TransINT: Embedding Implication Rules in Knowledge Graphs with Isomorphic Intersections of Linear Subspaces

So Yeon Min
MIT CSAIL
Cambridge, MA 02142
symin95@gmail.com

Preethi Raghavan
IBM Research
Cambridge, MA 02142
praghav@us.ibm.com

Peter Szolovits
MIT CSAIL
Cambridge, MA 02142
psz@mit.edu

Abstract

Knowledge Graphs (KG), composed of entities and relations, provide a structured representation of knowledge. For easy access to statistical approaches on relational data, multiple methods to embed a KG into $f(\text{KG}) \in \mathbb{R}^d$ have been introduced. We propose TransINT, a new KG embedding method that isomorphically preserves the implication ordering among relations in the embedding space. TransINT maps set of entities (tied by a relation) to continuous sets of vectors that are inclusion-ordered isomorphically to relation implications. We further suggest possibilities of learning semantic relatedness among sets of entities with magnitude of angle between the embedded continuous sets.

1 Introduction

Recently, learning distributed vector representations of multi-relational knowledge has become an active area of research [1, 8, 5, 14, 2]. These methods map components of a KG (entities and relations) to elements of $\mathbb{R}^d$and capture statistical patterns, regarding vectors close in distance as representing similar concepts. However, they lack common sense knowledge which are essential for reasoning [13, 4, 7]. For example, “parent" and “father" would be deemed similar by KG embeddings, but by common sense, "father $\Rightarrow$ parent" yet not the other way around.

Thus, one focus of current research is to bring common sense rules to KG embeddings[4, 13, 15]. Some methods impose hard geometric constraints and embed asymmetric orderings of knowledge[7, 11, 12]. However, they only embed hierarchy (unary *Is_a* relations), and cannot embed n-ary relations in KG's. Moreover, their hierarchy learning is largely incompatible with conventional relational learning, because they put hard constraints on distance to represent partial ordering, which is a common metric of similarity/ relatedness in relational learning.

We propose TransINT, a new KG embedding method that isomorphically preserves the implication ordering among relations in the embedding space. With rank-based parameter sharing, TransINT restrict entities tied by a relation to be embedded to vectors in a particular region of R^d. For example, we map any entities tied by *is_father_of* to vectors in a region that is part of the region for *is_parent_of*; so that we can automatically know that if John is a father of Tom, he is also his parent even if such a fact is missing in the KG. Such set-to-region mapping allows using angles between regions as semantic relatedness among sets - an extension of the line of thought as in word/ image embedding methods such as [6], [3] to relational embedding.

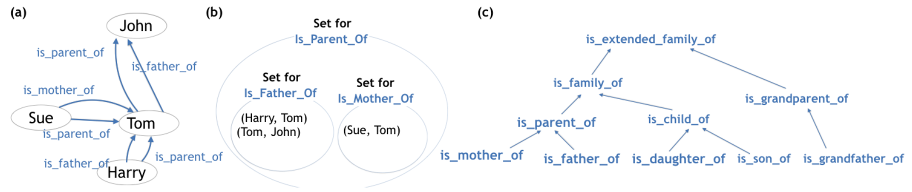

Figure 1: Two equivalent ways of expressing relations. (a): relations defined in a hypothetical KG. (b): relations defined in a set-theoretic perspective (Definition 1). Because *is_father_of* $\Rightarrow$ *is_parent_of*, the set for *is_father_of* is a subset of that for *is_parent_of* (Definition 2). (c): Hierarchical depiction of familial relations.

The main contributions of our work are: (1) A novel KG embedding such that implication rules in the original KG are guaranteed to unconditionally, not approximately, hold. (2) We introduce a novel parameter sharing regularization and negative example construction methods. (3) Our model suggests possibilities of learning semantic relatedness between groups of objects.

2 TransINT

In this section, we describe the intuition and justification of our method. We first define relation as sets, and revisit TransH as mapping relations to sets in $\mathbb{R}^d$. Finally, we propose TransINT, which connects the ordering of the two aforementioned sets. We put * next to definitions and theorems we propose/ introduce. Otherwise, we use existing definitions and cite them.

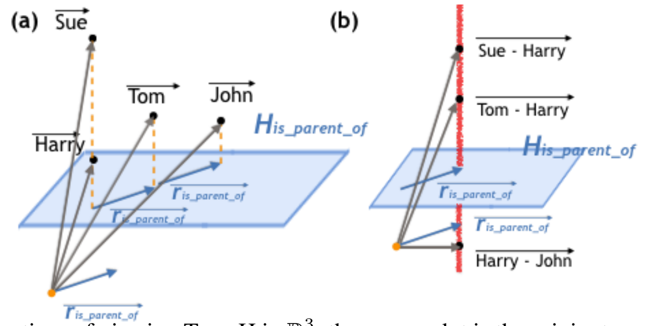
(a)
$\overrightarrow{Sue}$
$\overrightarrow{Tom}$
$\overrightarrow{John}$
$\overrightarrow{Harry}$
$H_{is\_parent\_of}$
$\overrightarrow{r_{is\_parent\_of}}$
$\overrightarrow{r_{is\_parent\_of}}$
$\overrightarrow{r_{is\_parent\_of}}$
(b)
$\overrightarrow{Sue - Harry}$
$\overrightarrow{Tom - Harry}$
$H_{is\_parent\_of}$
$\overrightarrow{r_{is\_parent\_of}}$
$\overrightarrow{r_{is\_parent\_of}}$
$\overrightarrow{Harry - John}$

