_{[i]}} \mathbf{f} = sgn(\mathbf{v}_{[i]} + \mathbf{r}_{[i]} - \mathbf{u}_{[i]}) - sgn(\mathbf{v}_{[i]} + \mathbf{r}_{[i]} - \mathbf{u}'_{[i]}). \quad (2)$$

This implies that if their signs are the same, i.e., $sgn(\mathbf{v}_{[i]} + \mathbf{r}_{[i]} - \mathbf{u}_{[i]}) = sgn(\mathbf{v}_{[i]} + \mathbf{r}_{[i]} - \mathbf{u}'_{[i]})$, then $\nabla_{\mathbf{v}_{[i]}} \mathbf{f} = 0$. This happens very frequently. If $\mathbf{u}_{[i]}$ and $\mathbf{u}'_{[i]}$ are two random values, for instance, the zero gradient probability will be 50%. One significance of the problem is that $\nabla_{\mathbf{v}_{[i]}} \mathbf{f}$ is mainly decided by two different values $\mathbf{u}_{[i]}$ and $\mathbf{u}'_{[i]}$ instead of $\mathbf{v}_{[i]}$. In our experiments, we could observe that 25%∼72% of gradients are zero.

This is still the case for many variations of `TransE` that involve matrix multiplications, such as `TransR`, `TransH`, `TransD`, and so forth. They use projection matrices to transform $\mathbf{v}$ and $\mathbf{u}$. Because of space constraints, we will prove for `TransR`, but our theorem can also be applied to others.

In `TransR`, $e(v, r, u) = \|M_r\mathbf{v}+\mathbf{r}-M_r\mathbf{u}\|$, where M_r is a transformation matrix associated with r. Thus, $\mathbf{f} = \|M_r\mathbf{v}+\mathbf{r}-M_r\mathbf{u}\|_1 - \|M_r\mathbf{v}+\mathbf{r}-M_r\mathbf{u}'\|_1$. The gradient of $\mathbf{f}$ w.r.t. $\mathbf{v}_{[i]}$ can be derived as follows:

$$\begin{aligned}\nabla_{\mathbf{v}_{[i]}} \mathbf{f} =& sgn(M_{r[:i]}\mathbf{v}_{[i]} + \sum_j \mathbf{r}_{[j]} - M_{r[:i]}\mathbf{u}_{[i]}) \\ &- sgn(M_{r[:i]}\mathbf{v}_{[i]} + \sum_j \mathbf{r}_{[j]} - M_{r[:i]}\mathbf{u}'_{[i]}),\end{aligned} \quad (3)$$

where $M_{r[:i]}$ is i-th column vector of the transformation matrix M_r.

Note that the two terms in Equation (3) are the same except for $\mathbf{u}_{[i]}$ and $\mathbf{u}'_{[i]}$. As shown, $\nabla_{\mathbf{v}_{[i]}} \mathbf{f} = 0$ under the same condition that the signs are equal in both terms. ■

Theorem 3.4: $\nabla_{\mathbf{u}_{[i]}} e(t^+) - e(t^-)$ can also be 0 when $t^+ = (v, r, u)$ and $t^- = (v', r, u)$, i.e., the source vertex v is changed to v' to create a negative triple t^-. (Its proof is almost the same as the previous one.)

[3] Click to see the implementation code.

[4] Click to see an example, where $\mathbf{v} = (a, d)$, $\mathbf{r} = (b, e)$, $\mathbf{u} = (c, f)$ and $\mathbf{u}' = (g, h)$, i.e., two dimensional vectors for simplicity.

The above two theorems state that, given $t^+ = (v, r, u)$, $\nabla_{\mathbf{v}_{[i]}}\mathcal{L}$ (resp. $\nabla_{\mathbf{u}_{[i]}}\mathcal{L}$) can be zero when the target (resp. source) vertex is perturbed in its negative triple sample.

