Schema-Aware Hyper-Relational Knowledge Graph Embeddings for Link Prediction

Yuhuan Lu, Dingqi Yang*, Pengyang Wang, Paolo Rosso, and Philippe Cudre-Mauroux

Abstract—Knowledge Graph (KG) embeddings have become a powerful paradigm to resolve link prediction tasks for KG completion. The widely adopted triple-based representation, where each triplet (h, r, t) links two entities h and t through a relation r, oversimplifies the complex nature of the data stored in a 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. Moreover, as the KG schema information has been shown to be useful for resolving link prediction tasks, it is thus essential to incorporate the corresponding hyper-relational schema in KG embeddings. Against this background, we propose sHINGE, a schema-aware hyper-relational KG embedding model, which learns from hyper-relational facts directly (without the transformation to the n-ary representation) and their corresponding hyper-relational schema in a KG. Our extensive evaluation shows the superiority of sHINGE on various link prediction tasks over KGs. In particular, compared to a sizeable collection of 21 baselines, sHINGE consistently outperforms the best-performing triple-based KG embedding method, hyper-relational KG embedding method, and schema-aware KG embedding method by 19.1%, 1.8%, and 12.9%, respectively.

Index Terms—Knowledge graph embedding, Hyper-relation, Schema, Link prediction

1 Introduction

KNOWLEDGE Graphs (KGs) leverage a graph-structured data model to integrate interrelated entities via relations that encode the underlying semantics between the entities, representing real-world facts. Using the widely adopted triple-based representation, a fact is represented as a 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). Modern KGs such as Freebase [1], Google's Knowledge Graph [2] or Wikidata [3], contains a large amount of high-quality facts empowering a large range of Web applications including semantic search [4], question-answering [5], query expansion [6], or recommendation systems [7]. However, these KGs 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* [8], even though this is a mandatory property of the schema [9]. Against this background, Knowledge Graph completion problems have been widely studied. A key problem in this context is to predict the missing links in a KG (a.k.a. link prediction). 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.

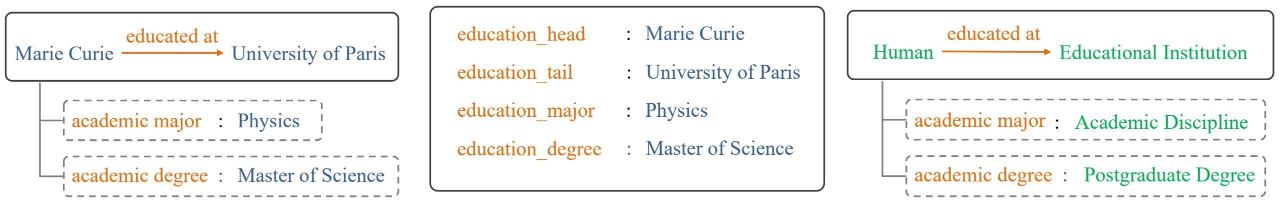
- *Yuhuan Lu, Dingqi Yang, and Pengyang Wang are with the State Key Laboratory of Internet of Things for Smart City and Department of Computer and Information Science, University of Macau, Macao SAR, China, E-mail: lu.yuhuan@connect.umac.mo, dingqiyang@um.edu.mo, pywang@um.edu.mo. Paolo Rosso and Philippe Cudre-Mauroux are with the University of Fribourg, Switzerland, E-mail: paolo@exascale.info, philippe.cudre-mauroux@unifr.ch.*
- **Corresponding author: Dingqi Yang (email: dingqiyang@um.edu.mo)*

Manuscript received April 19, 2005; revised August 26, 2015.

To resolve such link prediction tasks over KGs, Knowledge Graph embeddings have been shown as a powerful tool in the current literature [10]. 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 [11], 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 wide 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 [12] or n-ary [13] relational data), where each fact contains multiple relations and entities. Figure 1a 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. For example, more than 30% entities in Freebase are involved in such hyper-relational facts [12]. When learning KG embeddings, traditional methods transform those hyper-relational facts into triplets by either 1) keeping the base triplet only from a hyper-relational fact [14]; 2) creating additional triplets from a hyper-relational fact via reification [15]; or 3) creating

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.



Marie Curie
educated at
University of Paris
academic major : Physics
academic degree : Master of Science
education_head : Marie Curie
education_tail : University of Paris
education_major : Physics
education_degree : Master of Science
Human
educated at
Educational Institution
academic major : Academic Discipline
academic degree : Postgraduate Degree

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

Fig. 1: An example of a hyper-relational fact and its corresponding n-ary representation and schema information.

additional triplets from a hyper-relational fact via relation paths [12]. In our previous work [14], we have conducted extensive experiments using a hypothesis test showing that all these three transformations lead to significant performance degradation of up to 29.3% across different link prediction tasks and datasets.

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 [12], [13], [16], [17]. 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 1b 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 [12], [16], or learn from the relatedness between entity/relation pairs [13], [17]. However, key-value pairs (k, v) on a hyper-relational fact should not be treated identically as base triplet (h, r, t). As triplets still serve as the fundamental data structure in modern KGs, they preserve the essential information for link prediction. Thus, it is highly beneficial to directly capture the structure of the base triplets in hyper-relational facts [14], [18].

Moreover, to further exploit the hyper-relational facts, the schema information of the KG should be seamlessly incorporated into the KG embeddings. Specifically, schema information of KGs has been shown to be useful for resolving link prediction tasks [17], [19], [20]. Figure 1c shows the partial schema information on Wikidata related to the example about Marie Curie. When predicting {*Marie Curie*, *educated at*, ?}, its corresponding schema represented as an entity-typed triplet {*Human*, *educated at*, ?} suggests that the missing tail entity is likely to be of types *Educational Institution*, *Monastery*, or *chess club*, etc., according to Wikidata[2]. This can effectively help predict the missing entity by favoring the entities of these types. In the context of hyper-relational facts, the schema information on key-value pairs can further help the prediction. Following the previous example, the schema of the key-value pair {*academic degree*, *Master of Science*} is represented by an entity-typed key-value pair {*academic degree*, *Postgraduate Degree*}, which can serve as a strong clue to further favor the entity type *Educational Institution*. Because a *Postgraduate Degree* is more likely to be given by an *Educational Institution*, rather than by a *monastery* or a *chess club*. Therefore, such schema information is essential in learning high-quality KG embeddings for link prediction over hyper-relational facts.

Motivated by the above observation, we propose in this paper sHINGE, a schema-aware Hyper-relatIonal kNowledge Graph Embedding model. sHINGE is designed to directly learn from hyper-relational facts and their corresponding schema in a KG, capturing not only the primary structural information of the KG encoded in the triplets and their associated key-value pairs, but also the schema information represented by entity-typed triplets and their associated entity-typed key-value pairs. More precisely, for each hyper-relational fact, we first design two modules, each consisting of two convolutional neural network pipelines. On one hand, we learn from a base triplet (h, r, t) and associated key-value pairs together with the triplet itself (h, r, t, k, v), generating a triple-wise relatedness feature vector and quintuple-wise relatedness feature vectors, respectively. Afterward, we compute the fact 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. On the other hand, we learn from the corresponding entity-typed triplet(s) (h_type, r, t_type) and each associated entity-typed key-value pairs together with the entity-typed triplet itself $(h\_type, r, t\_type, k, v\_type)$, generating triple-wise schema relatedness feature vector(s) and quintuple-wise schema relatedness feature vector. Then, we also impose the same minimum operation as depicted above on these feature vectors to obtain the schema relatedness feature vector. Finally, based on the fact and schema relatedness feature vectors, we concatenate them into an overall relatedness feature vector and employ a fully connected projection to output the predicted score for the input hyper-relational fact. Our contributions are hence three-fold:

- We investigate the problem of schema-aware hyper-relational Knowledge Graph embedding, where each hyper-relational fact not only contains a base triplet together with an arbitrary number of key-value pairs, but also is associated with the corresponding hyper-relational

2. https://www.wikidata.org/wiki/Property:P69

schema information represented as entity-typed triplets and key-value pairs.

- We introduce sHINGE, a schema-aware Hyper-relatIonal kNowledge Graph Embedding model, designed to directly learn from hyper-relational facts and their corresponding schema information in a KG, capturing not only the primary structural information of the KG encoded in the triplets and their associated key-value pairs, but also the schema information encoded by entity-typed triplets and their associated entity-typed key-value pairs.
- We conduct a thorough evaluation of our method compared to a sizeable collection of 21 baselines on two real-world KG datasets using two link prediction tasks. Our results show that sHINGE consistently outperforms the best-performing triple-based KG embedding method, hyper-relational KG embedding method, and schema-aware KG embedding method by 19.1%, 1.8%, and 12.9%, respectively.

