Structure Is All You Need: Structural Representation Learning on Hyper-Relational Knowledge Graphs

Jaejun Lee [1] **Joyce Jiyoung Whang** [1]

Abstract

Hyper-relational knowledge graphs (HKGs) enrich knowledge graphs by extending a triplet to a hyper-relational fact, where a set of qualifiers adds auxiliary information to a triplet. While many HKG representation learning methods have been proposed, they often fail to effectively utilize the HKG's structure. This paper demonstrates that thoroughly leveraging the structure of an HKG is crucial for reasoning on HKGs, and a purely structure-based representation learning method can achieve state-of-the-art performance on various link prediction tasks. We propose MAYPL, which learns to initialize representation vectors based on the structure of an HKG and employs an attentive neural message passing consisting of fact-level message computation and entity-centric and relation-centric aggregations, thereby computing the representations based solely on the structure. Due to its structure-driven learning, MAYPL can conduct inductive inferences on new entities and relations. MAYPL outperforms 40 knowledge graph completion methods in 10 datasets, compared with different baseline methods on different datasets to be tested from diverse perspectives.

1. Introduction

Knowledge graphs (KGs) provide a structured representation of human knowledge, which eases organizing the relationships between concepts or objects (Ji et al., 2022). In KGs, information is represented in the form of a triplet (Toutanova & Chen, 2015; Bordes et al., 2013), e.g., (`Finding Nemo`, `set in`, `Sydney`). Since the simple triplet form oversimplifies information, hyper-relational knowledge graphs (HKGs) have been recently proposed to add auxiliary details by adding qualifiers to a triplet (Rosso et al., 2020), where a qualifier consists of a relation and an entity, e.g., (`country`, `Australia`). In HKGs, each piece of information is represented as a hyper-relational fact defined as a triplet and its qualifiers, e.g., $\big($(`Finding Nemo`, `set in`, `Sydney`),{(`country`, `Australia`), (`state`, `New South Wales`)}$\big)$.

Reasoning and inferring new knowledge on HKGs necessitates understanding and exploiting intricate relationships among entities and relations. However, most existing studies on HKGs do not sufficiently utilize the graph structure of HKGs. For example, transformer-based methods (Wang et al., 2021; Yu & Yang, 2021; Chung et al., 2023) process each hyper-relational fact (referred to as a 'fact' for brevity) individually, failing to capture the interconnectedness between different facts. Some other recent methods (Hu et al., 2023) use simple one-hop neighbourhood information to learn representations on HKGs. While StarE (Galkin et al., 2020) proposes a graph neural network (GNN) layer for encoding a fact, it turns out that the proposed GNN encoding is redundant and does not critically affect the overall performance (Yu & Yang, 2021). Recently, HAHE (Luo et al., 2023) introduces another GNN-based encoding, but it does not consider relations and the positions of entities within each fact. Like this, existing HKG methods incorporate only limited structural information and fail to utilize the structures of HKGs effectively.

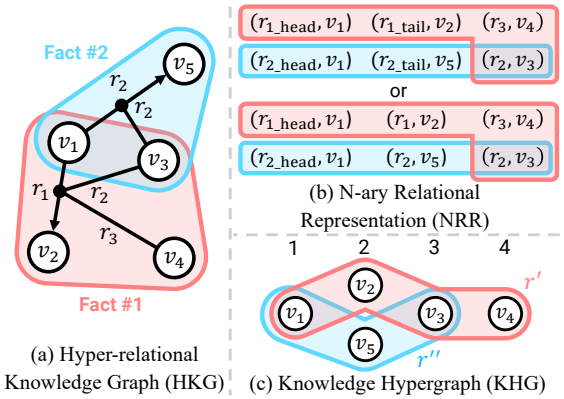
In this paper, we demonstrate that thoroughly leveraging an HKG's structure is crucial for successful reasoning on HKGs, and a purely structure-based representation learning can achieve state-of-the-art performance on a range of transductive and inductive link prediction tasks. We propose a structure-driven representation learning method, named MAYPL (Message pAssing framework for hYper-relational knowledge graph rePresentation Learning), that learns to initialize the representations of entities and relations based on their interconnections, co-occurrence, and positions within a fact. By compressing the information of which entities and relations are included and how they are connected within each fact, MAYPL computes fact-level messages, which are attentively aggregated to entities and relations. Along with the fact-level structure, the component-level connectivity is also considered in MAYPL's attentive neural message

[1]School of Computing, KAIST, Daejeon, South Korea. Correspondence to: Joyce Jiyoung Whang <jjwhang@kaist.ac.kr>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

Table 1: Comparison of MAYPL and existing methods.

	Know. Rep.				Ind. Inf.	
	KG	KHG	NRR	HKG	ent.	rel.
InGram (Lee et al., 2023b)	✓				✓	✓
RMPI (Geng et al., 2023)	✓				✓	✓
G-MPNN (Yadati, 2020)		✓			△	
HCNet (Huang et al., 2024)		✓			✓	
NaLP (Guan et al., 2019)			✓			
RAM (Liu et al., 2021b)			✓			
StarE (Galkin et al., 2020)				✓		
HAHE (Luo et al., 2023)				✓		
HyNT (Chung et al., 2023)				✓		
QBLP (Ali et al., 2021)				✓	✓	
MAYPL				✓	✓	✓



(a) Hyper-relational Knowledge Graph (HKG) (b) N-ary Relational Representation (NRR) (c) Knowledge Hypergraph (KHG)

Figure 1: HKGs cannot be transformed into NRRs or KHGs without losing information. Details are in Section 2.

passing mechanism. From initialization to link prediction, MAYPL utilizes only the structure of a given HKG. Since MAYPL learns how to compute, propagate, and aggregate messages over a given HKG's structure, MAYPL can also handle a new HKG that is different from a training HKG.

While real-world knowledge bases, such as Wikidata (Vrandečić & Krötzsch, 2014) and YAGO (Suchanek et al., 2007), take the form of HKGs, and many recently proposed methods directly handle HKGs (Galkin et al., 2020; Chung et al., 2023; Luo et al., 2023), some transform the original HKGs into knowledge hypergraphs (KHGs) (Yadati, 2020) or n-ary relational representations (NRRs) (Guan et al., 2019; Liu et al., 2021b), putting up with losing some information provided in the original HKGs. Table 1 compares MAYPL and other existing methods in terms of (i) the form of knowledge representations (Know. Rep.) and (ii) the capability of inductive inference (Ind. Inf.). For the inductive inference, we consider whether a method can handle new entities (ent.) and new relations (rel.). We provide the details of the compared methods in Section 2 and a complete comparison including more methods in Appendix A. MAYPL is the only HKG-handling method that can conduct inductive inference with new entities and relations. Since MAYPL learns how to compute the representations using an HKG's structure, what it learned can be seamlessly applied to a new HKG. Our main findings and contributions are:

- Employing an HKG's structure is crucial for HKG reasoning, and purely structure-based learning can successfully solve various link prediction problems.
- MAYPL is the first structure-oriented representation learning method for HKGs that can be applied in both transductive and inductive learning settings.
- By learning how to compute representations based solely on the structure of a given HKG, MAYPL can make inductive inferences where new entities and relations appear at inference time.
- MAYPL outperforms 40 different methods on 10 benchmark datasets, where each benchmark is introduced to test a different perspective of a method.[1]

2. Related Work

N-ary Relational Representations and Knowledge Hypergraphs NRRs (Guan et al., 2019; Liu et al., 2021b) and KHGs (Fatemi et al., 2020; Abboud et al., 2020) have been studied to add supplementary information to KGs. While each fact is represented using a primary triplet and a set of qualifiers in HKGs, NRRs (Wen et al., 2016) express each fact using a set of role-value pairs. On the other hand, in KHGs, each fact is converted into a tuple in the form of $r(v_1,\cdots,v_k)$, where the order of entities matters. Figure 1 shows the conversions from an HKG to an NRR or a KHG. In the original HKG shown in Figure 1(a), v_3 and v_5 are both incident to r_2 due to the facts $\big((v_1,r_1,v_2),\{(r_2,v_3),(r_3,v_4)\}\big)$ and $\big((v_1,r_2,v_5),\{(r_2,v_3)\}\big)$. However, on the NRR, shown in Figure 1(b), (v_1,r_2,v_5) is decomposed into (r_{2_head},v_1) and (r_{2_tail},v_5), and thus, v_5 is now incident to a different relation from v_3. Alternatively, one can decompose the triplet into (r_{2_head},v_1) and (r_2,v_5), in which case, one cannot identify the tail entity of r_2 since (r_2,v_3) also exists. In the KHG, shown in Figure 1(c), much information is also lost, e.g., the information that v_3 is incident to r_2 as a qualifier in both facts is lost. Therefore, an HKG cannot be transformed into an NRR or a KHG without losing information, which indicates that it is desirable to handle HKGs directly instead of converting them into other forms.

NaLP (Guan et al., 2019) and RAM (Liu et al., 2021b) have been proposed for NRRs, considering the relatedness between the pairs (Guan et al., 2023; 2019) or the role-aware modeling (Liu et al., 2021b). For KHGs, G-MPNN (Yadati,

[1]Our codes are available at https://github.com/bdi-lab/MAYPL/.

2020) and HCNet (Huang et al., 2024) have been proposed, which can partially solve inductive link prediction. While G-MPNN can handle new entities only one-hop distant from observed entities, HCNet can handle the case where all entities are new at inference time. Both G-MPNN and HCNet cannot handle new relations because they assume that all relations are observed during training time and learn representations specific to the observed relations.

Representation Learning on HKGs Many recently proposed HKG methods utilize transformers (Wang et al., 2021; Yu & Yang, 2021; Chung et al., 2023; Galkin et al., 2020), including StarE (Galkin et al., 2020) and HyNT (Chung et al., 2023). While some methods rarely take into account the graph structure of HKGs and focus on processing facts individually (Rosso et al., 2020; Guan et al., 2020; Yan et al., 2022; Xiong et al., 2023), others consider direct neighbors of entities (Hu et al., 2023) or a GNN-based encoding (Luo et al., 2023). For example, HAHE (Luo et al., 2023) updates entity representations using a GNN layer, though it ignores the information about relations and the positions of components. All these methods rely on learning representations of fixed entities and relations, which limits handling new entities or relations at inference time. Unlike these, MAYPL utilizes the fact-level, entity-level and relation-level connectivity structures along with their compositional and positional information, which enables inductive link prediction on an entirely new HKG with all new entities and relations.

Inductive KG Completion Inductive KG completion methods have been proposed to predict missing links in a new KG different from a training KG (Teru et al., 2020; Lee et al., 2023b; Geng et al., 2023). While most existing inductive KG methods assume that only entities are new and the relations are identical to the training set (Teru et al., 2020; Zhu et al., 2021), a few recent methods, such as InGram (Lee et al., 2023b) and RMPI (Geng et al., 2023), allow both entities and relations to differ from the training set, though they cannot handle HKGs. While QBLP (Ali et al., 2021) assumes new entities but not new relations in HKGs, it requires external text features of entities, and its inductive capability comes from these external features. In contrast, MAYPL can make inductive inferences on new entities and relations in HKGs without any extra features.

3. Problem Definitions

Definition 3.1 (Hyper-relational Knowledge Graph)**.** A hyper-relational knowledge graph is defined as $G = (\mathcal{V}, \mathcal{R}, \mathcal{H})$ where $\mathcal{V}$ is a set of entities, $\mathcal{R}$ is a set of relations, and $\mathcal{H}$ is a set of hyper-relational facts. A hyper-relational fact $h \in \mathcal{H}$ is defined as $h \in \mathcal{T} \times \mathcal{P}(\mathcal{Q})$ where $\mathcal{T} \subseteq \mathcal{V} \times \mathcal{R} \times \mathcal{V}$ is a set of triplets, $\mathcal{Q} \subseteq \mathcal{R} \times \mathcal{V}$ is a set of qualifiers, and $\mathcal{P}(\mathcal{Q})$ is the power set of $\mathcal{Q}$.

In a hyper-relational fact $\big((v_1, r_1, v_2), \{(r_2, v_3)\}\big)$, where (v_1, r_1, v_2) is a primary triplet and (r_2, v_3) is a qualifier, we indicate the position of each entity as follows: v_1 is a head entity, v_2 is a tail entity, and v_3 is a qualifier entity. Similarly, the position of a relation is specified as follows: r_1 is a primary relation and r_2 is a qualifier relation. Transductive link prediction on HKGs are described in Definition 3.2.

Definition 3.2 (Transductive Link Prediction on HKGs)**.** Given an HKG, $G = (\mathcal{V}, \mathcal{R}, \mathcal{H})$, $\mathcal{H}$ is decomposed into three pairwise disjoint sets, such as $\mathcal{H} = \mathcal{H}_{\text{tr}} \cup \mathcal{H}_{\text{val}} \cup \mathcal{H}_{\text{tst}}$, where $\mathcal{H}_{\text{tr}}$ is a training set, $\mathcal{H}_{\text{val}}$ is a validation set, and $\mathcal{H}_{\text{tst}}$ is a test set. A transductive link prediction on an HKG is to predict a missing entity in a hyper-relational fact $h \in \mathcal{H}_{\text{tst}}$.

On the other hand, in inductive link prediction, a model is trained on a training HKG and makes predictions on an inference HKG, as detailed in Definition 3.3.

