# HELIOS: Hyper-Relational Schema Modeling from Knowledge Graphs

# Yuhuan Lu

State Key Laboratory of Internet of Things for Smart City and Department of Computer and Information Science University of Macau, Macao, China lu.yuhuan@connect.umac.mo

# Weijian Yu

State Key Laboratory of Internet of Things for Smart City and Department of Computer and Information Science University of Macau, Macao, China mc14948@connect.um.edu.mo

# ABSTRACT

Knowledge graph (KG) schema, which prescribes a high-level structure and semantics of a KG, is significantly helpful for KG completion and reasoning problems. Despite its usefulness, open-domain KGs do not practically have a unified and fixed schema. Existing approaches usually extract schema information using entity types from a KG where each entity e can be associated with a set of types  $\{T_e\}$ , by either heuristically taking one type for each entity or exhaustively combining the types of all entities in a fact (to get entity-typed tuples, (*h\_type*, *r*, *t\_type*) for example). However, these two approaches either overlook the role of multiple types of a single entity across different facts or introduce nonnegligible noise as not all the type combinations actually support the fact, thus failing to capture the sophisticated schema information. Against this background, we study the problem of modeling hyper-relational schema, which is formulated as mixed hyperrelational tuples  $({T_h}, r, {T_t}, k_1, {T_{v_1}}, ...)$  with two-fold hyperrelations: each type set  $\{T\}$  may contain multiple types and each schema tuple may contain multiple key-type set pairs  $(k, \{T_v\})$ . To address this problem, we propose HELIOS, a hyper-relational schema model designed to subtly learn from such hyper-relational schema tuples by capturing not only the correlation between multiple types of a single entity, but also the correlation between types of different entities and relations in a schema tuple. We evaluate HELIOS on three real-world KG datasets in different schema prediction tasks. Results show that HELIOS consistently outperforms state-of-the-art hyper-relational link prediction techniques by 20.0-29.7%, and is also much more robust than baselines in predicting

MM '23, October 29-November 3, 2023, Ottawa, ON, Canada

# Bangchao Deng

State Key Laboratory of Internet of Things for Smart City and Department of Computer and Information Science University of Macau, Macao, China mc14969@um.edu.mo

# Dingqi Yang\*

State Key Laboratory of Internet of Things for Smart City and Department of Computer and Information Science University of Macau, Macao, China dingqiyang@um.edu.mo

types and relations across different positions in a hyper-relational schema tuple.

# CCS CONCEPTS

- Computing methodologies  $\rightarrow$  Knowledge representation and reasoning.

# **KEYWORDS**

Hyper-relation; Schema; Knowledge graph; Entity Type

#### **ACM Reference Format:**

Yuhuan Lu, Bangchao Deng, Weijian Yu, and Dingqi Yang. 2023. HELIOS: Hyper-Relational Schema Modeling from Knowledge Graphs. In *Proceedings* of the 31st ACM International Conference on Multimedia (MM '23), October 29-November 3, 2023, Ottawa, ON, Canada. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3581783.3612184

# **1** INTRODUCTION

Knowledge Graphs (KGs) [27], such as Freebase [4], Wikidata [63] or Google's Knowledge Graph [20], have become a promising data management paradigm powering a wide range of Web applications, such as semantic search [66], question-answering [68], or recommender systems [71]. Traditionally, KGs are represented as a set of triplets; each triplet (head, relation, tail), or (h,r,t) for short, represents a fact that encodes a relation connecting a head entity to a tail entity, such as (Apple, headquarter location, Cupertino). To better describe real-world facts, modern KGs often contain hyper-relational facts [13, 21, 22, 36, 47, 61, 72], where a base triplet (*h*, *r*, *t*) is further associated with an arbitrary number of key-value<sup>1</sup> pairs (k, v)describing additional information about the triplet, represented as  $(h, r, t, k_1, v_1, ...)$ . Figure 1 shows a real-world example on Wikidata (Apple, industry, software industry, in scope of, computer program, in scope of, operating system) involving four entities. To effectively make use of KGs, link prediction tasks [40, 57] have been widely adopted to solve KG completion and reasoning problems, such as (h, r, ?) or  $(h, ?, t, k_1, v_1, ...)$ , where the question mark represents the missing element (entity or relation) to be predicted. Existing

<sup>\*</sup>Corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

<sup>© 2023</sup> Copyright held by the owner/author(s). Publication rights licensed to ACM. ACM ISBN 979-8-4007-0108-5/23/10...\$15.00 https://doi.org/10.1145/3581783.3612184

<sup>&</sup>lt;sup>1</sup>Although a key-value pair indeed refers a relation-entity pair in this paper, the term key-value is used to differ from the elements in the base triplet, as it has been shown that the base triplet contains primary information about the KG while the key-value pairs contain secondary information describing the triplet [22, 47].

Yuhuan Lu, Bangchao Deng, Weijian Yu, and Dingqi Yang



Figure 1: A real-world example of (hyper-relational) facts and their associated entity types from Wikidata. Note that entity types listed in the blue box next to each entity are not exhaustive due to space limitation.

approaches to this problem usually learn to capture the structural information of the KG encoded in the facts, and generate a ranking list of all entities or relations for the missing element, where the top ones are the most plausible answers.

To effectively solve such link prediction tasks over a KG, the schema information of the KG, which prescribes a high-level structure and semantics that the KG follows or should follow [26], has been shown to be significantly useful [21, 36, 38, 48, 62]. For example, when predicting (Apple, headquarter location, ?), the corresponding schema represented as an entity-typed tuple (enterprise, *headquarter location*, ?) could suggest that the missing tail entity is likely to be of type city (according to pre-extracted schema rules such as RETA-Filter [47], for example); this can then serve as a strong clue to further favor the entities of this type in the prediction. In the current literature, such schema information can be used as a pre-processing filter to reduce the solution space of link prediction problems [48], as an additional input together with a fact for scoring the plausibility of the fact [21, 24, 28, 36, 43, 59, 65], or as a post-processing step to check the schema correctness of the predicted facts [62]. Therefore, a high-quality schema of the KG is essential to boost the performance of link prediction tasks, and these existing work [21, 36, 48, 62] assume knowing the golden schema of a KG. However, in practice, open-domain KGs such as Wikidata do not have a unified and fixed schema [1, 48, 73]. Even though some effort such as Schema.org has been made to create unified and shared schemas for structured data on the Web, such schemas still have a low coverage on the Web, and thus have not been widely adopted by modern KGs.

In the background, existing methods extract schema information using entity types from KGs [21, 24, 36, 48]. Specifically, each entity in a KG is often associated with one or multiple types as shown in the blue boxes in Figure 1. Existing approaches either heuristically take one type for each entity [21, 36] (according to the type popularity in the KG, for example), or exhaustively combine the types of all entities in a fact to extract schema rules [24, 48]. However, both approaches fail to consider the sophisticated schema information in KGs. First, assuming one type per entity oversimplifies the schema of the KGs and overlooks the role of multiple types of a single entity in different facts [59]. For example, in the facts (*Apple*,

headquarter location, Cupertino) and (Apple, logo image, Apple logo) as shown in Figure 1, the appropriate types for the entity Apple should be different, i.e., enterprise and brand, respectively. Second, the exhaustive combination of the types of all entities in a fact faces two challenges. On one hand, it introduces non-negligible noise to the schema, as only some of the type combinations actually support the fact while others may make less sense. For example, for the fact (Apple, board member, Arthur D. Levinson) in Figure 1, exhaustive combination (of all head and tail types together with the relation [48]) will get a tuple (brand, board member, human), which is obviously less reasonable than (corporation, board member, human). On the other hand, the exhaustive combination also incurs significant computational overhead. If one entity has m types on average, we will extract  $m^n$  entity-typed tuples for a hyper-relational fact, which grows exponentially with the number of entities *n* (a.k.a. arity) in the hyper-relational fact.