In stochastic gradient descent, each dimension of $\mathbf{v}$ or $\mathbf{u}$ is independently updated, i.e., $\mathbf{v}_{[i]} = \mathbf{v}_{[i]} - lr \cdot \nabla_{\mathbf{v}_{[i]}}\mathcal{L}$, where lr is a learning rate. Note that other dimensions except for i are not closely related to updating $\mathbf{v}_{[i]}$.

Thus, many true-negative pairs (that meet the specified zero-gradient condition in the above theorems) do not update $\mathbf{u}_{[i]}$ or $\mathbf{v}_{[i]}$. This causes two problems: i) a considerable amount of gradient computation is actually useless, and ii) vector representations may not be fully optimized.

When checking $\nabla_{\mathbf{v}_{[i]}}\mathcal{L}$ for each vertex v in $FB15K$ with `TransE`, `TransH`, `TransG`, and so on, we observed that 25%∼72% elements of the gradient matrix[5] are zero in average during the entire training period. This means that at most 75% elements of vector representations are not updated even after huge computation. Be careful that some elements of a vector representation can still be updated. However, many elements are not updated, which is suboptimal.

4 Proposed Method to Address Non-exclusive Relation Category Problem

To address the non-exclusive relation category problem, we i) convert a knowledge graph into a bipartite graph containing the same knowledge — however, we do not physically convert the graph but rather utilize an equivalent implementation trick that will be introduced later — and ii) learn multiple vector representations for each relation.

In particular, the bipartite graph conversion makes the multiple vector representation learning much easier than that without the bipartite conversion.

4.1 Bipartite graph conversion

For the conversion, we divide each vertex v into v_s and v_t: v_s will appear only in the source vertex position of triples and v_t in the target vertex position.

The main advantage of the bipartite graph approach is the decoupling of edges that are incident in the original knowledge graph. For example, two symmetric triples (u, r, v) and (v, r, u) can easily be embedded by `TransE` after the decoupling because $\|\mathbf{u_s} + \mathbf{r} - \mathbf{v_t}\|$ and $\|\mathbf{v_s} + \mathbf{r} - \mathbf{u_t}\|$ do not share any vertex representations. Therefore, this bipartite conversion is effective at embedding symmetric relations. After the conversion, however, the bipartite graph has twice as many vertices, which may delay the training process and increase the GPU memory requirement.

Instead, we only learn one linear layer and $\mathbf{v_s} \in \mathbb{R}^d$. Given a relation r and a vertex v, we require $\mathbf{v_t} = Linear(\mathbf{r} \oplus \mathbf{v_s})$, where $\oplus$ means the concatenation of two vectors, and $Linear(\cdot)$ is the linear layer we train whose input and output dimensions are $2d$ and d respectively. This linear layer trick implies that the bipartite conversion is specific to the relation r (because we do concatenate with $\mathbf{r}$), and $\mathbf{v_t}$ varies depending on r given a fixed vector $\mathbf{v_s}$.

The proposed bipartite conversion is not only to reduce the number of parameters that we have to learn but also to prevent the case that $\mathbf{v_s}$ (resp. $\mathbf{v_t}$) becomes any arbitrary vector representation when v (resp. t) does not appear in the source (resp. target) position of triples .

[5] An (i, j) element of the gradient matrix is $\nabla_{\mathbf{v}_{[j]}}\mathcal{L}$, where $\mathbf{v}$ is an i-th vector in the matrix.