Figure 2: Two perspectives of viweing TransH in $\mathbb{R}^3$; the orange dot is the origin, to emphasize that a vector is really a point from the origin but can be translated and considered equivalently. (a): first projecting $\vec{h}$ and $\vec{t}$ onto $H_{is\_family\_of}$, and then requiring $\overrightarrow{h_\perp} + \vec{r_j} \approx \overrightarrow{t_\perp}$ (b): first substracting $\vec{t}$ from $\vec{h}$, and then projecting the distance $(\overrightarrow{t-h})$ to $H_{is\_family\_of}$ and requiring $(\overrightarrow{t-h})_\perp \approx r_j$. The red line is unique because it is when $\overrightarrow{r_{is\_family\_of}}$ is translated to the origin.

2.1 Sets as Relations

Relations can be defined as sets.[] For example, in Figure 1a, 1b, *Is_Father_Of(Tom, Harry)* is equivalent to *(Tom, Harry)* $\in \mathbf{R}_{\text{Is\_Father\_Of}}$

Definition *(Relation Set):* Let r_i be a binary relation x, y entities. Then, $r_i(x, y)$ iff there exists some set $\mathbf{R_i}$ such that the pair $(x, y) \in \mathbf{R_i}$. $\mathbf{R_i}$ is called the **relation set** of r_i.[9]

Definition *(Logical Implication):* For two relations, r_1 implies r_2 (or $r_1 \Rightarrow r_2$) iff $\forall x, y$,

$$(x, y) \in \mathbf{R_1} \Rightarrow (x, y) \in \mathbf{R_2}$$

or equivalently,

$$\mathbf{R_2} \subset \mathbf{R_1}.[9]$$

For example, *Is_Father_Of* $\Rightarrow$ *Is_Parent_Of* by common sense. We can see in Figure 1b that, $\mathbf{R_{Is\_Father\_Of}} \subset \mathbf{R_{Is\_Parent\_Of}}$.

2.2 Background: TransE and TransH

Given a fact triple (h, r, t) in a given KG (i.e. *(Harry, is_father_of, Tom)*), TransE wants $\vec{h} + \vec{r} \approx \vec{t}$ where $\vec{h}, \vec{r}, \vec{t}$ are embeddings of h, r, t. In other words, the distance between two entity vectors is equal to a fixed relation vector. TransE applies well to 1-to-1 relations but has issues for N-to-1, 1-to-N and N-to-N relations, because the distance between two vectors are unique and thus two entities can only be tied with one relation.

To address this, TransH constraints the distance of entities in a multi-relational way, by decomposing distance with projection (Figure 2a). TransH first projects an entity vector into a hyperplane unique to each relation, and then requires their difference is some constant value. Like TransE, it embeds an entity to a vector. However, for each relation r_j, it assigns **two** components: a relation-specific hyperplane H_j and a fixed vector $\vec{r_j}$ on H_j. For each fact triple (h, r_j, t), TransH wants (Figure 2)

$$\overrightarrow{h_\perp} + \vec{r_j} \approx \overrightarrow{t_\perp} \cdots\cdots \quad \text{(Eq. 1)}$$

where $\overrightarrow{h_\perp}, \overrightarrow{t_\perp}$ are projections on $\vec{h}, \vec{t}$ onto H_j (Figure 2a).

Revisiting TransH We interpret TransH in a novel perspective. An equivalent way to put Eq.1 is to change the order of subtraction and projection:

$$\text{Projection of } (\overrightarrow{t-h}) \text{ onto } H_j \approx \vec{r_j}.$$

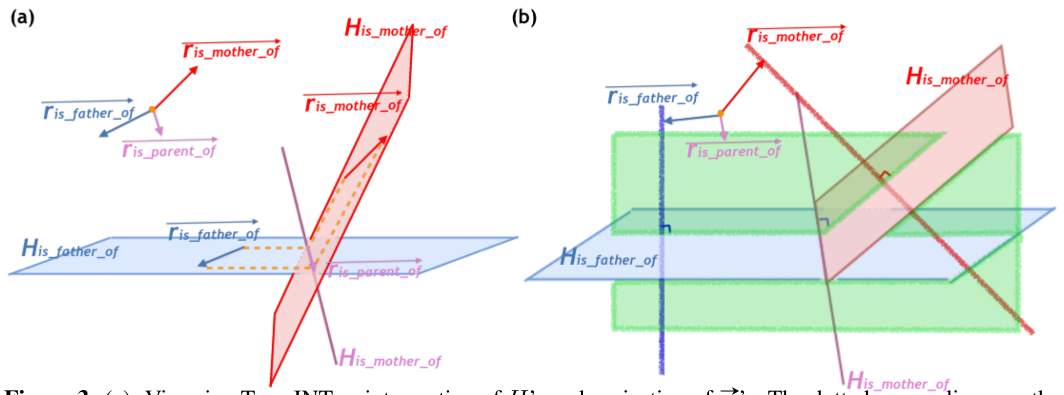
(a)
$H_{is\_mother\_of}$
$\overrightarrow{r_{is\_mother\_of}}$
$\overrightarrow{r_{is\_father\_of}}$
$\overrightarrow{r_{is\_parent\_of}}$
$\overrightarrow{r_{is\_mother\_of}}$
$H_{is\_father\_of}$
$\overrightarrow{r_{is\_father\_of}}$
$\overrightarrow{r_{is\_parent\_of}}$
$H_{is\_mother\_of}$
(b)
$\overrightarrow{r_{is\_mother\_of}}$
$\overrightarrow{r_{is\_father\_of}}$
$H_{is\_mother\_of}$
$\overrightarrow{r_{is\_parent\_of}}$
$H_{is\_father\_of}$
$H_{is\_mother\_of}$