Against this background, we study the problem of modeling hyper-relational schema from KGs. Instead of heuristically assuming one type per entity or exhaustively combining entity types when extracting schema information, we define hyper-relational schema<sup>2</sup> as mixed hyper-relational tuples  $({T_h}, r, {T_t}, k_1, {T_{v_1}}, ...)$ , where  $\{T_e\}$  represents the set of types for the entity e. The meaning of hyper-relations here are two folds: 1) each type set could contain an arbitrary number of entity types; and 2) each schema tuple involves a base triplet  $({T_h}, r, {T_t})$  associated with an arbitrary number of key-(type)set pairs  $(k, \{T_n\})$ . Moreover, different from existing works extracting schema information as hard rules for downstream tasks [1, 48, 62, 70], we formulate a schema prediction problem, predicting a missing element in a hyper-relational schema tuple, such as  $(?, r, \{T_t\})$  or  $(\{T_h\}, ?, \{T_t\}, k_1, \{T_{v_1}\})$ , where the missing elements are a set of types or a relation, respectively. To solve this problem, we propose HELIOS, a Hyper-rELatIOnal Schema model designed to learn from such hyper-relational schema tuples. Specifically, for a hyper-relational schema tuple  $(\{T_h\}, r, \{T_t\}, k_1, \{T_{v_1}\}, ...),$ it first captures the correlation between multiple types of a single entity accounting for the specific context of the schema tuple; we use Graph Attention Networks (GATs) to encode a type set into a contextualized type feature vector, which can dynamically capture the role of multiple types of the same entity across different facts (as evidenced by our experiments and case studies below). Afterward, it captures the correlation between types of different entities and relations in the schema tuple, by feeding the contextualized type features together with relations to a self-attention network with learnable edge biases discriminating connections between different elements in the schema tuple. HELIOS is trained using a masked training process, being able to predict any missing elements in a hyper-relational schema tuple. Our contributions can be summarized as follows:

• We revisit the drawbacks of existing approaches on extracting and using KG schema for link prediction, and propose to study a novel problem of modeling hyper-relational schema which are formulated as mixed hyper-relational schema tuples  $({T_h}, r, {T_t}, k_1, {T_{v_1}}, ...).$ 

<sup>&</sup>lt;sup>2</sup>Note that as a triple fact/schema tuple is indeed a special case of a hyper-relational fact/schema tuple without key-value pairs, we will use the hyper-relational fact/schema tuple referring to both of them unless specified otherwise.

- We propose HELIOS to subtly model the two-fold hyper-relations for solving schema prediction problems, capturing not only the correlation between multiple types of a single entity, but also the correlation between types of different entities and relations in a hyper-relational schema tuple.
- We conduct a thorough evaluation of HELIOS compared to a sizable collection of baselines on three real-world KG datasets. Results show that HELIOS consistently outperforms state-of-the-art hyper-relational link prediction techniques in both type and relation prediction tasks, yielding 29.7% and 20.0% improvement over the best-performing baselines, respectively. Moreover, HELIOS achieves much more robust performance than baselines in predicting types and relations at any positions in a hyper-relational schema tuple, showing 76.6% and 72.7% smaller coefficients of variation than the best-performing baselines, respectively.

# 2 RELATED WORK

# 2.1 (Hyper-relational) Link Prediction over KGs

Link prediction tasks [40, 57] have been widely used to solve KG completion and reasoning problems. Under the widely adopted triple representation of facts (h, r, t), early work resorted to ruleand feature-based relational learning [17, 35] or using hand-crafted features [31, 42]. Recently, KG embedding techniques [46] have been used to learn representations of entities/relations in a KG, which can then be effectively used for resolving link prediction tasks. These techniques fall into two broad categories [57]. First, translational distance models exploit distance-based scoring functions to create the embeddings, such as TransE [6] learning from triplets (h, r, t)to preserve  $h + r \approx t$ . Several works further improved TransE to capture richer KG structures, such as involving relation-specific hyperplanes [60] or spaces [14, 29, 34]. Second, semantic matching models exploit similarity-based scoring functions, such as RESCAL [41] which represents each entity as a vector and each relation as a matrix, and then uses a bilinear function to model the relation between two entities. Several works later extended RESCAL by reducing the complexity of the models [52, 67], by improving the model expressiveness [3], or by modeling non-linear relations using neural networks [2, 12, 39, 50, 54].

Some recent works have shown that the triple representation of a KG oversimplifies the complex nature of the data stored in the KG [23, 47], in particular for hyper-relational data where each fact contains multiple relations and entities, as shown in Figure 1. Some existing work adopted an n-ary representation for hyper-relational facts, i.e., a set of key-value (relation-entity) pairs [23, 36, 61, 72]. As a typical example in [21, 23], 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. There is also a variation of this n-ary representation [36, 61, 72], where a hyper-relational fact (h, r, t) with (k, v) is associated with a meta-relation, represented as an ordered list of keys (relations), such as  $R := (r_h, r_t, k)$ ; the fact is then represented as a list of ordered values associated with the meta-relation  $\{R, (h, t, v)\}$ . Using such n-ary representations, these techniques learn either the relatedness between entity-relation pairs [21, 23], or relatedness among all entities in a fact [36, 61, 72] for link prediction. However, recent studies [22, 47] revealed that the base triplets (h, r, t) serve

as the fundamental data structure in the KGs and preserve the essential information for link prediction (which should not be treated identically as key-value pairs (k, v)), and suggested learning directly from hyper-relational facts. Following this fashion, HINGE [47] and NeuInfer [22] use two different feature extraction pipelines for the base triplets and key-value pairs, respectively, and then merge the two feature vectors for link prediction; StarE [18], Hy-Transformer [69], GRAN [58], and QUAD [53] design Graph Neural Networks (GNNs) to encode the base triplets together with key-value pairs using transformer [55] for link prediction.

Our work differs from these hyper-relational link prediction techniques by focusing on hyper-relational schema tuples (with the two-fold hyper-relations) instead of hyper-relational facts. We show in our experiments that existing hyper-relational link prediction techniques fail to capture the sophisticated correlation of entity types in a hyper-relational schema tuple, resulting in suboptimal performance.

# 2.2 Schema of KGs

The schema of a KG prescribes a high-level structure and semantics that the KG follows or should follow [26]. For domain-specific KGs, their schema is usually manually defined and usually has a small scale. For example, the medical KG in [32] defines 9 entity types and 9 relations. Note that in traditional KGs, a relation is manually defined to connect two specific entity types only. However, open-domain KGs (e.g., Wikidata [63] and Freebase [4]) usually do not have a unified and fixed schema [1, 48, 73]; for some dynamic KGs, relevant and irrelevant entity types can even change over time [73]. Subsequently, KG schema (or ontology) extraction has been investigated (a.k.a., ontology learning, taxonomy extraction, or ontology population), for purposes including refining and completing an existing ontology [33, 64], adapting a known ontology to a new KG [16], or building an ontology from scratch [44, 45], from either a given KG [24] or text corpora [8]. Traditional ontology population approaches mostly require extensive manual efforts, extracting ontology rules from large annotated text corpora [19], which is significantly different from our problem settings. Recent solutions resort to graph embedding techniques for link prediction on ontology-level graphs [7] or on fact- and ontology-level graphs jointly [24]. These studies assume to have a golden ontology-level graph (i.e., schema) such as [10] and focus on the triple representation. In this paper, we study the schema modeling problem on fact-level KGs under the hyper-relational representation.

In the background of link prediction over KGs, schema information has been shown to be significantly helpful [21, 24, 36, 48, 62]. Existing approaches use such schema information in different stages of link prediction tasks: 1) as a pre-processing filter to reduce the solution space before link prediction [48]; 2) as an additional input together with a fact fed to link prediction techniques [21, 24, 28, 36, 43, 59, 65]; 3) as a post-processing step to verify the schema correctness of the predicted facts [62]. Existing work extract schema information resorting to entity types from KGs, by either heuristically assuming only one type [21, 36] for each entity or exhaustively combining the types of all entities in a fact [24, 48]. However, we argue that these approaches fail to consider the sophisticated schema information in KGs, and propose to study the hyper-relational schema



Figure 2: Overview of our proposed HELIOS with three modules: 1) Learning from type sets, 2) Learning from schema tuples, and 3) Prediction with mask.

modeling problem, making predictions over mixed hyper-relational schema tuples ( $\{T_h\}, r, \{T_t\}, k_1, \{T_{v_1}\}, ...$ ).

# **3 MODELING HYPER-RELATIONAL SCHEMA**

In this section, we introduce HELIOS, a hyper-relational schema embedding model learning directly from hyper-relational schema tuples. A couple of formal definitions are presented:

Definition 3.1. Entity type set. An entity type set is the collection of types  $\{T_e\}$  of an entity e in a given KG.

Definition 3.2. Hyper-relational schema tuple: A hyper-relational schema tuple contains a base triplet  $({T_h}, r, {T_t})$  associated with an arbitrary number of key-(type)set pairs  $(k_i, {T_{v_i}})$ .

