Beyond Triplets: Hyper-Relational Knowledge Graph Embedding for Link Prediction

Paolo Rosso
University of Fribourg, Switzerland

Dingqi Yang*
University of Fribourg, Switzerland
University of Macau, SAR China
{firstname.lastname}@unifr.ch

Philippe Cudré-Mauroux
University of Fribourg, Switzerland

ABSTRACT

Knowledge Graph (KG) embeddings are a powerful tool for predicting missing links in KGs. Existing techniques typically represent a KG as a set of triplets, where each triplet (h, r, t) links two entities h and t through a relation r, and learn entity/relation embeddings from such triplets while preserving such a structure. However, this triplet representation oversimplifies the complex nature of the data stored in the KG, in particular for hyper-relational facts, where each fact contains not only a base triplet (h, r, t), but also the associated key-value pairs (k, v). Even though a few recent techniques tried to learn from such data by transforming a hyper-relational fact into an n-ary representation (i.e., a set of key-value pairs only without triplets), they result in suboptimal models as they are unaware of the triplet structure, which serves as the fundamental data structure in modern KGs and preserves the essential information for link prediction. To address this issue, we propose HINGE, a hyper-relational KG embedding model, which directly learns from hyper-relational facts in a KG. HINGE captures not only the primary structural information of the KG encoded in the triplets, but also the correlation between each triplet and its associated key-value pairs. Our extensive evaluation shows the superiority of HINGE on various link prediction tasks over KGs. In particular, HINGE consistently outperforms not only the KG embedding methods learning from triplets only (by 0.81-41.45% depending on the link prediction tasks and settings), but also the methods learning from hyper-relational facts using the n-ary representation (by 13.2-84.1%).

CCS CONCEPTS

• **Computing methodologies** → **Natural language processing**; **Knowledge representation and reasoning**.

KEYWORDS

Knowledge graph embedding, Hyper-relation, Link prediction

ACM Reference Format:
Paolo Rosso, Dingqi Yang, and Philippe Cudré-Mauroux. 2020. Beyond Triplets: Hyper-Relational Knowledge Graph Embedding for Link Prediction. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3366423.3380257

*Corresponding author

WWW '20, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380257

1 INTRODUCTION

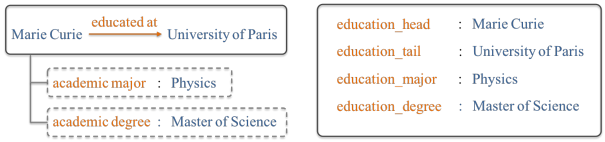
Knowledge Graphs (KGs), such as Freebase [2], Google's Knowledge Graph [12] or Wikidata [40], have become a key asset powering a large range of Web applications ranging from semantic search [41] to question-answering [46], query expansion [13], or recommendation systems [48]. KGs are typically represented through a set of triplets; each triplet *head, relation, tail*, or (h, r, t) for short, encodes a relation connecting a head entity and a tail entity, such as *Switzerland* (head) *hasCurrency* (relation) *Swiss franc* (tail). While modern KGs typically contain rich and high-quality data, they are also known to suffer from an incompleteness issue, i.e., missing facts. For example, 71% of all people from Freebase have no *place of birth* [26], even though this is a mandatory property of the schema [39]. Against this background, Knowledge Graph completion problems have been widely studied. A central problem in this context is to predict the missing links in a KG (a.k.a. link prediction). More precisely, given two elements of a triplet, the task is to predict the missing one, such as $(?, r, t)$, $(h, ?, t)$ or $(h, r, ?)$, where the question mark represents the missing entity/relation.

In the current literature, Knowledge Graph embeddings have been shown as a powerful tool for such link predictions [35]. The key idea of KG embeddings is to learn a latent (and low-dimensional) vector representation of entities/relations (i.e., entity/relation embeddings) from a set of triplets in a KG, while preserving the essential information for link prediction in the KG. For example, TransE [3], a typical KG embedding technique, models a relation as a vector-plus operation between two entities $h+r \approx t$; subsequently, when predicting the missing links, a new fact can be asserted by evaluating $||h + r - t||$.

Despite its broad adoption, the *triple-based* representation of a KG often oversimplifies the complex nature of the data stored in the KG, in particular for hyper-relational data (a.k.a. multi-fold [38] or n-ary [14] relational data), where each fact contains multiple relations and entities. Figure 1(a) shows an example about Marie Curie's education from Wikidata: it contains a base triplet: (h, r, t) {*Marie Curie*, *educated at*, *University of Paris*}, as well as further information associated with the triplet, represented as key-value (relation-entity) pairs[1] (k, v) including {*academic major*, *physics*}, {*academic degree*, *Master of Science*}, etc.

Such hyper-relational data is ubiquitous in KGs. Taking Freebase as an example, more than 30% of its entities are involved in such hyper-relational facts [38]. When learning KG embeddings, those hyper-relational facts have to be transformed into triplets by either 1) keeping the base triplet only from a hyper-relational

[1]We use the term key-value (k, v) denoting a relation-entity pair here to emphasize its difference from the triplet (h, r, t), even though h, t and v are entities while r and k are relations.



(a) A hyper-relational fact (b) N-ary representation of the fact

Figure 1: An example of a hyper-relational fact and its corresponding n-ary representation

fact, e.g., (h, r, t) in the above example, causing irreversible information loss; 2) creating additional triplets from a hyper-relational fact via reification [4], where an artificial entity is used to represent the base triplet $q := (h, r, t)$ and additional triplets are created as (q, k, v); or 3) creating additional triplets from a hyper-relational fact via relation paths [38], where we link h to v via an artificial relation $\widehat{rk}$ representing a relation path $r \rightarrow k$, resulting in additional triplets $(h, \widehat{rk}, v)$. Although the latter two transformations preserve the complete information of a hyper-relational fact, the extra entities/relations they artificially introduce confuse KG embeddings methods, preventing them from capturing key structural properties of the original input graph (see Section 4.3 for more detail).

Against this background, in this paper, we investigate the problem of hyper-relational Knowledge Graph embedding. In the current literature, a few recent studies consider such hyper-relational data [14, 38, 49]. These works consider a set of relations as a so-called n-ary (or multi-fold) relation, while the associated entities then become instances of that relation. Figure 1(b) shows an n-ary representation of the above example about Marie Curie. An n-ary relation "education" is extracted from the hyper-relational fact, containing the following four relations: *education_head*, *education_tail*, *education_major* and *education_degree*; the hyper-relational fact is then represented as a set of key-value (relation-entity) pairs only, i.e., {*education_head*:*Marie Curie*, *education_tail*:*University of Paris*, *education_major*:*Physics*, *education_degree*:*Master of Science*}. Subsequently, using such an n-ary representation, existing approaches to link prediction either learn to model the relatedness of entities [38, 49], or learn from the relatedness between entity/relation pairs [14]. However, the n-ary representation of a hyper-relational fact (as a set of key-value pairs without triplets) treats each key-value pair in the fact equally, which is not compatible with the schema used by modern KGs. Specifically, as triplets still serve as the fundamental data structure in modern KGs, they preserve the essential information for link prediction. In other words, key-value pairs (k, v) on a hyper-relational fact should not be treated identically as base triplet (h, r, t). In our experiments, we conduct a hypothesis test and experimentally show that embeddings learnt from the original base triplets (from a set hyper-relational facts) consistently and significantly outperform (by 58.3-125.27%) embeddings learnt from the same number of triplets created by a *null model*, where one triplet is extracted from the n-ary representation of each hyper-relational fact (represented a set of key-value pairs) via a randomly sampled relation path [38] (see Section 4.2 for more detail). Such an observation suggests that it is highly beneficial to directly capture the structure of the base triplets in hyper-relational facts.