Definition 3.3 (Inductive Link Prediction on HKGs)**.** Consider a training HKG, $G = (\mathcal{V}, \mathcal{R}, \mathcal{H})$ and an inference HKG, $\acute{G} = (\acute{\mathcal{V}}, \acute{\mathcal{R}}, \acute{\mathcal{H}})$, where $\mathcal{V} \neq \acute{\mathcal{V}}$ or $\mathcal{R} \neq \acute{\mathcal{R}}$. Note that $\mathcal{H}$ is a training set, and $\acute{\mathcal{H}}$ is decomposed into three pairwise disjoint sets, such as $\acute{\mathcal{H}} = \acute{\mathcal{H}}_{\text{fct}} \cup \acute{\mathcal{H}}_{\text{val}} \cup \acute{\mathcal{H}}_{\text{tst}}$, where $\acute{\mathcal{H}}_{\text{fct}}$ is a set of observed facts, $\acute{\mathcal{H}}_{\text{val}}$ is a validation set, and $\acute{\mathcal{H}}_{\text{tst}}$ is a test set. An inductive link prediction on an HKG is to predict a missing entity in $h \in \acute{\mathcal{H}}_{\text{tst}}$ using $\acute{\mathcal{H}}_{\text{fct}}$.

4. Structural Representation Learning on Hyper-Relational Knowledge Graphs

Figure 2 shows an overview of MAYPL consisting of a structure-driven initializer, attentive neural message passing, and link prediction. While the structure-driven initializer computes the initial representations of entities and relations based on their interconnected structure, the attentive neural message passing module encodes fact-level messages using the entity and relation representations based on the facts' composition information, which are attentively aggregated to entities and relations considering their positions. MAYPL predicts a missing query entity in an incomplete fact by selecting the entity with the highest similarity to the query.

4.1. Structure-driven Initializer of MAYPL

A simple message-passing-based initializer, referred to as a structure-driven initializer, is proposed, which exploits the interconnections, co-occurrence, and positions of entities and relations. In the initializer, the representation of an entity v is computed by considering the entities that co-occur with v in the same facts and the relations incident to v. Similarly, the representation of a relation r is computed based on the relations that co-occur with r in the same facts and the entities incident to r. In these processes, the positions of entities and relations within the facts and the positions of entities for their incident relations are considered. The

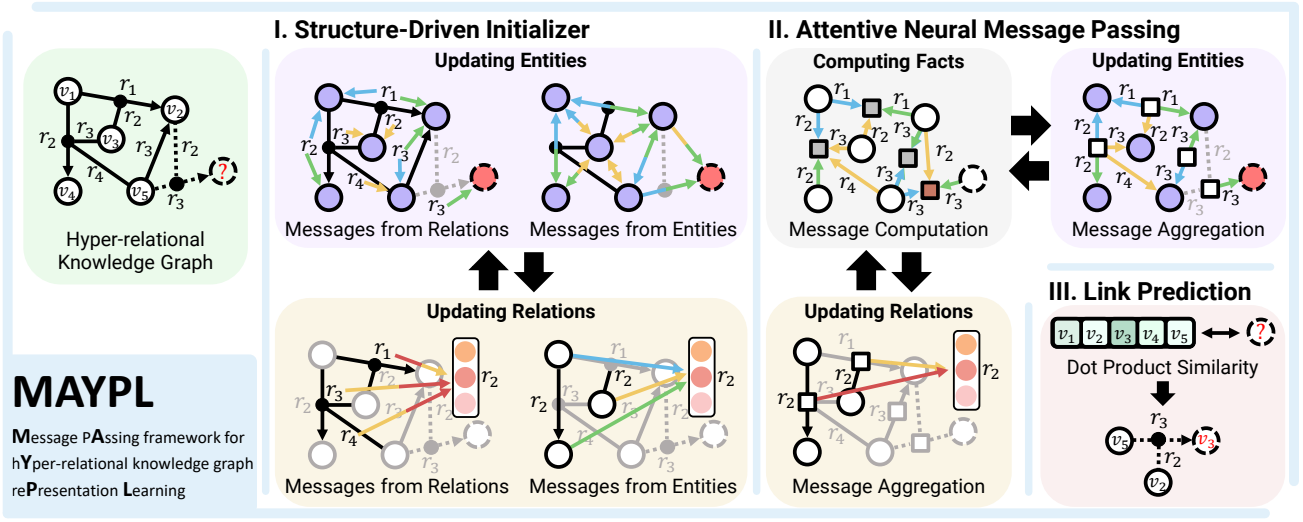


Figure 2: MAYPL consists of a structure-driven initializer, an attentive neural message passing, and link prediction. The structure-driven initializer initializes the entity and relation representations by exchanging messages among co-occurring entities and relations within the same facts and considering their positions. In attentive neural message passing, a fact's message is computed based on its entities and relations, and the entity and relation representations are updated by attentively aggregating the facts' messages to which they belong. Link prediction is performed using the dot product similarity.

initializer outputs the initial representations of entities denoted by $\mathbf{v}^{(0)} \in \mathbb{R}^{d_0}$ for $v \in \mathcal{V}$ and of relations denoted by $\mathbf{r}^{(0)} \in \mathbb{R}^{d_0}$ for $r \in \mathcal{R}$, where d_0 is the dimension of the initial representation.

Let the superscript $(\tilde{l})$ of a vector or a matrix indicate the $\tilde{l}$-th layer, where $\tilde{l} = 1, \cdots, \tilde{L}$ and $\tilde{L}$ is the total number of layers in the initializer. We set learnable vectors $\mathbf{v}_{\text{init}}, \mathbf{r}_{\text{init}} \in \mathbb{R}^{d_0}$ that are shared across all entities and relations, respectively. Let $\tilde{\mathbf{v}}^{(\tilde{l})} \in \mathbb{R}^{d_0}$ denote an entity representation vector of v in the initializer, and $\tilde{\mathbf{v}}^{(0)} = \mathbf{v}_{\text{init}}$. Similarly, let $\tilde{\mathbf{r}}^{(\tilde{l})} \in \mathbb{R}^{d_0}$ denote a relation representation vector of r in the initializer, and $\tilde{\mathbf{r}}^{(0)} = \mathbf{r}_{\text{init}}$. To indicate the position of an entity or a relation in a fact h, we define a position-indicator function $\lambda_h(z)$ that returns the position of z in h. If z is an entity, $\lambda_h(z)$ returns a head entity, tail entity, or qualifier entity. If z is a relation, $\lambda_h(z)$ returns a primary relation or qualifier relation. Similarly, $\tau_r(v)$ is defined to return the position of an entity v for a relation r: a head entity, tail entity, or qualifier entity.[2]

