HyperCL: A Contrastive Learning Framework for Hyper-Relational Knowledge Graph Embedding with Hierarchical Ontology

Yuhuan Lu, Weijian Yu, Xin Jing, Dingqi Yang[*]
State Key Laboratory of Internet of Things for Smart City and
Department of Computer and Information Science, University of Macau, China
{yc17462, mc14948, yc27431, dingqiyang}@um.edu.mo

Abstract

Knowledge Graph (KG) embeddings are essential for link prediction over KGs. Compared to triplets, hyper-relational facts consisting of a base triplet and an arbitrary number of key-value pairs, can better characterize real-world facts and have aroused various hyper-relational embedding techniques recently. Nevertheless, existing works seldom consider the ontology of KGs, which is beneficial to link prediction tasks. A few studies attempt to incorporate the ontology information, by either utilizing the ontology as constraints on entity representations or jointly learning from hyper-relational facts and the ontology. However, existing approaches mostly overlook the ontology hierarchy and suffer from the dominance issue of facts over ontology, resulting in suboptimal performance. Against this background, we propose a universal contrastive learning framework for hyper-relational KG embeddings (**HyperCL**), which is flexible to integrate different hyper-relational KG embedding methods and effectively boost their link prediction performance. HyperCL designs relation-aware Graph Attention Networks to capture the hierarchical ontology and a concept-aware contrastive loss to alleviate the dominance issue. We evaluate HyperCL on three real-world datasets in different link prediction tasks. Experimental results show that HyperCL consistently boosts the performance of state-of-the-art baselines with an average improvement of 3.1-7.4% across the three datasets.

1 Introduction

Knowledge Graphs (KGs) which represent a network of real-world entities and exhibit the relationship between them, have empowered a wide range of applications, such as question answering (Yih et al., 2015) or recommender systems (Zhang et al., 2016). KGs are generally expressed as a set of triplets; each triplet denoted by *(head, relation, tail)*, or *(h, r, t)* for short, encodes the connection from a head entity to a tail entity, such as *(Apple, headquarters location, Cupertino)* shown in Fig. 1. To better illustrate real-world facts, hyper-relational facts are developed in Freebase (Bollacker et al., 2008) and Wikidata (Wikidata, 2022), which consist of not only a base triplet *(h, r, t)*, but also an arbitrary number of key-value pairs *(k, v)* further describing the base triplet, represented as *(h, r, t,* k_1*,* v_1*, ...)*. Fig. 1 presents an example of hyper-relational facts on Wikidata *(Apple, industry, software industry, of, computer program, of, operating system)*.

To effectively make use of hyper-relational facts, recent studies have proposed various embedding methods to solve link prediction tasks over KGs. Most of them learn to capture the structural information encoded in hyper-relational facts with Convolutional Neural Networks (CNNs) (Rosso et al., 2020), Graph Neural Networks (GNNs) (Galkin et al., 2020), or Transformer (Wang et al., 2021). However, they often neglect the importance of modeling ontology in KGs, which has shown to be significantly useful (Rosso et al., 2021). For example, the entities *computer program* and *operating system* in Fig. 1 are hard to differentiate and tend to have similar representations by current embedding methods since they are affiliated to the same hyper-relational fact and have a common key *of*. Nevertheless, they can be more distinguishable through the ontology. As shown in Fig. 2, *computer program* and *operating system* respectively belong to different concepts *program* and *system*. Hence, it is beneficial to incorporate the ontology information into entity representations.

In this context, existing hyper-relational KG embedding methods employ the ontology information of KGs as type constraints or joint learning. Specifically, most specific concepts for entities are considered as entity types and used to compute the

[*]Corresponding author.

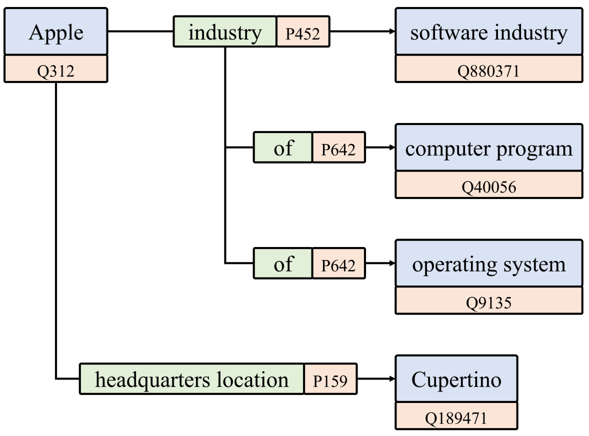

Figure 1: A real-world example of (hyper-relational) facts from Wikidata.

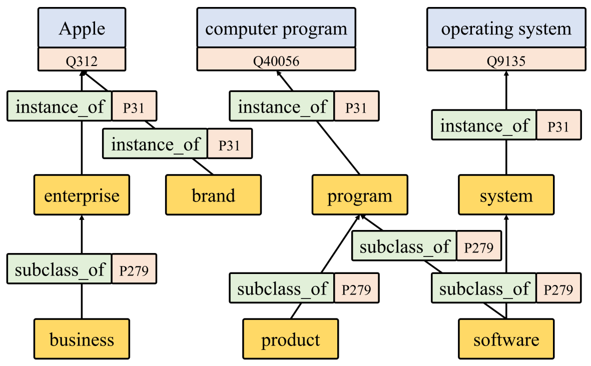

Figure 2: A hierarchical ontology of partial entities in Fig. 1. Yello blocks denote concepts. There are two kinds of ontology triplets, *(entity, instance_of, concept)* and *(concept, subclass_of, concept)*.

similarity between entities (Liu et al., 2021). The most specific concepts for entities are identified by the ontology relation *instance_of* in Fig. 2. For example, *enterprise* and *brand* are two entity types for the entity *Apple*. However, these existing methods do not model the hierarchical structure of the ontology and thus fail to capture the semantic relations between entity types, which is a strong clue for entity representations. For example, two specific concepts *program* and *system* in Fig. 2 belong to a common abstract concept *software*, which indicates the semantic relatedness between entities *computer program* and *operating system*. On the other hand, joint learning methods combine hyper-relational facts with the ontology of KGs, formulating a joint model to learn the representations of both entities and concepts (Lu et al., 2023b; Luo et al., 2023a). Yet, they are either inflexible to accommodate the hierarchical ontology as the type constraint methods or lack consideration of the dominance issue of hyper-relational facts over the ontology. Specifically, due to the highly imbalanced numbers of entities and concepts (the latter is usually much less than the former), the learning process is dominated by learning from the facts rather than from ontology (using GNNs for example), resulting in the information of ontology barely encoded into entities and thus causing the suboptimal performance (as evidenced by our experiments below that addressing the dominance issue can boost the link prediction performance by 2.5-5.8%).