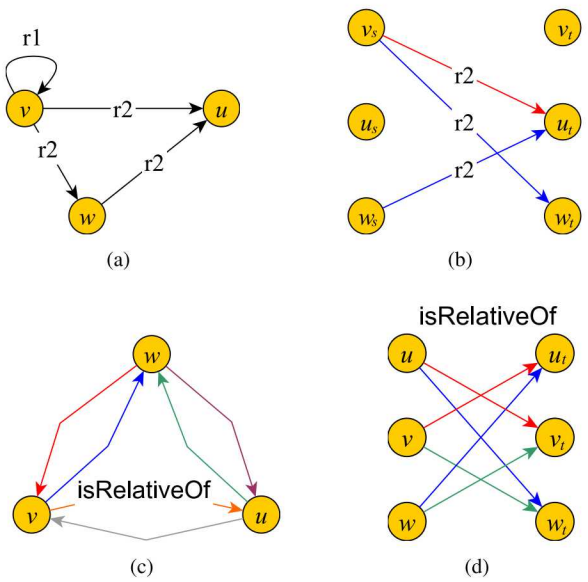

Fig. 2: (a) A knowledge graph where $r1$ is a reflexive relation and $r2$ is a transitive relation. (b) Its edge-colored bipartite graph for the relation $r2$ — note that edges have different colors when they share a vertex (i.e., incident edges). This edge-coloring task should be performed for each relation, and this coloring result implies that 2 vector representations are enough for $r2$. (c) The edge-coloring result of Figure 1.b without the bipartite conversion, where we need 6 colors. (d) The edge-coloring result of Figure 1.b after the bipartite conversion. Note that we need 3 colors. The bipartite graph conversion can reduce the number of colors required, which is the reason why we do the edge-coloring after the bipartite conversion.

4.2 Edge-coloring for multiple vector representations

The bipartite graph approach is effective to embed symmetric relations but not complete for others. As we discussed in Example 3.2, one vector representation for one relation may not be sufficient so we learn a set of multiple vector representations $\mathcal{R} = \{\mathbf{r_0}, \mathbf{r_1}, \cdots\}$ given a relation r. However, we have to suppress the use of too many vector representations for a relation for potential overfitting and efficiency issues. This creates an important trade-off issue. *We found that the minimum number of vector representations for a relation can be deterministically calculated, and its calculation in the bipartite graph is considerably easier than in the original knowledge graph.*

The edge-coloring problem is to color incident edges with different colors and is very well studied for bipartite

graphs [7]. Thus, we run the edge-coloring algorithm after constructing a bipartite graph. We perform the bipartite conversion and edge-coloring for each relation to know the exact number of colors (vector representations) needed to embed the relation. The number of used colors is the same as the number of vector representations for the relation such that all incident edges that have the same relation can be decoupled for embedding.

In particular, the bipartite graph conversion reduces the number of colors required in the edge coloring process, which is the reason why we run the edge coloring algorithm after the bipartite conversion. See Example 4.2 below to see how this is the case.

Example 4.1 (Edge-coloring after Bipartite graph conversion): In Figure 2.a, a knowledge graph that consists of four triples is presented: one triple for "r1" and three for "r2". Note that "r1" is reflexive and "r2" is transitive. After converting this knowledge graph into the bipartite graph in Figure 2.b and performing edge coloring, we know that 2 vector representations are sufficient to map all three triples that have "r2". In addition, note that u_s's degree is 0, which means that there are no training cases for it. However, we can actually learn its vector representation because $\mathbf{u_t}$ is calculated from $\mathbf{u_s}$ with the linear layer. □

Example 4.2 (Edge-coloring after Bipartite graph conversion): In Figure 2.d, you can see the edge-coloring result of the example triples in Figure 1.b, one of the most problematic cases in KGE. Using 3 colors, we can solve the edge-coloring problem, which means 3 vector representations are enough to embed all the triples with the "isRelativeOf" relation. Note that Figure 2.c requires 6 colors when we do not adopt the bipartite graph. Thus, our approach to perform the edge-coloring after the bipartite conversion is effective in reducing the number of vectors we have to learn for each relation.

Because we learn 3 vector representations for the relation, triples with different colors can be considered as different relations. Thus, triples in Figure 2.d are not any more symmetric and transitive during the embedding process. □

As shown in Figures 2.c and d, the bipartite graph conversion is effective to reduce the number of colors (vector representations) to be learned during the embedding process. This is one more reason why we use the combination of the bipartite graph conversion and edge-coloring approach.