Motivated by the above observation, we propose in this paper HINGE, a Hyper-relatIonal kNowledge Graph Embedding model. HINGE is designed to directly learn from hyper-relational facts in a KG, capturing not only the primary structural information of the KG encoded in the triplets, but also the correlation between each triplet and its associated key-value pairs (if any). More precisely, for each hyper-relational fact, we first design two convolutional neural network pipelines, which learn 1) from the base triplet (h, r, t), generating a triple-wise relatedness feature vector for h, r and t, and 2) from each key-value pair (k, v) associated with the triplet together with the triplet itself, generating a quintuple-wise relatedness feature vector between h, r, t, k and v, respectively. Afterward, we compute the overall relatedness feature vector for the hyper-relational fact by taking the minimum value along each feature dimension over the triple-wise relatedness feature vector and all the quintuple-wise relatedness feature vectors. The basic assumption behind this operation is that for a valid hyper-relational fact, both the relatedness for the base triplet (h, r, t) and the relatedness between each key-value pair (k, v) and the base triplet should be high; subsequently, the minimum relatedness along each feature dimension is expected to be high. Finally, we use a fully connected projection to output the predicted score from the overall relatedness feature vector for the input hyper-relational fact.

Our contributions are hence four-fold:

- We investigate the problem of hyper-relational Knowledge Graph embedding, where each fact contains not only a base triplet, but also associated key-value pairs;
- We discuss a key limitation of a commonly used representation scheme for hyper-relational data (i.e., using a set of key-value pairs only). We empirically verify its limitation, showing that triplets serve as the fundamental data structure underpinning modern KGs and indeed encode the essential information for link prediction;
- We introduce HINGE, a Hyper-relatIonal kNowledge Graph Embedding model, designed to directly learn from hyper-relational facts in a KG, capturing not only the primary structural information of the KG encoded in the triplets, but also the correlation between each triplet and its associated key-value pairs;
- We conduct a thorough evaluation of our method compared to a sizable collection of baselines on two real-world KG datasets. Our results show that compared to nine state-of-the-art KG embedding methods learning from triplets only, HINGE consistently achieves better performance, yielding an improvement of 0.81-41.45% on various link prediction tasks (i.e., head/tail or relation prediction) with different data transformation settings (e.g., keeping base triplet only, via reification or relation paths) over the best-performing baseline methods on individual tasks. Moreover, compared to methods learning from hyper-relational facts using an n-ary representation, HINGE shows improvements of 13.2-84.1% across different link prediction tasks over the best-performing baseline methods on individual tasks.

2 RELATED WORK

Graph embeddings have become a key paradigm to learn representations of nodes in a graph and facilitate downstream graph analysis tasks [5, 16, 45]. As a specific type of graphs, Knowledge

Graphs contain both semantic-enriched nodes (entities) and edges (relations). Therefore, KG embedding techniques learn representations of entities and relations in a KG by preserving the relations between entities [35]. In the following, we briefly discuss existing KG Embedding techniques learning from 1) triplets only, 2) triplets with other data, and 3) hyper-relational facts.

2.1 KG Embeddings from Triplets

In the current literature, most of the existing KG embedding techniques learn from a set of triplets (h, r, t) extracted from an input KG. These techniques can be classified into two broad categories, i.e., translational distance models and semantic matching models [28]. First, translational distance models exploit distance-based scoring functions to create the embeddings. One representative model of this family is TransE [3], which creates embeddings from triplets (h, r, t) such that the relation between the head and tail entities are preserved as $h + r \approx t$. Several works further improve TransE to capture richer KG structures—such as multi-mapping relations (one-to-many, many-to-one, or many-to-many)—using more sophisticated scoring function involving relation-specific hyperplanes [37] or spaces [9, 18, 22], for example. Second, semantic matching models exploit similarity-based scoring functions. One typical model in that context is RESCAL [27]. It represents each entity as a vector and each relation as a matrix, and uses a bilinear function to model the relation between two entities. Several works also extend RESCAL by putting a specific focus on reducing the model complexity [44], by capturing asymmetric relations [34], or by modeling non-linear relations using neural networks [1, 7, 25, 30, 31].

However, representing a KG *using triplets only* often oversimplifies the complex nature of the data stored in the KG, in particular for hyper-relational data, where each fact contains multiple relations and entities (see example above). Even though a hyper-relational fact can be transformed to triplets by either keeping the base triplets only or creating additional triplets via reification [4] or relation paths [38], none of these transformations is ideal for knowledge graph embeddings, as the former transformation setting incurs irreversible information loss in the KG embeddings while the latter two settings generate additional entities/relations distracting the KG embedding method from capturing the essential information for link prediction (see our experimental results in Section 4.3 for more detail on this). Therefore, it would be highly beneficial to learn KG embeddings directly from such hyper-relational facts.

2.2 KG Embeddings from Triplets with other Data

We also note that there are a few works on KG embeddings considering other data together with the triplets. According to the sources of such data, these works can be classified into two categories, i.e., data in the KG and third-party data. First, except triplets linking entities via relations, other data contained in a KG can be incorporated into KG embeddings. For example, multi-modal attributes associated with entities (a.k.a. literals), such as non-discrete numerical literals [11, 32] or text literal [20], have been shown to improve the KG embeddings on various tasks; images associated with entities have also been used to improve entity matching tasks (matching entities across different KGs) [24]. These works mainly focus on using multi-modal data to enrich the representation of entities, while triplets remain the only relational representation between entities, which differs from our work focusing on hyper-relational facts. Second, some related techniques learn entity/relation embeddings from triplets in a KG jointly with third-party data sources, in particular with text (e.g., Wikipedia articles) [6, 10, 15, 29, 33, 36, 42, 43, 47, 50]. These works focus on combining the advantages of a KG with further (textual) data sources to learn both entity/relation and word embeddings simultaneously, which is different from our work learning from a KG only while considering hyper-relational facts.

2.3 KG Embeddings from Hyper-Relational Facts

Some recent works on KG embeddings started to consider hyper-relational data (a.k.a. multi-fold or n-ary relational data) [14, 38, 49]. More precisely, these works transform a hyper-relational fact into an n-ary representation, i.e., a set of key-value (relation-entity) pairs while completely avoiding triplets. For example, in [14], a hyper-relational fact (h, r, t) with (k, v) is transformed into $\{r_h{:}h, r_t{:}t, k{:}v\}$ by converting the relation r into two keys r_h and r_t, associated with head h and tail t, respectively. Using this representation, these works learn the relatedness between entity/relation pairs for predicting missing links in KGs. Specifically, m-TransH [38] models the interaction between entities involved in each fact in order to perform link prediction on missing entities. RAE [49] further extends m-TransH by considering the relatedness between entities in each fact for performing instance reconstruction, i.e., predicting one or multiple missing entities in a fact. As these two works capture only the relatedness between entities and can thus only predict missing entities, NaLP [14] was later proposed to model the relatedness between key-value (relation-entity) pairs contained in each fact, which enables the prediction of either a missing key (relation) or a missing value (entity).

However, transforming a hyper-relational fact into an n-ary representation (i.e., as a set of key-value pairs) is inherently incompatible with the schema used by modern KGs, where triplets still serve as the fundamental data structure. In other words, key-value pairs (k, v) on a hyper-relational fact should not be treated identically to base triplets (h, r, t), as the latter actually preserves the essential information for link prediction in the KGs. We also empirically verify this point in our experiments (see Section 4.2 for more detail). Therefore, in this paper, we design HINGE to directly learn from the base triplets even for hyper-relational facts, while simultaneously learning from the associated key-value pairs also.

3 LEARNING FROM HYPER-RELATIONAL FACTS

In this section, we introduce HINGE, a hyper-relational KG embedding model learning directly from hyper-relational facts. We introduce a couple of formal definitions:

Definition 3.1. **Hyper-relational fact:** A hyper-relational fact contains a base triplet (h, r, t) and a set of associated key-value pairs (k_i, v_i), $i = 1, ..., n$.

Definition 3.2. **Triple fact**: A triple fact contains a triplet (h, r, t) only.

Based on these definitions, we start below by discussing our main design principle and goals, before presenting our proposed model in detail.

3.1 Design Principle and Goals

As triplets are indeed the fundamental data structure in modern KGs and thus preserve the essential information for link prediction in the KGs, our main design principle is to, on one hand, learn embeddings directly from this *primary* data source (triplets) in order to preserve the essential information for link prediction in the KG to a maximum extent, while on the other hand also learning the relatedness between a triplet and its associated key-value pairs (if any). Following this principle, we set the three following goals:

I) *Effectively learning from both triple facts and hyper-relational facts.* Specifically, as not all facts are hyper-relational in a KG (e.g., 30% of entities are involved in such hyper-relational facts in Freebase [38]), the embedding model should be highly effective at learning both from triple facts and from hyper-relational facts. In other words, the incorporation of key-value pairs from hyper-relational facts should not distract the main learning process from the triplets.