Figure 3: (a): Vieweing TransINT as intersection of H's and projection of $\vec{r}$'s. The dotted orange lines are the projection constraint. (b): Viewing TransINT in the relation space (Figure 2b) perspective. The blue line, red line, and the green plane is respectively *is_father_of, is_mother_of* and *is_parent_of*'s relation space - where $\overrightarrow{t-h}$'s of h, t tied by these relations can exist. The blue and the red line lie on the green plane - *is_parent_of*'s relation space includes the other two's.

This means that all entity vectors $(\vec{h}, \vec{t})$ such that their distance $\overrightarrow{t-h}$ belongs to the red line are considered to be tied by relation r_j (Figure 2b) i.e. $\mathbf{R_j} \approx$ the red line. For example,

$$\textit{(Tom, Sue)} \in \mathbf{R_j} \cong \overrightarrow{(Sue - Tom)} \in \text{the red line}$$

The red line is the set of all vectors whose projection onto H_j is the fixed vector $\vec{r_j}$. Thus, **TransH actually embeds a relation set in KG (figure 1b) to a continuous set in** $\mathbb{R}^d$. We call such sets **relation space**; in other words, a **relation space** of some relation r_i is the space where each (h, r_i, t)'s $\overrightarrow{h-t}$ can exist

Thus, in TransH,:

$$\begin{aligned} r_i(x, y) &\equiv (x, y) \in \mathbf{R_i} && \textbf{(relation in KG)} \\ &\cong (x - y) \in \text{relation space of } r_i && \textbf{(relation in } \mathbb{R}^\mathbf{d}\textbf{)} \end{aligned}$$

2.3 TransINT

Like TransH, TransINT embeds a relation r_j to a (subspace, vector) pair $(H_j, \vec{r_j})$. However, TransINT modifies the relation embeddings $(H_j, \vec{r_j})$ so that the relation spaces (i.e. red line of Figure 2b) are ordered by implication; we do so by intersecting the H_j's and projecting the $\vec{r_j}$'s (Figure 3a).

More generally, TransINT requires two hard geometric constraints on $(H_j, \vec{r_j})$'s that

For distinct relations r_i, r_j, require the following if and only if $r_i \Rightarrow r_j$:
Intersection Constraint: $H_j = H_i \cap H_j$.
Projection Constraint: Projection of $\vec{r_1}$ to H_j is $\vec{r_j}$.

where $\vec{H_i}, \vec{H_j}$ and $\vec{r_i}, \vec{r_j}$ are distinct. For example, we assign

$$H_{is\_parent\_of} = H_{is\_father\_of} \cap H_{is\_mother\_of}.$$

and require $\overrightarrow{r_{is\_father\_of}}$ and $\overrightarrow{r_{is\_mother\_of}}$ to have the same projection onto $H_{is\_parent\_of}$ (orange dottedd line of Figure 3a), which is $\overrightarrow{r_{is\_parent\_of}}$.

Our main result is that the two above constraints order relation spaces by inclusion isomorphically to implication ordering. Figure 3b graphically illustrates that *is_parent_of*'s relation space (green hyperplane) includes those of *is_father_of* (blue line) and *is_mother_of* (red line).[1]

In other words, the two constraints guarantee that an ordering isomorphic to implication holds in the embedding space: $\boxed{(r_i \Rightarrow r_j) \text{ iff } (r_i\text{'s rel. space} \subset r_j\text{'s rel. space})}$ For example, if *is_parent_of* subsumes *is_father_of*, the first relation space includes the latter's; the converse also holds (Figure 3). At first sight, it may look paradoxical that the H_j's and the relation spaces are inversely ordered; however, it is a natural consequence of the rank-based geometry in $\mathbb{R}^d$.

Main Theorem 1 (Isomorphism):* Let $\{(H_i, \vec{r_i})\}_n$ be the (subspace, vector) embeddings assigned to relations $\{\mathbf{R_i}\}_n$ by the *Intersection Constraint* and *the Projection Constraint*; P_i the projection matrix of H_i. Then, $(\{Sol(P_i, \vec{r_i})\}_n, \subset)$ is isomorphic to $(\{\mathbf{R_i}\}_n, \subset)$.