Against this background, we propose a universal contrastive learning framework for hyper-relational KG modeling (**HyperCL**), which is flexible to integrate different hyper-relational KG embedding methods. Specifically, we inherit the most prominent encoder-decoder architecture (Galkin et al., 2020; Luo et al., 2023b) as the backbone of our framework and devise a Concept-aware Contrastive Learning (**CCL**) module to enhance hyper-relational KG embedding methods of this architecture (four state-of-the-art encoder-decoder models are selected to verify the effectiveness of HyperCL in the experiments). The CCL module first captures the hierarchical structure of ontology; we use relation-aware Graph Attention Networks (GATs) to encode the sophisticated concept information, which accounts for the heterogeneous relationships between concepts and incorporates the heterogeneity into entity and concept representations. Afterward, it builds two views (i.e., an instance view for hyper-relational facts and an ontology view for the hierarchical ontology) of KGs. Finally, it develops a concept-aware contrastive loss to enforce the representations of the same entities across two views to be close to each other while those of different but semantically similar entities to be apart. This design thereby alleviates the dominance issue of hyper-relational facts by first decoupling the learning process from the facts and ontology and then connecting them via our contrastive loss. HyperCL is trained using a multi-task learning strategy, being able to accelerate the optimization process. Our contributions can be summarized as follows:

- We revisit the existing approaches that employ the ontology of KGs for hyper-relational KG embeddings, and discover two key limitations: 1) the ignorance of the ontology hierarchy; and 2) the dominance issue of facts over ontology.
- We propose HyperCL framework to subtly model both hyper-relational facts and the hierarchical ontology of KGs. A CCL module is designed,

where the relation-aware GATs are used to capture the hierarchical structure of ontology while the concept-aware contrastive loss is employed to alleviate the dominance issue, both of which enhance the hyper-relational KG embeddings.

- We conduct a thorough evaluation of HyperCL to demonstrate its effectiveness in boosting the link prediction performance of four hyper-relational KG embedding methods on three real-world KGs. Results show that HyperCL can consistently boost the performance of these methods with an average improvement of 3.1-7.4% across the three datasets.

2 Related Work

2.1 Hyper-Relational Facts Modeling

The triple representation of a KG oversimplifies the intricate structure of information stored in the KG (Rosso et al., 2020), especially for hyper-relational facts where each fact is composed of multiple entities and relations. Some previous works employed an n-ary representation for hyper-relational facts, i.e., a set of relation-entity pairs (Wen et al., 2016; Zhang et al., 2018; Guan et al., 2019; Fatemi et al., 2021; Liu et al., 2021; Wang et al., 2023a). Upon such n-ary representations, these approaches learn either relatedness between relation-entity pairs or relatedness among all entities in a fact. However, recent studies (Rosso et al., 2020) discovered that the base triplet of a hyper-relational fact preserves the essential information, and advised directly learning from hyper-relational facts. Following this suggestion, HINGE (Rosso et al., 2020), NeuInfer (Guan et al., 2020), and ShrinkE (Xiong et al., 2023) separately model base triplets and key-value pairs. GRAN (Wang et al., 2021) proposes a heterogeneous graph to distinguish between the relation-entity connections in base triplets and those in key-value pairs. HyNT (Chung et al., 2023) devises a context Transformer to learn the representations of numeric literals in either triplets or qualifiers. HyperFormer (Hu et al., 2023) encodes the local-level sequential information in hyper-relational facts with Transformers. MSeaHKG (Di and Chen, 2021), StarE (Galkin et al., 2020), HyT (Yu and Yang, 2021), QUAD (Shomer et al., 2022), and HAHE (Luo et al., 2023b) design GNNs to represent the base triplets together with key-value pairs.

Our work focuses on a different perspective to improve current hyper-relational KG embedding methods by subtly incorporating the ontology of KGs. To the best of our knowledge, this is the first universal framework for modeling both hyper-relational facts and ontology information.

2.2 Ontology of KGs

The ontology of a KG provides rich descriptions of the semantics of entities, which promotes the representation of the KG (Krompaß et al., 2015). Some recent studies utilized the concepts in ontology as entity types to constrain the representation of entities (Krompaß et al., 2015; Xie et al., 2016; Niu et al., 2020; Cui et al., 2021; Rosso et al., 2021; Yang et al., 2023; Li et al., 2023). RAM (Liu et al., 2021) extends the type-constraint mechanism to encompass hyper-relational facts, representing entity types through linear combinations of latent vectors. HELIOS (Lu et al., 2023a) investigates the problem of hyper-relational schema modeling with flat entity ontology. However, these previous works fail to accommodate the hierarchical structure of ontology. To address the above limitations, JOIE (Hao et al., 2019) and DGS (Iyer et al., 2022) develop a joint learning architecture to learn from both triplets and ontology. sHINGE (Lu et al., 2023b), tNaLP (Guan et al., 2021), and DHGE (Luo et al., 2023a) follow this fashion, using two pipelines to represent hyper-relational facts and the ontology, respectively. Nevertheless, sHINGE and tNaLP parallelly learn from multiple types for an entity, also neglecting the hierarchical nature of ontology; DHGE overlooks the dominance issue of facts over ontology, rendering the model training dominated by hyper-relational facts while the ontology information is barely encoded into entity representations.

We argue that the above concerns can be well addressed by our HyperCL framework.

3 Preliminaries

In this section, we introduce some important notions about the Hyper-relational Knowledge Graph (HKG) and present the latest encoder-decoder architecture for HKG embedding.

3.1 Hyper-Relational Knowledge Graphs

We formalize two views of the HKG and present the definition of the link prediction task on it.

Instance view of the HKG. The instance view of the HKG consists of an entity set $\mathcal{E}$ and an instance relation set $\mathcal{R}_I$. A hyper-relational fact from the instance view can be represented as a base triplet (h, r_I, t) with a set of associated key-value pairs $\{(k_i, v_i)\}_{i=1}^{n}$, where $h, t, v_i \in \mathcal{E}$ and $r_I, k_i \in \mathcal{R}_I$.

Ontology view of the HKG. The ontology view of the HKG is comprised of the same entity set $\mathcal{E}$, a concept set $\mathcal{C}$ and an ontology relation set $\mathcal{R}_O$. The ontology relation set can be further divided into two subsets $\mathcal{R}_{Oe}$ and $\mathcal{R}_{Oc}$, representing entity-concept relations and concept-concept relations, respectively. Accordingly, there exist two kinds of triplets in the ontology view, $(e, r_{Oe}, c) \in \mathcal{E} \times \mathcal{R}_{Oe} \times \mathcal{C}$ and $(c_i, r_{Oc}, c_j) \in \mathcal{C} \times \mathcal{R}_{Oc} \times \mathcal{C}$.