2 Related Works

Graph embeddings have become a key paradigm to learn representations of nodes in a graph and facilitate downstream graph analysis tasks [21], [22], [23]. 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 [10]. In the following, we briefly discuss existing KG Embedding techniques learning from 1) triplets only, 2) triplets with other data, 3) hyper-relational facts, and 4) schema of KGs.

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 [24]. First, translational distance models exploit distance-based scoring functions to create embeddings. One representative model of this family is TransE [11], 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 a more sophisticated scoring function involving relation-specific hyperplanes [25] or spaces [26], [27], [28], for example. Second, semantic matching models exploit similarity-based scoring functions. One typical model in that context is RESCAL [29]. 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 [30], by capturing asymmetric relations [31], or by modeling non-linear relations using neural networks [32], [33], [34], [35], [36].

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 [15] or relation paths [12], 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 [14]. 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, besides 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 [37], [38] or text literal [39], 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) [40]. 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) [41], [42], [43], [44], [45], [46], [47], [48], [49], [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 differs 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) [12], [13], [16]. 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 [13], 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 [12] models the interaction between entities involved in each fact in order to perform link prediction on missing entities. RAE [16] 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 [13] 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). NeuInfer [18] treats the information in the same n-ary fact discriminatively and represents each n-ary fact as a primary triplet associated with a set of descriptive key-value pair(s). HypE [51] improves entity embeddings with positional embeddings, which is employed to capture the semantics of an entity referring to its position in a relation. HyperMLN [52] interprets the path-reasoning process with first-order logic, simultaneously improving knowledge embeddings and logic rules optimization. Different from modeling hyper-relational facts in Euclidean space, PolygonE [53] represents n-ary relations in hyperbolic space, capturing the hierarchical structure. Tensor decomposition which yields satisfactory performance in triplet embeddings can also be generalized for hyper-relational facts. GETD [54] integrates Tucker decomposition with Tensor Ring decomposition, resulting in a favorable performance for hyper-relational facts with different arities. S2S [55] further extends GETD to learn from hyper-relational facts with mixed arity through embedding sharing techniques. Enlightened by the powerful ability of Transformer [56] in sequence modeling, some recent works represented a hyper-relational fact as a semantic sequence and applied a self-attention mechanism to capture sophisticated interactions between sequence elements. StarE [57] combines a message-passing-based graph encoder with a Transformer-based decoder to model hyper-relational facts with an arbitrary number of key-value pairs. GRAN [58] represents a hyper-relational fact as a heterogeneous graph and employs edge-biased attention to capture inter-vertex interactions. Hy-Transformer [59] extends StarE substituting the graph encoder by a lightweight relation/type embedding technique, improving the efficiency of link prediction. QUAD [60] also extends StarE by adopting two separate aggregators to encode the primary entity-typed triplets and associated key-type pairs, respectively.

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. Therefore, in this paper, we design sHINGE to directly learn from the base triplets even for hyper-relational facts, while simultaneously learning from the associated key-value pairs.

2.4 Schema-aware KG Embeddings

A real world KG is usually endowed with an additional ontological schema depicting the structural logic followed by the facts in the KG, which has the potential to improve the link prediction performance over the KG. Some recent works focused on the fusion of schema into KG embeddings. SIC [61] completes a KG by iteratively checking the correctness of candidate triplets against the KG schema. MTE [62] models the relations and schema in a KG simultaneously to enhance the representation ability, where the multi-types of each entity are formulated as a taxonomy tree and fed into a multi-type embedding layer to produce the type-level entity embedding. tNaLP [20] is an extension of NaLP [13], which learns type embeddings by introducing type constraints of roles and role-values in an unsupervised manner. RETA [19] is an end-to-end solution fully leveraging schema information encoded in triplets. It raises the matching of relation-tail pairs by considering the plausibility of both triplets and their corresponding schema. RAM [17] represents schema by linear combinations of basic vectors in a latent space, which promotes the semantically related entity types to have close representations.

However, modeling the schema of a KG as the n-ary representation of entity types in a hyper-relational fact cannot fully benefit from the rich semantics encoded in the schema. Similar to the case for hyper-relational facts, the base entity-typed triplet possesses the fundamental schema information of a hyper-relational fact and should be treated as the main structure (as we show in our experiments in Section 5.2). In other words, the entity-typed key-value pairs should serve as the companions for the base entity-typed triplet. Therefore, in this paper, we propose sHINGE to capture such hyper-relational schema by learning from both the base entity-typed triplets and associated entity-typed key-value pairs simultaneously.

3 Schema-aware Hyper-relational KG Embeddings

In this section, we introduce sHINGE, our proposed KG embedding model learning from both hyper-relational facts and their corresponding schema. We introduce several formal definitions:

Definition 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 2. **Triple fact**: A triple fact contains a triplet (h, r, t) only.

Definition 3. **Schema**: The ontology represents the semantics of facts in a KG. In this study, we formulate the schema for fact (h, r, t) as entity-typed triplet (h_type, r, t_type) while for an associated key-value pair (k_i, v_i) as (k_i, v_i_type).

Based on these definitions, Figure 2 illustrates our proposed model sHINGE. It is designed to directly learn from both hyper-relational facts and their corresponding schema in a KG, capturing not only the structural information of the KG characterized by hyper-relational facts, but also the schema information encoded in the hyper-relational facts. More precisely, sHINGE consists of three parts. The first part learns from the hyper-relational facts. 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 a unique hyper-relational relatedness feature vector. On the other hand, the second part learns the schema information from hyper-relational facts. For the same hyper-relational fact fed into the first part, it 1) learns to capture the corresponding schema information encoded in the base triplet, generating a set of triple-wise schema relatedness feature vectors, one for each entity-typed triplet (h_type, r, t_type), and 2) learns to capture the corresponding schema information encoded in each key-value pair associated with the base triplet together with its corresponding entity-typed triplets, generating a set of quintuple-wise schema relatedness vectors, one for each entity-typed hyper-relational fact $(h\_type, r, t\_type, k, v\_type)$. Likewise, it then 3) merges these relatedness feature vectors to a unique schema relatedness feature vector. Finally, the obtained hyper-relational and schema relatedness feature vectors are concatenated to generate a final prediction score. In the following, we present the details of these three modules.

3.1 Learning from Hyper-relational Facts

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 $\vec{\phi} \in \mathbb{R}^{1\times n_f(K-2)}$. The process of learning from a triplet can be formulated as:

$$\vec{\phi} = \text{Flatten}\left(\text{Conv}\left(\left[\vec{h}, \vec{r}, \vec{t}\right]^{\text{T}}, \mathbf{W}_{\phi}\right)\right) \tag{1}$$

where $[\cdot, \cdot]$ denotes the concatenation operation along the row of input vectors and $\mathbf{W}_\phi$ is the learnable parameters of CNN filters. This relatedness vector $\vec{\phi}$ can be used to characterize the plausibility of a (base) triplet (h, r, t) of being true.

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 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 $\vec{\psi}_i$, $i = 1, .., n$. The learning process from a key-value pair can be formulated as:

$$\vec{\psi}_i = \text{Flatten}\left(\text{Conv}\left(\left[\vec{h}, \vec{r}, \vec{t}, \vec{k}_i, \vec{v}_i\right]^{\text{T}}, \mathbf{W}_{\psi}\right)\right) \tag{2}$$

where $\mathbf{W}_\psi$ denotes the learnable parameters of CNN filters, which is shared by multiple key-value pairs. 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. Note that this module is not used for triple facts, as they do not contain any key-value pair.

Merging Relatedness Feature Vectors. In the previous two steps, 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 make final prediction for the input hyper-relational facts. To achieve this goal, we compute the fact 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 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 fact relatedness feature vector $\vec{h}_f$. This process can be formulated as:

$$\vec{h}_f = \text{Min}\left(\left[\vec{\phi}; \vec{\psi}_1; \vec{\psi}_2; \ldots; \vec{\psi}_n\right]\right) \tag{3}$$

where $[\cdot;\cdot]$ refers to the concatenation operation along the column of input vectors. 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 [13], [14].