II) *Leveraging key-value pairs when learning from the base triplets in hyper-relational facts.* For each hyper-relational fact, the base triplet is associated with a set of key-value pairs providing further information about the base triplet. As the base triplet remains the primary data source for modeling the hyper-relational fact, the embedding model should be designed to leverage key-value pairs to *improve* the learning process from the triplet.

III) *Learning from an arbitrary number of key-value pairs from a hyper-relational fact.* As the number of key-value pairs involved in hyper-relational facts varies, the embedding model should be able to effectively handle an arbitrary number of key-value pairs in a hyper-relational fact.

We built HINGE with those goals in mind, in order to effectively learn embeddings from both triple and hyper-relational facts.

3.2 HINGE

Figure 2 illustrates our proposed model HINGE. It is designed to directly learn from hyper-relational facts in a KG, capturing not only the primary structural information of the KG encoded in the triplets, but also the relatedness between each triplet and its associated key-value pairs (if any). More precisely, HINGE consists of three parts. For each hyper-relational fact containing a base triplet (h, r, t) and associated key-value pairs (k_i, v_i), $i = 1, ..., n$, it 1) learns from the base triplet (h, r, t), generating a triple-wise relatedness feature vector for h, r and t, and 2) learns from each key-value pair (k, v) associated with the base triplet together with the triplet itself, generating the quintuple-wise relatedness feature vector between h, r, t, k and v, respectively. Afterward, it 3) merges these relatedness feature vectors to generate a final prediction score for the input hyper-relational fact. In the following, we present the details of these three modules.

3.2.1 Learning from Triplets. In both triple or hyper-relational facts, (base) triplets encode the primary structural information of a KG, and thus capture essential information for link prediction in the KG. To learn from a (base) triplet (h, r, t), we resort to a Convolutional Neural Network (CNN) to model the intrinsic interaction between the three elements in the triplet, i.e., head h, relation r and tail t, in order to generate a triple-wise relatedness feature vector.

More precisely, as shown in Figure 2, we start by concatenating the three corresponding embedding vectors $\vec{h}, \vec{r}, \vec{t} \in \mathbb{R}^K$ (K is the embedding dimension) into an "image" $T \in \mathbb{R}^{3\times K}$, which is the input for a 2D convolutional layer with n_f filters of size 3×3. The filter of size 3 is chosen to capture the triple-wise relatedness between $\vec{h}, \vec{r}$ and $\vec{t}$. This layer returns n_f feature maps of size $K - 2$, which are then flattened into a triple-wise relatedness vector $\vec{\phi} \in \mathbb{R}^{1\times n_f(K-2)}$. This relatedness vector $\vec{\phi}$ can be used to characterize the plausibility of a (base) triplet (h, r, t) of being true.

3.2.2 Learning from Key-Value Pairs. Key-value pairs contain further information describing the associated base triplet in a hyper-relational fact, which suggests that learning from key-value pairs should be coupled with the corresponding triplet. Therefore, for each key-value pair (k_i, v_i) associated with the base triplet (h, r, t) in a hyper-relational fact, we also resort to a CNN to capture the interaction between each elements in the triplet and the key-value pair, i.e., h, r, t, k_i and v_i, in order to generate a quintuple-wise relatedness feature vector.

As shown in Figure 2 and similar to the case of learning from triplets, we first concatenate the five corresponding embedding vectors $\vec{h}, \vec{r}, \vec{t}, \vec{k}_i, \vec{v}_i \in \mathbb{R}^K$ into an "image" $H \in \mathbb{R}^{5\times K}$, and feed H to a 2D convolutional layer with n_f filters of size 5×3. The first dimension size 5 of the filter here is chosen to capture the quintuple-wise relatedness between $\vec{h}$, $\vec{r}$, $\vec{t}$, $\vec{k}_i$ and $\vec{v}_i$; the second dimension size 3 is chosen to match the filter size of the CNN for base triplets, in order to merge the resulting relatedness feature vectors (see below). This layer returns n_f feature maps of size $K - 2$, which is then flattened into the quintuple-wise relatedness vector $\vec{\psi}_i \in \mathbb{R}^{1\times n_f(K-2)}$. This relatedness vector $\vec{\psi}_i$ can be used to characterize the plausibility of the base triplet (h, r, t) associated with the key-value pair (k_i, v_i) being a true fact. This process is repeated for each key-value pair (k_i, v_i), $i = 1, .., n$, in the input hyper-relational fact containing n key-value pairs, resulting in n quintuple-wise relatedness vectors $\vec{\psi}_i$, $i = 1, .., n$. Note that this module is not used for triple facts, as they do not contain any key-value pair.

3.2.3 Merging Relatedness Feature Vectors for Prediction. In the previous two modules, for each hyper-relational fact, one triple-wise relatedness vector $\vec{\phi}$ is generated from the base triplet (h, r, t) while n quintuple-wise relatedness vectors $\vec{\psi}_i$ are generated from the n key-value pairs together with the base triplet. We now wish to merge these relatedness feature vectors in order to return a final score for the input hyper-relational facts.

To achieve this goal, we first compute the overall relatedness feature vector by taking the minimum value along each feature dimension over the triple-wise relatedness feature vector and all the quintuple-wise relatedness feature vectors. We concatenate the

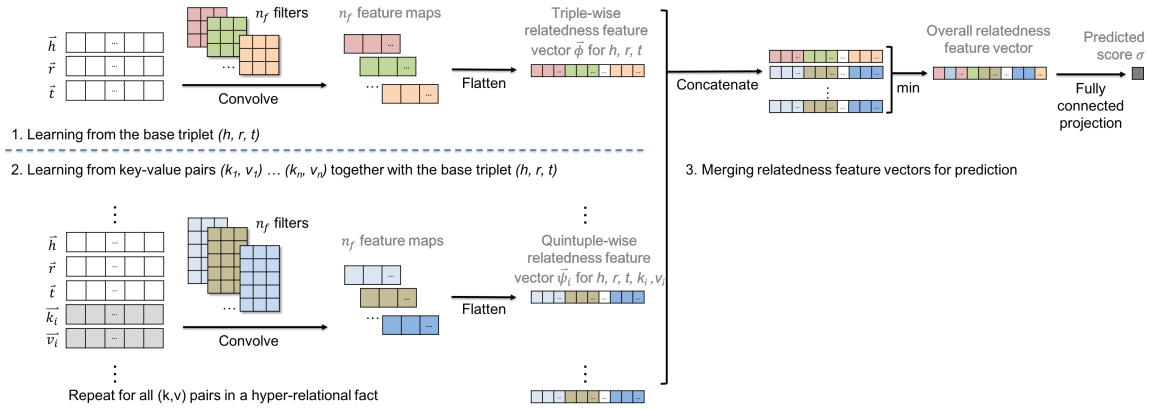


Figure 2: Overview of our proposed method HINGE

triple-wise relatedness feature vectors $\vec{\phi}$ and the n quintuple-wise relatedness vectors $\vec{\psi}_i$ into a matrix of size $(n+1) \times n_f(K-2)$, and compute the minimum value of this matrix along each column, resulting in the overall relatedness feature vector. The underlying assumption for this operation is that for a valid hyper-relational fact, both 1) the relatedness for the base triplet (h, r, t) and 2) the relatedness between each key-value pair (k, v) and the base triplet (h, r, t) should be high. While each entry of a triple-wise (or quintuple-wise) relatedness feature vector actually measures the relatedness between h, r, t (or between h, r, t, k_i v_i) under a certain filter, the minimum relatedness along each feature dimension is expected to be high. Similar ideas have also been successfully applied by previous works to merge relatedness scores in a neural network [14]. Finally, we use a fully connected projection to output the predicted score σ from the overall relatedness feature vector for the input hyper-relational fact.

3.3 HINGE and Design Goals

In this section, we discuss how our proposed model HINGE fits the goals we set above in Section 3.1.