Link prediction on the HKG. The task of link prediction on the HKG is to predict a missing element from hyper-relational facts in the instance view. For a hyper-relational fact, the missing one can be any entity in $\{h, t, v_1, v_2, \ldots, v_n\}$ or any relation in $\{r_I, k_1, k_2, \ldots, k_n\}$.

Since the ontology view only contains triplets, the term "hyper-relational fact" is specifically used to denote the facts in the instance view throughout this paper.

3.2 Encoder-Decoder Architecture for HKGs

The encoder-decoder architecture is the most prevalent and widely adopted framework for HKG embedding, which proves to be effective in link prediction (Galkin et al., 2020; Luo et al., 2023b). As shown in Fig. 3, this architecture (the grey part) is composed of an encoder (mostly GNNs) and a decoder (mostly Transformers). Specifically, the encoder captures the intricate relationship between entities $\mathcal{E}$ and relations $\mathcal{R}_I$ in the *instance view*, encoding the structural information to obtain the updated embeddings $\widehat{\mathcal{E}}$ and $\widehat{\mathcal{R}_I}$. The decoder extends the capabilities of the architecture by capturing the semantic correlation between entities and relations within each hyper-relational fact, generating the final output for link prediction.

In this work, we inherit the encoder-decoder architecture as the backbone of our framework, and integrate multiple state-of-the-art HKG embedding methods to validate the effectiveness of our framework in the experiments.

4 Methodology

This section introduces our universal contrastive learning framework (HyperCL), for hyper-relational KG embeddings. As shown in Fig. 3, our Concept-aware Contrastive Learning (CCL) module is proposed to be universally compatible with any encoder-decoder architecture. Specifically, our CCL consists of two key components: 1) relation-aware graph attention networks to obtain the updated entity embeddings in the ontology view $\widetilde{\mathcal{E}}$; 2) a concept-aware contrastive loss function that captures the shared information by both views to get the final entity representations $\overline{\mathcal{E}}$. In the following, we elaborate on the above two components and present a multi-task learning approach for model training.

4.1 Relation-Aware Graph Attention Layers

For the *ontology view*, we employ a graph encoder to capture the sophisticated hierarchical concept information and encode it into high-order entity embeddings. Since the vanilla Graph Neural Networks (GNNs) fail to accommodate the diverse ontology relation types, we refine the original Graph Attention Networks (GATs) (Veličković et al., 2017) with relation type embeddings to adaptively incorporate the heterogeneous relationships into node representations.

Specifically, the ontology view can be regarded as a graph $\mathcal{G}_O$, where each entity or concept is associated with a node in $\mathcal{G}_O$. Without loss of generality, we depict a single relation-aware graph attention layer in the following. Given a node i, its neighbors are denoted by $\mathcal{N}_i$. The aggregation of the first-hop structural information of i can be expressed as a linear combination of its neighboring nodes' representations:

$$\mathbf{h}_{\mathcal{N}_i} = \sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{h}_j \tag{1}$$

where $\mathbf{h}_j$ refers to the representation of node j and α_{ij} denotes the attention score from node i to node j, which is computed by:

$$\alpha_{ij} = \frac{\exp\left(a\left[\mathbf{W}_O\mathbf{h}_i \| \mathbf{W}_O\mathbf{h}_j \| \mathbf{W}_b\mathbf{b}_{r(i,j)}\right]\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(a\left[\mathbf{W}_O\mathbf{h}_i \| \mathbf{W}_O\mathbf{h}_k \| \mathbf{W}_b\mathbf{b}_{r(i,k)}\right]\right)} \tag{2}$$

where a represents the attention mechanism that applies a single layer of feed-forward neural network with the LeakyReLU activation function. $\mathbf{W}_O$ and $\mathbf{W}_b$ are learnable parameters. $r(i, j)$ denotes the relation type between node i and node j and $\mathbf{b}_{r(i,j)}$ denotes the embedding of $r(i, j)$. Then the representation of node i is updated by:

$$\mathbf{h}_i^{(l)} = \sigma\left(\mathbf{W}_l^{(l)}\left(\mathbf{h}_i^{(l-1)} + \mathbf{h}_{\mathcal{N}_i}^{(l-1)}\right)\right) \tag{3}$$

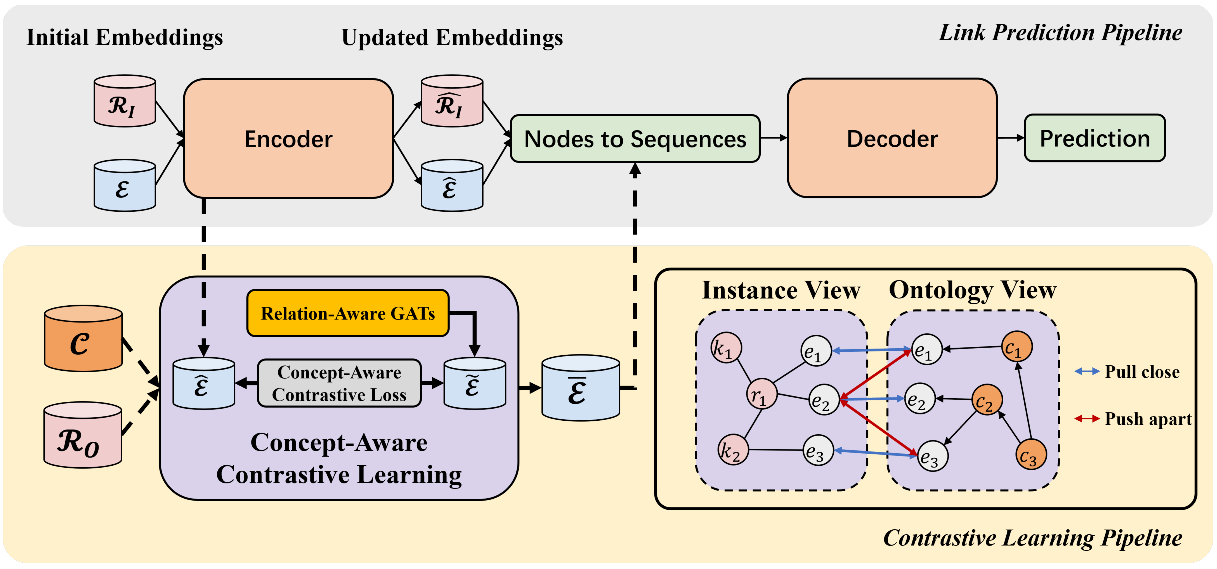

Figure 3: Overview of our HyperCL framework. The grey box denotes the link prediction pipeline with the encoder-decoder architecture while the yellow box represents the pipeline of our Concept-aware Contrastive Learning (CCL) module. The details of the contrastive learning are depicted in the bottom right. In this framework, $\mathcal{E}$ is the input entity embedding set, $\widehat{\mathcal{E}}$ is the updated entity embedding set in the instance view, $\widetilde{\mathcal{E}}$ is the updated entity embedding set in the ontology view, and $\overline{\mathcal{E}}$ is the final entity embedding set updated by CCL module. Likewise, $\mathcal{R}_I$ is the input instance relation set, $\widehat{\mathcal{R}_I}$ is the updated instance relation set, and $\mathcal{R}_O$ is the input ontology relation set.

where $\mathbf{W}_l^{(l)}$ is the learnable parameter at the l-th layer and σ refers to the activation function. Through multi-layer message passing and information aggregation, we can obtain the final embeddings of entities in the ontology view $\widetilde{\mathcal{E}}$.