3.2 Learning from Schema

Learning from Entity-typed Triplets. Schema information encoded in triplets is crucial for evaluating the semantic plausibility and in turn enhancing the link prediction performance. Thus, the schema information is also extracted by CNN to capture the interactions between the elements in the entity-typed triplet. With the similar procedure to learning from triples, a triple-wise schema relatedness vector $\vec{\xi}_j \in$

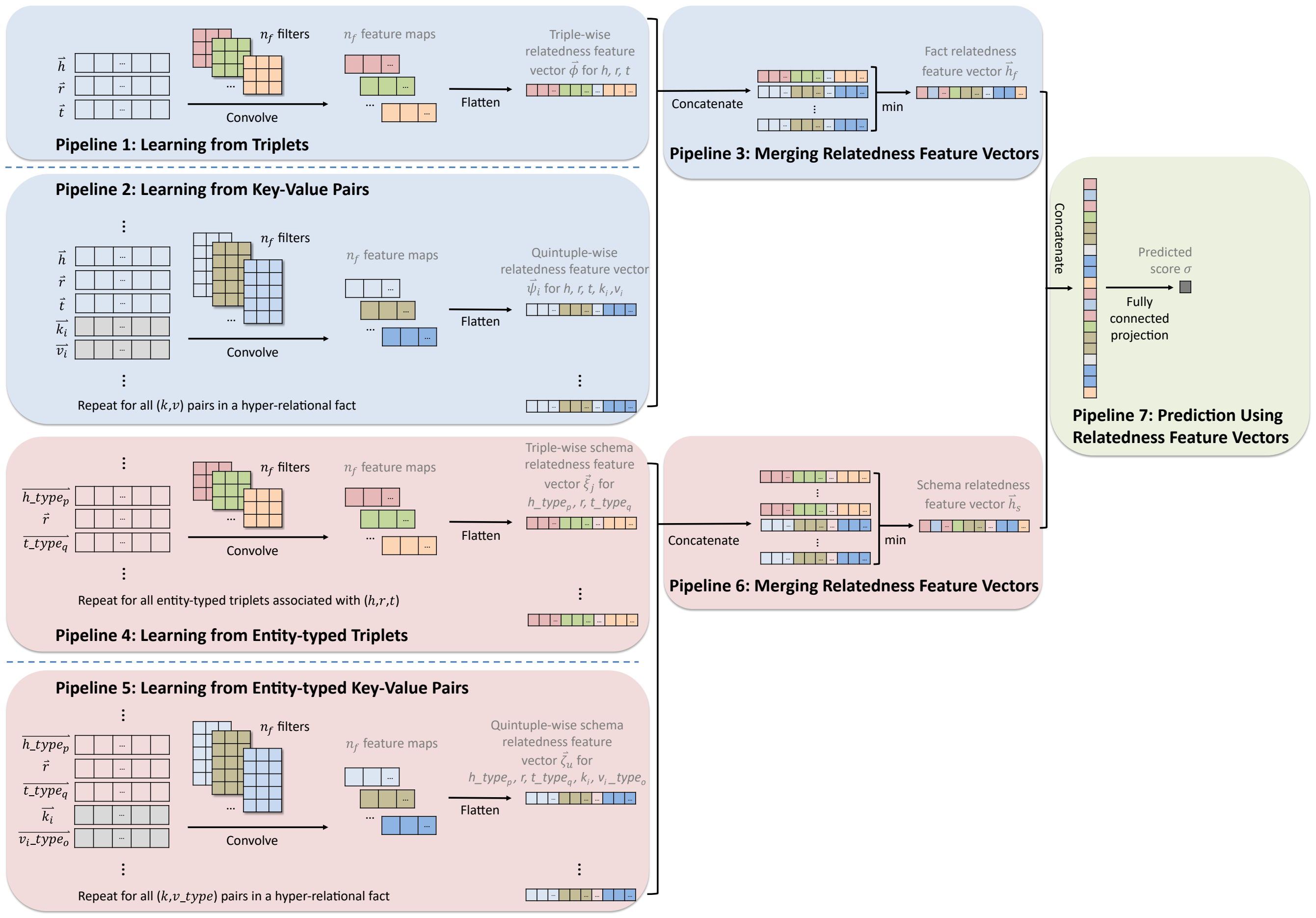

Pipeline 1: Learning from Triplets
n_f filters
n_f feature maps
Convolve
Flatten
Triple-wise relatedness feature vector $\vec{\phi}$ for h, r, t
Pipeline 2: Learning from Key-Value Pairs
Quintuple-wise relatedness feature vector $\vec{\psi}_i$ for h, r, t, k_i, v_i
Repeat for all (k,v) pairs in a hyper-relational fact
Pipeline 3: Merging Relatedness Feature Vectors
Concatenate
min
Fact relatedness feature vector $\vec{h}_f$
Pipeline 4: Learning from Entity-typed Triplets
Triple-wise schema relatedness feature vector $\vec{\xi}_j$ for h_type_p, r, t_type_q
Repeat for all entity-typed triplets associated with (h,r,t)
Pipeline 5: Learning from Entity-typed Key-Value Pairs
Quintuple-wise schema relatedness feature vector $\vec{\zeta}_u$ for h_type_p, r, t_type_q, k_i, $v_i\_type_o$
Repeat for all (k,v_type) pairs in a hyper-relational fact
Pipeline 6: Merging Relatedness Feature Vectors
Schema relatedness feature vector $\vec{h}_s$
Pipeline 7: Prediction Using Relatedness Feature Vectors
Predicted score σ
Fully connected projection

Fig. 2: Overview of our proposed method sHINGE. The blue, red and green backgrounds correspond to the module 1, 2 and 3 (Section 3.1, 3.2 and 3.3), respectively.

$\mathbb{R}^{1\times n_f(K-2)}$ is obtained given a specific input entity-typed triplet (h_type_p, r, t_type_q). The learning process from an entity-typed triplet is akin to Formula (1), which is shown as follows:

$$\vec{\xi_j} = \text{Flatten}\left(\text{Conv}\left(\left[\overrightarrow{h\_type}_p, r, \overrightarrow{t\_type}_q\right]^{\mathrm{T}}, \mathbf{W}_\xi\right)\right) \quad (4)$$

where $\mathbf{W}_\xi$ denotes the learnable parameters of CNN filters. This relatedness feature vector $\vec{\xi_j}$ facilitates the plausibility characterization of a base triplet being true. Here, $\overrightarrow{h\_type}_p$ and $\overrightarrow{t\_type}_q$ denote a given head type and tail type, respectively. Assume that $1 \le p \le m_1$ and $1 \le q \le m_2$, we obtain $m_1 m_2$ relatedness feature vectors $\vec{\xi_j}$, $j = 1, .., m_1 m_2$ after the iterative convolutional operations.

Learning from Entity-typed Key-Value Pairs. Inheriting the idea of key-value pairs learning process, the entity-typed key-value pairs are coupled with the corresponding entity-typed triplet to learn the further schema information. For each entity-typed key-value pair $(k_i, v_i\_type_o)$ associated with the entity-typed triplet (h_type_p, r, t_type_q), the CNN with the same structure as that of learning from key-value pairs is applied to capture the interaction between each element in the entity-typed triplet and the entity-typed key-value pair, i.e., h_type_p, r, t_type_q, k_i and $v_i\_type_o$, and then generate the quintuple-wise relatedness vector $\vec{\zeta_u} \in \mathbb{R}^{1\times n_f(K-2)}$. The learning process is similar to Formula (2), which is presented as follows :

$$\vec{\zeta_u} = \text{Flatten}\left(\text{Conv}\left(\left[\overrightarrow{h\_type}_p, r, \overrightarrow{t\_type}_q, \vec{k_i}, \overrightarrow{v_i\_type}_o\right]^{\mathrm{T}}, \mathbf{W}_\zeta\right)\right) \quad (5)$$

where $\mathbf{W}_\zeta$ is the learnable parameters of CNN filters. Here, $\overrightarrow{v_i\_type}_o$ refers to a given value type. This relatedness feature vector $\vec{\zeta_u}$ is beneficial to the plausibility characterization of the base triplet (h, r, t) associated with the key-value pair (k_i, v_i) being true. Assume that $1 \le o \le c$, we obtain $m_1 m_2 nc$ relatedness feature vectors $\vec{\zeta_u}$, $u = 1, .., m_1 m_2 nc$ after the repeated process of convolution.

Merging Relatedness Feature Vectors Referring to the merging process of the first module, the schema relatedness feature vector is calculated by taking the minimum value along each feature dimension over the triple-wise schema relatedness feature vector and all the quintuple-wise schema relatedness feature vectors. More precisely, the $m_1 m_2$ triple-wise schema relatedness feature vectors $\vec{\xi_j}$ and the $m_1 m_2 nc$ quintuple-wise schema relatedness feature vectors $\vec{\zeta_u}$ are concatenated into a matrix of size $(m_1 m_2 + m_1 m_2 nc) \times n_f(K-2)$, and then the minimum operation is applied to this matrix along each column, producing the schema relatedness feature vector $\vec{h}_s$. This process can be formulated as:

$$\vec{h}_s = \mathrm{Min}\left(\left[\vec{\xi}_1; \vec{\xi}_2; \ldots; \vec{\xi}_{m_1 m_2}; \vec{\zeta}_1; \vec{\zeta}_2; \ldots; \vec{\zeta}_{m_1 m_2 nc}\right]\right) \quad (6)$$

3.3 Prediction Using Relatedness Feature Vectors

Based on the above two modules, we obtain a fact relatedness feature vector $\vec{h}_f$ and a schema relatedness feature vector $\vec{h}_s$, both of which have the same size of $1 \times n_f(K-2)$. Subsequently, these two vectors are concatenated into the overall relatedness feature vector of size $2 \times n_f(K-2)$. 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. The pseudocode of the proposed method is presented in Algorithm 1.