In the initializer, an entity v's representation is computed by aggregating the messages of the entities that co-occur with v in facts and the relations that are incident to v:

$$\tilde{\mathbf{v}}^{(\tilde{l})} = \mathbf{AGGR}^{(\tilde{l})}_{\texttt{ent}}\left(\{\tilde{\mathbf{u}}^{(\tilde{l}-1)} | u \in \mathcal{V}_v\}, \{\tilde{\mathbf{r}}^{(\tilde{l}-1)} | r \in \mathcal{R}_v\}\right)$$

where $\mathcal{V}_v$ is the set of entities that co-occur with v in facts and $\mathcal{R}_v$ is the set of relations incident to v. The message aggregation function for entities, $\mathbf{AGGR}^{(\tilde{l})}_{\texttt{ent}}$, is defined by

$$\mathbf{AGGR}^{(\tilde{l})}_{\texttt{ent}} = \frac{1}{\sum_{u \in \mathcal{V}_v} |\mathcal{H}_u \cap \mathcal{H}_v|} \sum_{u \in \mathcal{V}_v} \sum_{h \in \mathcal{H}_u \cap \mathcal{H}_v} \widetilde{\boldsymbol{U}}^{(\tilde{l})}_{\lambda_h(v)} \widetilde{\boldsymbol{W}}^{(\tilde{l})}_{\lambda_h(u)} \tilde{\mathbf{u}}^{(\tilde{l}-1)} + \frac{1}{|\mathcal{R}_v|} \sum_{r \in \mathcal{R}_v} \widetilde{\boldsymbol{A}}^{(\tilde{l})}_{\tau_r(v)} \tilde{\mathbf{r}}^{(\tilde{l}-1)}$$

where $\mathcal{H}_v$ is the set of facts that an entity v belongs to, $\widetilde{\boldsymbol{U}}^{(\tilde{l})}_{\lambda_h(v)}, \widetilde{\boldsymbol{W}}^{(\tilde{l})}_{\lambda_h(u)} \in \mathbb{R}^{d_0 \times d_0}$ are learnable matrices that are learned depending on the positions of the entities v and u in h, respectively, and $\widetilde{\boldsymbol{A}}^{(\tilde{l})}_{\tau_r(v)}$ is a learnable matrix that is learned depending on the position of v with respect to r.

Similarly, a relation r's representation is computed by aggregating the messages of relations that co-occur with r in facts and the entities that are incident to r:

$$\tilde{\mathbf{r}}^{(\tilde{l})} = \mathbf{AGGR}^{(\tilde{l})}_{\texttt{rel}}\left(\{\tilde{\mathbf{y}}^{(\tilde{l}-1)} | y \in \mathcal{R}_r\}, \{\tilde{\mathbf{v}}^{(\tilde{l}-1)} | v \in \mathcal{V}_r\}\right)$$

where $\mathcal{R}_r$ is the set of relations that co-occur with r in facts and $\mathcal{V}_r$ is the set of entities that are incident to r. The message aggregation function for relations, $\mathbf{AGGR}^{(\tilde{l})}_{\texttt{rel}}$, is

$$\mathbf{AGGR}^{(\tilde{l})}_{\texttt{rel}} = \frac{1}{\sum_{y \in \mathcal{R}_r} |\mathcal{H}_y \cap \mathcal{H}_r|} \sum_{y \in \mathcal{R}_r} \sum_{h \in \mathcal{H}_y \cap \mathcal{H}_r} \widehat{\boldsymbol{U}}^{(\tilde{l})}_{\lambda_h(r)} \widehat{\boldsymbol{W}}^{(\tilde{l})}_{\lambda_h(y)} \tilde{\mathbf{y}}^{(\tilde{l}-1)} + \frac{1}{|\mathcal{V}_r|} \sum_{v \in \mathcal{V}_r} \widehat{\boldsymbol{A}}^{(\tilde{l})}_{\tau_r(v)} \tilde{\mathbf{v}}^{(\tilde{l}-1)}$$

where $\mathcal{H}_r$ is the set of facts that a relation r belongs to, $\widehat{\boldsymbol{U}}^{(\tilde{l})}_{\lambda_h(r)}, \widehat{\boldsymbol{W}}^{(\tilde{l})}_{\lambda_h(y)} \in \mathbb{R}^{d_0 \times d_0}$ are learnable matrices that are

[2] While we assume that z is unique in h in $\lambda_h(z)$ for brevity, our implementation correctly returns the position of z even if z is not unique; the same also applies to $\tau_r(v)$. Refer to our codes.

learned depending on the positions of the relations r and y in h, respectively, and $\widehat{\boldsymbol{A}}^{(\tilde{l})}_{\tau_r(v)}$ is a learnable matrix that is learned depending on the position of v with respect to r.

Motivation and Principles Our structure-driven initializer is designed to compute the initial representations of entities and relations purely using the structure of an HKG, contributing to MAYPL's inductive inference capability. From a single learnable vector $\mathbf{v}_{\text{init}}$ shared across all entities at the beginning, $\widetilde{\mathbf{v}}^{(\tilde{l})}$ is updated for $v \in \mathcal{V}$ using $\mathbf{AGGR}^{(\tilde{l})}_{\texttt{ent}}$; what is learned during this process is how to aggregate messages from the co-occurred entities and incident relations of each entity. By introducing different learnable projection matrices for different positions of entities and relations, the position information is also considered in this aggregation. Similarly, for relations, $\mathbf{AGGR}^{(\tilde{l})}_{\texttt{rel}}$ learns to aggregate messages from the co-occurred relations and incident entities of each relation. Both $\mathbf{AGGR}^{(\tilde{l})}_{\texttt{ent}}$ and $\mathbf{AGGR}^{(\tilde{l})}_{\texttt{rel}}$ can be directly applied to a new HKG that is different from a training HKG since $\mathbf{AGGR}^{(\tilde{l})}_{\texttt{ent}}$ and $\mathbf{AGGR}^{(\tilde{l})}_{\texttt{rel}}$ have learned how to aggregate messages on an HKG using its structure. After $\widetilde{L}$ layers, $\widetilde{\mathbf{v}}^{(\widetilde{L})}$ is computed for $v \in \mathcal{V}$ and $\widetilde{\mathbf{r}}^{(\widetilde{L})}$ is computed for $r \in \mathcal{R}$, which are fed into the following attentive neural message passing module.

4.2. Attentive Neural Message Passing

After initializing the representations of entities and relations, MAYPL updates the entity and relation representations by message computation and aggregation. Specifically, MAYPL considers (i) which entities and relations comprise a fact when computing a fact's message, (ii) which facts an entity belongs to when updating an entity representation, and (iii) which facts a relation belongs to when updating a relation representation. In this process, not only these composition and connectivity information but also the positions of the entities and relations within each fact are considered. Let $\mathbf{v}^{(l)} \in \mathbb{R}^{d_l}$ denote an entity representation of v, where d_l is the dimension at the l-th layer, $l = 1, \cdots, L$, and L is the total number of layers. Let $\mathbf{r}^{(l)} \in \mathbb{R}^{d_l}$ denote a relation representation of r. Note that $\mathbf{v}^{(0)} = \widetilde{\mathbf{v}}^{(\widetilde{L})}$ and $\mathbf{r}^{(0)} = \widetilde{\mathbf{r}}^{(\widetilde{L})}$.