There is still one problem. Because a relation r is represented by multiple vector representations $\mathcal{R} = \{\mathbf{r_0}, \mathbf{r_1}, \cdots\}$, it is not clear which vector we have to use to embed triples. Each relation vector representation $\mathbf{r_i}$ may provide very different energy level for a triple (v, r, u). We need a stable method to have one energy level from multiple relation vector representations.

To this end, given a triple (v, r, u), we use the softmax-based ensemble to determine its energy level as follows:

$$P(\mathbf{r_i} \mid \mathbf{x}) = \frac{e^{x^\intercal \cdot \mathbf{w}_{\mathbf{r_i}}}}{\sum_{j=1}^{|\mathcal{R}|} e^{x^\intercal \cdot \mathbf{w}_{\mathbf{r_j}}}}, \tag{4}$$

and

$$\begin{aligned} e(v, r, u) &= \sum_{i=1}^{|\mathcal{R}|} P(\mathbf{r_i} \mid \mathbf{v_s} \oplus \mathbf{u_t}) \cdot \|\mathbf{v_s} + \mathbf{r_i} - \mathbf{u_t}\| \\ &= \sum_{i=1}^{|\mathcal{R}|} P(\mathbf{r_i} \mid \mathbf{v_s} \oplus \mathbf{u_t}) \cdot \|\mathbf{v_s} + \mathbf{r_i} - Linear(\mathbf{r_i} \oplus \mathbf{u_s})\|. \end{aligned} \tag{5}$$

`TransG` also learns multiple vectors for each relation but mainly uses only one $\mathbf{r_i}$ that is the closest to $\mathbf{v}$ to embed the triple, i.e., $\min_i \|\mathbf{r_i} - \mathbf{v}\|$. Our softmax-based definition does not have this restriction.

It has already been proven that a simple online edge-coloring algorithm performs well [7] and that Δ (the max degree) colors are sufficient to color all edges. This algorithm reads and colors edges one-by-one. Given k possible colors, an edge e is marked as *failure* if all colors are already consumed by e's incident edges. This simple coloring algorithm's competitive ratio is approximately 48%[6] for bipartite graphs. In practice, the competitive ratio is higher than that of the simple coloring algorithm. We prefer the online algorithm for its low time-complexity $\mathcal{O}(|\mathcal{E}|)$ and good performance.

The number of successfully colored edges is a good estimate of successful embedding. For example, assume that almost all edges are successfully colored with only 3 colors, whereas $\Delta = 100$ colors are required for complete coloring; this is the case in many knowledge graphs because the degree distribution follows a power-law distribution[7]. In this case, we can let the embedding algorithm learn 3 vector representations rather than the wasteful 100 vectors. In Table 1, the detailed coloring results are summarized. In many relations, 20 colors were sufficient, i.e., its coloring success rate is $\geq 80\%$. We do not prefer the perfect edge-coloring to prevent over-fitting problems. We found that the number of colors that is enough to color around 80% of edges leads to the best embedding result.

5 Proposed Method to Address Zero Gradient Problem

The ZGP occurs because of the opposite signs of the true and negative triples' energy. The following loss function is free from this problem.

$$\mathcal{L}_{new} = \sum_{t^+ \in \mathcal{E}} \sum_{t^- \in \mathcal{N}(t^+)} \max\left(0, \frac{e(t^+)}{e(t^-)} - \gamma\right), \tag{6}$$

where $\gamma \leq 1$.

The proposed loss function based on ***energy ratio*** requires that a true triple's energy should be at least $\frac{1}{\gamma}$ times smaller than that of a negative triple. The proposed `TransB` uses this loss function with the energy definition in Equation (5).

[6]Given k colors, at least n edges can be colored by the online algorithm, where $n = 0.48 \times m$ and m is the number of edges successfully colored by the optimal algorithm.

[7]This means that there are many low-degree vertices and a small number of very high-degree vertices.