4.2 Concept-Aware Contrastive Loss

After receiving the updated entity embeddings $\hat{\mathcal{E}}$ and $\widetilde{\mathcal{E}}$ from the instance and ontology views respectively, a concept-aware contrastive loss is developed to pull the representations of the same entity across two views together while separates apart those of different but semantically similar entities, thereby strengthening the distinction between entity representations and alleviating the dominance of instance view information. Furthermore, the concept-aware contrastive loss is developed to distinguish positive and hard-negative samples (entities under the same/similar concept). This design is motivated by the fact that the easy-negative samples can be effectively distinguished by minimizing the link prediction loss as existing approaches do (e.g., StarE (Galkin et al., 2020), HyT (Yu and Yang, 2021), QUAD (Shomer et al., 2022), and HAHE (Luo et al., 2023b)), while our concept-aware contrastive loss complementarily focuses on the hard-negative samples. A similar idea has also been adopted by (Yang et al., 2023), where a fast-thinking process efficiently filters out easy-negative samples, while a slow-thinking process focuses on distinguishing hard-negative samples.

Specifically, a concept-aware batch selection strategy is proposed, ensuring all entities in a batch belong to a common concept. For a concept in the ontology view, the Breadth-First Search (BFS) algorithm is used to collect entities belonging to the concept. As shown in Fig. 4, the BFS starts from the three concepts c_i, c_j and c_k, and attains their corresponding entity sets $\{e_i, e_j\}$, $\{e_k, e_m, e_p\}$ and $\{e_i, e_j, e_k, e_m, e_p\}$, respectively. Then batches are selected from these entity sets while meeting the requirement that one batch can only be randomly sampled from one entity set. To prevent overfitting, we implement a size threshold to select batches from entity sets whose sizes are larger than the threshold. Compared to the traditional random sampling-based batch selection, our strategy has three advantages: 1) enforces contrastive learning to focus on separating entities with similar semantics (common concepts), thereby facilitating the distinct representation of each entity; 2) sets up a size threshold to concentrate on those hard to be distinguished entities, thus improving the efficiency of contrastive learning; 3) maintains inherent distribution biases of concepts while implementing batch selection. For each entity e_i in a batch, we hold the two views of the same entity as a positive pair $(\hat{\mathbf{e}}_i, \tilde{\mathbf{e}}_i)$. On the other hand, any other entity e_j

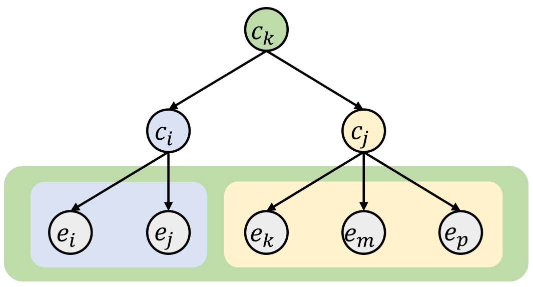

Figure 4: An example of the entity sets obtained by the Breadth-First Search (BFS) algorithm. Different colors indicate different concepts and their corresponding entity sets.

in the same batch is deemed a negative entity and is used to construct the negative pairs $(\hat{\mathbf{e}_i}, \widetilde{\mathbf{e}_j})$ and $(\hat{\mathbf{e}_j}, \widetilde{\mathbf{e}_i})$. Finally, an extended InfoNCE loss (Wang et al., 2023b) is utilized as the contrastive loss:

$$\mathcal{L}_{CL}(i) = - \log \frac{\exp\left(\frac{s(\hat{\mathbf{e}_i},\widetilde{\mathbf{e}_i})}{\tau_{cl}}\right)}{\sum_{j\in\mathcal{H}_i\cup\{i\}}\left(\exp\left(\frac{s(\hat{\mathbf{e}_i},\widetilde{\mathbf{e}_j})}{\tau_{cl}}\right)+\exp\left(\frac{s(\hat{\mathbf{e}_j},\widetilde{\mathbf{e}_i})}{\tau_{cl}}\right)\right)} \tag{4}$$

where $s(\cdot)$ is a cosine similarity metric to measure the similarity between two vectors, $\mathcal{H}_i$ is the set of negative entities for e_i and τ_{cl} is the temperature hyperparameter controlling the strength of penalties on negative entities.

4.3 Multi-task Training

The CCL module subtly connects entity embeddings in both views in a self-supervised manner. To ensure the separability and flexibility of CCL, only the updated entity representations in the instance view are fed into the subsequent modules in HyperCL for link prediction, generating a link prediction loss $\mathcal{L}_{LP}$, similar to previous HKG embedding methods. Hence, the overall loss function is defined as:

$$\mathcal{L} = \mathcal{L}_{LP} + \lambda\mathcal{L}_{CL} \tag{5}$$

where λ is a hyperparameter trading off the two losses.

Given the difficulty in identifying the most suitable λ, we employ a multi-task training strategy by alternating the training procedures of link prediction and contrastive learning. The corresponding parameters in the two pipelines are updated alternatively until the link prediction pipeline reaches convergence. Note that during the training phase of the contrastive learning pipeline, any entity in the same batch can be used as the negative entity for others. The code of HyperCL is publicly available online[1].

5 Experiments

In this section, we present the experimental setup, results, and discussion, answering the following questions. **RQ1**: Can HyperCL consistently boost the link prediction performance of different hyper-relational KG embedding methods? **RQ2**: What's the impact of the concept-aware contrastive loss on link prediction performance? **RQ3**: What's the impact of modeling the hierarchical structure of ontology on link prediction performance?

5.1 Experimental Setup

5.1.1 Datasets

We conduct experiments on three commonly used hyper-relational KG datasets **JF17K** (Wen et al., 2016), **WikiPeople** (Guan et al., 2019), and **WD50K** (Galkin et al., 2020), where the data provider already splits the training and test datasets. As these datasets do not contain ontology information, we crawl concepts from their corresponding data sources (Freebase and Wikidata). For Freebase, we extract concepts directly from the entity node depicted as "/type/object", where the hierarchical concepts for an entity are also exhibited. For Wikidata, we first collect concepts through the property "instance_of" for each entity and then extract deeper concepts through the property "subclass_of" for each concept, until no deeper concepts are found. Table 1 shows the statistics of our datasets.