Computing Facts' Messages

Given a fact $h \in \mathcal{H}$, let us decompose it as a set of its relation-entity pairs. For example, $\big((v_1, r_1, v_2), \{(r_2, v_3)\}\big)$ is decomposed by $\{(r_1, v_1), (r_1, v_2), (r_2, v_3)\}$. Let $p = (r, v) \in (\mathcal{R} \times \mathcal{V})_h$ denote a pair of a relation r and an entity v in h, where v is incident to r and $(\mathcal{R} \times \mathcal{V})_h$ is the set of relation-entity pairs for h. Then, a message $\mathbf{p}^{(l)}$ of p is computed by

$$\mathbf{p}^{(l)} = \boldsymbol{P}^{(l)}_{\lambda_h(v)} \left(\left(\boldsymbol{W}^{(l)}_{\lambda_h(v)} \mathbf{v}^{(l-1)} \right) \odot \left(\boldsymbol{U}^{(l)}_{\lambda_h(v)} \mathbf{r}^{(l-1)} \right) \right)$$

where $\boldsymbol{P}^{(l)}_{\lambda_h(v)} \in \mathbb{R}^{d_l \times d_l}$, $\boldsymbol{W}^{(l)}_{\lambda_h(v)}, \boldsymbol{U}^{(l)}_{\lambda_h(v)} \in \mathbb{R}^{d_l \times d_{l-1}}$ are learnable projection matrices which are learned depending on the position of v in h, and $\odot$ is the Hadamard product. By considering $\mathbf{p}^{(l)}$ for $\forall p \in (\mathcal{R} \times \mathcal{V})_h$, a message of h is computed as follows:

$$\mathbf{h}^{(l)} = \frac{1}{|(\mathcal{R} \times \mathcal{V})_h|} \sum_{p \in (\mathcal{R} \times \mathcal{V})_h} \mathbf{p}^{(l)} \tag{1}$$

which implies that a message of h is computed by the position-aware aggregation of its relation-entity pairs.

Update of Entity Representations

For an entity $v \in \mathcal{V}$, let us consider a pair of its incident relation and the corresponding fact. Let $q = (r, h)_v \in (\mathcal{R} \times \mathcal{H})_v$ denote a pair of a relation r and a fact h where v is incident to r in h and $(\mathcal{R} \times \mathcal{H})_v$ is the set of relation-fact pairs for v. A message $\mathbf{q}^{(l)}$ of q is computed by

$$\mathbf{q}^{(l)} = \boldsymbol{Q}^{(l)}_{\lambda_h(v)} \left(\mathbf{h}^{(l)} \odot \left(\boldsymbol{A}^{(l)}_{\lambda_h(v)} \mathbf{r}^{(l-1)} \right) \right) \tag{2}$$

where $\boldsymbol{Q}^{(l)}_{\lambda_h(v)} \in \mathbb{R}^{d_l \times d_l}$, $\boldsymbol{A}^{(l)}_{\lambda_h(v)} \in \mathbb{R}^{d_l \times d_{l-1}}$ are learnable matrices learned depending on the position of v in h. Note that $\mathbf{q}^{(l)}$ is a message for v, which reflects the representations of its incident relation and the corresponding fact. Finally, an entity representation $\mathbf{v}^{(l)}$ of v is computed by an attentive aggregation of $\mathbf{q}^{(l)}$ for $\forall q \in (\mathcal{R} \times \mathcal{H})_v$ as follows:

$$\mathbf{v}^{(l)} = \sum_{q \in (\mathcal{R} \times \mathcal{H})_v} \alpha^{(l)}_{q,v} \boldsymbol{B}^{(l)} \mathbf{q}^{(l)}, \tag{3}$$

$$\alpha^{(l)}_{q,v} = \frac{\texttt{exp}\left(\mathbf{a}^{(l)} \cdot \sigma \left(\boldsymbol{Q}^{(l)} \mathbf{v}^{(l-1)} + \boldsymbol{K}^{(l)} \mathbf{q}^{(l)} \right) \right)}{\sum_{k \in (\mathcal{R} \times \mathcal{H})_v} \texttt{exp}\left(\mathbf{a}^{(l)} \cdot \sigma \left(\boldsymbol{Q}^{(l)} \mathbf{v}^{(l-1)} + \boldsymbol{K}^{(l)} \mathbf{k}^{(l)} \right) \right)}$$

where $\boldsymbol{Q}^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}, \boldsymbol{K}^{(l)}, \boldsymbol{B}^{(l)} \in \mathbb{R}^{d_l \times d_l}$ are learnable projection matrices, $\mathbf{a}^{(l)} \in \mathbb{R}^{d_l}$ is a learnable vector, and $\sigma(\cdot)$ is an activation function. Note that $\alpha^{(l)}_{q,v}$ is the relative importance of q to v among all relation-fact pairs of v.

Update of Relation Representations

Similarly, for a relation $r \in \mathcal{R}$, we consider a pair of its incident entity and the corresponding fact. Let $\bar{q} = (v, h)_r \in (\mathcal{V} \times \mathcal{H})_r$ denote a pair of an entity v and a fact h where r is incident to v in h and $(\mathcal{V} \times \mathcal{H})_r$ is the set of entity-fact pairs for r. Then, a message $\bar{\mathbf{q}}^{(l)}$ of $\bar{q}$ is computed by

$$\bar{\mathbf{q}}^{(l)} = \bar{\boldsymbol{Q}}^{(l)}_{\lambda_h(r)} \left(\mathbf{h}^{(l)} \odot \left(\bar{\boldsymbol{A}}^{(l)}_{\lambda_h(v)} \mathbf{v}^{(l-1)} \right) \right) \tag{4}$$

where $\bar{\boldsymbol{Q}}^{(l)}_{\lambda_h(r)} \in \mathbb{R}^{d_l \times d_l}$ is a learnable matrix learned depending on the position of r in h, and $\bar{\boldsymbol{A}}^{(l)}_{\lambda_h(v)} \in \mathbb{R}^{d_l \times d_{l-1}}$ is a learnable matrix learned depending on the position of

v in h. Note that $\bar{\mathbf{q}}^{(l)}$ is a message for r, which reflects the representations of its incident entity and the corresponding fact. Finally, a relation representation $\mathbf{r}^{(l)}$ of r is computed by an attentive aggregation of $\bar{\mathbf{q}}^{(l)}$ for $\forall \bar{q} \in (\mathcal{V} \times \mathcal{H})_r$:

$$\mathbf{r}^{(l)} = \sum_{\bar{q} \in (\mathcal{V} \times \mathcal{H})_r} \bar{\alpha}^{(l)}_{\bar{q},r} \bar{\boldsymbol{B}}^{(l)} \bar{\mathbf{q}}^{(l)}, \tag{5}$$

$$\bar{\alpha}^{(l)}_{\bar{q},r} = \frac{\exp\left(\bar{\mathbf{a}}^{(l)} \cdot \sigma\left(\bar{\boldsymbol{Q}}^{(l)} \mathbf{r}^{(l-1)} + \overline{\boldsymbol{K}}^{(l)} \bar{\mathbf{q}}^{(l)}\right)\right)}{\sum_{\bar{k} \in (\mathcal{V} \times \mathcal{H})_r} \exp\left(\bar{\mathbf{a}}^{(l)} \cdot \sigma\left(\bar{\boldsymbol{Q}}^{(l)} \mathbf{r}^{(l-1)} + \overline{\boldsymbol{K}}^{(l)} \bar{\mathbf{k}}^{(l)}\right)\right)}$$

where $\bar{\boldsymbol{Q}}^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$, $\overline{\boldsymbol{K}}^{(l)}, \bar{\boldsymbol{B}}^{(l)} \in \mathbb{R}^{d_l \times d_l}$ are learnable projection matrices, $\bar{\mathbf{a}}^{(l)} \in \mathbb{R}^{d_l}$ is a learnable vector, and $\sigma(\cdot)$ is an activation function. Note that $\bar{\alpha}^{(l)}_{\bar{q},r}$ is the relative importance of $\bar{q}$ to r among all entity-fact pairs of r.