Based on the above definitions, we introduce our HELIOS model as shown in Figure 2. Specifically, HELIOS consists of three parts. For an input hyper-relational schema tuple, it 1) learns from the type set to capture the correlation between multiple types for each entity, generating a contextualized type feature vector, 2) learns from the hyper-relational schema tuple to capture the correlation between contextualized type feature vectors and relations, and 3) makes predictions with mask tokens for resolving our schema prediction tasks. We present the three modules in detail below.

#### 3.1 Learning from Type Sets

In a hyper-relational schema tuple, each entity type set could contain an arbitrary number of types as shown in Figure 1. To accommodate the variation of the size of the type set and capture the dynamic correlation between multiple types of the type set across different hyper-relational schema tuples, we propose a two-level type set embedding paradigm inspired by the contextualized word embedding techniques [5, 25, 49]. Specifically, for a type set, we use a Graph Attention Network (GAT) [56] in the first level to model the intrinsic interaction between its contained types and generate a static type feature vector of the type set. In the second level, another GAT is applied to dynamically adjust the importance of each type in the type set accounting for their role in the specific context of a hyper-relational schema tuple, by incorporating the static type feature vectors of other type sets in the hyper-relational schema tuple into modeling, and thus generate the final contextualized type feature vector of the type set.

In the first level, for each type set  $\{T_e\}$ , we construct a fullyconnected graph  $\mathcal{G}_e$  with self-loops, where each type  $e\_type_p \in$  $\{T_e\}$  is a node p in  $\mathcal{G}_e$ ; the fully-connected graph gives the maximum flexibility to GAT to learn to capture the correlation between multiple types of the same entity. Without loss of generality, we depict a single graph attention layer in the following. To encode the correlation between types, an attention mechanism a is applied to every pair of nodes  $e\_type_p, e\_type_q \in \{T_e\}$  in the graph  $\mathcal{G}_e$  to obtain the attention weights indicating the importance of node qto node p:

$$e_{pq} = a \left( \mathbf{W} \overrightarrow{e_t type}_p, \mathbf{W} \overrightarrow{e_t type}_q \right)$$
(1)

where  $e_{type_p}, e_{type_q} \in \mathbb{R}^F$  refer to the input embeddings of node p and q, respectively. F denotes the embedding size and  $\mathbf{W} \in \mathbb{R}^{F \times F}$  denotes the shared linear transformation to enhance the representability of node features. The attention mechanism  $a : \mathbb{R}^F \times \mathbb{R}^F \to \mathbb{R}$  is inherited from [56], employing a single layer feedforward neural network with the LeakyReLU [37] as the non-linear activation function. To normalize the attention weights across all choices of node q, a softmax function is utilized:

$$\alpha_{pq} = \operatorname{softmax}_{q} (e_{pq}) = \frac{\exp(e_{pq})}{\sum_{m \in N_{p}} \exp(e_{pm})}$$
(2)

where  $N_p$  denotes the set of neighbors of node p. Since  $G_e$  is a fully-connected graph with self-loops,  $N_p$  corresponds to all node sets in the graph, including node p itself. Afterward, the normalized attention weights are used to compute the linear combination of input embeddings to update the embedding of node p:

$$\vec{e_{type}}'_{p} = \sigma\left(\sum_{q \in \mathcal{N}_{p}} \alpha_{pq} \mathbf{W} \vec{e_{type}}_{q}\right)$$
(3)

where  $\sigma(\cdot)$  refers to the ELU [9] activation function. Here, **W** is the same linear transformation parameter as that in Eq. (1). Through a multi-layer GAT with the number of layers  $L_t$ , the profound feature of each type is learnt. Finally, we use the sum readout function, i.e., sum all type features to get the static type feature vector  $s_e$  with respect to the type set.

In the second level, we also build a fully-connected graph  $\mathcal{G}'_e$  with self-loops for each type set  $\{T_e\}$ . Here  $\mathcal{G}'_e$  not only contains nodes denoting types in  $\{T_e\}$  but also nodes denoting static type feature vectors of other type sets in the same hyper-relational schema tuple obtained from the first level. Similar to the first level, we also resort to a GAT of  $L'_t$  layers to model the complex correlation between nodes, which can dynamically adjust the importance of types in

HELIOS: Hyper-Relational Schema Modeling from Knowledge Graphs

MM '23, October 29-November 3, 2023, Ottawa, ON, Canada

 $\{T_e\}$  with the consideration of other static type feature vectors, resulting in the final contextualized type feature vector  $s'_e$ .

# 3.2 Learning from Schema Tuples

After being processed by the previous module, each type set  $\{T_e\}$  is encoded into a contextualized type feature vector  $s'_{e}$ . Subsequently, a hyper-relational schema tuple  $(\{T_h\}, r, \{T_t\}, k_1, \{T_{v_1}\}, \ldots)$  is transformed into  $(s'_{h}, r, s'_{t}, k_{1}, s'_{v_{1}}, ...)$ . To capture the correlation between types of different entities and relations in the schema tuple, we feed the contextualized type features together with relations to a self-attention network with learnable edge biases discriminating connections between different elements in the schema tuple. We adopt a masked training process, making HELIOS able to predict any missing (masked) elements in a hyper-relational schema tuple. As shown in Figure 2, the missing type set in the hyper-relational schema tuple is replaced by a [MASK] token. Then the schema tuple is fed into a self-attention network to learn the inherent correlation between elements in the schema tuple. Without loss of generality, we present a single self-attention layer below. Following the traditional fully-connected attention paradigm [55], for an input element  $\vec{u_i}$  in the schema tuple  $u_i \in \{s'_h, r, s'_t, k_1, s'_{v_1}, ...\}, \vec{u_i}$  is first projected into attention query, key and value<sup>3</sup>  $\mathbf{W}^Q \vec{u_i}, \mathbf{W}^K \vec{u_i}, \mathbf{W}^V \vec{u_i} \in \mathbb{R}^F$ by linear transformation parameters  $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V \in \mathbb{R}^{F \times F}$ . To measure the similarity between elements, a scaled dot-product is applied:

$$\beta_{ij} = \frac{\left(\mathbf{W}^Q \vec{u}_i\right)^\top \left(\mathbf{W}^K \vec{u}_j + \vec{c}_{ij}^K\right)}{\sqrt{F}} \tag{4}$$

where  $\vec{c}_{ij}^{K}$  (and also  $\vec{c}_{ij}^{V}$  below) refers to learnable edge biases on attention key (and value)<sup>3</sup> [51, 58]. In a self-attention layer, a hyperrelational schema tuple can also be regarded as a fully-connected graph, and edge biases specify different edge classes between elements to discriminate connections between different elements in the schema tuple. In the background, inspired by [58], edge biases are divided into five categories according to edge classes, namely  $(s'_{h}, r), (s'_{t}, r), (r, k), (k, s'_{v})$  and others not included in the above categories. Notably, edge biases are independent of edge direction. For instance,  $(\dot{s_h}, r)$  and  $(r, \dot{s_h})$  belong to the same edge class and will share the same edge bias. With the above definition of edge biases, the detailed formation of edge-biased fully-connected attention is shown in Figure 2 as an adjacency matrix. Such an attention mechanism can facilitate the representation of self-attention by modeling the relative position of elements in the schema tuple [51, 58]. After deriving the similarity scores  $\beta_{ij}$ , a softmax function is used to normalize the scores and the edge biases  $\vec{c}_{ij}^V$  on the attention value<sup>3</sup> is also added when updating the embedding of element  $\vec{u_i}$ 

$$\vec{u}_{i}^{\prime} = \sum_{j=1}^{N} \frac{\exp\left(\beta_{ij}\right)}{\sum_{k=1}^{N} \exp\left(\beta_{ik}\right)} \left(\mathbf{W}^{V} \vec{u}_{j} + \vec{c}_{ij}^{V}\right)$$
(5)

where *N* denotes the number of elements in the schema tuple. Using this self-attention layer with learnable edge biases, a multi-layer self-attention network can generate the informative embedding of [MASK] token to predict the missing element.

|--|

Dataset	JF17K	WikiPeople	WD50K
#Types	613	2127	4186
#Relations	501	178	531
#Types per type set	5.35	1.26	1.35
#Training tuples	76379	294439	166429
Triple+Hyper (%)	57.9%+42.1%	97.4%+2.6%	86.2%+13.8%
#Test tuples	6144	9472	46158
Triple+Hyper (%)	42.4%+57.6%	97.2%+2.8%	86.9%+13.1%

#### 3.3 Prediction with Mask

The previous module outputs the final embedding of [MASK] token, denoted as  $\vec{x}_M$  therein. To make predictions, a single layer of linear transformation with softmax function is used to generate the prediction of the type set:

$$\mathbf{p} = \operatorname{softmax} \left( \mathbf{W}_M \vec{x}_M + \vec{b}_M \right) \tag{6}$$

where  $\mathbf{W}_M$  corresponds to the weight matrix of the input embedding layer with respect to type set and  $\vec{b}_M$  is a learnable type set bias. The final output **p** is a probability distribution over all types in KG. Such prediction scores target elements against all candidates simultaneously, which accelerates the convergence speed and improves the prediction efficiency [12]. Note that for predicting missing a relation, the learnable parameters  $\mathbf{W}_M$  and  $\vec{b}_M$  in Eq. (6) correspond to the weight matrices of the input embedding layer of relation and relation bias, respectively.