3.4 Model Training Process

To train the model parameters, we minimize a softplus loss. More precisely, following [13], [31], our loss function is defined as the negative log-likelihood of the logistic model:

$$\sum_{\omega \in \Omega} log(1 + e^{-\sigma(\omega)}) + log(1 + e^{\sigma(\omega')}) \quad (7)$$

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), $i = 1, ..., n$, 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 sHINGE model for the true fact ω and the negative fact ω', respectively.

The loss function 7 is minimized using the Adam stochastic optimizer [63], and the model parameters are learnt via back propagation. Specifically, we use rectified linear units (ReLU) as the non-linearity activation function [64] and batch normalization [65] after the two CNN layers for fast training.

Algorithm 1: Link prediction using sHINGE.

```
Input:
    A hyper-relational fact, $(h, r, t, k_i, v_i, \ldots)$;
    The corresponding schema,
    $(h\_type, r, t\_type, k_i, v_i\_type, \ldots)$;
    The entity embedding set, $\mathbf{E}$;
    The relation embedding set, $\mathbf{R}$;
    The type embedding set, $\mathbf{T}$.
Output:
    The plausibility score $\sigma$.

    $\vec{\phi} \leftarrow$ Formula (1)
    while $i \leq n$ do
        $\vec{\psi}_i \leftarrow$ Formula (2)
    end while
    $\vec{h}_f \leftarrow$ Formula (3)
    while $j \leq m_1 m_2$ do
        $\vec{\xi}_j \leftarrow$ Formula (4)
    end while
    while $u \leq m_1 m_2 nc$ do
        $\vec{\zeta}_u \leftarrow$ Formula (5)
    end while
    $\vec{h}_s \leftarrow$ Formula (6)
    Concatenate $\vec{h}_f$ and $\vec{h}_s$
    $\sigma \leftarrow$ Fully connected projection of $\left[\vec{h}_f, \vec{h}_s\right]^{\mathrm{T}}$
```

4 Discussion

4.1 Complexity of sHINGE

In this section, we discuss the complexity of the proposed sHINGE. Given the number of training facts N, its time complexity is $\mathcal{O}(N \cdot \overline{e\_type}^3 \cdot \overline{n} \cdot n_f \cdot K)$, where $\overline{e\_type}$ denotes the average number of types per entity, and $\overline{n}$ denotes the average number of key-value pairs per entity.

Note that the cubic term $\overline{e\_type}^3$ is caused by the learning process of entity-typed key-value pairs, where a relatedness feature vector is learnt from quintuples $(h\_type, r, t\_type, k, v\_type)$, requiring iterating all combinations of h_type, t_type, and v_type. To alleviate this complexity issue, we can limit the number of types learnt per entity in practice, i.e., setting an upper bound for $\overline{e\_type}$ to limit the complexity. In practice, it has been shown that a small number of entity types are enough to capture the schema relatedness for KG completion tasks [19]. Subsequently, in this paper, we learn from the top one type for each entity, where the types are ranked according to their popularity in the whole KG. In this context, as $\overline{e\_type} = 1$, the time complexity of sHINGE $\mathcal{O}(N \cdot \overline{e\_type}^3 \cdot \overline{n} \cdot n_f \cdot K)$ simplifies to $\mathcal{O}(N \cdot \overline{n} \cdot n_f \cdot K)$.

In the future, we plan to address the complexity issue from two aspects: 1) investigate more on scalable methods (via sampling techniques for example) to reduce such a cubic complexity when learning from hyper-relational schema; and 2) explore type fusion methods to feed an aggregated type into sHINGE for each entity.

4.2 Learning from Facts with Missing Information

Missing information is common in modern KGs. For hyper-relational KG embeddings, the missing information can be in the two following cases: 1) some entities in a fact are not associated with any types; and 2) some facts are not associated with any key-value pairs (i.e., triple fact). In this context, sHINGE is designed to flexibly accommodate such facts with missing information. Specifically, sHINGE separately computes the relatedness feature vectors with respect to triplets, key-value pairs and their corresponding schema information, and then merges these relatedness feature vectors for the final prediction. It can flexibly handle the two cases: 1) When an entity has no type information in a KG, we assign an ”unknown” type to the entity, and then keep the same processing pipeline for prediction. Notably, the ”unknown” type is utilized as a common type for entities with missing type information as ”Schema.org”[3], which employs the type ”Thing” to encapsulate all entities on the Internet. This ensures that sHINGE is able to learn

3. https://schema.org/docs/full.html: ”Schema.org” provides the schema for structured data on the Internet and has been widely adopted by many world-class technology companies, such as Google, Microsoft and Pinterest.

from the facts with missing entity types. 2) When a fact has no key-value pairs, sHINGE only calculates the triple-wise relatedness feature vector in the first module and triple-wise schema relatedness feature vector in the second module (pipelines 1 and 4 in Fig. 2, respectively). Subsequently, these two feature vectors are fed into the module 3 (pipeline 7 in Fig. 2) to produce the final prediction score.

4.3 Entities with Hierarchical Types

The underlying hierarchical structure of entity types could also be useful for KG link prediction. There are two categories of treatment for the hierarchy of entity types. The first category explicitly models the hierarchy of types to support embedding. HAKE [66] employs the polar coordinate system to represent an entity at different levels of the hierarchy, while TaRP [67] assigns different weights to entity types according to their positions in the hierarchy. The second category implicitly models hierarchical types and produces aggregated type information. DHGE [68] integrates hierarchical types by a hypergraph neural network, while AttEt [69] proposes neighborhood attention to aggregate hierarchical types with type-specific attention weights. In this study, sHINGE handles entity types equally. However, the treatment for hierarchical types mentioned above can be easily incorporated into sHINGE due to its flexible network structure. The fusion of the type hierarchy will be considered in our future work.

4.4 Negative Sampling Learning Scheme

We discuss and compare the two mainstream learning schemes for hyper-relational KGs, i.e., 1) negative-sampling-based learning methods adopted by NaLP [13], NeuInfer [18], HypE [51], HyperMLN [52], GETD [54], as well as our sHINGE; and 2) self-attention-based methods (with full softmax) adopted by GRAN [58], StarE [57], Hy-Transformer [59] and QUAD [60], in the following three aspects:

- *Full softmax approximation using negative sampling*. The softmax function is commonly used in these self-attention-based methods to distinguish the most likely element for each missing element in a hyper-relational fact. However, the computation of the full softmax function faces great computational challenges for large-scale KGs with a large number of candidate entities and relations for one missing element. By contrast, negative sampling is an efficient technique to approximate the full softmax function and is often used on large-scale problems [70].
- *More than one missing element and fact-level scoring*. Self-attention-based methods are often limited to predicting only one missing element for a given hyper-relational fact due to the mask mechanism [58], while negative-sampling-based methods are more flexible to predict any number of missing elements for a fact as it is capable of evaluating a fact-level plausibility score for any given fact [19]. In other words, negative sampling methods offer more flexibility in knowledge representation and better interpretability of link prediction [71].
- *Arity-sensitive complexity.* Self-attention-based methods usually adopt a padding mechanism in the training process so as to accommodate different fact arities; subsequently, the size of input depends on the largest arity in a dataset. However, for modern KGs, only a few facts have a very large arity, which makes such a padding mechanism inefficient because a large number of inputs are indeed padded [57], [58], [59], [60]. In practice, these methods often set an upper bound for the maximum arity for a bounded complexity. In contrast, as discussed in Section 4.1, the complexity of sHINGE does not depend on the maximum fact arity in the KG.

Therefore, we advocate for the negative-sampling-based methods in this paper, even though self-attention-based methods sometimes show promising performance in some tasks [58].

5 EXPERIMENTS

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

5.1 Experimental Setup

5.1.1 Dataset

We conduct experiments on two hyper-relational datasets ***JF17K*** [12] and ***WikiPeople*** [13], 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 [12] 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 [13]. 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.

In practice, open-domain KGs usually do not have a unified and fixed schema. Even though some effort such as Schema.org has been made to create the unified and shared schema for structured data on the Web, such schema still has a low coverage on the Web, and thus has not been widely adopted by modern KGs. To fill the shortfall of schema in JF17K and WikiPeople, we extract the entity type information from their corresponding data sources (Freebase and Wikidata) following the procedure illustrated in [19]. For Freebase, entity types are crawled through the entity node described as "/type/object". For Wikidata, entity types are extracted from the property "instance_of" for each entity. Notably, all types are treated equally in this study, and the hierarchy of entity types remains as the future work, as pointed out in the previous discussion section.