5.1.2 Baselines

We consider a sizeable collection of state-of-the-art techniques from two categories. The first category includes model learning from hyper-relational facts only: **m-TransH** (Wen et al., 2016); **RAE** (Zhang et al., 2018); **NaLP** (Guan et al., 2019); **NeuInfer** (Guan et al., 2020); **HINGE** (Rosso et al., 2020); **ShrinkE** (Xiong et al., 2023); **GRAN** (Wang et al., 2021); **MSeaHKG** (Di and Chen, 2021); **HyNT** (Chung et al., 2023); **HyperFormer** (Hu et al., 2023); **StarE** (Galkin et al., 2020); **HyT** (Yu and Yang, 2021); **QUAD** (Shomer et al., 2022); **HAHE** (Luo et al., 2023b). The second category includes model learning from both hyper-relational facts and ontology of KGs: **RAM** (Liu et al., 2021); **tNaLP**

[1] https://github.com/UM-Data-Intelligence-Lab/HyperCL_code

Dataset	Entities	Relations	Concepts	Training	Test	Facts (Hyper%)	Arity
JF17K	28,645	501	748	76,379	24,568	100,947 (45.9%)	2-6
WikiPeople	34,839	178	5,396	294,439	37,712	332,151 (2.6%)	2-7
WD50K	47,156	532	9,370	166,435	46,159	212,594 (13.6%)	2-67

Table 1: Statistics of the datasets. The columns respectively denote the number of entities, relations, concepts, training facts, test facts, all facts (the ratio of hyper-relational facts), and the range of arity.

Method	**JF17K**						**WikiPeople**						**WD50K**					
	All entities			**All relations**			**All entities**			**All relations**			**All entities**			**All relations**		
	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
m-TransH	0.124	0.081	0.193		N/A		0.167	0.162	0.354		N/A		0.074	0.072	0.198		N/A	
RAE	0.307	0.211	0.486		N/A		0.193	0.175	0.388		N/A		0.132	0.118	0.243		N/A	
NaLP	0.364	0.287	0.519	0.827	0.729	0.896	0.327	0.265	0.449	0.875	0.838	0.929	0.223	0.162	0.337	0.775	0.702	0.896
NeuInfer	0.489	0.418	0.625	0.889	0.793	0.922	0.349	0.281	0.506	0.906	0.852	0.954	0.235	0.178	0.355	0.816	0.759	0.924
HINGE	0.519	0.445	0.682	0.903	0.865	0.959	0.367	0.305	0.488	0.935	0.895	0.976	0.245	0.181	0.362	0.878	0.812	0.963
ShrinkE	0.554	0.459	0.702		N/A		0.472	0.415	0.589		N/A		0.336	0.259	0.478		N/A	
GRAN	0.652	0.579	0.798	0.996	0.993	0.999	0.496	0.426	0.619	0.959	0.944	0.976	0.361	0.287	0.504	0.945	0.917	0.983
MSeaHKG	0.579	0.481	0.718	0.932	0.887	0.979	0.393	0.301	0.562	0.836	0.792	0.953	0.324	0.239	0.481	0.825	0.778	0.917
HyNT	0.634	0.558	0.783	0.992	0.988	0.995	0.457	0.376	0.597	0.948	0.928	0.973	0.337	0.271	0.464	0.907	0.881	0.948
HyperFormer	0.651	0.587	0.784		N/A		0.471	0.356	0.645		N/A		0.365	0.285	0.514		N/A	
StarE	0.584	0.504	0.741		N/A		0.394	0.290	0.593		N/A		0.315	0.240	0.458		N/A	
HyT	0.592	0.513	0.750		N/A		0.399	0.298	0.588		N/A		0.314	0.241	0.453		N/A	
QUAD	0.585	0.504	0.747		N/A		0.379	0.272	0.583		N/A		0.316	0.245	0.451		N/A	
HAHE	0.657	0.585	0.798	0.996	0.994	0.999	0.495	0.421	0.623	0.959	0.944	0.977	0.379	0.305	0.521	0.940	0.914	0.977
RAM	0.394	0.328	0.572		N/A		0.461	0.402	0.569		N/A		0.287	0.226	0.425		N/A	
tNaLP	0.370	0.292	0.528	0.834	0.733	0.906	0.333	0.272	0.457	0.886	0.842	0.937	0.230	0.169	0.344	0.789	0.715	0.912
sHINGE	0.528	0.459	0.701	0.918	0.876	0.973	0.372	0.312	0.499	0.947	0.905	0.984	0.248	0.185	0.370	0.885	0.819	0.971
DHGE	0.556	0.467	0.718	0.927	0.884	0.979	0.457	0.406	0.572	0.918	0.872	0.964	0.305	0.231	0.501	0.896	0.847	0.958
HyperCL+StarE	0.602	0.528	0.768		N/A		0.415	0.309	0.618		N/A		0.336	0.266	0.487		N/A	
HyperCL+HyT	0.617	0.534	0.778		N/A		0.419	0.316	0.615		N/A		0.338	0.263	0.488		N/A	
HyperCL+QUAD	0.605	0.529	0.776		N/A		0.396	0.288	0.605		N/A		0.339	0.268	0.498		N/A	
HyperCL+HAHE	**0.673**	**0.604**	**0.818**	**0.997**	**0.995**	**0.999**	**0.509**	**0.437**	**0.644**	**0.963**	**0.947**	**0.984**	**0.395**	**0.321**	**0.539**	**0.956**	**0.930**	**0.993**

Table 2: Overall link prediction performance (*All entities* and *All relations*). "N/A" denotes the case that the method cannot be applied to the task (namely m-TransH, RAE, ShrinkE, StarE, HyT, QUAD, and RAM are unable to predict missing relations). "All entities" means the link prediction over all predictable positions of entities when applicable. Thus, the reported results of StarE, HyT, and QUAD are head/tail predictions.
All baselines are implemented in our environment with their original hyperparameter settings.

(Guan et al., 2021); **sHINGE** (Lu et al., 2023b); **DHGE** (Luo et al., 2023a). Detailed descriptions of baselines are in Appendix A.

Among these baselines, we instantiate our HyperCL with four state-of-the-art encoder-decoder techniques StarE, HyT, QUAD, and HAHE to validate its effectiveness.

5.1.3 Evaluation Metrics

Link prediction is a typical task for evaluating the performance of KG embedding. For the missing entity (or relation) in a test fact, a ranking list of entities (or relations) is predicted. In this ranking list, in addition to the ground truth from the test fact, other entities (or relations) might also be true; we thus adopt the filtered setting (Bordes et al., 2013) to remove them from the ranking list. Mean Reciprocal Rank (MRR), Hits@1, and Hits@10 are used as evaluation metrics. We report the average results in predicting all entities and all relations separately.