TABLE I: Edge-coloring results of Next-Fit, an online bipartite graph edge-coloring algorithm [7], with $k = 20$. On average, 87.5% and 73.9% of edges were successfully colored with 20 colors. Relations are sorted by the coloring success rate.

$FB15K$			$WN18$		
Relation	Max Degree	Success Rate	Relation	Max Degree	Success Rate
Average Case	-	87.5%	Average Case	-	73.9%
webpage/category	3613	0.67%	member_of_domain_usage	229	31.7%
webpage/topic	3590	0.67%	instance_hyponym	471	56.1%
location/constraints	843	56%	instance_hypernym	473	56.3%
people/lived	332	71.7%	member_of_domain_topic	341	58.9%
Other relations	119	≥80%	Other relations	382	≥82.7%

Input: Knowledge Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathsf{L})$, Max Epoch max_epoch, Initial Learning Rate δ
Output: Feature Matrix M for vertices and relations

1 $\mathsf{M} \leftarrow d \times n$ matrix with random initialization
2 Randomly initialize the weight $\mathbf{W} \in \mathbb{R}^{2d \times d}$ and bias $\mathbf{b} \in \mathbb{R}^d$ of $Linear(\cdot)$
3 **while** $epoch \leq max\_epoch$ **do**
4 $\delta \leftarrow$ search_then_coverage$(\delta, epoch)$
5 $\mathcal{E} \leftarrow$ random_permute$(\mathcal{E})$
6 **foreach** $\mathcal{E}_{batch} \in minibatch\_split(\mathcal{E})$ **do**
7 **foreach** $t^+ \in \mathcal{E}_{batch}$ **do**
8 $\mathcal{N}(t^+) \leftarrow$ sample_negative_triples(t^+)
9 $\mathsf{M} = \mathsf{M} - \delta \times \nabla_{\mathsf{M}} \mathcal{L}_{new}$
10 $\mathbf{W} = \mathbf{W} - \delta \times \nabla_{\mathbf{W}} \mathcal{L}_{new}$
11 $\mathbf{b} = \mathbf{b} - \delta \times \nabla_{\mathbf{b}} \mathcal{L}_{new}$
12 $epoch++$
13 **return** M

Algorithm 1: The proposed KGE algorithm. All red parts are closely related to our contributions.

Theorem 5.1: $\mathcal{L}_{new}$ does not have the problem of zero gradients.

Proof: We sketch the idea of the proof. Whereas $x - y = (x \pm \epsilon) - (y \pm \epsilon)$, $\frac{x}{y}$ is not equal to $\frac{x \pm \epsilon}{y \pm \epsilon}$ even though the difference ϵ is very small. In general, the number of training true-negative triple pairs is large. Thus, the sum of many small non-zero gradients can contribute to the embedding process. ■

`ComplEx` [15] also proposed a new loss function based on log-likelihood — they did not theoretically prove what are existing methods' problems though. However, we theoretically identify what is the most critical problem in the energy margin based loss and our `TransB` based on the proposed energy ratio loss outperforms `ComplEx` in all cases with non-trivial margins.

6 Embedding Algorithm

We present the learning algorithm in Algorithm 1. Key parts are highlighted in red. Let M be a $d \times n$ feature matrix, where d is the dimension of the feature space and $n \leq |\mathcal{V}| + \Delta|\mathcal{E}|$ is the number of vector representations of vertices and relations that we have to learn, i.e., each column of M is a vector representation. In addition to them, we also have to learn one linear layer to convert $\mathbf{v_s}$ into $\mathbf{v_t}$ but its space overheads are negligible in comparison with the space of the vector representations.

We randomly initialize M and the weight and bias of the linear layer (lines 1 and 2). At every epoch, we randomly permute the entire triple list (line 5). This procedure is performed to prevent the learning procedures from being overly fitted to a certain input sequence. After dividing the entire triple list into many smaller mini batches, we create a set of derived negative triples (line 8). At lines 9, 10, and 11, we update parameters to learn based on our proposed methods Finally, we do not use a constant learning rate. The learning rate δ decays every epoch according to the search_then_coverage approach [5].