5.1.2 Baselines

We compare sHINGE against a sizeable collection of state-of-the-art KG embedding techniques from three categories. First, models learning from triplets only:

- Translational distance models: **TransE** [11] learns to preserve the relation between two entities as $h + r \approx t$. **TransH** [25] extends TransE to better capture multi-mapping relations by introducing relation-specific hyperplanes. **TransR** [26] introduces relation-specific projections to also better capture multi-mapping relations.

TABLE 1: Statistics of the datasets

Dataset		JF17K		WikiPeople	
#Entity		28,645		34,839	
#Relation		322		375	
#Type		652		395	
#Fact		100,947		325,504	
#Triple Fact w/ types		46,210		265,573	
#Triple Fact w/o types		8,417		51,544	
#Hyper Fact w/ types		40,725		7,534	
#Hyper Fact w/o types		5,595		826	
#Fact (training)	Triple only	44,210	57.8%	280,520	97.4%
	Hyper only	32,169	42.2%	7,389	2.6%
	Total	76,379	100%	287,918	100%
#Fact (test)	Triple only	10,417	42.4%	36,597	97.4%
	Hyper only	14,151	57.6%	971	2.6%
	Total	24,568	100%	37,586	100%

TransD [27] 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** [29] 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** [30] simplifies Rescal by representing each relation embedding as a diagonal matrix. **ComplEx** [31] further extends DistMult in the complex space in order to better model both symmetric and asymmetric relations. **Analogy** [72] models explicitly analogical structures in multi-relational KG embeddings. **ConvE** [33] 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 [73], 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.

Second, models learning from hyper-relational facts. As discussed in Section 4.4, we focus on negative-sampling-based methods and exclude self-attention-based models in our experiments even though the latter sometimes show promising performance in some tasks.

- **m-TransH** [12] models the interaction between entities involved in each n-ary fact. Each fact is represented as a list of ordered values associated with a meta-relation. Using this representation, m-TransH can only be applied to perform the link prediction task on missing entities.
- **RAE** [16] 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. Similar to m-TransH, RAE can only be used to predict missing entities.
- **NaLP** [13] models the relatedness between key-value (relation-entity) pairs contained in each n-ary fact. It represents each hyper-relational fact as a set of key-value pairs 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).
- **NaLP-Fix** is our 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.
- **NeuInfer** [18] represents each hyper-relational fact as a primary triplet associated with a set of descriptive key-value pairs. Then it further employs a fully-connected neural network model to perform the prediction tasks on hyper-relational facts.
- **HypE** [51] enhances entity embedding performance with positional embeddings. Positional embeddings are decoupled from entity modeling and are utilized to represent entities based on their positions in relations.
- **HyperMLN** [52] is an explainable model that interprets the path-reasoning process combined with first-order logic. Logic rules can improve the knowledge embeddings, while the semantics captured by the embedding model benefits the generation of logic rules in turn.
- **GETD** [54] combines Tucker decomposition with Tensor Ring decomposition to fully express hyper-relational facts with any size or arity.
- **HINGE** [14] is the original version of sHINGE, which views an n-ary fact as a primary triplet and its associated key-value pairs, capturing triple-wise and quintuple-wise relatedness for (h, r, t) and (h, r, t, k_i, v_i) in a hyper-relational fact using convolution neural networks.

Third, models learning from the schema of KG:

- **SIC** [61] iteratively applies different types of knowledge graph completion models to produce multiple triplets and then evaluate these triplets by a schema-correctness metric.
- **RAM** [17] constructs a latent space that represents types as linear combinations of basic vectors and thus types with similar semantics will present close representations in the space. Subsequently, a pattern matrix is explored to evaluate the compatibility between the embeddings of types and related entities.
- **tNaLP** [20] is the extension of NaLP with the consideration of schema information. In tNaLP, type constraints are calculated as the type compatibility using a neural network model. The learnt type constraints are imposed on the fact embedding pipeline to improve the prediction performance.
- **sHINGE** is our proposed method. 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 implementation of sHINGE and used datasets are available here[4].

The embedding size is set to 100 for all methods, if not specified otherwise.

4. https://github.com/RyanLu32/sHINGE/

5.1.3 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 [13]. 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.

5.2 Link Prediction Performance Comparison

In this experiment, we compare the link prediction performance of sHINGE and HINGE (the original version of sHINGE, without learning from schema information) against all baselines on different tasks. Table 2 shows the results on both datasets. The best-performing method from each category of techniques is highlighted. In the following, we discuss the results and our key findings.

5.2.1 Comparison to Baselines Learning from Triplets Only

We observe that both HINGE and sHINGE consistently outperform all baselines learning from triplets only. Specifically, sHINGE achieves the best performance in most cases, showing an improvement of 19.1% over the best-performing baselines on average (23.1% on head/tail prediction and of 15.1% on relation prediction).

One exception is for the head/tail prediction on WikiPeople, where the improvement is marginal (1.9%). We further find that the best-performing baseline in this case is ConvE, which shows a tiny advantage in MRR but performs badly in Hit@1 compared to sHINGE (see Table 2). Note that similar to HINGE and sHINGE, 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% hyper-relational facts in both training and test datasets), where both HINGE, sHINGE and ConvE perform well. In contrast, on 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 and sHINGE significantly outperform ConvE by leveraging key-value pairs in the hyper-relational facts. In addition, we also highlight that ConvE is specifically designed for head/tail prediction only, and is not applicable to the relation prediction task.

5.2.2 Comparison to Baselines Learning from Hyper-Relational Facts

We observe that NeuInfer, HypE, HyperMLN, HINGE and sHINGE perform significantly better than other n-ary representation baselines (m-TransH, RAE, NaLP and NaLP-Fix). *This verifies the superiority of preserving the triple structure over the n-ary representation when learning hyper-relational facts.*

Moreover, among the methods preserving the triple structure of hyper-relational facts, HINGE (and sHINGE) perform better than NeuInfer. This is attributed to the fact that NeuInfer uses a vanilla fully-connected layer for learning the relatedness feature vectors, while HINGE (and sHINGE) employs 2D CNNs, capturing the rich interactions between entities, relations and key-value pairs. Compared to its original version HINGE, sHINGE further learning from KG schema information yields 1.8% improvement on average (1.7% and 1.9% improvements on head/tail prediction and relation prediction, respectively).

5.2.3 Comparison to Baselines Learning from KG Schema

Among the methods learning from hyper-relational facts with schema information, sHINGE consistently outperforms all other baselines on both datasets, showing an improvement of 12.9% over the best-performing baselines on average (15.8% on head/tail prediction and of 10.1% on relation prediction).

Moreover, we observe that tNaLP and sHINGE consistently outperform their original versions without learning from KG schema, NaLP and HINGE, by 43.7% and 1.8% on average, respectively. *This verifies the usefulness of KG schema information in performing link prediction tasks.*

Finally, compared to tNaLP which learns from schema information under the n-ary representation of hyper-relational facts, sHINGE achieves higher performance. *This implies that when learning from schema information, the triple structure is also more effective than the n-ary representation.*

5.3 Key/Value Prediction Performance Comparison

In this experiment, we compare the performance of sHINGE and HINGE on a key/value prediction task against all applicable baselines. Our evaluation protocol is similar to the one from the link prediction task.

Table 3 shows the results. We discover that both HINGE and sHINGE consistently outperform all baselines learning from hyper-relational facts with or without schema information. The improvement is particularly large when predicting values on JF17K dataset due to the larger proportion of hyper-relational facts in JF17K than WikiPeople. This demonstrates that HINGE (and sHINGE) have an advantage in representing hyper-relational facts. In general, sHINGE achieves the best performance in most cases, yielding an improvement of 3.6% over its original version

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

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
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	
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
NeuInfer	0.4233	0.5576	0.3503	0.4962	0.8737	0.3314	0.2378	0.3524	0.1781	0.6502	0.8395	0.5517
HypE	0.4527	0.5319	0.3805	0.8621	0.9033	0.8374	0.4236	0.5748	0.3019	0.8103	0.8577	0.7562
HyperMLN	0.4708	0.5698	0.4015	0.9017	0.9486	0.8752	0.4529	0.6033	0.3561	0.8764	0.9125	0.8358
GETD	0.4629	0.5536	0.3822		N/A		0.4383	0.5839	0.3391		N/A	
HINGE	**0.4711**	**0.5811**	**0.4079**	**0.9494**	**0.9971**	**0.9164**	**0.4548**	**0.6229**	**0.3649**	**0.9282**	**0.9769**	**0.8966**
SIC	0.3378	0.4089	0.3044	0.6923	0.7281	0.6735	0.2842	0.3677	0.2695	0.6740	0.7136	0.6108
RAM	0.4529	0.5530	0.3728	0.8611	0.9522	0.8217	0.3309	0.5077	0.3190	0.8724	0.9231	0.7743
tNaLP	0.4648	0.5792	0.3801	0.8724	0.9831	0.8051	0.3207	0.4583	0.2690	0.8533	0.9571	0.7928
sHINGE	**0.4780**	**0.5860**	**0.4251**	**0.9506**	**0.9977**	**0.9172**	**0.4582**	**0.6279**	**0.3718**	**0.9664**	**0.9961**	**0.9430**

TABLE 3: Key/Value prediction performance on both WikiPeople and JF17K.