[1]Proofs and details are present in the appendix

In actual implementation and training, TransINT requires something less strict than $P_i(\overrightarrow{t-h}) = \vec{r_i}$:

$$P_i(\overrightarrow{t-h}) - \vec{r_i} \approx \vec{0} \equiv ||P_i(\overrightarrow{t-h} - \vec{r_i})||_2 < \epsilon,$$

for some non-negative and small ϵ. This bounds $\overrightarrow{t-h} - \vec{r_i}$ to regions with thickness 2ϵ, centered around $Sol(P_i, \vec{r_i})$ (Figure 6). We prove that isomorphism still holds with this weaker requirement.

Definition* $(Sol_\epsilon(P, k))$: Given a projection matrix P, the solution space of $||P\vec{x} - \vec{k}||_2 < \epsilon$ is denoted as $\mathbf{Sol}_\epsilon(\mathbf{P}, \vec{k})$.

Main Theorem 2* *(Margin-aware Isomorphism)**: For all non-negative scalar ϵ, $(\{Sol_\epsilon(P_i, \vec{r_i})\}_n, \subset)$ is isomorphic to $(\{\mathbf{R_i}\}_n, \subset)$.

To intuitively explain our formulation, the lesser constraint the space to be projected onto, the more freedom a vector is given; which is analogous to that, for example, is_family_of puts more freedom on who can be tied by it than is_father_of. (Figure 1b).

2.4 Training TransINT

We construct negative examples (wrong fact triplets) and train with a margin-based loss, as in TransE and TransH. We introduce a novel parameter sharing scheme to tie the basis of the H_r's. This scheme automatically makes traversing through a particular triple also execute training with all triple that it implies (even if that triple itself is missing in the KG), eliminating the need to manually create missing triples that are true by implication rule. We discuss details in the appendix.

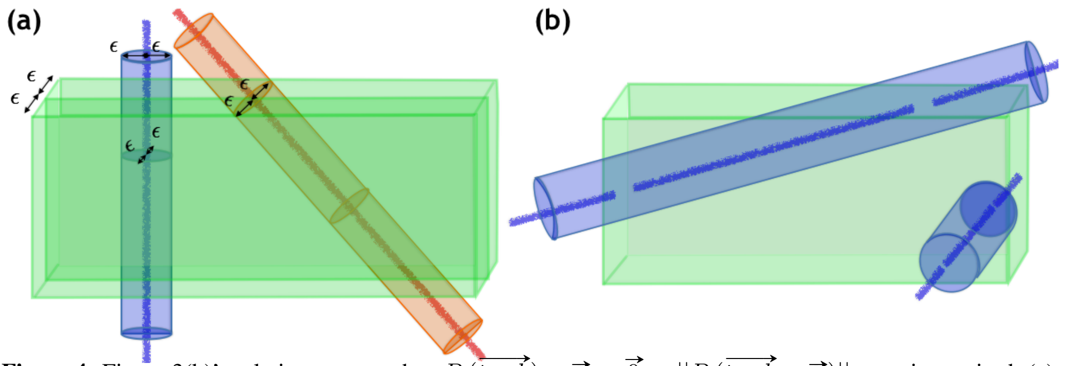

Figure 4: Figure 3(b)'s relation spaces when $P_i(\overrightarrow{t-h}) - \vec{r_i} \approx \vec{0} \equiv ||P_i(\overrightarrow{t-h} - \vec{r_i})||_2 < \epsilon$ is required. (a): Each relation space now becomes regions with thickness ϵ, centered around figure 3(b)'s relation space. (b): Relationship of the angle and area of overlap between two relation spaces. With respect to the green region, the nearly perpendicular cylinder overlaps much less with it than the other cylinder with much closer angle.

3 Future Work: Semantically Interpreting Overlap Between Regions

Traditional embedding methods that map an object (i.e. words, images) to a singleton vector learn soft tendencies between embedded vectors, such as semantic similarity [6], [3]. A common metric for such tendency is cosine similarity, or angle between two embddings. TransINT extends such line of thought to semantic relatedness between groups of objects - as angle between regions of embeddings. In Figure 6b, one can observe that the closer the angle between two regions, the larger the overlap in area. For entities h and t to be tied by both relations r_1, r_2, $t - h$ has to belong to the intersection of their relation spaces. We hypothesize that if the angle between two relation spaces is near 0, then two relations are semantically very similar. For example, relations such as *was_born_in* and *has_citizenship_of* do not imply one another but are closely related. In such a case, both a majority of V_1 and V_2 belong to the intersection. Conversely, if the angle between two relations are near 0, the area of the intersection takes up the majority of both relation spaces. We will investigate this as an ongoing work.

4 Conclusion

We presented a new KG embedding method that embed sets of entities (tied by relations) to continuous sets in $\mathbb{R}^d$. We show that the inclusion ordering of the embedded sets are isomorphic to the implication ordering maong relations. We further suggest possibilities of applying our model for mining semantic similairty among sets of entities.

References

[1] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 2787–2795, USA, 2013. Curran Associates Inc.

[2] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In *AAAI*, 2011.