5.1.4 Hyperparameters and Environment

Our HyperCL is trained for 300 epochs with early stopping on the benchmark hardware (Intel Xeon5320@2.20GHz, 256GB RAM@3200Hz, NVIDIA GeForce RTX 3090 24GB, Ubuntu 22.04). The hyperparameter settings for each dataset are shown in Appendix B and the training details are presented in Appendix C. Note that HyperCL only incurs a marginal computational overhead of up to 10.4% in the training time, compared to the baseline models (see Appendix C for details).

5.2 Overall Performance (RQ1)

Table 2 shows the overall performance on all three datasets and we highlight the best-performing result on each task and for each dataset. We observe that HyperCL+HAHE consistently outperforms all baselines, achieving 3.4% and 0.6% improvement on average over the best-performing baselines in predicting entities and relations, respectively.

Notably, we observe that our HyperCL framework consistently improves the link prediction performance of the corresponding base KG embedding methods StarE, HyT, QUAD, and HAHE in all tasks on all datasets, which demonstrates the effectiveness of HyperCL. In particular, HyperCL enhances the performance of the above four baselines with an average improvement of 3.1-7.4% across different datasets in predicting entities. An-

Method	**JF17K**						**WikiPeople**						**WD50K**					
	All entities			**All relations**			**All entities**			**All relations**			**All entities**			**All relations**		
	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
HyperCL+StarE	0.602	0.528	0.768		N/A		0.415	0.309	0.618		N/A		0.336	0.266	0.487		N/A	
w/o loss	0.589	0.510	0.750		N/A		0.401	0.295	0.607		N/A		0.322	0.246	0.467		N/A	
w/o concept	0.595	0.519	0.761		N/A		0.409	0.305	0.614		N/A		0.331	0.262	0.475		N/A	
w/o hierarchy	0.593	0.517	0.760		N/A		0.408	0.305	0.611		N/A		0.329	0.261	0.468		N/A	
HyperCL+HyT	0.617	0.534	0.778		N/A		0.419	0.316	0.615		N/A		0.338	0.263	0.488		N/A	
w/o loss	0.598	0.515	0.760		N/A		0.405	0.303	0.597		N/A		0.318	0.247	0.459		N/A	
w/o concept	0.610	0.530	0.765		N/A		0.414	0.311	0.607		N/A		0.335	0.260	0.479		N/A	
w/o hierarchy	0.612	0.531	0.771		N/A		0.415	0.313	0.608		N/A		0.335	0.261	0.477		N/A	
HyperCL+QUAD	0.605	0.529	0.776		N/A		0.396	0.288	0.605		N/A		0.339	0.268	0.498		N/A	
w/o loss	0.588	0.505	0.753		N/A		0.384	0.275	0.591		N/A		0.315	0.243	0.455		N/A	
w/o concept	0.599	0.525	0.766		N/A		0.394	0.287	0.598		N/A		0.336	0.267	0.489		N/A	
w/o hierarchy	0.596	0.520	0.764		N/A		0.395	0.288	0.601		N/A		0.334	0.258	0.494		N/A	
HyperCL+HAHE	0.673	0.604	0.818	0.997	0.995	0.999	0.509	0.437	0.644	0.963	0.947	0.984	0.395	0.321	0.539	0.956	0.930	0.993
w/o loss	0.659	0.587	0.800	0.996	0.995	0.998	0.497	0.425	0.621	0.957	0.941	0.978	0.383	0.309	0.522	0.945	0.920	0.981
w/o concept	0.664	0.594	0.806	0.997	0.995	0.999	0.507	0.436	0.640	0.962	0.946	0.983	0.387	0.313	0.531	0.952	0.927	0.986
w/o hierarchy	0.663	0.592	0.803	0.997	0.995	0.999	0.507	0.436	0.639	0.963	0.947	0.983	0.390	0.317	0.531	0.953	0.928	0.988

Table 3: Ablation results. Three HyperCL variants are combined with four encoder-decoder architecture-based baselines (namely StarE, HyT, QUAD, and HAHE).

other interesting observation is that while tNaLP and sHINGE are extensions of NaLP and HINGE with the consideration of the ontology of KGs, they only achieve a little improvement since they neglect the hierarchical structure of the ontology. Moreover, DHGE considers the hierarchical ontology but still yields suboptimal performance compared to HyperCL-enhanced baselines. This is attributed to its joint modeling of hyper-relational facts and the hierarchical ontology, which introduces the dominance issue as illustrated in Section 1. The above observation further verifies that properly incorporating ontology is critical for performance improvement, which is the key merit of HyperCL.

5.3 Ablation Study

The concept-aware contrastive loss and relation-aware GATs are two essential components of our HyperCL. We consider three variants to quantify their impact on link prediction performance and demonstrate their utility. Each variant is tuned individually with different optimal hyperparameters (see Appendix B for details) and the best prediction results are reported.

5.3.1 Impact of the Concept-Aware Contrastive Loss (RQ2)

We devise two variants of HyperCL to demonstrate the effectiveness of the concept-aware contrastive loss and its concept-aware batch selection strategy, respectively. The first variant removes the whole contrastive loss and directly integrates the ontology into the original encoder with relation-aware GATs, thus generating a joint model for knowledge embedding while maintaining the hierarchical structure of the ontology. This variant is denoted as **w/o loss**. The second variant replaces the concept-aware batch selection in the contrastive loss with random sampling-based batch selection, denoted as **w/o concept**.

Table 3 presents the results. We observe that the concept-aware contrastive loss and the concept-aware batch selection strategy both contribute to the link prediction improvement. In particular, the concept-aware contrastive loss enhances the performance of four baselines with an average improvement of 2.5-5.8% across different datasets, accounting for 75% performance improvement of the complete HyperCL framework (compared to the average improvement of 3.1-7.4% by the complete HyperCL framework in Section 5.2). *This implies that the dominance issue is a key factor resulting in the suboptimal performance of joint learning models, which can be largely mitigated by our HyperCL.* In addition, the concept-aware batch selection strategy also improves the performance of baselines, implying that separating semantically similar entities indeed benefits contrastive learning.

5.3.2 Impact of the Hierarchical Ontology (RQ3)

We further devise a variant of HyperCL to verify the effectiveness of the hierarchical ontology, where only the most specific concepts are reserved while other deeper concepts are discarded. Thus, the ontology of KGs loses its hierarchical information, denoted by **w/o hierarchy**.