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
m-TransH	0.1429	0.4258	0.1277		N/A		0.2695	0.5831	0.3374		N/A	
RAE	0.1273	0.4025	0.1078		N/A		0.2737	0.5892	0.3406		N/A	
NaLP	0.4297	0.5543	0.3508	0.5272	0.8903	0.3485	0.2976	0.4083	0.2415	0.6579	0.8525	0.5617
NaLP-Fix	0.4481	0.5629	0.3703	0.8315	0.9796	0.7318	0.3644	0.5017	0.3196	0.7815	0.9154	0.6930
NeuInfer	0.4538	0.5819	0.3705	0.6274	0.9306	0.4581	0.3785	0.5263	0.3272	0.7890	0.9017	0.7041
HypE	0.4613	0.5420	0.3865	0.8874	0.9151	0.8563	0.4726	0.6468	0.3637	0.8315	0.8809	0.7834
HyperMLN	0.4838	0.5921	0.4052	0.9225	0.9519	0.8972	0.4814	0.6328	0.3850	0.9088	0.9217	0.8406
GETD	0.4835	0.5719	0.3964		N/A		0.4813	0.6417	0.3866		N/A	
HINGE	**0.4928**	**0.6377**	**0.4192**	**0.9479**	**0.9850**	**0.9222**	**0.5490**	**0.6770**	**0.4715**	**0.9988**	**0.9996**	**0.9975**
SIC	0.3527	0.4489	0.3156	0.7488	0.7903	0.7024	0.3091	0.3812	0.2854	0.7187	0.7620	0.6531
RAM	0.4921	0.6083	0.4037	0.9054	0.9627	0.8536	0.4012	0.6084	0.3699	0.9201	0.9572	0.8103
tNaLP	0.4879	0.6217	0.4082	0.9238	0.9825	0.9104	0.3923	0.5273	0.3147	0.9182	0.9824	0.8453
sHINGE	**0.5494**	**0.6748**	**0.4903**	**0.9580**	**0.9910**	**0.9292**	**0.5578**	**0.6941**	**0.4818**	**0.9993**	**1.0000**	**0.9991**

TABLE 4: The impact of schema on link prediction performance.

Fact Type	Schema	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	w/o	0.4765	0.5874	0.3937	0.9493	0.9979	0.9145	0.2641	0.4965	0.1572	0.8723	0.9846	0.7965
	w/	**0.4842**	**0.6287**	**0.4508**	**0.9906**	**0.9989**	**0.9573**	**0.2941**	**0.5271**	**0.1733**	**0.9264**	**0.9889**	**0.8856**
Hyper	w/o	0.3213	0.4888	0.2322	0.9432	1.0000	0.8876	0.5850	0.7172	0.5112	0.9841	0.9929	0.9785
	w/	**0.3326**	**0.5049**	**0.2456**	**0.9503**	**1.0000**	**0.9014**	**0.6518**	**0.7933**	**0.5512**	**1.0000**	**1.0000**	**0.9969**

TABLE 5: The impact of schema on key/value prediction performance.

Schema	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
w/o	0.5352	0.6447	0.4800	0.9202	0.9720	0.8817	0.5346	0.6582	0.4613	0.9625	0.9731	0.9658
w/	**0.5814**	**0.7156**	**0.5090**	**0.9730**	**0.9970**	**0.9581**	**0.5578**	**0.6941**	**0.4818**	**0.9993**	**1.0000**	**0.9991**

HINGE on average (6.8% and 0.4% improvements on value prediction and key prediction, respectively).

5.4 The Impact of Schema on Link Prediction Performance

In this experiment, we look into the impact of schema through the breakdown of link prediction performance on different categories of facts (triple/hyper-relational facts, facts with/without schema information). Note that a fact without schema information refers to the fact that contains at least one entity without type, where we assign an "unknown" type to the entity.

Table 4 shows the results. We observe that facts with schema information consistently achieve better performance than facts without schema information, which indicates that the schema information indeed helps the link prediction in KGs. In addition, we also find that while the impact of schema information on triple facts is higher than that on hyper-relational facts on WikiPeople, we have a completely opposite observation on JF17K, i.e., the impact of schema information on triple facts is obviously lower than that on hyper-relational facts. This can be explained by the dataset statistics. Where 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).

5.5 The Impact of Schema on Key/Value Prediction Performance

In this experiment, we study the influence of schema by comparing the performance of sHINGE on the key-value pairs without and with schema information.

Table 5 shows the results. We observe that key-value pairs with schema information consistently outperform that without schema information, showing a 16.8% improvement on the value prediction task, and an 8.4% improvement on the key prediction task on WikiPeople (7.3% and 6.1% on JF17K, respectively). This further demonstrates the effectiveness of schema information on key-value prediction.

5.6 Parameter Sensitivity Study

We study the impact of three key parameters in sHINGE, i.e., the number of filters n_f used in the CNNs, the embedding dimension K and the number of types learnt per entity s.

First, by fixing the embedding dimension $K = 100$ and the number of types $s = 1$, we vary the number of filters n_f from 10 to 800, and plot its impact on the link prediction performance for both datasets in Figure 3. We observe that when increasing n_f, the performance rises dramatically in the beginning, and then flattens out when $n_f \geq 400$ in most cases.

Second, by fixing the number of filters $n_f = 400$ and the number of types $s = 1$, 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 increases rapidly in the beginning, and then remains stable when $K \geq 100$ in most cases.

Finally, by fixing the embedding dimension $K = 100$ and the number of filters $n_f = 400$, we vary the number of types learnt per entity s from 1 to 3, and present its impact on the link prediction performance for both datasets in Figure 5. We discover that learning from the top one type for each entity achieves the best performance in general on both datasets. Therefore, we set the number of filters $n_f = 400$, the embedding dimension $K = 100$, and the number of types learnt per entity $s = 1$, in all previous experiments.

6 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, resulting in suboptimal models due to their ignorance of the triplet structure. Moreover, most existing approaches do not consider the hyper-relational schema information of KG, which as we show in this paper is also critical to resolving link prediction tasks. Against this background, we proposed sHINGE, a schema-aware Hyper-relatIonal kNowledge Graph Embedding model. It captures not only the primary structural information of the KG encoded in the triplets and their associated key-value pairs, but also the schema information encoded by entity-typed triplets and their associated entity-typed key-value pairs. Our extensive evaluation shows the superiority of sHINGE on various link prediction tasks over KGs using two real-world KG datasets. In particular, compared to a sizeable collection of 21 baselines, sHINGE consistently outperforms the best-performing triple-based KG embedding method, hyper-relational KG embedding method, and schema-aware KG embedding method by 19.1%, 1.8%, and 12.9%, respectively.

In the future, we plan to further investigate the hyper-relational KG embedding problem by considering high-order relations over KGs using the combination of graph neural networks and self-attention layers.

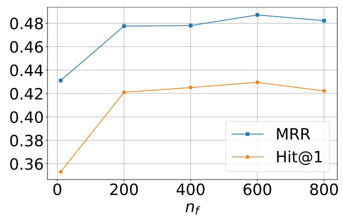
Acknowledgments

This project has received funding from the University of Macau (SRG2021-00002-IOTSC) and The Science and Technology Development Fund, Macau SAR (0047/2022/A1, SKL-IOTSC(UM)-2021-2023), European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement 683253/GraphInt). This work was performed in part at SICC which is supported by SKL-IOTSC, University of Macau.

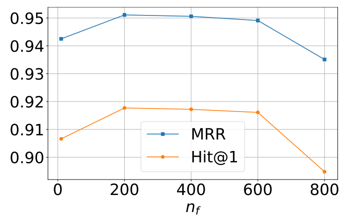
References

[1] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *ACM SIGMOD/PODS*. ACM, 2008, pp. 1247–1250.