[3] Andrea Frome, Gregory S. Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. In *NIPS*, 2013.

[4] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. Jointly embedding knowledge graphs and logical rules. In *EMNLP*, 2016.

[5] Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 4284–4295. Curran Associates, Inc., 2018.

[6] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

[7] Maximilian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *NIPS*, 2017.

[8] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *ICML*, 2011.

[9] Robert Roth Stoll. *Set theory and logic*. Courier Corporation, 1979.

[10] Gilbert Strang. *Linear algebra and its applications*. Thomson, Brooks/Cole, Belmont, CA, 2006.

[11] Ivan Vendrov, Jamie Ryan Kiros, Sanja Fidler, and Raquel Urtasun. Order-embeddings of images and language. *CoRR*, abs/1511.06361, 2015.

[12] Luke Vilnis, Xiang Li, Shikhar Murty, and Andrew McCallum. Probabilistic embedding of knowledge graphs with box lattice measures. *arXiv preprint arXiv:1805.06627*, 2018.

[13] Quan Wang, Bin Wang, and Li Guo. Knowledge base completion using embeddings and rules. In *IJCAI*, 2015.

[14] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, 2014.

[15] Zhuoyu Wei, Jun Zhao, Kang Liu, Zhenyu Qi, Zhengya Sun, and Guanhua Tian. Large-scale knowledge base completion: Inferring via grounding network sampling over selected instances. In *CIKM*, 2015.

Appendix

A Proof For TransINT's Isomorphic Guarantee

Here, we provide the proofs for Main Theorems 1 and 2. We also explain some concepts necessary in explaining the proofs. We put * next to definitions and theorems we propose/ introduce. Otherwise, we use existing definitions and cite them.

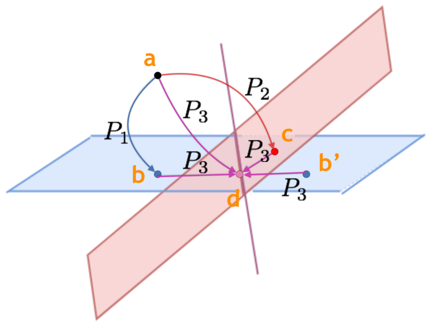

Figure 5: Projection matrices of subspaces that include each other.

A.1 Linear Subspace and Projection

We explain in detail elements of $\mathbb{R}^d$ that were intuitively discussed. In this and later sections, we mark all lemmas and definitions that we newly introduce with *; those not marked with * are accompanied by reference for proof. We denote all $d \times d$ matrices with capital letters (ex) A) and vectors with arrows on top (ex) $\overrightarrow{b}$).

A.1.1 Linear Subspace and Rank

The linear subspace given by $A(x - \overrightarrow{b}) = 0$ (A is $d \times d$ matrix and $b \in \mathbb{R}^d$) is the set of $x \in \mathbb{R}^d$ that are solutions to the equation; its rank is the number of constraints $A(x - \overrightarrow{b}) = 0$ imposes. For example, in $\mathbb{R}^3$, a hyperplane is a set of $\overrightarrow{x} = [\,x_1, x_2, x_3]\, \in \mathbb{R}^3$ such that $ax_1 + bx_2 + cx_3 - d = 0$ for some scalars a, b, c, d; because vectors are bound by one equation (or its "A" only really contains one effective equation), a hyperplane's rank is 1 (equivalently $rank(A) = 1$). On the other hand, a line in $\mathbb{R}^3$ imposes to 2 constraints, and its rank is 2 (equivalently $rank(A) = 2$).

Consider two linear subspaces H_1, H_2, each given by $A_1(\overrightarrow{x} - \overrightarrow{b_1}) = 0, A_2(\overrightarrow{x} - \overrightarrow{b_2}) = 0$. Then,

$$(H_1 \subset H_2) \Leftrightarrow (A_1(\overrightarrow{x} - \overrightarrow{b_1}) = 0 \Rightarrow A_2(\overrightarrow{x} - \overrightarrow{b_2}) = 0)$$

by definition. In the rest of the paper, denote H_i as the linear subspace given by some $A_i(\overrightarrow{x} - \overrightarrow{b_i}) = 0$.

A.1.2 Properties of Projection

Invariance For all $\overrightarrow{x}$ on H, projecting $\overrightarrow{x}$ onto H is still $\overrightarrow{x}$; the converse is also true.

Lemma 1 $P\overrightarrow{x} = \overrightarrow{x} \Leftrightarrow \overrightarrow{x} \in H$ [10].

Orthogonality Projection decomposes any vector $\overrightarrow{x}$ to two orthogonal components - $P\overrightarrow{x}$ and $(I - P)\overrightarrow{x}$ (Figure 4). Thus, for any projection matrix P, $I - P$ is also a projection matrix that is orthogonal to P (i.e. $P(I - P) = 0$) [10].

Lemma 2 Let P be a projection matrix. Then $I - P$ is also a projection matrix such that $P(I - P) = 0$ [10].

The following lemma also follows.

Lemma 3 $||P\overrightarrow{x}|| \leq ||P\overrightarrow{x} + (I - P)\overrightarrow{x}|| = ||\overrightarrow{x}||$ ([10]).