#### 3.4 Model Training

In a hyper-relational schema tuple  $({T_h}, r, {T_t}, k_1, {T_{v_1}}, ...) \in \mathbb{R}^N$ , we mask every element in turn to expand the number of training instances from 1 to N. Thus, each instance corresponds to a missing (masked) element to be predicted. Such an expansion enables the model to smoothly learn to predict any missing (masked) elements in a hyper-relational schema tuple. To train the model parameters, we minimize a cross-entropy loss using Adam optimizer [30].

# 4 EXPERIMENTS

#### 4.1 Experimental Setup

4.1.1 Datasets. We conduct experiments on three commonly used hyper-relational KG datasets *JF17K* [47], *WikiPeople* [47], and *WD50K* [18], where the training and test datasets are already split by the data providers. As these datasets do not contain information about entity types, we crawl the entity type information from their corresponding data sources (Freebase and Wikidata) referring to [48]. For Freebase, we extract entity types directly from the entity types by crawling through the property "instance\_of" for each entity. Note that we treat all types equally in this work, and leave the hierarchy of entity types as future work. For each hyper-relational fact (*h*, *r*, *t*, *k*<sub>1</sub>, *v*<sub>1</sub>, ...), we replace the entity by its type set to obtain our hyper-relational schema tuples ({*T<sub>h</sub>*}, *r*, {*T<sub>t</sub>*}, *k*<sub>1</sub>, {*T<sub>v<sub>1</sub>*}, ...). In the training process, we skip all schema tuples that appear in the test to avoid data leakage. Table 1 shows the statistics of our datasets.</sub>

4.1.2 Baselines. We compare HELIOS against a sizable collection of state-of-the-art techniques of hyper-relational link prediction over

 $<sup>^3</sup>Note that attention key and value are completely irrelevant to the key-value pairs in a hyper-relational schema tuple.$ 

KGs, including: NaLP [23] learns the relatedness between relationtype pairs under the n-ary representation of entity-typed tuples; HINGE [47] captures both the triple-wise and quintuple-wise relatedness between elements in entity-typed tuples; NeuInfer [22] models both primary entity-typed triplet and its associated keytype pairs; RAM [36] captures the latent compatibility between the meta-relation and all involved entity types by a pattern matrix; GRAN [58] represents the entity-typed tuple as a heterogeneous graph and captures the inter-vertex interactions via self-attention mechanism; StarE [18] transforms an entity-typed tuple into a directed heterogeneous graph and extract the inter-vertex interaction using a GNN encoder; Hv-Transformer [69] extends StarE substituting the graph encoder by a light-weight relation/type embedding technique; QUAD [53] also extends StarE by adopting two separate aggregators to encode the primary entity-typed triplets and associated key-type pairs, respectively. Detailed hyperparameter settings of baselines and HELIOS are in Appx. A.1. The code and data of HELIOS are available online here<sup>4</sup>.

4.1.3 Dataset Transformation. As discussed in the introduction, our hyper-relational schema tuples  $({T_h}, r, {T_t}, k_1, {T_{v_1}}, ...)$  need to be transformed to entity-typed tuples by replacing each type set by a single type  $(h_type, r, t_type, k_1, v_1\_type, ...)$  for the baseline methods. There are two common settings as follows.

- **One type** kept for each entity [21, 36]. In this setting, we keep only the most popular type for each entity, where the popularity of types refers to their frequency in the dataset. Subsequently, we obtain one entity-typed tuple from each schema tuple.
- Exhaustive combination of types [24, 48]. In this setting, for one schema tuple such as  $(\{T_h\}, r, \{T_t\}, k_1, \{T_{v_1}\})$ , we generate a set of entity-typed tuples of size  $|\{T_h\}| \cdot |\{T_t\}| \cdot |\{T_{v_1}\}|$  by exhaustively combining types of all entities such as  $\{(h\_type, r, t\_type, k_1, v_1\_type)|h\_type \in \{T_h\}, t\_type \in \{T_t\}, v_1\_type \in \{T_{v_1}\}\}$ .

We denote our hyper-relational schema tuples as Original setting.

4.1.4 Evaluation Tasks and Metrics. Schema prediction tasks predict a missing element in a hyper-relational schema tuple, such as  $(?, r, \{T_t\}, k_1, \{T_{v_1}\})$  or  $(\{T_h\}, ?, \{T_t\}, k_1, \{T_{v_1}\})$ , where the missing elements are a type set or a relation, respectively. For the missing type set (or relation) in a test schema tuple, a ranking list of types (or relation) is predicted. In this ranking list, in addition to the ground truth from the test tuple, other type sets (or relation) might also be true; we thus adopt the filtered setting introduced by [6], removing them from the ranking list. As a type set usually contains multiple types as ground truth (also for one type and exhaustive settings), we report Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG) for a fair comparison. We report the average results in predicting: 1) All type sets  $\{T_h\}$ ,  $\{T_t\}$  and  $\{T_v\}$ , and their statistic breakdown; 2) Head and tail type H/T type sets  $\{T_h\}$  and  $\{T_t\}$ ; 3) Value type sets  $\{T_v\}$ ; 4) All relation r and k, and their statistic breakdown; 5) **Primary relation** r; 6) **Key** k.

Note that for the dataset transformation *exhaustive* setting when the size of type sets  $|\{T\}|$  is large, a particular problem for evaluation occurs. As baseline techniques can only make predictions on a single entity-typed tuple, for each test schema tuple with one missing element, we generate a set of entity-typed tuples and each

<sup>4</sup>https://github.com/UM-Data-Intelligence-Lab/HELIOS\_code

baseline technique thus generates a set of ranking lists for the missing element, while we have a unique ground truth for that missing element. Therefore, to bridge this gap, we first aggregate these ranking lists using Reciprocal Rank Fusion [11], and then evaluate the aggregated ranking list against the ground truth.

# 4.2 Schema Prediction Performance

We compare the schema prediction performance of HELIOS against all baselines under both data transformation settings. For each dataset transformation setting, we highlight the best-performing results on each task and for each dataset.

4.2.1 Overall Performance Comparison (All type and All relation). Table 2 shows the overall performance on all three datasets. We observe that HELIOS consistently outperforms all baseline techniques, achieving 29.7% and 20.0% improvement on average over the best baselines in predicting types and relations, respectively. Interestingly, we observe that comparing the two data transformation settings, there is no clear advantage of one over the other. While One type may overlook useful schema information, *Exhaustive* may introduce noisy entity-typed tuples; both lead to degraded schema prediction performance to some extent. Our HELIOS learns directly from entity type sets, resulting in the best performance.

In terms of computational overhead, our HELIOS also outperforms baselines with *exhaustive* settings. For example, on our benchmark hardware (Intel Xeon5320@2.20GHz, 128GB RAM@3200Hz, NVIDIA GeForce RTX 3090 24GB, Ubuntu 18.04), the training process of HELIOS takes 23.0 seconds per epoch on JF17K dataset, while the best-performing baselines GRAN and RAM spend 364.7 and 177.5 seconds per epoch, respectively. This is because the 76,379 training tuples on JF17K dataset are transformed to 1,235,401 (~16x more) tuples with the exhaustive setting.

Moreover, we show two case studies of the attention weights of types learnt by HELIOS in Figure 3. First, in Figure 3a, the most supportive entity-typed tuples for the two facts are *(corporation, board member, human)* and *(technology company, product or material produced, electronics)*, respectively, despite the two facts have the same head entity *Apple*. This implies that the role of multiple types of a single entity varies in different facts. Second, in Figure 3b, the most supportive entity-typed tuple for the two facts are *(corporation, board member, human)* and *(comprehensive academic and research university, member of, coalition)*, respectively, even though their head entities share the same type *corporation*. In other words, the same type *corporation* shows different levels of importance in these two facts. In summary, in both cases, HELIOS effectively identifies the most supportive entity-typed tuples, which cannot be captured by existing approaches with *One type* or *Exhaustive* settings.