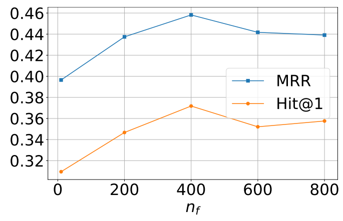
[2] Google, https://www.google.com/intl/bn/insidesearch/features/search/knowledge.html, 2014.



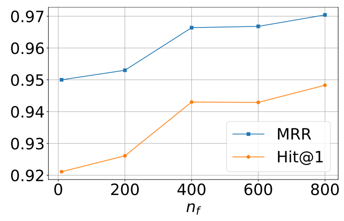
(a) Head/Tail Prediction on WikiPeople.



(b) Relation Prediction on WikiPeople.

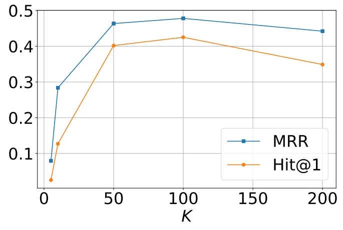


(c) Head/Tail Prediction on JF17K.

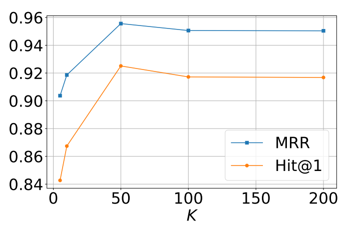


(d) Relation Prediction on JF17K.

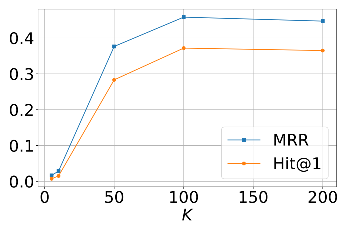
Fig. 3: Impact of the number of filters n_f.



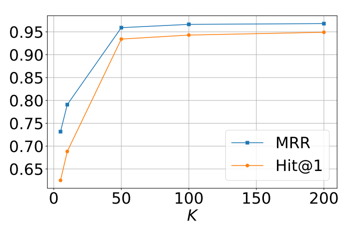
(a) Head/Tail Prediction on WikiPeople.



(b) Relation Prediction on WikiPeople.

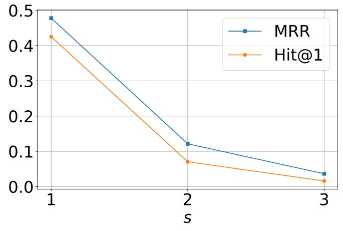


(c) Head/Tail Prediction on JF17K.

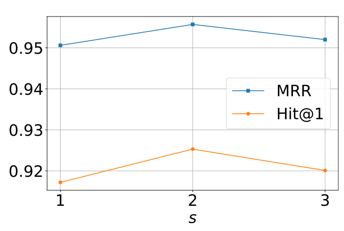


(d) Relation Prediction on JF17K.

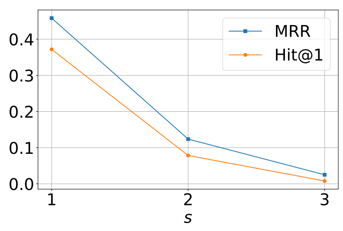
Fig. 4: Impact of the embedding dimension K.



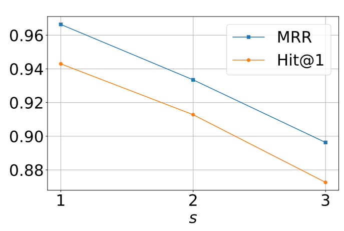
(a) Head/Tail Prediction on WikiPeople.



(b) Relation Prediction on WikiPeople.



(c) Head/Tail Prediction on JF17K.



(d) Relation Prediction on JF17K.

Fig. 5: Impact of the number of types learnt per entity s.

[3] Wikidata, http://wikidata.org/, 2012.

[4] C. Xiong, R. Power, and J. Callan, "Explicit semantic ranking for academic search via knowledge graph embedding," in *WWW*, 2017, pp. 1271–1279.

[5] S. W.-t. Yih, M.-W. Chang, X. He, and J. Gao, "Semantic parsing via staged query graph generation: Question answering with knowledge base," in *ACL and IJCNLP*, 2015, pp. 1321–1331.

[6] J. Graupmann, R. Schenkel, and G. Weikum, "The spheresearch engine for unified ranked retrieval of heterogeneous xml and web documents," in *VLDB*. VLDB Endowment, 2005, pp. 529–540.

[7] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *KDD*. ACM, 2016, pp. 353–362.

[8] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, "A review of relational machine learning for knowledge graphs," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 11–33, 2015.

[9] R. West, E. Gabrilovich, K. Murphy, S. Sun, R. Gupta, and D. Lin, "Knowledge base completion via search-based question answering," in *WWW*. ACM, 2014, pp. 515–526.

[10] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *TKDE*, vol. 29, no. 12, pp. 2724–2743, 2017.

[11] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *NIPS*, 2013, pp. 2787–2795.

[12] J. Wen, J. Li, Y. Mao, S. Chen, and R. Zhang, "On the representation and embedding of knowledge bases beyond binary relations," in *IJCAI*. AAAI Press, 2016, pp. 1300–1307.

[13] S. Guan, X. Jin, Y. Wang, and X. Cheng, "Link prediction on n-ary relational data," in *WWW*. ACM, 2019, pp. 583–593.

[14] P. Rosso, D. Yang, and P. Cudré-Mauroux, "Beyond triplets: hyper-relational knowledge graph embedding for link prediction," in *WWW*, 2020, pp. 1885–1896.

[15] D. Brickley, R. V. Guha, and B. McBride, "Rdf schema 1.1," *W3C recommendation*, vol. 25, pp. 2004–2014, 2014.

[16] R. Zhang, J. Li, J. Mei, and Y. Mao, "Scalable instance reconstruction in knowledge bases via relatedness affiliated embedding," in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 1185–1194.

[17] Y. Liu, Q. Yao, and Y. Li, "Role-aware modeling for n-ary relational knowledge bases," in *WWW*, 2021, pp. 2660–2671.

[18] S. Guan, X. Jin, J. Guo, Y. Wang, and X. Cheng, "Neuinfer: Knowledge inference on n-ary facts," in *ACL*, 2020, pp. 6141–6151.

[19] P. Rosso, D. Yang, N. Ostapuk, and P. Cudré-Mauroux, "Reta: A schema-aware, end-to-end solution for instance completion in knowledge graphs," in *WWW*, 2021, pp. 845–856.

[20] S. Guan, X. Jin, J. Guo, Y. none Wang, and X. Cheng, "Link prediction on n-ary relational data based on relatedness evaluation," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[21] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616–1637, 2018.

[22] R. Hussein, D. Yang, and P. Cudré-Mauroux, "Are meta-paths necessary? revisiting heterogeneous graph embeddings," in *CIKM*, 2018, pp. 437–446.

[23] D. Yang, P. Rosso, B. Li, and P. Cudre-Mauroux, "Nodesketch: Highly-efficient graph embeddings via recursive sketching," in *KDD*, 2019, pp. 1162–1172.

[24] P. Rosso, D. Yang, and P. Cudré-Mauroux, "Knowledge graph embeddings," in *Encyclopedia of Big Data Technologies.*, 2019.

[25] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes." in *AAAI*, vol. 14, 2014, pp. 1112–1119.

[26] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion." in *AAAI*, vol. 15, 2015, pp. 2181–2187.

[27] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in *ACL and IJCNLP*, vol. 1, 2015, pp. 687–696.

[28] T. Ebisu and R. Ichise, "Toruse: Knowledge graph embedding on a lie group," in *AAAI*, 2018.

[29] M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data." in *ICML*, vol. 11, 2011, pp. 809–816.

[30] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in *ICLR*, 2015.

[31] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *ICML*, 2016, pp. 2071–2080.

[32] R. Socher, D. Chen, C. D. Manning, and A. Ng, "Reasoning with neural tensor networks for knowledge base completion," in *NIPS*, 2013, pp. 926–934.

[33] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in *AAAI*, 2017, pp. 1811–1818.

[34] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *ESWC*. Springer, 2018, pp. 593–607.

[35] I. Balazevic, C. Allen, and T. M. Hospedales, "Hypernetwork knowledge graph embeddings," *arXiv preprint arXiv:1808.07018*, 2018.

[36] D. Q. Nguyen, T. Vu, T. D. Nguyen, D. Q. Nguyen, and D. Phung, "A capsule network-based embedding model for knowledge graph completion and search personalization," *arXiv preprint arXiv:1808.04122*, 2018.

[37] A. Garcia-Duran and M. Niepert, "Kblrn: End-to-end learning of knowledge base representations with latent, relational, and numerical features," *arXiv preprint arXiv:1709.04676*, 2017.