Projection onto an included space If one subspace H_1 includes H_2, the order of projecting a point onto them does not matter. For example, in Figure 3, a random point $\overrightarrow{a}$ in R^3 can be first projected onto H_1 at $\overrightarrow{b}$, and then onto H_3 at $\overrightarrow{d}$. On the other hand, it can be first projected onto H_3 at $\overrightarrow{d}$, and then onto H_1 at still $\overrightarrow{d}$. Thus, the order of applying projections onto spaces that includes one another does not matter.

If we generalize, we obtain the following two lemmas (Figure 6):

Lemma 4* Every two subspaces $H_1 \subset H_2$ if and only if $P_1P_2 = P_2P_1 = P_1$.

proof) By Lemma 1, if $H_1 \subset H_2$, then $P_2\vec{x} = \vec{x} \quad \forall \vec{x} \in H_1$. On the other hand, if $H_1 \not\subset H_2$, then there is some $\vec{x} \in H_1, \vec{x} \notin H_2$ such that $P_2\vec{x} \neq \vec{x}$. Thus,

$$\begin{aligned} H_1 \subset H_2 &\Leftrightarrow \forall \vec{x} \in H_1, \quad P_2\vec{x} = \vec{x} \\ &\Leftrightarrow \forall \vec{y}, \quad P_2(P_1\vec{y}) = P_1\vec{y} \Leftrightarrow P_2P_1 = P_1. \end{aligned}$$

Because projection matrices are symmetric ([10]),

$$P_2P_1 = P_1 = {P_1}^T = {P_1}^T{P_2}^T = P_1P_2. \blacksquare$$

Lemma 5* For two subspaces H_1, H_2 and vector $\vec{k} \in H_2$,

$$H_1 \subset H_2 \Leftrightarrow Sol(P_2, \vec{k}) \subset Sol(P_1, P_1\vec{k}).$$

proof) $Sol(P_2, \vec{k}) \subset Sol(P_1, P_1\vec{k})$ is equivlaent to $\forall \vec{x} \in \mathbb{R}^d, P_2\vec{x} = \vec{k} \Rightarrow P_1\vec{x} = P_1\vec{k}$.

By Lemma 4, if $H_1 \subset H_2 \Leftrightarrow P_1P_2 = P1$. Since $\vec{k} \in P_2$, $P_2\vec{x} = \vec{k} \Leftrightarrow P_2(x - \vec{k}) = \vec{0} \Leftrightarrow P_1(P_2\vec{x} - \vec{k}) = \vec{0} \Leftrightarrow P_1P_2\vec{x} = P_1\vec{k} \Leftrightarrow P_1\vec{x} = P_1\vec{k}. \blacksquare$

Partial ordering If two subspaces strictly include one another, projection is uniquely defined from lower rank subspace to higher rank subspace, but not the other way around. For example, in Figure 3, a point $\vec{a}$ in R^3 (rank 0) is always projected onto H_1 (rank 1) at point $\vec{b}$. Similarly, point $\vec{b}$ on H_1 (rank 1) is always projected onto similarly, onto H_3 (order 2) at point d. However, "inverse projection" from H_3 to H_1 is not defined, because not only $\vec{b}$ but other points on H_1 (such as $\vec{b'}$) project to H_3 at point $\vec{d}$; these points belong to $Sol(P_3, \vec{d})$. In other words, $Sol(P_1, \vec{b}) \subset Sol(P_3, \vec{d})$. This is the key intuition for isomorphism , which we prove in the next chapter.

A.2 Proof for Isomorphism

Now, we prove that TransINT's two constraints (section 2.3) guarantee isomorphic ordering in the embedding space.

Two posets are isomorphic if their sizes are the same and there exists an order-preseving mapping between them.[] Thus, any two posets $(\{A_i\}_n, \subset), (\{B_i\}_n, \subset)$ are isomorphic if $|\{A_i\}_n| = |\{B_i\}_n|$ and

$$\forall i, j \quad A_i \subset A_j \Leftrightarrow B_i \subset B_j$$

***Main Theorem 1** (Isomorphism):* Let $\{(H_i, \vec{r_i})\}_n$ be the (subspace, vector) embeddings assigned to relations $\{\mathbf{R_i}\}_n$ by the *Intersection Constraint* and *the Projection Constraint*; P_i the projection matrix of H_i. Then, $(\{Sol(P_i, \vec{r_i})\}_n, \subset)$ is isomorphic to $(\{\mathbf{R_i}\}_n, \subset)$.

proof) Since each $Sol(P_i, \vec{r_i})$ is distinct and each $\mathbf{R_i}$ is assigned exactly one $Sol(P_i, \vec{r_i})$, $|\{Sol(P_i, \vec{r_i})\}_n| = |\{I_i\}_n|$.①

Now, let's show

$$\forall i, j, \quad R_i \subset R_j \Leftrightarrow Sol(P_i, \vec{r_i}) \subset Sol(P_j, \vec{r_j}).$$

Because the $\forall i, j$, intersection and projection constraints are true iff $R_i \subset R_j$, enough to show that the two constraints hold iff $Sol(P_i, \vec{r_i}) \subset Sol(P_j, \vec{r_j}$.