As shown in Table 3, the hierarchical structure of ontology facilitates the hyper-relational knowledge embedding of baselines and thus improves their link prediction performance. Moreover, we have further investigated the link prediction improvements and found that entities with deeper concepts benefit more from the hierarchical ontology. Specif-

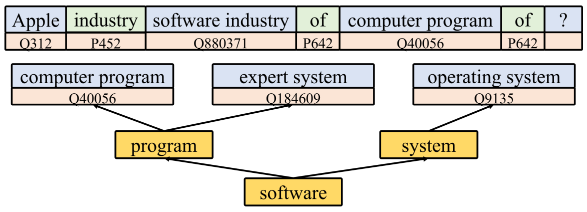

Figure 5: Case study on the importance of the hierarchical ontology on link prediction on a hyper-relational fact. The question mark denotes the missing value (entity). The lower part is a partial hierarchical ontology related to this fact.

ically, entities with 3 or more layers of concepts gain an average improvement of 5.7% across different datasets while entities with less than 3 layers of concepts only earn an average improvement of 0.8%. To intuitively demonstrate the advantage of modeling the hierarchical ontology, we conduct a case study of predicting the missing value of a hyper-relational fact in Fig. 5. The missing entity should possess similar semantics with *computer program* since they are affiliated with the same base triplet *(Apple, industry, software industry)* and have the common key *of*. The ground-truth entity is *operating system*, which is correctly predicted by HyperCL+HAHE. However, the w/o hierarchy variant answers *expert system* since it belongs to the same specific concept *program* with *computer program*. This implies that the loss of the hierarchical ontology narrows the view of HyperCL and makes it overlook possible candidates for link prediction, such as the entity *operating system* that belongs to the same high-level concept *software* as *computer program* does.

6 Conclusion

In this paper, we propose HyperCL, a universal contrastive learning framework for hyper-relational knowledge graph embedding, which considers the hierarchical structure of the ontology of KGs and alleviates the dominance issue of hyper-relational facts over the ontology. Experimental results show that HyperCL consistently boosts the performance of state-of-the-art baselines with an average improvement of 3.1-7.4% across three datasets, demonstrating the effectiveness of HyperCL.

In the future, we plan to further extend HyperCL to other KG embedding architectures and investigate graph augmentation techniques for concept-aware contrastive learning.

7 Limitations

As shown in the experiments, HyperCL is more powerful in predicting entities than relations. Besides the reason that the problem space of relations is usually much smaller than that of entities, this is also partially attributed to the current framework not taking into account the contrastive learning of relations. In the future, we will consider extending our framework to contrastive relations as a unified architecture.

8 Ethics Statement

This paper investigates the problem of knowledge graph link prediction, aiming at hyper-relational knowledge graph completion with ontology information to empower a wide range of web applications, such as question answering, recommender systems, and query expansion. The KG datasets used in this paper are all publicly available. Therefore, we believe it does not raise any ethical issues.

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

Chanyoung Chung, Jaejun Lee, and Joyce Jiyoung Whang. 2023. Representation learning on hyper-relational and numeric knowledge graphs with transformers. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 310–322.

Zijun Cui, Pavan Kapanipathi, Kartik Talamadupula, Tian Gao, and Qiang Ji. 2021. Type-augmented relation prediction in knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7151–7159.

Shimin Di and Lei Chen. 2021. Message function search for hyper-relational knowledge graph.

Bahare Fatemi, Perouz Taslakian, David Vazquez, and David Poole. 2021. Knowledge hypergraphs: prediction beyond binary relations. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 2191–2197.

Mikhail Galkin, Priyansh Trivedi, Gaurav Maheshwari, Ricardo Usbeck, and Jens Lehmann. 2020. Message passing for hyper-relational knowledge graphs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7346–7359.

Saiping Guan, Xiaolong Jin, Jiafeng Guo, Yuanzhuo Wang, and Xueqi Cheng. 2020. Neuinfer: Knowledge inference on n-ary facts. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 6141–6151.

Saiping Guan, Xiaolong Jin, Jiafeng Guo, Yuanzhuo Wang, and Xueqi Cheng. 2021. Link prediction on n-ary relational data based on relatedness evaluation. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):672–685.

Saiping Guan, Xiaolong Jin, Yuanzhuo Wang, and Xueqi Cheng. 2019. Link prediction on n-ary relational data. In *The world wide web conference*, pages 583–593.

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 *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1709–1719.

Zhiwei Hu, Víctor Gutiérrez-Basulto, Zhiliang Xiang, Ru Li, and Jeff Z Pan. 2023. Hyperformer: Enhancing entity and relation interaction for hyper-relational knowledge graph completion. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 803–812.

Roshni G Iyer, Yunsheng Bai, Wei Wang, and Yizhou Sun. 2022. Dual-geometric space embedding model for two-view knowledge graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 676–686.

Denis Krompaß, Stephan Baier, and Volker Tresp. 2015. Type-constrained representation learning in knowledge graphs. In *The Semantic Web-ISWC 2015: 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I 14*, pages 640–655. Springer.

Zhao Li, Xin Liu, Xin Wang, Pengkai Liu, and Yuxin Shen. 2023. Transo: a knowledge-driven representation learning method with ontology information constraints. *World Wide Web*, 26(1):297–319.

Yu Liu, Quanming Yao, and Yong Li. 2021. Role-aware modeling for n-ary relational knowledge bases. In *Proceedings of the Web Conference 2021*, pages 2660–2671.

Yuhuan Lu, Bangchao Deng, Weijian Yu, and Dingqi Yang. 2023a. Helios: Hyper-relational schema modeling from knowledge graphs. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 4053–4064.

Yuhuan Lu, Dingqi Yang, Pengyang Wang, Paolo Rosso, and Philippe Cudre-Mauroux. 2023b. Schema-aware hyper-relational knowledge graph embeddings for link prediction. *IEEE Transactions on Knowledge and Data Engineering*.

Haoran Luo, E Haihong, Ling Tan, Gengxian Zhou, Tianyu Yao, and Kaiyang Wan. 2023a. Dhge: dual-view hyper-relational knowledge graph embedding for link prediction and entity typing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 6467–6474.

Haoran Luo, Yuhao Yang, Yikai Guo, Mingzhi Sun, Tianyu Yao, Zichen Tang, Kaiyang Wan, Meina Song, Wei Lin, et al. 2023b. Hahe: Hierarchical attention for hyper-relational knowledge graphs in global and local level. *arXiv preprint arXiv:2305.06588*.

Guanglin Niu, Bo Li, Yongfei Zhang, Shiliang Pu, and Jingyang Li. 2020. Autoeter: Automated entity type representation for knowledge graph embedding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1172–1181.

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*, pages 1885–1896.

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 *Proceedings of the Web Conference 2021*, pages 845–856.

Harry Shomer, Wei Jin, Juanhui Li, Yao Ma, and Jiliang Tang. 2022. Learning representations for hyper-relational knowledge graphs. *arXiv preprint arXiv:2208.14322*.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Chenxu Wang, Xin Wang, Zhao Li, Zirui Chen, and Jianxin Li. 2023a. Hyconve: A novel embedding model for knowledge hypergraph link prediction with convolutional neural networks. In *Proceedings of the ACM Web Conference 2023*, pages 188–198.