First, to fit goal I, we adopt two separate neural network pipelines to learn from 1) the base triplet, and from 2) the key-value pairs together with the base triplet. In case of triple facts, only the first module is used while the second module is not activated. In case of hyper-relational facts (with key-value pairs), both the first and the second modules are used. Subsequently, the CNN for base triplets (used in the first module) is independent from the key-value pairs, which allows it to capture the primary structural information of the KG encoded in the triplets, and thus to preserve the essential information for link prediction in a KG to a maximum extent.

Second, to fit goal II, we merge, via a "min" operation, the relatedness feature vectors learnt from 1) the base triplet, and 2) the key-value pairs together with the base triplet. Such an operation ensures that the CNN for key-value pairs (used in the second module) effectively help to evaluate the plausibility of a hyper-relational fact. More precisely, considering a hyper-relational fact containing a base triplet and an associated key-value pair, we obtain a triple-wise relatedness vector $\vec{\phi}$ and a quintuple-wise relatedness vectors $\vec{\psi}$. If the key-value pair are highly related to the base triplet (i.e., $\vec{\psi}$ has high values), the merge operation (via the "min") ensures that the values in the overall relatedness feature vector are determined mostly by the values of $\vec{\phi}$ (i.e., relatedness for the base triplet). In other words, the final predicted score depends mostly on the base triplet. In contrast, if the key-value pair is poorly related to the base triplet (i.e., $\vec{\psi}$ has low values), the merge ("min") operation ensures that the values in the overall relatedness feature vector are low (i.e., they are determined mainly based on the values of $\vec{\psi}$). In other words, the overall relatedness of the hyper-relational fact is low, because the key-value pair is poorly related to the base triplet, which further implies that this fact is less probably a true fact.

Third, goal III is automatically fulfilled with our merge operation. For the case of multiple key-value pairs (k_i, v_i), $i = 1, \ldots, n$, associated with the base triplet, *each* key-value pair (k_i, v_i) is combined with the base triplet to generate the corresponding quintuple-wise relatedness feature vector $\vec{\psi}_i$. Subsequently, our merging (via the "min") operation is able to take an arbitrary number of quintuple-wise relatedness feature vectors as input.

3.4 Model Training Process

To train the model parameters, we minimize a softplus loss. More precisely, following [14, 34], our loss function is defined as the negative log-likelihood of the logistic model:

$$\sum_{\omega \in \Omega} log(1 + e^{-\sigma(\omega)}) + log(1 + e^{\sigma(\omega')}) \tag{1}$$

where Ω is the input set of hyper-relational facts. For each hyper-relational fact ω containing (h, r, t) and the associated (k_i, v_i), $i = 1, \ldots, n$, one negative sample ω' is generated by randomly corrupting one entity (h, t, or v_i) or relation (r or k_i). $\sigma(\omega)$ and $\sigma(\omega')$ denote the predicted score of our HINGE model for the true fact ω and the negative fact ω', respectively.

The loss function 1 is minimized using the Adam stochastic optimizer [19], and the model parameters are learnt via back propagation. Specifically, we use rectified linear units (ReLU) as the non-linearity activation function [21] and batch normalization [17] after the two CNN layers for fast training.

Table 1: Statistics of the datasets

Dataset		JF17K		WikiPeople	
#Entity		28,645		34,839	
#Relation		322		375	
#Fact (training)	Only triple	44,210	57.8%	280,520	97.4%
	Only hyper-relational	32,169	42.2%	7,389	2.6%
	Total	76,379	100%	287,918	100%
#Fact (test)	Only triple	10,417	42.4%	36,597	97.4%
	Only hyper-relational	14,151	57.6%	971	2.6%
	Total	24,568	100%	37,586	100%

4 EXPERIMENTS

In this section, we evaluate our proposed model HINGE on various link prediction tasks. In the following, we start by presenting our experimental setup, followed by our results and discussions.

4.1 Experimental Setup

4.1.1 Dataset. We conduct experiments on two hyper-relational datasets ***JF17K*** [38] and ***WikiPeople*** [14], extracted from two popular KGs, i.e., Freebase and Wikidata, respectively. Each of these two datasets contains both triple facts and hyper-relational facts. While *JF17K* was filtered from Freebase to have a significant presence of hyper-relational facts (see [38] for more detail), *WikiPeople* is extracted from Wikidata and focuses on entities of type *human* without any specific filtering to improve the presence of hyper-relational facts [14]. As the original WikiPeople dataset also contains literals (used as tails) in some facts, we filter out these non-entity literals and the corresponding facts. Table 1 shows the main statistics of these datasets.

4.1.2 Baselines. We compare HINGE against a sizable collection of state-of-the-art Knowledge Graph embedding techniques from two categories.

The first category includes models learning from triplets only.

- Translational distance models: **TransE** [3] learns to preserve the relation between two entities as $h + r \approx t$. **TransH** [37] extends TransE to better capture multi-mapping relations by introducing relation-specific hyperplanes. **TransR** [22] introduces relation-specific projections to also better capture multi-mapping relations. **TransD** [18] extends TransR by decomposing the projection matrix into a product of two vectors. These models minimize a margin-based ranking objective function, where we empirically set the margin $b = 1$ with the L_2-norm. In addition, we set the learning rate to 0.001 with a stochastic gradient descent optimizer, the number of negative samples to 1, and the batch size to 128.
- Semantic matching models: **Rescal** [27] represents each entity as a vector and each relation as a matrix, and uses a bilinear function to model the relation between a pair of entities. **DistMult** [44] simplifies Rescal by representing each relation embedding as a diagonal matrix. **ComplEx** [34] further extends DistMult in the complex space in order to better model both symmetric and asymmetric relations. **Analogy** [23] models explicitly analogical structures in multi-relational KG embeddings. **ConvE** [7] adopts a 2D CNN to capture richer interactions between entity and relation embeddings. We set the margin $b = 1$ with the L_2-norm for Rescal. For DistMult, ComplEx and Analogy, we set the learning rate to 0.1 with Adagrad optimizer [8], the number of negative samples to 1, and the batch size to 128. For ConvE, we set the learning rate to 0.003, the batch size to 128, the dropout to 0.2, and the label smoothing value to 0.1.

The second category includes models learning from the n-ary representation of hyper-relational facts.

- **m-TransH** [38] models the interaction between entities involved in each n-ary fact. Specifically, each hyper-relational fact (h, r, t) with (k_i, v_i), $i = 1, ..., n$ is associated with a so-called meta-relation, represented as an ordered list of keys (relations), such as $R := (r_h, r_t, k_1, ..., k_n)$; the fact is then represented as a list of ordered values associated with the meta-relation $\{R, (h, t, v_1, ..., v_n)\}$. Using this representation, m-TransH is built on top of TransH to capture multi-fold relations between entities in a meta-relation. As it learns only from sets of entities in meta-relations (without considering the exact relations in each meta-relation), it can be applied to perform the link prediction task on missing entities only. Following suggestions from the original paper, we empirically set its hyper-parameters as follows: the margin to 0.5, the weight to 0.001, and the threshold to 0.01.
- **RAE** [49] extends m-TransH by explicitly considering the pairwise relatedness between entities in n-ary facts. Using the same n-ary representation of hyper-relational facts, RAE further learns from the pairwise relatedness between entities in each n-ary fact in order to perform instance reconstruction, i.e., predicting one or multiple missing entities. Similar to m-TransH, RAE can only be used to predict missing entities. We search for optimal parameters by adopting the techniques described in [14] on each dataset, and report the results with the optimal settings.
- **NaLP** [14] models the relatedness between key-value (relation-entity) pairs contained in each n-ary fact. It represents each hyper-relational fact (h, r, t) with (k_i, v_i), $i = 1, ..., n$, as a set of key-value pairs $\{r_h{:}h, r_t{:}t, k_i{:}v_i\}$, $i = 1, ..., n$ by converting the relation r into two keys r_h and r_t, associating with head h and tail t, respectively. Using this representation, NaLP learns from the pairwise relatedness between key-value pairs via a neural network pipeline, which enables the prediction of both missing keys (relations) or missing values (entities). Note that in NaLP, a commonly adopted negative sampling process is used, which randomly corrupts one key or value in a true fact. However, this process is not fully adaptable to its n-ary represention of hyper-relational facts, in particular for keys r_h and r_t. For example, when corrupting the key r_h by a randomly sampled r'_h $(r \neq r')$, the negative fact becomes $\{r'_h{:}h, r_t{:}t, k_i{:}v_i\}$, $i = 1, ..., n$. This is unrealistic as r'_h is not compatible with r_t while only *one* relation r (or r') can be assumed between h and t in a hyper-relational fact. Therefore, we propose a variant of NaLP fixing this issue. Following the suggestion from the original paper, we adopt the optimal hyper-parameters reported on each dataset.
- **NaLP-Fix** is a variant of NaLP with a fixed negative sampling process. Specifically, when corrupting the key r_h by a randomly sampled r'_h $(r \neq r')$, we also corrupt r_t by r'_t, resulting in a negative fact $\{r'_h{:}h, r'_t{:}t, k_i{:}v_i\}$, $i = 1, ..., n$. Subsequently for this negative fact, only a single relation r' links h and t, which is a realistic case. Similarly, when corrupting r_t, we also corrupt r_h

in the same way. We keep using the same hyper-parameters as for NaLP.