First, let's show $\mathbf{R_i} \subset \mathbf{R_i} \Rightarrow Sol(P_i, \vec{r_i}) \subset Sol(P_j, \vec{r_j})$. From the *Intersection Constraint*, $\mathbf{R_i} \subset \mathbf{R_i} \Rightarrow H_j \subset H_i$. By Lemma 5, $Sol(P_i, \vec{r_i}) \subset Sol(P_j, P_j\vec{r_i})$. From the *Projection Constraint*, $\vec{r_j} = P_j\vec{r_i}$. Thus, $Sol(P_i, \vec{r_i}) \subset Sol(P_j, P_j\vec{r_i}) = Sol(P_j, \vec{r_j}). \cdots\cdots$ ②

Now, let's show the converse; enough to show that if $Sol(P_i, \vec{r_i}) \subset Sol(P_j, \vec{r_j})$, then the intersection and projection constraints hold true.

$$\begin{aligned} &Sol(P_i, \vec{r_i}) \subset Sol(P_j, \vec{r_j}) \\ &\Leftrightarrow \forall \vec{x}, \quad P_i\vec{x} = \vec{r_i} \Rightarrow P_j\vec{x} = \vec{r_j}) \end{aligned}$$

If $P_i\vec{x} = \vec{r_i}$,

$$\begin{aligned} \forall \vec{x}, \quad P_jP_i\vec{x} = P_j\vec{r_i} \\ \forall \vec{x}, \quad P_j\vec{x} = \vec{r_j} \end{aligned}$$

both have to be true. For any $\vec{x} \in H_i$, or equivalently, if $\vec{x} = P_i\vec{y}$ for some $\vec{y}$, then the second equation becomes $\forall\vec{y}, \quad P_jP_i\vec{y} = \vec{r_j}$, which can be only compatible with the first equation if $\vec{r_j} = P_j\vec{r_i}$, since any vector's projection onto a subspace is unique. (Projection Constraint)

Now that we know $\vec{r_j} = P_j\vec{r_i}$, by Lemma 5, $H_i \subset H_j$ (intersection constraint).$\cdots$ ③ From ①, ②, ③, the two posets are isomorphic.$\blacksquare$

In actual implementation and training, TransINT requires something less strict than $P_i(\overrightarrow{t-h}) = \vec{r_i}$:

$$P_i(\overrightarrow{t-h}) - \vec{r_i} \approx \vec{0} \equiv ||P_i(\overrightarrow{t-h-\vec{r_i}})||_2 < \epsilon,$$

for some non-negative and small ϵ. This bounds $\overrightarrow{t-h-\vec{r_i}}$ to regions with thickness 2ϵ, centered around $Sol(P_i, \vec{r_i})$ (Figure 5). We prove that isomorphism still holds with this weaker requirement.

Definition* $(Sol_\epsilon(P, k))$: Given a projection matrix P, the solution space of $||P\vec{x} - \vec{k}||_2 < \epsilon$ is denoted as $\mathbf{Sol}_\epsilon(\mathbf{P}, \vec{k})$.

Main Theorem 2 (Margin-aware Isomorphism): For all non-negative scalar ϵ, $(\{Sol_\epsilon(P_i, \vec{r_i})\}_n, \subset)$ is isomorphic to $(\{\mathbf{R_i}\}_n, \subset)$.

proof) Enough to show that $(\{Sol_\epsilon(P_i, \vec{r_i})\}_n, \subset)$ and $(\{Sol(P_i, \vec{r_i})\}_n, \subset)$ are isomorphic for all ϵ.

First, let's show

$$Sol(P_i, \vec{r_i}) \subset Sol(P_j, \vec{r_j}) \Rightarrow Sol_\epsilon(P_i, \vec{r_i}) \subset Sol_\epsilon(P_j, \vec{r_j}).$$

By Main Theorem 1 and Lemma 4,

$$Sol(P_i, \vec{r_i}) \subset Sol(P_j, \vec{r_j}) \Leftrightarrow \vec{r_j} = P_j\vec{r_i}, P_j = P_jP_i.$$

Thus, for all vector $\vec{b}$,

$$\begin{aligned}
&P_i(x - \vec{r_i}) = \vec{b} \\
&\Leftrightarrow P_jP_i(\vec{x} - \vec{r_i}) = P_j\vec{b} \\
&\Leftrightarrow P_j(\vec{x} - \vec{r_i}) = P_j\vec{b} (\because \text{Lemma 4}) \\
&\Leftrightarrow P_j(\vec{x} - \vec{r_j}) = P_j\vec{b} (\because P_j\vec{r_j} = \vec{r_j} = P_j\vec{r_i})
\end{aligned}$$

Thus, if $||P_i(\vec{x} - \vec{r_i})|| < \epsilon$, then $||P_j(\vec{x} - \vec{r_j})|| = ||P_j(P_i(\vec{x} - \vec{r_i}))|| < ||P_j(P_i(\vec{x} - \vec{r_i})) + (I - P)(P_i(\vec{x} - \vec{r_i}))|| = ||P_i(\vec{x} - \vec{r_i})|| < \epsilon$. $\cdots$ ①