[38] Y. Tay, L. A. Tuan, M. C. Phan, and S. C. Hui, "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, 2017, pp. 1029–1038.

[39] A. Kristiadi, M. A. Khan, D. Lukovnikov, J. Lehmann, and A. Fischer, "Incorporating literals into knowledge graph embeddings," *arXiv preprint arXiv:1802.00934*, 2018.

[40] Y. Liu, H. Li, A. Garcia-Duran, M. Niepert, D. Onoro-Rubio, and D. S. Rosenblum, "Mmkg: Multi-modal knowledge graphs," in *European Semantic Web Conference*. Springer, 2019, pp. 459–474.

[41] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph and text jointly embedding," in *EMNLP*, 2014, pp. 1591–1601.

[42] H. Zhong, J. Zhang, Z. Wang, H. Wan, and Z. Chen, "Aligning knowledge and text embeddings by entity descriptions," in *EMNLP*, 2015, pp. 267–272.

[43] W. Fang, J. Zhang, D. Wang, Z. Chen, and M. Li, "Entity disambiguation by knowledge and text jointly embedding," in *CoNLL*, 2016, pp. 260–269.

[44] I. Yamada, H. Shindo, H. Takeda, and Y. Takefuji, "Joint learning of the embedding of words and entities for named entity disambiguation," in *CoNLL*, 2016, pp. 250–259.

[45] X. Han, Z. Liu, and M. Sun, "Neural knowledge acquisition via mutual attention between knowledge graph and text," in *AAAI*, 2018.

[46] K. Toutanova, D. Chen, P. Pantel, H. Poon, P. Choudhury, and M. Gamon, "Representing text for joint embedding of text and knowledge bases," in *EMNLP*, 2015, pp. 1499–1509.

[47] I. Yamada, H. Shindo, H. Takeda, and Y. Takefuji, "Learning distributed representations of texts and entities from knowledge base," *TACL*, vol. 5, pp. 397–411, 2017.

[48] M. Yu and M. Dredze, "Improving lexical embeddings with semantic knowledge," in *ACL*, vol. 2, 2014, pp. 545–550.

[49] J. Cheng, Z. Wang, J.-R. Wen, J. Yan, and Z. Chen, "Contextual text understanding in distributional semantic space," in *CIKM*. ACM, 2015, pp. 133–142.

[50] P. Rosso, D. Yang, and P. Cudre-Mauroux, "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)*, 2019.

[51] B. Fatemi, P. Taslakian, D. Vazquez, and D. Poole, "Knowledge hypergraphs: prediction beyond binary relations," in *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 2020, pp. 2191–2197.

[52] Z. Chen, X. Wang, C. Wang, and J. Li, "Explainable link prediction in knowledge hypergraphs," in *Proceedings of the 31st ACM international conference on information & knowledge management*, 2022, pp. 262–271.

[53] S. Yan, Z. Zhang, X. Sun, G. Xu, S. Li, Q. Liu, N. Liu, and S. Wang, "Polygone: Modeling n-ary relational data as gyro-polygons in hyperbolic space," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 4, 2022, pp. 4308–4317.

[54] Y. Liu, Q. Yao, and Y. Li, "Generalizing tensor decomposition for n-ary relational knowledge bases," in *Proceedings of the web conference 2020*, 2020, pp. 1104–1114.

[55] S. Di, Q. Yao, and L. Chen, "Searching to sparsify tensor decomposition for n-ary relational data," in *Proceedings of the Web Conference 2021*, 2021, pp. 4043–4054.

[56] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[57] M. Galkin, P. Trivedi, G. Maheshwari, R. Usbeck, and J. Lehmann, "Message passing for hyper-relational knowledge graphs," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 7346–7359.

[58] Q. Wang, H. Wang, Y. Lyu, and Y. Zhu, "Link prediction on n-ary relational facts: A graph-based approach," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 2021, pp. 396–407.

[59] D. Yu and Y. Yang, "Improving hyper-relational knowledge graph completion," *arXiv preprint arXiv:2104.08167*, 2021.

[60] H. Shomer, W. Jin, J. Li, Y. Ma, and J. Tang, "Learning representations for hyper-relational knowledge graphs," *arXiv preprint arXiv:2208.14322*, 2022.

[61] K. Wiharja, J. Z. Pan, M. J. Kollingbaum, and Y. Deng, "Schema aware iterative knowledge graph completion," *Journal of Web Semantics*, vol. 65, p. 100616, 2020.

[62] Y. Xue, J. Jin, A. Song, Y. Zhang, Y. Liu, and K. Wang, "Relation-based multi-type aware knowledge graph embedding," *Neurocomputing*, vol. 456, pp. 11–22, 2021.

[63] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[64] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.

[65] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[66] Z. Zhang, J. Cai, Y. Zhang, and J. Wang, "Learning hierarchy-aware knowledge graph embeddings for link prediction," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 03, 2020, pp. 3065–3072.

[67] Z. Cui, P. Kapanipathi, K. Talamadupula, T. Gao, and Q. Ji, "Type-augmented relation prediction in knowledge graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 8, 2021, pp. 7151–7159.

[68] H. Luo, E. Haihong, L. Tan, G. Zhou, T. Yao, and K. Wan, "Dhge: Dual-view hyper-relational knowledge graph embedding for link prediction and entity typing," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 5, 2023, pp. 6467–6474.

[69] J. Zhuo, Q. Zhu, Y. Yue, Y. Zhao, and W. Han, "A neighborhood-attention fine-grained entity typing for knowledge graph completion," in *Proceedings of the fifteenth ACM international conference on web search and data mining*, 2022, pp. 1525–1533.

[70] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013, pp. 3111–3119.

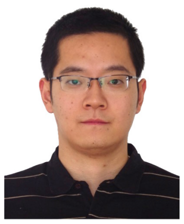
[71] G. Niu, B. Li, Y. Zhang, and S. Pu, "Cake: A scalable commonsense-aware framework for multi-view knowledge graph completion," in *Proceedings of the 60th Annual Meeting of the Associ-*

ation for Computational Linguistics (Volume 1: Long Papers), 2022, pp. 2867–2877.

[72] H. Liu, Y. Wu, and Y. Yang, "Analogical inference for multi-relational embeddings," in *ICML*, 2017, pp. 2168–2178.

[73] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

Yuhuan Lu received the B.Eng. and M.S. degrees in transportation engineering from Sun Yat-Sen University, Guangzhou, China, in 2017 and 2020, respectively. He is currently a Ph.D. student with the State Key Laboratory of Internet of Things for Smart City and Department of Computer and Information Science, University of Macau, Macao, and also a research assistant with the School of Intelligent Systems Engineering, Sun Yat-Sen University, Guangzhou, China. His research interests lie in Graph Embedding, Urban Computing and Intelligent Transportation Systems.



Dingqi Yang is an Associate Professor with the State Key Laboratory of Internet of Things for Smart City and Department of Computer and Information Science, University of Macau. He received his Ph.D. degree in Computer Science from Pierre and Marie Curie University and Institut Mines-TELECOM/TELECOM SudParis in France, where he won both the CNRS SAMOVAR Doctorate Award and the Press Mention in 2015. Before joining the University of Macau, he worked as a senior researcher at the University of Fribourg in Switzerland. His research interests include big data analytics, ubiquitous computing, and smart city.

Pengyang Wang is an Assistant Professor in the State Key Lab of Smart Cities and Internet-of-Things at the University of Macau. He obtained his Ph.D. in Computer Science from the University of Central Florida. His research interests are in data mining, machine learning and big data analytics. Pengyang has received "Global Top 100 Chinese Rising Stars in Artificial Intelligence", one Best Student Paper Runner-up award of SIGKDD 2018, and one Best Paper Runner-up award of SIGSPATIAL 2020. His research work has been featured by Synced and UCF Today, and also highlighted by the Natural Science Foundation (NSF) of the U.S.

Paolo Rosso received his Ph.D. degree in Computer Science from the University of Fribourg in Switzerland where he worked on Knowledge Graph embeddings. Before starting his Ph.D., he did the Erasmus traineeship at High Performance Computing Center in Stuttgart (Germany) where he developed a platform to benchmark different architectures for High Performance Computing. His research interests are Semantic Web and Machine Learning.

Philippe Cudre-Mauroux is a Full Professor and the director of the eXascale Infolab at the University of Fribourg in Switzerland. He received his Ph.D. from the Swiss Federal Institute of Technology EPFL, where he won both the Doctorate Award and the EPFL Press Mention. Before joining the University of Fribourg he worked on information management infrastructures for IBM Watson Research, Microsoft Research Asia, and MIT. His research interests are in next-generation, Big Data management infrastructures for non-relational data. Webpage: http://exascale.info/phil