For our HINGE model, we empirically set the number of filters n_f in both CNNs to 400, the batch size to 128, and the learning rate to 0.0001. The embedding size is set to 100 for all methods, if not specified otherwise. The implementation of HINGE and used datasets are available here[2].

4.1.3 Dataset Configuration. As discussed in the introduction, hyper-relational facts need to be transformed into triplets for the models that can learn from triplets only. There are three common settings for such a transformation.

- *Basic*: Only the base triplet (h, r, t) from a hyper-relational fact is kept, while removing all its associated key-value pairs. This setting causes irreversible information loss.
- *Relation Path* [38]: For each hyper-relational fact containing a base triplet (h, r, t) and associated key-value pairs (k_i, v_i), $i = 1, ..., n$, each pair of entities are linked via an artificially created relation path. For example, h is linked to v_i via a virtual relation $\widehat{rk_i}$ representing a relation path $r \rightarrow k_i$, resulting in an additional triplet $(h, \widehat{rk_i}, v_i)$. This setting creates extra relations and facts.
- *Reification* [4]: For each hyper-relational fact containing key-value pairs (k_i, v_i), $i = 1, ..., n$, an artificial entity is used to represent the base triplet $q := (h, r, t)$, and then additional triplets are created as (q, k_i, v_i), $i = 1, .., n$. This setting also creates extra entities and triplets.

In addition, we denote the original hyper-relational facts as the *Original* setting.

4.1.4 Evaluation Tasks and Metrics. Link prediction is a typical task for Knowledge Graph completion. Given two elements of a triplet in a (hyper-relational) fact, the task is to predict the missing one, such as $(?, r, t)$, $(h, ?, t)$ or $(h, r, ?)$, where the question mark represents the missing entity/relation. In this paper, we conduct experiments in all of these three cases, i.e., predicting a missing head, relation, or tail. We describe our evaluation protocols below by taking the case of predicting missing heads $(?, r, t)$ as an example. For the triplet $(?, r, t)$ in one test (hyper-relational) fact, we replace the missing head with all the entities, resulting in a set of candidate (hyper-relational) facts. Among those candidate facts, in addition to the testing fact itself, other corrupted facts might also be true facts (i.e., existing in the training/test datasets); these facts are thus removed from the candidate facts. Afterward, the resulting candidate facts are fed into an embedding model to output predicted scores. By ranking the candidate facts according to their corresponding scores, we generate a predicted ranking list of entities for the missing head. By repeating the evaluation process over all test facts in the test dataset, we report Mean Reciprocal Rank (*MRR*), $Hits@10$ and $Hits@1$, which are widely used metrics for link prediction tasks [14]. The same evaluation protocol and metrics also apply to predicting missing relations $(h, ?, t)$ and tails $(h, r, ?)$. As predicting missing heads or tails is essentially predicting missing entities, we report average results on these two cases (denoted as "Head/Tail Prediction"), while we report individual results for relation prediction.

[2]https://github.com/eXascaleInfolab/HINGE_code/

4.2 Limitation of N-Ary Representation

In this experiment, we experimentally show the limitation of the n-ary representation of a hyper-relational fact (represented as a set of key-value pairs without triplets). Specifically, most of the existing works on learning KG embeddings from hyper-relational facts [14, 38, 49] transform a hyper-relational fact into a set of key-value (relation-entity) pairs while completely avoiding triplets. For example, NaLP [14] transforms a hyper-relational fact (h, r, t) with (k_i, v_i), $i = 1, ..., n$, into a set of key-value pairs $\{r_h{:}h, r_t{:}t, k_i{:}v_i\}$, $i = 1, ..., n$ by converting relation r into two keys r_h and r_t, associated with head h and tail t, respectively. Subsequently in the embedding learning process, these key-value pairs are treated equally. However, we argue that as triplets still serve as the fundamental data structure in the modern KGs, they preserve the essential information for link prediction in KGs. In other words, key-value pairs (k, v) on a hyper-relational fact should not be treated equally as base triplets (h, r, t).

To verify this point, we define an extra setting, the *null model*, for dataset transformation. The null model reconstructs one triplet from each n-ary relational fact $\{r_h{:}h, r_t{:}t, k_i{:}v_i\}$, $i = 1, ..., n$, via a randomly sampled relation path [38], such as $(h, \widehat{r_h k_i}, v_i)$. Note that if we only sample the relation path $\widehat{r_h r_t}$, we reconstruct the original base triplet $(h, \widehat{r_h r_t}, t)$ (corresponding to the *Basic* setting). With this null model, we make a null hypothesis: *The information for link prediction encoded by the original base triplets is not greater than the triplets created by the null model.* We test this null hypothesis by performing link prediction tasks using all nine baseline models learning from triplets on the two sets of triplets, i.e., on the original *basic* triplets and the triplets created by the *null model*.

Table 2 shows the results on both datasets. Comparing the results from the two data transformation settings *basic* and *null model*, we clearly observe that the performance from the original base triplets is consistently and significantly better than the performance from the triplets created by the null model, with an average improvement[3] of 77.5% in head/tail prediction and 58.3% in relation prediction on the WikiPeople dataset (125.2% and 114.9% on JFK17K, respectively). To further verify this point, we conduct one-tailed paired t-test on the results using the original base triplets and the results using the triplets created by the null model from the same set of nine baselines, for each metric and each link prediction task. The test results consistently reject the null hypothesis at the 0.01 significance level (p-value $\ll 0.01$). Therefore, we verify that the information encoded by the original base triplets is significantly greater than the triplets created by the null model for link prediction. In other words, compared to the key-value pairs, the base triplets in hyper-relational facts preserve the essential information for link prediction in KGs; this finding indeed corresponds to the fundamental design principle behind our new technique HINGE.

4.3 Link Prediction Performance Comparison

In this experiment, we compare the link prediction performance of HINGE against all baselines under different dataset transformation settings. Table 2 shows the results on both datasets. For each dataset transformation setting, we highlight the best-performing method on each task and for each dataset. In the following, we discuss the results and highlight our key findings.

[3]referring to the average improvement on different metrics throughout this paper.

Table 2: Link prediction performance on both WikiPeople and JF17K.