4.2.2 Entity Type Prediction Performance Comparison (**H/T type** and **Value type**). Table 3 shows the entity type prediction performance. We observe that HELIOS consistently achieves the best performance, yielding 29.4% and 42.0% improvement over the bestperforming baselines in predicting H/T and value types, respectively. Moreover, compared to the baselines that can predict both H/T and value types, HELIOS achieves robust and stable performance in predicting types at any positions (i.e., head/tail or value) in a hyper-relational schema tuple, while the baselines yield different Table 2: Overall schema prediction performance (*All type* and *All relation*). "N/A" denotes the case that the method cannot be applied to the task (namely RAM, StarE, Hy-Transformer, and QUAD are unable to predict missing relations). "-" refers to the case that the method fails to run. Subsequently, NaLP, HINGE and NeuInfer run out of memory due to the expensive permutation process of n-ary representations; the training process of QUAD runs over time (we set a time limit of 48h for all methods). Moreover, due to the low efficiency of the evaluation code of NaLP, HINGE and NeuInfer, their results are evaluated on 10% of the original test data.

Dataset		JF17K				WikiPeople				WD50K			
Transformation	Method	All	type	All re	lation	All	type	All re	lation	All	type	All re	lation
Setting		MAP	NDCG	MAP	NDCG	MAP	NDCG	MAP	NDCG	MAP	NDCG	MAP	NDCG
	NaLP	0.1421	0.3700	0.4797	0.5771	0.2721	0.3565	0.1926	0.2499	0.1295	0.2536	0.1941	0.3274
	HINGE	0.1447	0.3677	0.6696	0.7302	0.4032	0.5146	0.5225	0.6205	0.1916	0.3306	0.5447	0.6420
	GRAN	0.1878	0.4103	0.8117	0.8416	0.5937	0.6398	0.6174	0.7351	0.3615	0.4906	0.6399	0.7100
Onotro	NeuInfer	0.1312	0.3931	0.9736	0.9802	0.3483	0.4669	0.3055	0.4598	0.1963	0.2326	0.3262	0.4556
One type	RAM	0.1979	0.4365	N/A	N/A	0.2900	0.3198	N/A	N/A	0.4103	0.5358	N/A	N/A
	StarE	0.3886	0.6088	N/A	N/A	0.6306	0.7228	N/A	N/A	0.4711	0.5979	N/A	N/A
	Hy-Transfomer	0.4322	0.6460	N/A	N/A	0.7206	0.7681	N/A	N/A	0.6791	0.7662	N/A	N/A
	QUAD	0.4523	0.6617	N/A	N/A	0.4675	0.5984	N/A	N/A	0.4738	0.5959	N/A	N/A
	NaLP	-	-	-	-	0.2612	0.3491	0.1058	0.2683	0.1680	0.2967	0.1954	0.3304
	HINGE	-	-	-	-	0.1699	0.3210	0.6961	0.7628	0.2771	0.4182	0.6681	0.7442
	GRAN	0.2868	0.5103	0.8497	0.8916	0.1627	0.3090	0.3607	0.4845	0.2170	0.3477	0.6480	0.7196
Exhaustive	NeuInfer	-	-	-	-	0.1600	0.2682	0.3145	0.4642	0.1810	0.3494	0.2699	0.4052
	RAM	0.3347	0.4275	N/A	N/A	0.4737	0.5653	N/A	N/A	0.3671	0.4728	N/A	N/A
	StarE	0.5307	0.6793	N/A	N/A	0.8805	0.9038	N/A	N/A	0.4145	0.5360	N/A	N/A
	Hy-Transfomer	0.4333	0.6141	N/A	N/A	0.8379	0.8783	N/A	N/A	0.4207	0.5441	N/A	N/A
	QUAD	-	-	N/A	N/A	0.8992	0.9230	N/A	N/A	0.4536	0.5708	N/A	N/A
Original	HELIOS	0.9636	0.9808	0.9932	0.9946	0.9558	0.9669	0.9258	0.9426	0.8484	0.8903	0.9048	0.9268







(b) The importance of the same type (corporation) varies in different facts.

Figure 3: Case studies on type attention weights. The table next to each entity shows its top three types according to their attention weights learnt by HELIOS (the type list is not exhaustive due to space limitation). The full name of the entity SPARC (Q647039) is Scholarly Publishing and Academic Resources Coalition. performance when predicting head/tail or value types. To further study this point, Table 5 shows the coefficient of variation [15] on the performance of predicting types at different positions. HELIOS has a coefficient of variation of 4.71%, which is 76.6% smaller than the best-performing baseline GRAN with *One type* setting showing a coefficient of variation of 20.13%.

4.2.3 Relation Prediction Performance Comparison (**Primary relation** and **Key**). Table 4 shows the relation prediction performance. Our HELIOS achieves the best performance in most cases, except on WikiPeople where NeuInfer with One type setting and GRAN with *Exhaustive* setting are slightly better than ours when predicting keys. Compared to the best-performing baselines, HELIOS yields an improvement of 21.1% on primary relation prediction, while a slight drop of 2.1% on key prediction. However, HELIOS is much more robust than the baselines when predicting relations at different positions (primary relation or key) with a much smaller coefficient of variation of 4.09%, which is 72.7% smaller than the best-performing baseline as shown in Table 5.

# 4.3 Additional Experiments in Appendix.

Due to space limitations, we include more experiment results in the Appendix, including ablation studies in Appx. A.2, and the performance of HELIOS across different categories of schema tuples in Appx. A.4.

## 5 CONCLUSION

In this paper, by revisiting the drawbacks of existing approaches on extracting and using KG schema for link prediction, we propose to study a novel problem of modeling hyper-relational schema,

Table 3: Entity type prediction performance (H/T type and Value type). Note that StarE, Hy-Transfomer, and QUAD can only predict head and tail types, but not value types; its performance on H/T type is thus the same as on All type in Table 2.

Dataset			JF1	7K		WikiPeople				WD50K			
Transformation	Method	H/T	type	Value	e type	H/T	type	Value	e type	H/T	type	Value	e type
Setting		MAP	NDCG	MAP	NDCG	MAP	NDCG	MAP	NDCG	MAP	NDCG	MAP	NDCG
	NaLP	0.1021	0.3289	0.2486	0.4796	0.2683	0.3525	0.4846	0.5856	0.1055	0.2320	0.4159	0.5111
	HINGE	0.1206	0.3433	0.2088	0.4328	0.4028	0.5140	0.4261	0.5502	0.1691	0.3122	0.4586	0.5493
	GRAN	0.1819	0.4077	0.2035	0.4172	0.5886	0.6332	0.7123	0.7965	0.3321	0.4666	0.6807	0.7500
One trme	NeuInfer	0.1156	0.3100	0.2031	0.4481	0.3391	0.4584	0.4702	0.5485	0.1874	0.2246	0.3518	0.4276
One type	RAM	0.1863	0.4291	0.2287	0.4563	0.2867	0.3165	0.4587	0.5090	0.3895	0.5191	0.6350	0.7176
	StarE	0.3886	0.6088	N/A	N/A	0.6306	0.7228	N/A	N/A	0.4711	0.5979	N/A	N/A
	Hy-Transfomer	0.4322	0.6460	N/A	N/A	0.7206	0.7681	N/A	N/A	0.6791	0.7662	N/A	N/A
	QUAD	0.4523	0.6617	N/A	N/A	0.4675	0.5984	N/A	N/A	0.4738	0.5959	N/A	N/A
	NaLP	-	-	-	-	0.2585	0.3458	0.4124	0.5349	0.1353	0.2677	0.7072	0.7789
	HINGE	-	-	-	-	0.1659	0.3175	0.3955	0.5216	0.2552	0.4006	0.5533	0.6399
	GRAN	0.2802	0.4871	0.4018	0.6182	0.1576	0.3047	0.4519	0.5528	0.2117	0.3440	0.2746	0.3887
Fyhoustivo	NeuInfer	-	-	-	-	0.1599	0.2680	0.3642	0.5272	0.1561	0.2775	0.4166	0.5467
Exhaustive	RAM	0.3276	0.4018	0.4538	0.4789	0.4775	0.5689	0.2582	0.3617	0.3629	0.4688	0.4517	0.5523
	StarE	0.5307	0.6793	N/A	N/A	0.8805	0.9038	N/A	N/A	0.4145	0.5360	N/A	N/A
	Hy-Transfomer	0.4333	0.6141	N/A	N/A	0.8379	0.8783	N/A	N/A	0.4207	0.5441	N/A	N/A
	QUAD	-	-	N/A	N/A	0.8992	0.9230	N/A	N/A	0.4536	0.5708	N/A	N/A
Original	HELIOS	0.9634	0.9805	0.9642	0.9816	0.9579	0.9684	0.8348	0.8808	0.8411	0.8851	0.9274	0.9472

Table 4: Relation prediction performance (*Primary relation* and *Key*). We exclude the methods (StarE, Hy-Transfomer and QUAD) that cannot predict relations.