7 Experiments

In this section, we present our experimental environments and results. We have tested with two popular knowledge graphs and 13 baseline KGE methods (`SE`, `SME`, `TransE`, `TransH`, `TransR`, `lppTransD`, `STransE`, `TransG`, `HolE`, `ComplEx`, `SSP`, `ConvE`, and `IRN`).

7.1 Experimental Environments

7.1.1 Datasets.: We use two standard benchmark datasets. The FB15K dataset contains 483K training triples and 50K test queries for popular 15K vertices and 1.3K relations of `FreeBase`. WN18 contains 31K training triples and 4K test queries for 14K vertices and 18 relations of `WordNet`.

7.1.2 Evaluation metrics.: "Mean Rank" and "Hits@10" are considered as two standard metrics. "Mean Rank" is the average rank of correct answers of all testing queries (thus, a low value is preferred), and "Hits@10" is the ratio of testing queries whose answers are one of the top-10 lowest energy vertices (thus, a high value is preferred).

These two evaluation metrics can be further classified into two types: "raw" and "filtered". "Filtered" means that true triples are not considered when calculating rankings because true triples contained by training or validation sets have low energy. In the filtered metric, therefore, only previously unknown answer candidates are considered. In the raw metric, we consider all answer candidates regardless of whether they are already known or not.

7.1.3 Baseline KGE methods.: We consider many KGE methods, such as `TransH`, `TransR`, `TransD`, `lppTransD`, `TransG`, `TransF`, `HolE`, `ComplEx`, `SSP`, `DistMult`, `ANALOGY`,

TABLE II: Raw Mean Rank and Raw Hits@10 for question answering. The best value is indicated in bold font. The proposed method with various parameter setups are marked in blue. We mainly compare with translation-based methods in this table. Recall that the ZGP occurs frequently in translation-based methods. Thus, we show the efficacy of our new loss function in those translation-based methods.

$FB15K$			$WN18$		
Method	Raw Mean Rank	Raw Hits@10	Method	Raw Mean Rank	Raw Hits@10
SME	274	30.7%	SME	526	54.7%
SE	273	28.8%	SE	1,011	68.5%
TransE	243	34.9%	TransE	263	75.4%
TransE-ZGP	193	48.8%	TransE-ZGP	315	78.52%
TransH	212	45.7%	TransH	318	75.4%
TransH-ZGP	176	49.9%	TransR	232	78.3%
TransR	198	48.2%	TransR-ZGP	219	80.1%
TransG	**152**	54.9%	*lpp*TransD	283	80.5%
TransB-10	193	54.2%	TransB-10	221	81.9%
TransB-20	190	**55.0%**	TransB-20	**218**	**82.9%**

RUGE, RGCN, and so forth. lppTransD is an enhancement of TransD by [21]. In TransH, two slightly different negative triple construction strategies were proposed: "unif" and "bern". We apply both strategies and chose the best one for all TransE's variations. Note that all these baseline methods use the loss function (1) that is weak from the ZGP.

For our TransB, we test the configurations of 10 and 20 vectors to embed a relation. Recall that in Table I coloring with 20 colors (vectors) show good performance in both knowledge graphs. We denote each configuration as TransB-10 or TransB-20.

7.1.4 Hardware and parameter configurations.: We run the experiments on a cluster of machines running Linux with a Xeon 3.4 GHz CPU, 128 GB of RAM, and a Tesla K80 GPU.

We set the feature space dimension as $d = 100$, the initial learning rate as $\delta = 0.1$ for $FB15K$ and $\delta = 0.01$ for $WN18$, and the energy ratio margin as $\gamma = \{0.3, 0.5, 0.8\}$ for our experiments. The decay rate of the search_then_coverage learning rate scheduler is set to 0.000015, and max_epoch is 1000. In this setting, the learning rate is decreased by at least two orders of magnitude around the end of learning procedures.