Summary When encoding a fact in Eq.(1), each fact is decomposed into a set of relation-entity pairs that comprise the fact, and the fact's message is computed by aggregating the pairs' representations. Each entity's representation is updated by attentively aggregating the facts' messages to which the entity belongs and the representations of the corresponding relations, shown in Eq.(2) and Eq.(3). Each relation's representation is updated by attentively aggregating the facts' messages to which the relation belongs and the representations of the corresponding entities, shown in Eq.(4) and Eq.(5). In all these computations and updates, the positions of entities and relations are considered.

4.3. Link Prediction on HKGs

For link prediction during training, an incomplete fact is made by dropping an entity from a complete fact in a training set. For an incomplete fact h, let x denote the missing entity, also called a query entity, and assume that x is incident to relation r. The initial representation of x is computed by the structure-driven initializer: starting from $\widetilde{\mathbf{x}}^{(0)} = \mathbf{v}_{\text{init}}$, it is updated by $\widetilde{\mathbf{x}}^{(\tilde{l})} = \mathbf{AGGR}^{(\tilde{l})}_{\text{ent}}(\{\widetilde{\mathbf{v}}^{(\tilde{l}-1)} | v \in \mathcal{V}_x\}, \{\widetilde{\mathbf{r}}^{(\tilde{l}-1)}\})$, where $\tilde{l} = 1, \cdots, \tilde{L}$ and $\mathcal{V}_x$ is the set of entities that co-occur with x in h. Then, given $\mathbf{x}^{(0)} = \widetilde{\mathbf{x}}^{(\tilde{L})}$, the attentive neural message passing is applied by computing the message of h by $\mathbf{h}^{(l)} = \frac{1}{|(\mathcal{R} \times \mathcal{V})_h|} \sum_{p \in (\mathcal{R} \times \mathcal{V})_h} \mathbf{p}^{(l)}$, followed by $\mathbf{x}^{(l)} = \boldsymbol{B}^{(l)} \left(\boldsymbol{Q}^{(l)}_{\lambda_h(x)} \left(\mathbf{h}^{(l)} \odot \left(\boldsymbol{A}^{(l)}_{\lambda_h(x)} \mathbf{r}^{(l-1)}\right)\right)\right)$, where $l = 1, \cdots, L$ and all the model parameters are explained in Section 4.2. By computing the dot product similarity between $\mathbf{x}^{(L)}$ and each entity representation, MAYPL predicts the missing entity as the entity with the highest similarity; the cross-entropy is used for the training loss.

The process of link prediction at inference time is identical to what is described above, except each incomplete fact is made from a test set instead of a training set. In inductive link prediction, an inference HKG, $\acute{G}$, differs from a training HKG, G (Definition 3.3). At inference time, MAYPL utilizes the fact set $\acute{\mathcal{H}}_{\text{fct}}$ of $\acute{G}$ to initialize the representations of entities and relations (Section 4.1), update them (Section 4.2), and perform link prediction (Section 4.3). Note that all the model parameters needed in this process are learned during training; the learned parameters can be directly applied to the inference HKG that can be different from a training HKG because, from initialization to link prediction, what MAYPL learns is how to compute and aggregate messages purely based on the structure of an HKG.

5. Experiments

We use diverse baseline methods presented in Appendix A and different benchmark datasets detailed in Appendix B. All results of baselines are obtained from the baselines' original papers or from the papers that introduced the datasets (Galkin et al., 2020; Guan et al., 2019; Ali et al., 2021; Yadati, 2020; Lee et al., 2023b). In Appendix C, we present which methods are run on which datasets and which results are obtained from which papers. In all tables of experimental results, '-' indicates that no result is available for a model on a benchmark from either the model's original paper or the benchmark's paper; the best result is **boldfaced**, the second-best result is underlined, and the third-best result is *italic*. The implementation and experimental details of MAYPL are described in Appendix D. We measure link prediction performance using three standard metrics: Mean Reciprocal Rank (MRR), Hit10, and Hit1.

5.1. Transductive Link Prediction on HKGs

Table 2 shows the transductive link prediction results on HKGs using WD50K (Galkin et al., 2020) and WikiPeople$^-$ (Wang et al., 2021) datasets, where all methods are trained using the training set and evaluated on the validation set for hyperparameter tuning and then trained again using the union of the training and validation sets with the selected hyperparameters and tested to predict missing entities on the facts in the test set. Note that this is the standard evaluation protocol for these datasets (Galkin et al., 2020; Wang et al., 2021), and we also follow this protocol. Models predict entities in the primary triplets (Tri) or all entities in facts (All). Also, Table 3 shows the results on WikiPeople (Guan et al., 2019), where the models predict all entities. Overall, MAYPL outperforms the baselines on these transductive link prediction tasks.

5.2. Inductive Link Prediction on KGs

Table 4 shows inductive link prediction results on KGs using NL-50, WK-50, and FB-50 datasets, where all entities are new, and half of the triplets include new relations at an inference KG (Lee et al., 2023b). We observe that MAYPL outperforms the baseline methods with a considerable gap

Table 2: Transductive link prediction on WD50K and WikiPeople$^-$. Models predict entities in the primary triplets (Tri) or all entities in facts (All).

		WD50K			WikiPeople$^-$		
		MRR	Hit10	Hit1	MRR	Hit10	Hit1
Tri	NaLP	0.177	0.264	0.131	-	-	-
	HINGE	0.243	0.377	0.176	-	-	-
	StarE	0.349	0.496	0.271	0.491	0.648	0.398
	GRAN	-	-	-	*0.503*	0.620	*0.438*
	Hy-Transformer	0.356	0.498	0.281	0.501	0.634	0.426
	HyperFormer	*0.366*	*0.514*	*0.288*	0.473	*0.646*	0.361
	HAHE	0.368	0.516	0.291	0.509	0.639	**0.447**
	ShrinkE	0.345	0.482	0.275	0.485	0.601	0.431
	HyNT	0.357	0.501	0.281	-	-	-
	MAYPL	**0.381**	**0.544**	**0.297**	**0.519**	**0.657**	0.444
All	HAHE	0.402	0.546	**0.327**	0.495	0.631	0.420
	HyNT	*0.383*	*0.527*	*0.308*	-	-	-
	MAYPL	**0.411**	**0.572**	0.326	**0.521**	**0.659**	**0.446**

Table 3: Transductive link prediction on all entities in facts in WikiPeople.