Dataset Transformation Setting	Method	WikiPeople						JF17K					
		Head/Tail Prediction			Relation Prediction			Head/Tail Prediction			Relation Prediction		
		MRR	Hit@10	Hit@1	MRR	Hit@10	Hit@1	MRR	Hit@10	Hit@1	MRR	Hit@10	Hit@1
Null Model	TransE	0.2021	0.4377	0.1031	0.2060	0.2626	0.1683	0.1916	0.3553	0.1079	0.6240	0.7356	0.5522
	TransH	0.1998	0.4400	0.0992	0.2203	0.2909	0.1785	0.1916	0.3529	0.1093	**0.6382**	**0.7543**	**0.5623**
	TransR	0.2009	0.4355	0.1015	0.1485	0.1972	0.1196	0.1921	0.3603	0.1085	0.6201	0.7207	0.5543
	TransD	0.1083	0.3416	0.0074	0.2985	0.4816	0.2089	0.0626	0.1890	0.0028	0.2155	0.2894	0.1718
	Rescal	0.1325	0.2986	0.0626	**0.6561**	**0.8174**	**0.5474**	0.1095	0.2031	0.0630	0.5679	0.6781	0.4954
	DistMult	0.1144	0.3083	0.0385	0.4281	0.5520	0.3315	0.0997	0.2145	0.0475	0.0512	0.2255	0.0323
	ComplEx	0.1050	0.2932	0.0329	0.3719	0.4467	0.3093	0.0828	0.1824	0.0382	0.1283	0.3277	0.0474
	Analogy	0.1144	0.2991	0.0406	0.4178	0.4903	0.3535	0.0917	0.1952	0.0438	0.1167	0.2495	0.0581
	ConvE	**0.2383**	**0.4548**	**0.1470**		N/A		**0.2270**	**0.4174**	**0.1357**		N/A	
Basic	TransE	0.3242	0.6064	0.1216	0.3482	0.4200	0.2734	0.2556	0.4529	0.1576	0.8574	0.9064	0.8270
	TransH	0.3206	0.6029	0.1155	0.3724	0.4448	0.2980	0.2570	0.4564	0.1619	**0.8618**	**0.9134**	**0.8328**
	TransR	0.3264	0.6090	0.1236	0.2446	0.4996	0.1651	0.2806	0.4974	0.1791	0.8431	0.8924	0.8137
	TransD	0.2200	0.5414	0.0205	0.5657	0.8804	0.4160	0.1343	0.3105	0.0501	0.6803	0.7872	0.6189
	Rescal	0.2772	0.4915	0.1404	**0.7936**	**0.9023**	**0.7306**	0.1709	0.3340	0.0952	0.7887	0.8491	0.7480
	DistMult	0.2468	0.5087	0.0645	0.6008	0.6776	0.5479	0.1752	0.3531	0.0955	0.2779	0.5340	0.1381
	ComplEx	0.2466	0.4944	0.0648	0.5676	0.6135	0.5367	0.1669	0.3307	0.0906	0.2380	0.3445	0.1765
	Analogy	0.2521	0.5033	0.0688	0.5984	0.6386	0.5699	0.1776	0.3471	0.0996	0.2667	0.4247	0.1773
	ConvE	**0.4781**	**0.6533**	**0.3666**		N/A		**0.3190**	**0.5470**	**0.2129**		N/A	
Relation Path	TransE	0.3191	0.6067	0.1160	0.2773	0.3379	0.2444	0.2832	0.4826	0.1832	0.8251	0.8940	0.7814
	TransH	0.3198	0.6084	0.1155	0.2399	0.3267	0.1906	0.2863	0.4899	0.1850	0.8179	0.8897	0.7738
	TransR	0.3214	0.6086	0.1167	0.1593	0.2154	0.1272	0.3075	0.5170	0.2051	0.6866	0.7779	0.6383
	TransD	0.2083	0.5228	0.0146	0.3344	0.5053	0.2443	0.1279	0.3204	0.0362	0.4333	0.5561	0.3676
	Rescal	0.2637	0.4780	0.1255	**0.7535**	**0.8673**	**0.6895**	0.1692	0.3188	0.0966	**0.8336**	**0.8957**	**0.7914**
	DistMult	0.2400	0.4987	0.0588	0.5787	0.6429	0.5360	0.2261	0.4084	0.1368	0.2817	0.5551	0.1574
	ComplEx	0.2415	0.4861	0.0672	0.4975	0.5415	0.4716	0.2193	0.3930	0.1352	0.2126	0.3361	0.1477
	Analogy	0.2443	0.4936	0.0688	0.5139	0.5555	0.4887	0.2244	0.3986	0.1413	0.2523	0.4308	0.1625
	ConvE	**0.4700**	**0.6537**	**0.3527**		N/A		**0.3665**	**0.5876**	**0.2574**		N/A	
Reification	TransE	0.3207	0.5977	0.1224	0.3253	0.3850	0.2747	0.2285	0.3806	0.1503	**0.8793**	**0.9187**	**0.8559**
	TransH	0.3244	0.6011	0.1242	0.3160	0.3873	0.2694	0.2302	0.3815	0.1538	0.8774	0.9218	0.8506
	TransR	0.3225	0.6002	0.1225	0.2396	0.2968	0.1817	0.2838	0.4722	0.1914	0.8751	0.9157	0.8510
	TransD	0.2123	0.5253	0.0195	0.5293	0.8611	0.3821	0.0950	0.2121	0.0366	0.5562	0.6610	0.5010
	Rescal	0.2751	0.4815	0.1430	**0.7684**	**0.8816**	**0.7053**	0.1354	0.2608	0.0759	0.6958	0.7620	0.6556
	DistMult	0.2276	0.4867	0.0519	0.5902	0.6611	0.5422	0.1523	0.2888	0.0875	0.1135	0.3251	0.0332
	ComplEx	0.2365	0.4795	0.0614	0.5375	0.5882	0.5039	0.1325	0.2552	0.0760	0.1311	0.2451	0.0717
	Analogy	0.2478	0.4901	0.0718	0.5838	0.6277	0.5562	0.1329	0.2594	0.0742	0.1548	0.2983	0.0852
	ConvE	**0.4657**	**0.6434**	**0.3559**		N/A		**0.3469**	**0.5410**	**0.2500**		N/A	
Original	m-TransH	0.0633	0.3006	0.0633		N/A		0.2060	0.4627	0.2060		N/A	
	RAE	0.0586	0.3064	0.0586		N/A		0.2153	0.4668	0.2153		N/A	
	NALP	0.4084	0.5461	0.3311	0.4818	0.8516	0.3198	0.2209	0.3310	0.1650	0.6391	0.8215	0.5472
	NALP-Fix	0.4202	0.5564	0.3429	0.8200	0.9757	0.7197	0.2446	0.3585	0.1852	0.7469	0.8921	0.6665
	HINGE	**0.4763**	**0.5846**	**0.4154**	**0.9500**	**0.9977**	**0.9159**	**0.4489**	**0.6236**	**0.3611**	**0.9367**	**0.9894**	**0.9014**

4.3.1 Comparison with Baselines Learning from Triplets Only. We observe that our HINGE model consistently outperforms all baselines learning from triplets only, for all three dataset transformation settings. Specifically, the *Basic* setting simply discards the key-value pairs, causing irreversible information loss, while the *Relation Path* and *Reification* settings create extra entities/relations which indeed distract the embedding models from capturing the essential information for link prediction from the input KG. We compute the average improvement of HINGE over the best-performing baselines for each dataset transformation settings in Table 3. We observe that HINGE yields significant improvement in most cases, showing improvements of up to 41.45% on head/tail prediction and of up to 24.65% on relation prediction.

One exception is for the head/tail prediction on WikiPeople with the *Basic* setting, where the improvement is marginal (0.81%). We further find that the best-performing baseline in this case is ConvE, which is indeed the most competitive baseline in head/tail prediction in all cases (see Table 2). Note that similar to our HINGE, ConvE also uses a 2D CNN layer for feature extraction from entity/relation embeddings in triplets, yielding good performance on head/tail prediction. The marginal improvement can be explained by the dominance of triple facts in WikiPeople dataset (97.4% triple facts vs 2.6%

Table 3: Improvement of HINGE over the best-performing baselines learning from triplets only. The best performing baselines are highlight in Table 2. The *Null Model* setting is excluded due to its very low performance.

Dataset	WikiPeople		JF17K	
Transformation Setting	Head/Tail Prediction	Relation Prediction	Head/Tail Prediction	Relation Prediction
Basic	0.81%	18.55%	41.45%	8.41%
Relation Path	2.85%	24.65%	22.96%	12.24%
Reification	3.28%	22.22%	29.71%	6.52%

hyper-relational facts in both training and test datasets), where both HINGE and ConvE perform well. In contrast, on the JF17K dataset, which contains a significant portion of hyper-relational facts (57.8% triple facts vs 42.2% hyper-relational facts in the training dataset and 42.4% triple facts vs 57.6% hyper-relational facts in the test dataset), HINGE significantly outperforms ConvE by leveraging key-value pairs in the hyper-relational facts (while ConvE learns from a transformed dataset using one of our data transformation settings). In addition, we also highlight that ConvE is specifically designed for head/tail prediction only, and is not applicable to the relation prediction task.