7.2 Experimental Results

We discuss the key results from various perspectives. Our main target is question answering. Triple classification to predict whether a triple is true or negative is another task to evaluate KGE.

7.2.1 Question answering.: We first discuss the efficacy of the proposed loss Equation (6), which is free from the ZGP. For this purpose, we changed the loss functions of TransE, TransH, TransR, and some other translation-based methods to the proposed one — in this evaluation, we do not compare with non-translation methods because the ZGP frequently occurs for tanslation methods. Note that TransE-ZGP immediately shows significant performance enhancements. In *FB15K*, TransE-ZGP achieves the best Mean Rank of 193 among all baseline methods except TransG and outperforms TransH and TransR for Hits@10 (48.8 of TransE-ZGP vs. 45.7 of TransH and 48.2 of TransR). This result supports that the ZGP had caused serious problems in the existing methods. TransH-ZGP and TransR-ZGP also exhibited non-trivial performance improvements. In Table II, some algorithms are omitted in a dataset because their patterns are the same as in the other dataset. TransB-20 shows the best Hits@10 performance among all methods for the datasets. In particular, it outperforms all other translation-based methods for $WN18$ for both metrics.

In Table III, we compare with advanced baseline methods in terms of filtered metrics. This evaluation includes all of translation and non-translation methods. TransG is one of the most advanced translation methods and ComplEx, HolE, SSP, and so forth are based on their novel architectures different from translation methods. KGE methods removed from Table III are inferior to the listed methods. In general, SSP and IRN shows the best performance among all baseline methods. Our TransB-20 beats them and shows the best performance in half cases.

7.2.2 Triple classification.: Triple classification is to predict whether a test triple is true or false knowledge. lppTransD[8] showed the best performance (85.3% accuracy) among all the baseline methods for $FB15K$, and TransB-20 achieved 89.9% accuracy.

8 Conclusion

We tackled two problems of KGE: non-exclusive relation categories and zero gradients. To address them, we convert a knowledge graph into a bipartite graph without any loss of knowledge representation. We also learn multiple vector representations and adopt a softmax function to determine which of them to use to embed a triple. The optimal number of vector representations is determined after solving the edge-coloring problem in the converted bipartite knowledge graph. Finally, we discover that many gradients are zero in the current

[8]We thank authors of lppTransD for sharing their triple classification dataset.

TABLE III: Filtered Mean Rank and Filtered Hits@10. We show the comparison with all advanced methods, such as TransG, STransE, HolE, ComplEx, SSP, ConvE, and IRN. Note that many of them are not translation-based methods.

Database	Method	Filtered Mean Rank	Filtered Hits@10
$FB15K$	HolE	N/A	73.9%
	ComplEx	N/A	84.0%
	SSP	82	79.0%
	ConvE	64	87.3%
	STransE	69	79.7%
	IRN	38	**92.7%**
	TransG	50	88.2%
	TransF	62	82.3%
	DistMult	N/A	82.4%
	ANALOGY	N/A	85.4%
	RUGE	N/A	86.5%
	RGCN	N/A	82.5%
	RGCN+	N/A	84.2%
	TransB-10	47	89.1%
	TransB-20	**37**	90.9%
$WN18$	HolE	N/A	94.9%
	ComplEx	N/A	94.7%
	SSP	**156**	93.2%
	ConvE	504	95.5%
	STransE	206	93%
	IRN	249	95.3%
	TransG	345	94.9%
	TransF	198	95.3%
	DistMult	N/A	93.6%
	ANALOGY	N/A	94.7%
	RGCN	N/A	95.5%
	RGCN+	N/A	96.4%
	TransB-10	371	94.3%
	TransB-20	356	**96.9%**

energy margin loss function, and we propose an energy ratio loss function that does not have the zero gradient problem.