Dataset			JF17K			WikiPeople				WD50K			
Transformation	Method	Primary	relation	K	ey	Primary	relation	K	ey	Primary	relation	K	ey
Setting		MAP	NDCG	MAP	NDCG	MAP	NDCG	MAP	NDCG	MAP	NDCG	MAP	NDCG
	NaLP	0.3854	0.4952	0.7309	0.7955	0.1849	0.2431	0.5337	0.6366	0.1596	0.2969	0.6041	0.6905
One type	HINGE	0.5541	0.6357	0.9774	0.9821	0.5190	0.6176	0.7189	0.7848	0.5113	0.6156	0.9424	0.9560
	GRAN	0.6750	0.7568	0.9937	0.9945	0.6017	0.7190	0.8234	0.8817	0.5803	0.6623	0.9627	0.9687
	NeuInfer	0.9553	0.9665	0.9981	0.9986	0.2837	0.4428	0.9226	0.9409	0.2345	0.3816	0.8720	0.8960
	NaLP	-	-	-	-	0.0970	0.2609	0.6073	0.6845	0.1549	0.2949	0.7072	0.7789
Eukoustino	HINGE	-	-	-	-	0.6950	0.7619	0.7577	0.8149	0.6445	0.7259	0.9662	0.9750
Exhaustive	GRAN	0.7751	0.8248	0.9197	0.9342	0.3415	0.4690	0.9025	0.9238	0.5860	0.6702	0.9843	0.9876
	NeuInfer	-	-	-	-	0.3131	0.4631	0.6555	0.7958	0.2348	0.3748	0.6763	0.6817
Original	HELIOS	0.9881	0.9906	1.0000	1.0000	0.9283	0.9445	0.8556	0.8888	0.8867	0.9121	0.9839	0.9875

Table 5: Coefficient of variation on the performance of predicting types (relations) at different positions H/T and value types (primary relations and keys) in hyper-relational tuples.

Data transformation setting	Method	Туре	Relation
	NaLP	49.75%	57.88%
	HINGE	27.85%	30.27%
One type	GRAN	20.13%	24.03%
	NeuInfer	31.22%	44.94%
	RAM	23.50%	N/A
	NaLP	56.97%	80.04%
	HINGE	44.23%	14.95%
Exhaustive	GRAN	29.69%	32.29%
	NeuInfer	52.94%	49.24%
	RAM	22.64%	N/A
Original	HELIOS	4.71%	4.09%

which is formulated as mixed hyper-relational schema tuples. To address this problem, we propose HELIOS, a hyper-relational schema model designed to subtly learn from the two-fold hyper-relations for solving schema prediction problems, by capturing not only the correlation between multiple types of a single entity, but also the correlation between types of different entities and relations in a hyper-relational schema tuple. We evaluate HELIOS on three real-world KG datasets in different schema prediction tasks. Results show that HELIOS consistently outperforms state-of-the-art hyper-relational link prediction techniques by 20.0-29.7%, and is also much more robust than baselines in predicting types and relations across different positions in a hyper-relational schema tuple.

In the future, we plan to further consider the hierarchy of entity types, and also build end-to-end solutions for schema-enhanced link prediction.

# ACKNOWLEDGMENTS

This project has received funding from the University of Macau (SRG2021-00002-IOTSC) and the Science and Technology Development Fund, Macau SAR (0038/2021/AGJ, 0047/2022/A1, SKL-IOTSC(UM)-2021-2023). This work was performed in part at SICC which is supported by SKL-IOTSC, University of Macau.

HELIOS: Hyper-Relational Schema Modeling from Knowledge Graphs

MM '23, October 29-November 3, 2023, Ottawa, ON, Canada

#### REFERENCES

- Vevake Balaraman, Simon Razniewski, and Werner Nutt. 2018. Recoin: relative completeness in Wikidata. In WWW Companion. 1787–1792.
- [2] Ivana Balazevic, Carl Allen, and Timothy M Hospedales. 2018. Hypernetwork Knowledge Graph Embeddings. arXiv preprint arXiv:1808.07018 11731 (2018).
- [3] Ivana Balažević, Carl Allen, and Timothy M Hospedales. 2019. Tucker: Tensor factorization for knowledge graph completion. In *EMNLP-IJCNLP*). 5185--5194.
- [4] 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.
- [5] Rishi Bommasani, Kelly Davis, and Claire Cardie. 2020. Interpreting pretrained contextualized representations via reductions to static embeddings. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. 4758– 4781.
- [6] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In Advances in neural information processing systems. 2787–2795.
- [7] Muhao Chen, Yingtao Tian, Xuelu Chen, Zijun Xue, and Carlo Zaniolo. 2018. On2vec: Embedding-based relation prediction for ontology population. In Proceedings of the 2018 SIAM International Conference on Data Mining. SIAM, 315-323.
- [8] Philipp Cimiano. 2006. Ontology learning and population from text: algorithms, evaluation and applications. Vol. 27. Springer Science & Business Media.
- [9] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. 2015. Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint arXiv:1511.07289 (2015).
- [10] ConceptNet. 2022. https://conceptnet.io/.
- [11] Gordon V Cormack, Charles LA Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In SIGIR. 758-759.
- [12] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In Proceedings of the AAAI conference on artificial intelligence, Vol. 32.
- [13] Shimin Di, Quanming Yao, and Lei Chen. 2021. Searching to sparsify tensor decomposition for n-ary relational data. In *Proceedings of the Web Conference* 2021. 4043-4054.
- [14] Takuma Ebisu and Ryutaro Ichise. 2018. Toruse: Knowledge graph embedding on a lie group. In AAAI.
- [15] Brian S Everitt and Anders Skrondal. 2010. The Cambridge dictionary of statistics. (2010).
- [16] Stefano Faralli, Alexander Panchenko, Chris Biemann, and Simone Paolo Ponzetto. 2017. The ContrastMedium algorithm: Taxonomy induction from noisy knowledge graphs with just a few links. In ACL. Association for Computational Linguistics.
- [17] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. 2013. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In WWW. 413–422.
- [18] Mikhail Galkin, Priyansh Trivedi, Gaurav Maheshwari, Ricardo Usbeck, and Jens Lehmann. 2020. Message passing for hyper-relational knowledge graphs. arXiv preprint arXiv:2009.10847 (2020).
- [19] Claudio Giuliano and Alfio Gliozzo. 2008. Instance-based ontology population exploiting named-entity substitution. In Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008). 265–272.
- [20] Google. 2012. Introducing the Knowledge Graph: things, not strings. https: //blog.google/products/search/introducing-knowledge-graph-things-not/.
- [21] Saiping Guan, Xiaolong Jin, Jiafeng Guo, Yuanzhuo none Wang, and Xueqi Cheng. 2021. Link Prediction on N-ary Relational Data Based on Relatedness Evaluation. *IEEE Transactions on Knowledge and Data Engineering* (2021).
- [22] Saiping Guan, Xiaolong Jin, Jiafeng Guo, Yuanzhuo Wang, and Xueqi Cheng. 2020. Neuinfer: Knowledge inference on n-ary facts. In ACL. 6141–6151.
- [23] Saiping Guan, Xiaolong Jin, Yuanzhuo Wang, and Xueqi Cheng. 2019. Link prediction on n-ary relational data. In The World Wide Web Conference. 583–593.
- [24] Junheng Hao, Muhao Chen, Wenchao Yu, Yizhou Sun, and Wei Wang. 2019. Universal representation learning of knowledge bases by jointly embedding instances and ontological concepts. In KDD. 1709–1719.
- [25] Valentin Hofmann, Janet Pierrehumbert, and Hinrich Schütze. 2021. Dynamic Contextualized Word Embeddings. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). 6970–6984.
- [26] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutiérrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan F. Sequeda, Steffen Staab, and Antoine Zimmermann. 2021. Knowledge Graphs. Number 22 in Synthesis Lectures on Data, Semantics, and Knowledge. Morgan & Claypool. https://doi.org/10.2200/ S01125ED1V01Y202109DSK022
- [27] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli,

Sebastian Neumaier, et al. 2021. Knowledge graphs. ACM Computing Surveys (CSUR) 54, 4 (2021), 1–37.