Another interesting observation is that, compared to the *Basic* setting, the *Relation Path* and *Reification* settings yield worse results in general. For example, on WikiPeople, compared to the *Basic* setting, the *Relation Path* setting shows an average performance drop of 3.4% on head/tail prediction and 20.5% on relation prediction (the *Reification* setting shows 2.2% and 5.7% performance drop, respectively). This further verifies that even though the *Relation Path* and *Reification* settings preserve the complete information of a hyper-relational fact, the extra created entities/relations indeed distract the KG embeddings from capturing essential information for link prediction in the input KG.

4.3.2 Comparison with Baselines Learning from Hyper-Relational Facts. We observe that our proposed model HINGE consistently outperforms all other baselines learning from hyper-relational facts. Among the methods learning from the n-ary representation of hyper-relational facts (i.e., m-TransH, RAE and NaLP), NaLP shows the best performance, as it learns the relatedness between key-value (relation-entity) pairs while m-TransH and RAE learn from entities only. Note that m-TransH and RAE result in very low performance on WikiPeople, which is probably due to the weak presence of hyper-relational facts in WikiPeople while and m-TransH and RAE are designed for hyper-relational facts. Moreover, compared to NaLP, NaLP-Fix (with our fixed negative sampling process) consistently shows better performance, with a slight improvement of 2.8% in head/tail prediction, and a tremendous improvement of 69.9% in relation prediction on WikiPeople (10.4% and 15.8% on JFK17K, respectively). This verifies the effectiveness of our fixed negative sampling process, in particular for relation prediction. Finally, compared to the best-performing baseline NaLP-Fix, HINGE shows a 13.2% improvement on the head/tail prediction task, and a 15.1% improvement on the relation prediction task on WikiPeople (84.1% and 23.8% on JF17K, respectively).

In addition, we also find that the baseline methods learning from hyper-relational facts (i.e., m-TransH, RAE, NaLP and our NaLP-Fix) yield, surprisingly, worse performance in many cases than the best-performing baselines learning from triplets only. This can be explained by the fact that their n-ary representation of hyper-relational facts indeed ignores the triplet structure, by converting a hyper-relational fact (h, r, t) with (k_i, v_i), $i = 1, ..., n$, into a set of key-value pairs $\{r_h{:}h, r_t{:}t, k_i{:}v_i\}$, $i = 1, ..., n$. However, as the triplet structure serves as the fundamental data structure in KGs and thus preserves essential information for link prediction in the KGs, even though these methods can learn from key-value pairs associated with triplets in hyper-relational facts, their ignorance of the triplet structure results in subpar performance.

4.4 Link Prediction Performance on Triple and Hyper-Relational Facts

In this experiment, we look into the breakdown of link prediction performance on triple and hyper-relational facts. We compare HINGE with NaLP-Fix only, as it is the best-performing baseline learning from hyper-relational facts.

Table 4 shows the results. We observe that HINGE consistently achieves the best performance on both types of facts. In addition, we also find that while the performance of both methods on triple facts is higher than on hyper-relational facts on WikiPeople, we have a completely opposite observation on JF17K, i.e., the performance on triple facts is obviously lower than on hyper-relational facts. This can be explained by the dataset statistics. Where the JF17K dataset has a significant presence of hyper-relational facts (42.2% and 73.6% in the training and test datasets, respectively), WikiPeople contains much fewer hyper-relational facts (2.6% in both the training and test datasets).

4.5 Key/Value Prediction Performance

In this experiment, we study the performance of HINGE on a key/value prediction task. Specifically, as a hyper-relational fact may contains a set of key-value pairs (k_i, v_i), with $i = 1, ..., n$, associated with the base triplet (h, r, t), this task tries to predict a missing key $(?, v_i)$ or a missing value $(k_i, ?)$ in a hyper-relational fact. Our evaluation protocol is similar to the one from the link prediction task. Taking the case of predicting a missing key $(?, v_i)$ as an example, we first replace the missing key with all possible relations, resulting in a set of candidate hyper-relational facts. After filtering out the other true facts (existing in the training/test datasets) except the test fact itself, we feed the remaining candidate hyper-relational facts to an embedding model to output predicted scores. By ranking the candidate facts according to their scores, we report *MRR*, $Hits@10$ and $Hits@1$. The same evaluation protocol and metrics also apply to predicting a missing value $(k_i, ?)$. Similar to the previous experiment, we compare HINGE only with NaLP-Fix, which is the best-performing baseline learning from hyper-relational facts.

Table 5 shows the results. We observe that HINGE consistently outperforms NaLP-Fix, showing a 6.0% improvement on the value prediction task, and a 29.4% improvement on the key prediction task on WikiPeople (28.7% and 28.6% on JF17K, respectively).

Table 4: Link prediction performance on triple and hyper-relational facts.

Fact Type	Method	WikiPeople						JF17K					
		Head/Tail Prediction			Relation Prediction			Head/Tail Prediction			Relation Prediction		
		MRR	Hit@10	Hit@1	MRR	Hit@10	Hit@1	MRR	Hit@10	Hit@1	MRR	Hit@10	Hit@1
Triple Fact	NaLP-Fix	0.4216	0.5592	0.3436	0.8057	0.9728	0.7022	0.0901	0.1802	0.0464	0.6005	0.8253	0.4806
	HINGE	**0.4765**	**0.5874**	**0.3937**	**0.9493**	**0.9979**	**0.9145**	**0.2641**	**0.4965**	**0.1572**	**0.8723**	**0.9846**	**0.7965**
Hyper-relational Fact	NaLP-Fix	0.3050	0.4757	0.2154	0.7605	0.9476	0.6517	0.3420	0.4760	0.2693	0.8542	0.9381	0.8067
	HINGE	**0.3213**	**0.4888**	**0.2322**	**0.9432**	**1.0000**	**0.8876**	**0.5850**	**0.7172**	**0.5112**	**0.9841**	**0.9929**	**0.9785**

Table 5: Key/Value prediction performance on hyper-relational facts.

Method	WikiPeople						JF17K					
	Value Prediction			Key Prediction			Value Prediction			Key Prediction		
	MRR	Hit@10	Hit@1	MRR	Hit@10	Hit@1	MRR	Hit@10	Hit@1	MRR	Hit@10	Hit@1
NaLP-Fix	0.5301	0.6377	0.4581	0.7121	0.8713	0.6317	0.4251	0.5789	0.3426	0.7643	0.8970	0.6945
HINGE	**0.5552**	**0.6647**	**0.5000**	**0.9402**	**0.9910**	**0.9007**	**0.5506**	**0.6880**	**0.4714**	**0.9986**	**0.9996**	**0.9978**

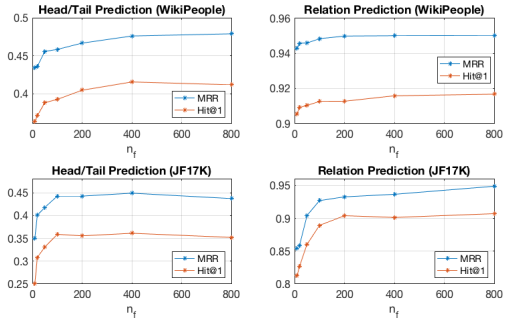


Figure 3: Impact of the number of filters n_f

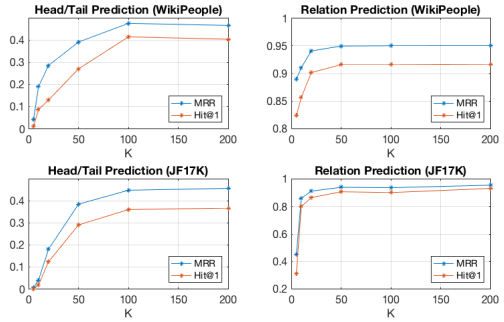


Figure 4: Impact of the embedding dimension K

4.6 Parameter Sensitivity Study

Finally, we study the impact of two key parameters in HINGE, i.e., the number of filters n_f used in the CNNs, and the embedding dimension K. First, by fixing the embedding dimension $K = 100$, we vary the number of filters n_f from 10 to 800 on a log scale, and show its impact on the link prediction tasks for both datasets in Figure 3. We observe that when increasing n_f, the performance starts to increase, and then flatten out when $n_f \geq 400$ in most cases. Second, by fixing the number of filters $n_f = 400$, we vary the embedding dimension K from 5 to 200 on a log scale, and show its impact on the link prediction tasks for both datasets in Figure 4. Similar to the case of n_f, we observe that when increasing K, the performance starts to increase, and then flatten out when $K \geq 100$ in most cases. Therefore, we set the number of filters $n_f = 400$ and the embedding dimension $K = 100$, in all previous experiments.