Throughout experiments following standard evaluation protocols, we proved that the proposed TransB distinctively outperforms other baseline methods for a couple of tasks in all datasets.

ACKNOWLEDGEMENT

This work was supported by the National Research Council of Science & Technology (NST) grant by the Korea government (MSIP) (No. CRC-15-05-ETRI).

This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energys National Nuclear Security Administration under contract DE-NA-0003525.

REFERENCES

[1] A. Bordes, X. Glorot, J. Weston, and Y. Bengio. A semantic matching energy function for learning with multi-relational data - Application to word-sense disambiguation. *Machine Learning*, 94(2):233–259, 2014.

[2] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko. Translating Embeddings for Modeling Multi-relational Data. In *Proceedings of NIPS'13*, pages 2787–2795, 2013.

[3] A. Bordes, J. Weston, R. Collobert, and Y. Bengio. Learning Structured Embeddings of Knowledge Bases. In *Proceedings of AAAI'11*, San Francisco, USA, 2011.

[4] M. Chen and C. Zaniolo. Learning multi-faceted knowledge graph embeddings for natural language processing. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 5169–5170, 2017.

[5] C. Darken and J. Moody. Note on learning rate schedules for stochastic optimization. In *Proceedings of NIPS'90*, pages 832–838, 1990.

[6] K. Do, T. Tran, and S. Venkatesh. Knowledge Graph Embedding with Multiple Relation Projections. *ArXiv e-prints*, Jan. 2018.

[7] M. Favrholdt and N. Nielsen. On-line edge-coloring with a fixed number of colors. *Algorithmica*, 35(2):176–191, 2003.

[8] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of ACL'15*, pages 687–696, 2015.

[9] G. Ji, K. Liu, S. He, and J. Zhao. Knowledge graph completion with adaptive sparse transfer matrix. In *Proceedings of AAAI'16*, AAAI'16, pages 985–991. AAAI Press, 2016.

[10] Y. Jia, Y. Wang, X. Jin, and X. Cheng. Path-specific knowledge graph embedding. *Knowledge-Based Systems*, 151:37 – 44, 2018.

[11] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of AAAI'15*, AAAI'15, pages 2181–2187, 2015.

[12] D. Q. Nguyen. An overview of embedding models of entities and relationships for knowledge base completion. *ArXiv e-prints*, Mar. 2017.

[13] M. Nickel, L. Rosasco, and T. Poggio. Holographic Embeddings of Knowledge Graphs. *ArXiv e-prints*, Oct. 2015.

[14] Y. Shen, P. Huang, M. Chang, and J. Gao. Modeling large-scale structured relationships with shared memory for knowledge base completion. In *Proceedings of the 2nd Workshop on Representation Learning for NLP, Rep4NLP@ACL 2017, Vancouver, Canada, August 3, 2017*, pages 57–68, 2017.

[15] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard. Complex Embeddings for Simple Link Prediction. *ArXiv e-prints*, June 2016.

[16] Q. Wang, Z. Mao, B. Wang, and L. Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

[17] Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of AAAI'14*, pages 1112–1119. AAAI Press, 2014.

[18] H. Xiao, M. Huang, Y. Hao, and X. Zhu. Transg : A generative mixture model for knowledge graph embedding. *CoRR*, abs/1509.05488, 2015.

[19] H. Xiao, M. Huang, L. Meng, and X. Zhu. SSP: semantic space projection for knowledge graph embedding with text descriptions. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3104–3110, 2017.

[20] H. Xiao, M. Huang, and X. Zhu. From one point to a manifold: Knowledge graph embedding for precise link prediction. In *Proceedings of IJCAI'16*, pages 1315–1321. IJCAI/AAAI Press, 2016.

[21] H. Yoon, H. Song, S. Park, and S. Park. A translation-based knowledge graph embedding preserving logical property of relations. In *Proceedings of NAACL'16*, pages 907–916, 2016.