- [28] Prachi Jain, Pankaj Kumar, Soumen Chakrabarti, et al. 2018. Type-sensitive knowledge base inference without explicit type supervision. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). 75–80.
- [29] 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.
- [30] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014).
- [31] Ni Lao, Tom Mitchell, and William Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In EMNLP. 529–539.
- [32] Linfeng Li, Peng Wang, Jun Yan, Yao Wang, Simin Li, Jinpeng Jiang, Zhe Sun, Buzhou Tang, Tsung-Hui Chang, Shenghui Wang, et al. 2020. Real-world data medical knowledge graph: construction and applications. *Artificial intelligence in medicine* 103 (2020), 101817.
- [33] Na Li, Zied Bouraoui, and Steven Schockaert. 2019. Ontology completion using graph convolutional networks. In *International Semantic Web Conference*. Springer, 435–452.
- [34] 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.
- [35] Yushan Liu, Yunpu Ma, Marcel Hildebrandt, Mitchell Joblin, and Volker Tresp. 2022. TLogic: Temporal Logical Rules for Explainable Link Forecasting on Temporal Knowledge Graphs. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 36. 4120–4127.
- [36] Yu Liu, Quanming Yao, and Yong Li. 2021. Role-Aware Modeling for N-ary Relational Knowledge Bases. In WWW. 2660-2671.
- [37] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, Vol. 30. Citeseer, 3.
- [38] Pasquale Minervini, Claudia d'Amato, Nicola Fanizzi, and Floriana Esposito. 2016. Leveraging the schema in latent factor models for knowledge graph completion. In Proceedings of the 31st annual ACM symposium on applied computing. 327–332.
- [39] 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).
- [40] 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.
- [41] 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.
- [42] Mathias Niepert. 2016. Discriminative gaifman models. In NIPS. 3405-3413.
- [43] Guanglin Niu, Bo Li, Yongfei Zhang, Shiliang Pu, and Jingyang Li. 2020. AutoETER: Automated Entity Type Representation for Knowledge Graph Embedding. arXiv preprint arXiv:2009.12030 (2020).
- [44] Giulio Petrucci, Marco Rospocher, and Chiara Ghidini. 2018. Expressive ontology learning as neural machine translation. *Journal of Web Semantics* 52 (2018), 66–82.
- [45] Marcin Pietrasik and Marek Reformat. 2020. A simple method for inducing class taxonomies in knowledge graphs. In *European Semantic Web Conference*. Springer, 53–68.
- [46] Paolo Rosso, Dingqi Yang, and Philippe Cudré-Mauroux. 2018. Knowledge graph embeddings. In Encyclopedia of Big Data Technologies. 1–7.
- [47] 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. 1885–1896.
- [48] Paolo Rosso, Dingqi Yang, Natalia Ostapuk, and Philippe Cudré-Mauroux. 2021. RETA: A Schema-Aware, End-to-End Solution for Instance Completion in Knowledge Graphs. In WWW. 845–856.
- [49] Arpita Roy and Shimei Pan. 2021. Incorporating extra knowledge to enhance word embedding. In Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence. 4929–4935.
- [50] 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.
- [51] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. arXiv preprint arXiv:1803.02155 (2018).
- [52] Baoxu Shi and Tim Weninger. 2017. ProjE: Embedding projection for knowledge graph completion. In AAAI. 1236–1242.
- [53] Harry Shomer, Wei Jin, Juanhui Li, Yao Ma, and Jiliang Tang. 2022. Learning Representations for Hyper-Relational Knowledge Graphs. arXiv preprint arXiv:2208.14322 (2022).
- [54] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*. 926–934.

Yuhuan Lu, Bangchao Deng, Weijian Yu, and Dingqi Yang

- [55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. Advances in neural information processing systems 30 (2017).
- [56] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. arXiv preprint arXiv:1710.10903 (2017).
- [57] 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.
- [58] Quan Wang, Haifeng Wang, Yajuan Lyu, and Yong Zhu. 2021. Link prediction on n-ary relational facts: A graph-based approach. arXiv preprint arXiv:2105.08476 (2021).
- [59] Shensi Wang, Kun Fu, Xian Sun, Zequn Zhang, Shuchao Li, and Shiyao Yan. 2022. Representation learning of knowledge graphs with the interaction between entity types and relations. *Neurocomputing* 508 (2022), 305–314.
- [60] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes.. In AAAI, Vol. 14. 1112–1119.
- [61] 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.
- [62] Kemas Wiharja, Jeff Z Pan, Martin J Kollingbaum, and Yu Deng. 2020. Schema aware iterative Knowledge Graph completion. *Journal of Web Semantics* 65 (2020), 100616.
- [63] Wikidata. 2022. http://wikidata.org/.
- [64] Fei Wu and Daniel S Weld. 2008. Automatically refining the wikipedia infobox ontology. In Proceedings of the 17th international conference on World Wide Web.

635-644.

- [65] Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2016. Representation Learning of Knowledge Graphs with Hierarchical Types. In IJCAI. 2965–2971.
- [66] Chenyan Xiong, Russell Power, and Jamie Callan. 2017. Explicit semantic ranking for academic search via knowledge graph embedding. In WWW. 1271–1279.
- [67] 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*.
- [68] 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.
- [69] Donghan Yu and Yiming Yang. 2021. Improving hyper-relational knowledge graph completion. arXiv preprint arXiv:2104.08167 (2021).
- [70] Eva Zangerle, Wolfgang Gassler, Martin Pichl, Stefan Steinhauser, and Günther Specht. 2016. An empirical evaluation of property recommender systems for Wikidata and collaborative knowledge bases. In OpenSym. 1–8.
- [71] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In KDD. ACM, 353–362.
- [72] 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. 1185–1194.
- [73] Amal Zouaq and Felix Martel. 2020. What is the schema of your knowledge graph? leveraging knowledge graph embeddings and clustering for expressive taxonomy learning. In Proceedings of the international workshop on semantic big data. 1–6.

HELIOS: Hyper-Relational Schema Modeling from Knowledge Graphs

# A APPENDIX

# A.1 Baseline details and Hyperparameter Settings

We implement the baselines with both dataset transformation settings, namely *One type* and *Exhaustive*, by reference to their best hyperparameter settings on link prediction tasks.

- NaLP<sup>5</sup> represents each entity-typed tuple (*h\_type*, *r*, *t\_type*) with  $(k_i, v_i\_type)$ , i = 1, ..., n, as a set of key-type pairs  $(r_h:h\_type)$ ,  $r_t$ : t type,  $k_i$ :  $v_i$  type) and then a neural network pipeline is constructed to predict the missing keys (relations) or missing types. Notably, NaLP adopts negative sampling to improve the training process. However, this process is not fully applicable to the n-ary representation of an entity-typed tuple because  $r_h$  and  $r_t$  can be inconsistent in this process and this is unreasonable. Therefore, a variant of NaLP named NaLP-fix is utilized to address this issue, which replaces  $r_h$  and  $r_t$  simultaneously in the negative sampling. Following the suggestion of the authors, we set the batch size, embedding size, training epoch and learning rate to {128, 100, 500, 0.00005} for all three datasets with both One type and Exhaustive transformation settings. Furthermore, we set the dimension of fully-connected layer to {1000, 1200, 1200} for JF17K, WikiPeople and WD50K datasets, respectively.
- **HINGE**<sup>6</sup> regards an n-ary entity-typed tuple as a primary entity-typed triplet and its associated key-type pairs. To capture both the triple-wise and quintuple-wise relatedness for (*h\_type*, *r*, *t\_type*) and (*h\_type*, *r*, *t\_type*, *k*, *v\_type*), two convolutional neural network pipelines are utilized. Following the suggestions from the original paper, we set the batch size, embedding size, learning rate and the number of convolution filers to {128, 100, 0.0001, 400} for all three datasets with both *One type* and *Exhaustive* transformation settings. Additionally, the training epoch is set to {600, 400, 400} for JF17K, WikiPeople and WD50K datasets, respectively.
- NeuInfer<sup>7</sup> represents each entity-typed tuple as a primary entitytyped triplet associated with a set of key-type pairs. Subsequently, a fully-connected neural network model is used to make predictions on entity-typed tuples. For both *One type* and *Exhaustive* transformation settings, we set the batch size, embedding size and learning rate to {128, 100, 0.00005} for all three datasets. The training epoch is reduced to {500, 500, 1000} for JF17K, WikiPeople and WD50K datasets, respectively, to prevent overfitting. As a pivotal hyperparameter in NeuInfer, the number of fully-connected layers capturing relatedness in the primary entity-typed triplet is set to {2, 1, 1} for JF17K, WikiPeople and WD50K datasets, respectively.
- **RAM**<sup>8</sup> explores the compatibility between roles (relations) to generate a meta-relation by representing relations as linear combinations of basic vectors in a latent space. Then it further captures the compatibility between the meta-relation and all involved entity types by a pattern matrix. Following the suggestion made by the authors, the batch size, training epoch, learning rate and the number of basic latent vectors are set to {64, 200, 0.005, 10}