	WikiPeople		
	MRR	Hit10	Hit1
RAE	0.172	0.320	0.102
NaLP	0.338	0.466	0.272
HypE	0.282	0.487	0.148
NeuInfer	0.350	0.467	0.282
RAM	0.380	0.539	0.279
S2S	0.372	0.533	0.277
PolygonE	0.431	0.568	0.334
HyConvE	0.362	0.501	0.275
MSeaKG	0.392	0.553	0.290
tNaLP	0.350	0.471	0.288
HCNet	0.421	0.565	0.344
HyCubE	*0.448*	0.592	*0.368*
HJE	0.450	*0.582*	0.375
MAYPL	**0.488**	**0.635**	**0.405**

Table 4: Inductive link prediction performance on KGs using NL-50, WK-50, and FB-50 datasets.

	NL-50			WK-50			FB-50		
	MRR	Hit10	Hit1	MRR	Hit10	Hit1	MRR	Hit10	Hit1
GraIL (Teru et al., 2020)	0.162	0.288	0.104	-	-	-	-	-	-
CoMPILE (Mai et al., 2021)	0.194	0.330	0.125	-	-	-	-	-	-
SNRI (Xu et al., 2022)	0.130	0.187	0.095	-	-	-	-	-	-
INDIGO (Liu et al., 2021a)	0.167	0.217	0.134	-	-	-	-	-	-
RMPI (Geng et al., 2023)	0.185	0.307	0.109	-	-	-	-	-	-
CompGCN (Vashishth et al., 2020)	0.003	0.005	0.000	0.003	0.002	0.001	0.004	0.006	0.002
NodePiece (Galkin et al., 2022)	0.037	0.079	0.013	0.008	0.013	0.002	0.021	0.048	0.006
NeuralLP (Yang et al., 2017)	0.101	0.190	0.064	0.025	0.054	0.007	0.088	0.184	0.043
DRUM (Sadeghian et al., 2019)	0.107	0.193	0.070	0.017	0.046	0.002	0.101	0.191	0.061
BLP (Daza et al., 2021)	0.041	0.093	0.011	0.041	0.092	0.013	0.078	0.156	0.037
QBLP (Ali et al., 2021)	0.048	0.097	0.020	0.035	0.080	0.011	0.071	0.147	0.030
NBFNet (Zhu et al., 2021)	0.225	0.346	0.161	0.062	0.105	*0.036*	0.130	0.259	0.071
RED-GNN (Zhang & Yao, 2022)	0.179	0.280	0.115	0.058	0.093	0.033	0.129	0.251	0.072
InGram (Lee et al., 2023b)	0.281	0.453	0.193	*0.068*	0.135	0.034	0.117	0.218	0.067
HyRel (Yang et al., 2024)	0.321	0.520	0.222	*0.068*	*0.138*	*0.036*	*0.178*	*0.333*	*0.101*
InKGE-EIPR (Mu et al., 2024)	*0.315*	**0.529**	*0.217*	0.076	0.164	0.038	0.204	**0.376**	0.123
MAYPL	**0.343**	*0.508*	**0.262**	**0.109**	**0.230**	**0.054**	**0.205**	0.361	**0.129**

on NL-50 and WK-50 and is comparable to the best baseline in FB-50. The full results on all 12 datasets from (Lee et al., 2023b) are reported in Table 18 in Appendix E. The results show that MAYPL's structure-oriented representation learning is effective in handling new entities and relations.

5.3. Inductive Link Prediction on HKGs

Table 5 shows inductive link prediction results on HKGs using WD20K(100)v1 (Ali et al., 2021) and WD20K(100)v2 (Ali et al., 2021) datasets, where models predict entities in the primary triplets. Also, Table 6 shows the results on WP-IND (Yadati, 2020) and MFB-IND (Yadati, 2020), where models predict all entities in facts; the baselines' performances are reported using Hit3 instead of Hit10 in (Yadati, 2020), and thus, MAYPL's performance is also measured using Hit3. MAYPL significantly outperforms all methods in all metrics on these datasets, showing MAYPL's remarkable inductive capability on HKGs.

5.4. Ablation Studies and Qualitative Analysis

Table 7 shows ablation studies of MAYPL on WikiPeople$^-$ (WP$^-$), WK-50, and WD20K(100)v2 (WDv2), where MRRs are reported. We consider the following variations: (i) introduce learnable vectors for individual entities and relations instead of using the structure-driven initializer in Section 4.1, (ii) remove the attentive neural message passing in Section 4.2, (iii) let all facts equally contribute to the update of entities and relations without considering their

Table 5: Inductive link prediction performance on the primary triplets in WD20K(100)v1 and WD20K(100)v2.

	WD20K(100)v1			WD20K(100)v2		
	MRR	Hit10	Hit1	MRR	Hit10	Hit1
BLP	0.057	0.123	0.019	0.040	0.092	0.015
CompGCN	0.104	0.184	*0.058*	0.026	0.053	0.007
StarE	0.113	*0.213*	0.062	*0.051*	0.129	0.014
QBLP (w/o qual.)	0.055	0.126	0.015	0.049	0.097	*0.026*
QBLP (w/ qual.)	*0.107*	0.245	0.040	0.067	*0.120*	0.035
MAYPL	**0.486**	**0.662**	**0.396**	**0.298**	**0.518**	**0.195**

Table 6: Inductive link prediction performance on all entities of facts in WP-IND and MFB-IND.

	WP-IND			MFB-IND		
	MRR	Hit3	Hit1	MRR	Hit3	Hit1
HGNN	0.072	0.112	0.045	0.121	0.114	0.076
HyperGCN	0.075	0.111	0.049	0.118	0.117	0.074
G-MPNN-sum	0.177	0.191	0.108	0.124	0.123	0.071
G-MPNN-max	*0.200*	*0.214*	*0.125*	*0.268*	*0.283*	*0.191*
HCNet	0.414	0.451	0.352	0.368	0.417	0.223
MAYPL	**0.468**	**0.529**	**0.363**	**0.550**	**0.582**	**0.469**

Table 7: Ablation Studies of MAYPL. MRRs are reported.

		WP$^-$	WK-50	WDv2
(i)	Learn $\mathbf{v}^{(0)}, \mathbf{r}^{(0)}$	0.388	0.061	0.112
(ii)	$l = 0$	0.109	0.029	0.144
(iii)	$\mathbf{a}^{(l)} = \bar{\mathbf{a}}^{(l)} = \mathbf{0}$	0.504	0.086	0.210
(iv)	$\mathbf{r}^{(l)} = \bar{\boldsymbol{A}}^{(l)}\mathbf{r}^{(l-1)}$	0.519	0.062	0.248
(v)	Rep. Hadamard Prod. w/ Concat.	0.518	0.090	0.265
(vi)	Diff. Para. for $\mathbf{x}$ and $\mathbf{v}$	0.520	0.081	0.179
	MAYPL	**0.521**	**0.109**	**0.298**

Table 8: Top 3 similar entities/relations to a target in WP$^-$ based on the initial representations returned by the structure-driven initializer and the final representations of MAYPL.