Hao Wang, Yao Xu, Cheng Yang, Chuan Shi, Xin Li, Ning Guo, and Zhiyuan Liu. 2023b. Knowledge-adaptive contrastive learning for recommendation. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 535–543.

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*.

Jianfeng Wen, Jianxin Li, Yongyi Mao, Shini Chen, and Richong Zhang. 2016. On the representation and embedding of knowledge bases beyond binary relations. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 1300–1307.

Wikidata. 2022. http://wikidata.org/.

Ruobing Xie, Zhiyuan Liu, Maosong Sun, et al. 2016. Representation learning of knowledge graphs with hierarchical types. In *IJCAI*, volume 2016, pages 2965–2971.

Bo Xiong, Mojtaba Nayyer, Shirui Pan, and Steffen Staab. 2023. Shrinking embeddings for hyper-relational knowledge graphs. *arXiv preprint arXiv:2306.02199*.

Dingqi Yang, Bingqing Qu, Paolo Rosso, and Philippe Cudre-Mauroux. 2023. Fast and slow thinking: A two-step schema-aware approach for instance completion in knowledge graphs. *IEEE Transactions on Knowledge and Data Engineering*.

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 *Proceedings of the Joint Conference of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on Natural Language Processing of the AFNLP*.

Donghan Yu and Yiming Yang. 2021. Improving hyper-relational knowledge graph completion. *arXiv preprint arXiv:2104.08167*.

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

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*, pages 1185–1194.

Appendix

A Baseline Details

The first category includes model learning from hyper-relational facts only: **m-TransH** (Wen et al., 2016) models the interaction between entities involved in each n-ary fact; **RAE** (Zhang et al., 2018) extends m-TransH by explicitly considering the pairwise relatedness between entities in n-ary facts; **NaLP** (Guan et al., 2019) learns the relatedness between relation-entity pairs under the n-ary representation of hyper-relational facts; **NeuInfer** (Guan et al., 2020) models both primary triplet and its associated key-value pairs; **HINGE** (Rosso et al., 2020) captures both the triple-wise and quintuple-wise relatedness between elements in hyper-relational facts; **ShrinkE** (Xiong et al., 2023) models the primary triplets as a spatial-functional transformation from the head into a relation-specific box; **GRAN** (Wang et al., 2021) represents the hyper-relational facts as a heterogeneous graph and captures the inter-vertex interactions via self-attention mechanism; **MSeaHKG** (Di and Chen, 2021) develops a generic message-passing function to encode hyper-relational facts; **HyNT** (Chung et al., 2023) devises a context Transformer to learn the representations of numeric literals in either triplets or qualifiers; **HyperFormer** (Hu et al., 2023) encodes the local-level sequential information in hyper-relational facts with Transformers; **StarE** (Galkin et al., 2020) transforms a hyper-relational fact into a directed heterogeneous graph and extract the inter-vertex interaction using a GNN encoder; **HyT** (Yu and Yang, 2021) extends StarE substituting the graph encoder by a light-weight relation/entity embedding technique; **QUAD** (Shomer et al., 2022) also extends StarE by adopting two separate aggregators to encode the primary triplets and associated key-value pairs, respectively; **HAHE** (Luo et al., 2023b) adopts global and local hypergraph attention to represent hyper-relational facts. The second category includes model learning from both hyper-relational facts and ontology of KGs: **RAM** (Liu et al., 2021) captures the latent compatibility between the meta-relation and all involved entities by a pattern matrix; **tNaLP** (Guan et al., 2021) extends NaLP with the consideration of schema information; **sHINGE** (Lu et al., 2023b) models the hyper-relational schema information to enhance link prediction performance; **DHGE** (Luo et al., 2023a) jointly learns from hyper-relational facts and the hierarchical ontology of KGs.

B Hyperparameter Settings

Three key hyperparameters of HyperCL are the number of relation-aware graph attention layers L, the threshold of concept-aware batch selection β, and the temperature of contrastive loss τ_{cl}. We use the grid search method to identify the optimal hyperparameter setting for each HyperCL-combined model. The range of candidate values for hyperparameters L, β, and τ_{cl} are {1, 2, 3, 4, 5}, {1024, 2048, 4096, 8192, 12288}, and {0.5, 0.6, 0.7, 0.8,

Method	JF17K			WikiPeople			WD50K		
	L	β	τ_{cl}	L	β	τ_{cl}	L	β	τ_{cl}
HyperCL+StarE	2	4096	0.7	2	4096	0.7	2	4096	0.7
w/o loss	2	4096	0.7	2	4096	0.7	2	4096	0.7
w/o concept	2	4096	0.7	2	4096	0.7	2	4096	0.7
w/o hierarchy	2	4096	0.7	2	4096	0.7	2	4096	0.7
HyperCL+HyT	2	4096	0.7	2	8192	0.7	2	4096	0.7
w/o loss	2	4096	0.7	2	8192	0.7	2	4096	0.7
w/o concept	2	4096	0.7	2	8192	0.7	2	4096	0.7
w/o hierarchy	2	4096	0.7	2	8192	0.7	2	4096	0.7
HyperCL+QUAD	3	4096	0.6	3	8192	0.7	3	4096	0.6
w/o loss	3	4096	0.6	3	8192	0.7	3	4096	0.6
w/o concept	3	4096	0.6	3	8192	0.7	3	4096	0.6
w/o hierarchy	3	4096	0.6	3	8192	0.7	3	4096	0.6
HyperCL+HAHE	2	8192	0.7	2	8192	0.7	2	8192	0.7
w/o loss	2	8192	0.7	2	8192	0.7	2	8192	0.7
w/o concept	2	8192	0.8	2	8192	0.8	2	8192	0.7
w/o hierarchy	2	8192	0.8	2	8192	0.8	2	8192	0.7

Table 4: The optimal hyperparameter settings for all HyperCL-combined baselines and their variants.

0.9}, respectively. Afterward, the optimal hyperparameter setting of a model is selected from exhaustive hyperparameter combinations by comparing the link prediction performance under the different combinations. The final hyperparameter settings for all models are shown in Table 4.

C Training Details

All models in Table 4 are trained 300 epochs with early stopping on each dataset with their corresponding optimal hyperparameter settings. The average increased training time due to HyperCL on three datasets JF17K, WikiPeople, and WD50K is 1.2h, 2.9h, and 1.9h, respectively. In comparison, the training time of the most efficient baseline HAHE on JF17K, WikiPeople, and WD50K is 11.5h, 28.4h, and 22.5h, respectively. Therefore, HyperCL only incurs a marginal computational overhead (up to 10.4% in training time).