<sup>7</sup>https://github.com/gsp2014/NeuInfer <sup>8</sup>https://github.com/liuyuaa/RAM for all three datasets with the **One type** transformation setting. For **Exhaustive** transformation, we set the training epoch to {10, 200, 30} for JF17K, WikiPeople and WD50K datasets, respectively. Meanwhile, the learning rate is set to 0.0002 for JF17K dataset, while other hyperparameters remain the same as for *One type* transformation.

- **GRAN**<sup>9</sup> represents the entity-typed tuple as a heterogeneous graph and captures the inter-vertex interactions by a self-attention mechanism, which simultaneously exploits the global and local dependencies in an entity-typed tuple. For *One type* transformation, wet set the batch size, embedding size, training epoch and learning rate to {1024, 256, 160, 0.0005} for all three datasets. Also, the number of hidden layers is set to {12, 6, 12} for JF17K, WikiPeople and WD50K datasets, respectively. For *Exhaustive* transformation, we change the training epoch from 160 to 60 for JF17K dataset.
- **StarE**<sup>10</sup> depicts the entity-typed tuple as a directed multi-relational graph and employs a message-passing based graph encoder to capture the heterogeneous inter-vertex interactions. Afterward, a transformer-based decoder is applied to predict the missing tail types. Following the suggestions from the original paper, the batch size, embedding size, graph layers and learning rate are set to {128, 200, 2, 0.0001} for all three datasets with the *One type* transformation setting. Moreover, the training epoch is set to {400, 500, 400} for JF17K, WikiPeople and WD50K datasets, respectively. For *Exhaustive* transformation, the batch size is enlarged to 1024 for JF17K dataset.
- **Hy-Transformer**<sup>11</sup> is a variant of StarE, which substitutes the graph encoder with light-weight relation/type embedding techniques for efficient computing. Moreover, a key-type pair-oriented training measure is added to enhance the prediction ability. For both *One type* and *Exhaustive* transformation settings, we set the learning rate and graph layers to {0.0001, 2} for all three datasets. Then, the batch size is set to {1024, 1024, 512} for JF17K, WikiPeople and WD50K datasets, respectively. Concurrently, the training epoch is set to {400, 500, 400} for JF17K, WikiPeople and WD50K datasets, respectively.
- **QUAD**<sup>12</sup> further extends StarE by adopting two separate aggregators to encode the primary entity-typed triplet and associated key-type pairs, respectively. Then the entity-typed triplet and associated key-type pairs are masked separately and fed into the transformer-based decoder to make predictions. Following the suggestion from the authors, we set the batch size, embedding size, training epoch, and the number of qualifier layers and triplet layers to {128, 200, 500, 2, 2} for all three datasets with the **One type** transformation setting. For **Exhaustive** transformation, the batch size is changed to 1024 for JF17K dataset.
- **HELIOS**<sup>13</sup> is our proposed model directly learning from hyperrelational schema tuples. With the *Original* transformation setting, the batch size, embedding size, the number of GAT layers, the number of self-attention layers, the number of attention heads

<sup>&</sup>lt;sup>5</sup>https://github.com/eXascaleInfolab/HINGE\_code/tree/master/NALP

<sup>&</sup>lt;sup>6</sup>https://github.com/eXascaleInfolab/HINGE\_code/tree/master/HINGE

 $<sup>^{9}</sup> https://github.com/PaddlePaddle/Research/tree/master/KG/ACL2021\_GRAN$ 

<sup>&</sup>lt;sup>10</sup>https://github.com/migalkin/StarE

<sup>&</sup>lt;sup>11</sup>https://github.com/PlusRoss/Hy-Transformer

<sup>12</sup> https://github.com/HarryShomer/QUAD

<sup>&</sup>lt;sup>13</sup>https://github.com/UM-Data-Intelligence-Lab/HELIOS\_code

Table 6: Overall schema prediction performance comparison between HELIOS and its variants.

		JF1	7K			WikiI	People		WD50K			
Method	All type		All relation		All type		All relation		All type		All relation	
	MAP	NDCG	MAP	NDCG	MAP	NDCG	MAP	NDCG	MAP	NDCG	MAP	NDCG
HELIOS(w/o edge biases)	0.9592	0.9790	0.9717	0.9787	0.9242	0.9368	0.8771	0.9043	0.8234	0.8711	0.8552	0.8886
HELIOS(w/o 2nd GAT)	0.9630	0.9811	0.9939	0.9952	0.9529	0.9644	0.8753	0.9032	0.8418	0.8854	0.9047	0.9259
HELIOS	0.9636	0.9808	0.9932	0.9946	0.9558	0.9669	0.9258	0.9426	0.8484	0.8903	0.9048	0.9268

Table 8: Performance across different categories of test tuples. Note that we exclude JF17K dataset, as it contains too few tuples of SType-HTuple.

Tasks	Test tuple	Wiki	People	WD50K		
	categories	MAP	NDCG	MAP	NDCG	
	SType-TTuple	0.9668	0.9702	0.8568	0.8857	
All Tyme	MType-TTuple	0.9572	0.9725	0.8031	0.8649	
Ап туре	SType-HTuple	0.8408	0.8733	0.9081	0.9253	
	MType-HTuple	0.7862	0.8548	0.9295	0.9521	
All Relation	SType-TTuple	0.8830	0.9475	0.8588	0.8903	
	MType-TTuple	0.9142	0.9418	0.8961	0.9227	
	SType-HTuple	0.8227	0.8233	0.9477	0.9604	
	MType-HTuple	0.9786	0.9769	0.9832	0.9885	

Table 7: Statistics of the test schema tuples in different categories.

Test tuple categories	JF17K	WikiPeople	WD50K
SType-TTuple	50	6306	19219
MType-TTuple	2555	2899	20898
SType-HTuple	1	135	1010
MType-HTuple	3538	132	5012

and the learning rate are set to  $\{1024, 256, 4, 6, 4, 0.0001\}$  for all three datasets.

# A.2 Ablation Studies

To delve into the effectiveness of each component in HELIOS, we conduct ablation studies on two critical components, i.e., the edge biases and the second GAT. The results are shown in Table 6. HE-LIOS performs better than its ablation variants, resulting in up to 5.0% (in predicting relations on WD50K) and 5.1% (in predicting relations on WD50K) and second GAT, respectively.

# A.3 Statistics of the datasets with different categories of test tuple

Table 7 presents the statistics of test schema tuples in different categories. As the JF17K dataset contains only one hyper-relational tuple with a single type (SType-HTuple), we exclude JF17K and focus only on WikiPeople and WD50K datasets to study the performance of HELIOS across different categories of schema tuples in Section A.4. Note that following StarE [18], the maximum arity for WD50K dataset is set to 7 in the experiments.

# A.4 Performance across Different Categories of Schema Tuples

We investigate the performance of HELIOS across different categories of hyper-relational schema tuples. Specifically, the meaning of hyper-relations here is two folds. First, according to the size of type sets, the test schema tuples can be classified into two categories: single type (SType) where all type sets contain only one type, and multiple types (MType) where at least one type set contains multiple types. Second, according to the number of key-(type)set pairs, the test schema tuples can be classified into two categories: triple tuple (TTuple) without any key-(type)set pairs, and hyperrelational tuple (HTuple) with key-(type)set pairs. Subsequently, we consider the combinations of these two classifications, resulting in four categories of test tuples, and compare their performance as shown in Table 8. We observe that for type prediction tasks, MType underperforms SType in general (by 1.6% on average), as the prediction on multiple types is more difficult than on single types. In contrast, for relation prediction tasks, MType consistently outperforms SType by 6.9% on average, implying that knowing more types can help the prediction of relations in schema tuples.