5 CONCLUSION

Existing Knowledge Graph embedding techniques mostly represent a KG as a set of triplets, and then learn entity/relation embeddings from such triplets while preserving the essential information for link prediction in the KG. However, this triplet representation oversimplifies the complex nature of the data stored in the KG, in particular for hyper-relational facts, where each fact contains not only a base triplet (h, r, t), but also its associated key-value pairs (k, v). Even though a few recent techniques learn from such data using an n-ary representation (i.e., a set of key-value pairs only without any triplet), they result in suboptimal models due to their ignorance of the triplet structure, which, as we show in this paper, is the fundamental structure in modern KGs and encodes essential information for link prediction. Against this background, we proposed HINGE, a hyper-relational KG embedding model. It captures not only the primary structural information of the KG from the triplets, but also the correlation between each triplet and its associated key-value pairs. Our extensive evaluation shows the superiority of HINGE on various link prediction tasks over KGs using two real-world KG datasets. In particular, HINGE consistently outperforms not only the KG embedding methods learning from triplets only (by 0.81-41.45% depending on the link prediction tasks and settings), but also the methods learning from hyper-relational facts using n-ary representations (by 13.2-84.1%).

In the future, we plan to investigate the hyper-relational KG embedding problem further by taking into consideration other types of data in a KG, such as numerical and text literals, or multi-modal data such as images.

6 ACKNOWLEDGEMENT

This work was supported by the Swiss National Science Foundation under grant number 407540_167320.

REFERENCES

[1] Ivana Balazevic, Carl Allen, and Timothy M Hospedales. 2018. Hypernetwork Knowledge Graph Embeddings. *arXiv preprint arXiv:1808.07018* (2018).

[2] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *ACM SIGMOD/PODS*. ACM, 1247–1250.

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*. 2787–2795.

[4] Dan Brickley, Ramanathan V Guha, and Brian McBride. 2014. RDF Schema 1.1. *W3C recommendation* 25 (2014), 2004–2014.

[5] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. 2018. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering* 30, 9 (2018), 1616–1637.

[6] Jianpeng Cheng, Zhongyuan Wang, Ji-Rong Wen, Jun Yan, and Zheng Chen. 2015. Contextual text understanding in distributional semantic space. In *CIKM*. ACM, 133–142.

[7] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2017. Convolutional 2d knowledge graph embeddings. In *AAAI*. 1811–1818.

[8] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, Jul (2011), 2121–2159.

[9] Takuma Ebisu and Ryutaro Ichise. 2018. Toruse: Knowledge graph embedding on a lie group. In *AAAI*.

[10] Wei Fang, Jianwen Zhang, Dilin Wang, Zheng Chen, and Ming Li. 2016. Entity disambiguation by knowledge and text jointly embedding. In *CoNLL*. 260–269.

[11] Alberto Garcia-Duran and Mathias Niepert. 2017. Kblrn: End-to-end learning of knowledge base representations with latent, relational, and numerical features. *arXiv preprint arXiv:1709.04676* (2017).

[12] Google. 2014. https://www.google.com/intl/bn/insidesearch/features/search/knowledge.html.

[13] Jens Graupmann, Ralf Schenkel, and Gerhard Weikum. 2005. The SphereSearch engine for unified ranked retrieval of heterogeneous XML and web documents. In *VLDB*. VLDB Endowment, 529–540.

[14] Saiping Guan, Xiaolong Jin, Yuanzhuo Wang, and Xueqi Cheng. 2019. Link Prediction on N-ary Relational Data. In *The World Wide Web Conference*. ACM, 583–593.

[15] Xu Han, Zhiyuan Liu, and Maosong Sun. 2018. Neural knowledge acquisition via mutual attention between knowledge graph and text. In *AAAI*.

[16] Rana Hussein, Dingqi Yang, and Philippe Cudré-Mauroux. 2018. Are Meta-Paths Necessary? Revisiting Heterogeneous Graph Embeddings. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 437–446.

[17] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[18] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *ACL and IJCNLP*, Vol. 1. 687–696.

[19] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[20] Agustinus Kristiadi, Mohammad Asif Khan, Denis Lukovnikov, Jens Lehmann, and Asja Fischer. 2018. Incorporating literals into knowledge graph embeddings. *arXiv preprint arXiv:1802.00934* (2018).

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

[22] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion.. In *AAAI*, Vol. 15. 2181–2187.

[23] Hanxiao Liu, Yuexin Wu, and Yiming Yang. 2017. Analogical inference for multi-relational embeddings. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. 2168–2178.

[24] Ye Liu, Hui Li, Alberto Garcia-Duran, Mathias Niepert, Daniel Onoro-Rubio, and David S Rosenblum. 2019. MMKG: Multi-modal Knowledge Graphs. In *European Semantic Web Conference*. Springer, 459–474.

[25] Dai Quoc Nguyen, Thanh Vu, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2018. A Capsule Network-based Embedding Model for Knowledge Graph Completion and Search Personalization. *arXiv preprint arXiv:1808.04122* (2018).

[26] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2015. A review of relational machine learning for knowledge graphs. *Proc. IEEE* 104, 1 (2015), 11–33.

[27] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A Three-Way Model for Collective Learning on Multi-Relational Data.. In *ICML*, Vol. 11. 809–816.

[28] Paolo Rosso, Dingqi Yang, and Philippe Cudré-Mauroux. 2019. Knowledge Graph Embeddings. In *Encyclopedia of Big Data Technologies*. https://doi.org/10.1007/978-3-319-63962-8_284-1

[29] Paolo Rosso, Dingqi Yang, and Philippe Cudre-Mauroux. 2019. Revisiting Text and Knowledge Graph Joint Embeddings: The Amount of Shared Information Matters!. In *Proceedings of the 2018 IEEE International Conference on Big Data (Big Data)*.

[30] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*. Springer, 593–607.

[31] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*. 926–934.

[32] 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 2017 ACM on Conference on Information and Knowledge Management*. ACM, 1029–1038.

[33] Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *EMNLP*. 1499–1509.

[34] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*. 2071–2080.

[35] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *TKDE* 29, 12 (2017), 2724–2743.

[36] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph and text jointly embedding. In *EMNLP*. 1591–1601.

[37] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes.. In *AAAI*, Vol. 14. 1112–1119.

[38] Jianfeng Wen, Jianxin Li, Yongyi Mao, Shini Chen, and Richong Zhang. 2016. On the representation and embedding of knowledge bases beyond binary relations. In *IJCAI*. AAAI Press, 1300–1307.

[39] Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd international conference on World wide web*. ACM, 515–526.

[40] Wikidata. 2012. http://wikidata.org/.

[41] Chenyan Xiong, Russell Power, and Jamie Callan. 2017. Explicit semantic ranking for academic search via knowledge graph embedding. In *Proceedings of the 26th international conference on world wide web*. International World Wide Web Conferences Steering Committee, 1271–1279.

[42] Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. In *CoNLL*. 250–259.

[43] Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2017. Learning Distributed Representations of Texts and Entities from Knowledge Base. *TACL* 5 (2017), 397–411. https://www.transacl.org/ojs/index.php/tacl/article/view/1065

[44] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR*.

[45] Dingqi Yang, Paolo Rosso, Bin Li, and Philippe Cudre-Mauroux. 2019. NodeSketch: Highly-Efficient Graph Embeddings via Recursive Sketching. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1162–1172.

[46] Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL and IJCNLP*. 1321–1331.

[47] Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *ACL*, Vol. 2. 545–550.

[48] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 353–362.

[49] Richong Zhang, Junpeng Li, Jiajie Mei, and Yongyi Mao. 2018. Scalable instance reconstruction in knowledge bases via relatedness affiliated embedding. In *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 1185–1194.

[50] Huaping Zhong, Jianwen Zhang, Zhen Wang, Hai Wan, and Zheng Chen. 2015. Aligning knowledge and text embeddings by entity descriptions. In *EMNLP*. 267–272.