Now, let's show the converse. Assume $||P_i(\vec{x} - \vec{r_i})|| < \epsilon$ for some i. Then,

Thus, for $||P_i(\vec{x} - \vec{r_i})|| < \epsilon$ to bound $||P_j(\vec{x} - \vec{r_j})||$ at all for all $\vec{x}$,

$$P_j(I - P_i) = 0, P_j(\vec{r_i} - \vec{r_j}) = 0$$

need to hold. By Lemma 4 and 5,

$$\begin{aligned}
&P_j = P_jP_i \Leftrightarrow H_j \subset H_i \\
&\Leftrightarrow Sol(P_i, \vec{r_i}) \subset Sol(P_j, P_j\vec{r_i}) = Sol(P_j, \vec{r_j}) \cdot\cdot ②
\end{aligned}$$

$|\{Sol_\epsilon(P_i, \vec{r_i})\}_n| = |\{Sol(P_i, \vec{r_i})\}_n|$ holds obviously; each $Sol(P_i, \vec{r_i})$ has a distinct $Sol_\epsilon(P_i, \vec{r_i})$ and each $Sol_\epsilon(P_i, \vec{r_i})$ also has a distinct "center" $(Sol(P_i, \vec{r_i})) \cdot\cdot$ ③

From ①, ②, ③, the two sets are isomorphic. $\blacksquare$

B Initialization and Training

The intersection and projection constraints can be imposed with parameter sharing. We describe how shared parameters are initialized and trained.

B.1 Parameter Sharing Initializaion

From initialization, we bind parameters so that they satisfy the two constraints. For each entity e_j, we assign a d-dimensional vector $\vec{e_j}$. To each $\mathbf{R}_i$, we assign $(H_i, \vec{r_i})$ (or $(A_i, \vec{r_i})$) with parameter sharing. We first construct the H's.

Intersection constraint We define the H's top-down, first defining the intersections and then the subspaces that go through it. To the head $\mathbf{R}_h$, assign a_h linearly independent rows for the basis of H_h. Then, to each $\mathbf{R}_i$ that is not a head, additionally assign a_i rows linearly independent to the bases of all of its parents, and construct H_i with its bases and the bases of all of its parents. Projection matrices can be uniquely constructed given the bases ([10]).

Now, we initlialize the $\vec{r_i}$'s.

Projection Constraint To the head $\mathbf{R}_h$, pick any random $x_h \in \mathbb{R}^d$ and assign $\vec{r_h} = P_h x$. To each non-head $\mathbf{R}_i$ whose parent is $\mathbf{R}_p$, assign $\vec{r_i} = \vec{r_p} + (I - P_p)(P_i)x_i$ for some random x_i. This results in

$$P_p\vec{r_i} = P_p\vec{r_p} + P_p(I - P_p)(P_i)\vec{x_i} = \vec{r_p} + \vec{0} = \vec{r_p}$$

for any parent, child pair.

B.1.1 Training

We construct negative examples (wrong fact triplets) and train with a margin-based loss, following the same protocols as in TransE and TransH.

Training Objective We adopt the same loss function as in TransH. For each fact triplet (h, r_i, t), we define the score function

$$f(h, r_i, t) = ||P_i(\overrightarrow{t-h}) - \vec{r_i}||_2$$

and train a margin-based loss L which is aggregates f's and discriminates between correct and negative examples.

$$L = \sum_{(h,r_i,t)\in G} max(0, f(h, r_i, t)^2 + \gamma - f(h', r_i', t')^2)$$

where G is the set of all triples in the KG and (h', r_i', t') is a negative triple made from corrupting (h, r_i, t). We minimize this objective with stochastic gradient descent.

Automatic Grounding of Positive Triples The parameter sharing scheme guarantees two advantages during all steps of training. First, the intersection and projection constraint are met not only at initialization but always.

Second, traversing through a particular (h, r_i, t) also automatically executes training with (h, r_p, t) for any $r_i \Rightarrow r_p$. For example, by traversing *(Tom, is_father_of, Harry)* in the KG, the model automatically also traverses *(Tom, is_parent_of, Harry)*, *(Tom, is_family_of, Harry)*, even if the two triples are missing in the KG. This is because $P_pP_i = P_p$ with the given initialization (section 4.1.1) and thus,

$$f(h, r_p, t) = ||P_p(\overrightarrow{t-h}) - \vec{r_p}||_2{}^2 = ||P_p(P_i((\overrightarrow{t-h}) - \vec{r_i}))||_2{}^2$$
$$\leq ||(P_p + (I - P_p))P_i((\overrightarrow{t-h}) - \vec{r_i}))||_2{}^2 = ||(P_i((\overrightarrow{t-h}) - \vec{r_i}))||_2{}^2 = f(h, r_i, t)$$

In other words, training $f(h, r_i, t)$ towards less than ϵ automatically guarantees, or has the effect of training $f(h, r_p, t)$ towards less than ϵ. This enables the model to be automatically trained with what exists in the KG, eliminating the need to manually create missing triples that are true by implication rule.