Target		Initializer	Final Rep.
`Vancouver`	1	`Venice`	`Toronto`
	2	`Budapest`	`Victoria`
	3	`Gothenburg`	`Ottawa`
`computer scientist`	1	`psychologist`	`mathematician`
	2	`professeur des universités`	`programmer`
	3	`inventor`	`AI researcher`
`family`	1	`death type`	`sibling`
	2	`citizen of`	`family name`
	3	`ethnic group`	`father`

relative importance in Section 4.2, (iv) compute relation representations by a simple projection instead of the relation aggregation in Section 4.2, (v) replace the Hadamard products in computing messages of facts, entities, and relations with concatenation in Section 4.2 and Section 4.3, (vi) use different parameters to compute the query representation and the other entity representations in Section 4.3. While varying some modules more critically affects performance in inductive settings (WK-50 and WDv2) than the transductive setting (WP$^-$), the variations degrade the performance, validating each module's contribution in MAYPL.[3]

[3]More detailed explanations about these ablation studies are available in Appendix F. Also, Appendix G provides additional case studies about MAYPL.

Table 8 shows the top 3 most similar entities or relations to a target in WP$^-$, where it shows similar entities for entities, i.e., `Vancouver` or `computer scientist`, and similar relations for a relation, i.e., `family`. Given a target, the top similar entities/relations are selected by computing the dot-products between the target and entities/relations using the initial representations returned by the structure-driven initializer (Initializer) or the final representations (Final Rep.) of MAYPL. While the initializer enables roughly selecting entities/relations of a similar type or conceptually relevant ones to the target, the ones chosen by the final representations become more relevant and semantically closer to the target. This shows that our initializer computes reasonable initial representations, which are effectively refined by the following attentive neural message passing in MAYPL.

6. Conclusion and Discussion

MAYPL learns representations based on how facts, entities, and relations are connected, positioned, and organized in HKGs. Due to its structure-driven learning, MAYPL can effectively compute representations on a new HKG consisting of new entities and relations unobserved during training time. Experimental results show that MAYPL outperforms 40 different baseline methods on 10 benchmark datasets in varied settings, i.e., transductive link prediction on HKGs, inductive link prediction on KGs, and inductive link prediction on HKGs. MAYPL's success implies that thoroughly learning and exploiting the structure of an HKG is necessary and sufficient for learning representations on HKGs.

Limitations and Future Work Within MAYPL, the procedure of message computation and aggregation requires enough computing resources as described in Appendix D. We plan to further optimize our implementations and apply cost-effective attention computation, e.g., replace Bahdanau-style attention (Bahdanau et al., 2015) with Luong-style attention (Luong et al., 2015), for increasing MAYPL's scalability. In Table 9, we compare MAYPL and other methods on WikiPeople$^-$ regarding the training time, the number of parameters, and MRR, where we present all baselines

Table 9: The training time, the number of parameters, and MRR of MAYPL and other methods on WikiPeople$^-$.

	Training Time	# parameters	MRR
StarE	4 days	8.2M	0.491
GRAN	10h	>8.9M	0.503
Hy-Transformer	5h	7.8M	0.501
HyperFormer	N/A	67.0M	0.473
HAHE	12h	30.2M	0.509
MAYPL	20h	10.5M	0.519

whose training times and parameter sizes are available. We believe MAYPL can be applied to question answering (QA) tasks (Chen et al., 2021), serving as an additional source of knowledge for pretrained language models. The theoretical analysis of MAYPL (Lee et al., 2024) and its extension to various forms of knowledge graphs (Lee et al., 2023a; Chung & Whang, 2023) remain as future work. Furthermore, by extending MAYPL, we plan to develop foundation models for KG reasoning (Galkin et al., 2024) applicable to a wide range of real-world use cases.

Impact Statement

Knowledge representation learning is one of the key techniques of many modern AI applications. We believe our work can be applied to various applications, such as generative models, natural language processing (Chen et al., 2021), robotics (Kwak et al., 2022), and information retrieval, and advance the machine learning fields requiring grounding knowledge and structure-infused representation vectors. Given that our work eases the automatic completion of missing knowledge, possible undesirable misuses of our technique include attempts to predict and disclose others' private information with malicious intentions. Such risks should be monitored, and the readers of this paper are advised to be aware that our research should not be applied to such activities.

Acknowledgements

This work was supported by the NRF of Korea grant funded by MSIT (70% from RS-2025-00559066, Responsible Multimodal Graph AI) and the IITP grant funded by MSIT (30% from RS-2022-II220369, (Part 4) Development of AI Technology to support Expert Decision-making that can Explain the Reasons/Grounds for Judgment Results based on Expert Knowledge).

References

Abboud, R., İsmail İlkan Ceylan, Lukasiewicz, T., and Salvatori, T. A box embedding model for knowledge base completion. In *Proceedings of the 34th Conference on Neural Information Processing Systems*, pp. 9649–9661, 2020.

Ali, M., Berrendorf, M., Galkin, M., Thost, V., Ma, T., Tresp, V., and Lehmann, J. Improving inductive link prediction using hyper-relational facts. In *Proceedings of the 20th International Semantic Web Conference*, pp. 74–92, 2021.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. In *arXiv preprint*, 2016.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

Bordes, A., Usunier, N., García-Durán, A., Weston, J., and Yakhnenko, O. Translating embeddings for modeling multi-relational data. In *Proceedings of the 27th Conference on Neural Information Processing Systems*, pp. 2787–2795, 2013.

Brody, S., Alon, U., and Yahav, E. How attentive are graph attention networks? In *Proceedings of the 10th International Conference on Learning Representations*, 2022.

Chen, S., Liu, X., Gao, J., Jiao, J., Zhang, R., and Ji, Y. HittER: Hierarchical transformers for knowledge graph embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 10395–10407, 2021.

Chung, C. and Whang, J. J. Learning representations of bi-level knowledge graphs for reasoning beyond link prediction. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence*, pp. 4208–4216, 2023.

Chung, C., Lee, J., and Whang, J. J. 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*, pp. 310–322, 2023.

Daza, D., Cochez, M., and Groth, P. Inductive entity representations from text via link prediction. In *Proceedings of The Web Conference 2021*, pp. 798–808, 2021.

Di, S. and Chen, L. Message function search for knowledge graph embedding. In *Proceedings of the ACM Web Conference 2023*, pp. 2633–2644, 2023.

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

Fatemi, B., Taslakian, P., Vazquez, D., and Poole, D. Knowledge hypergraphs: Prediction beyond binary relations. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, pp. 2191–2197, 2020.

Feng, Y., You, H., Zhang, Z., Ji, R., and Gao, Y. Hypergraph neural networks. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, pp. 3558–3565, 2019.

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

Galkin, M., Denis, E., Wu, J., and Hamilton, W. L. NodePiece: Compositional and parameter-efficient representations of large knowledge graphs. In *Proceedings of the 10th International Conference on Learning Representations*, 2022.

Galkin, M., Yuan, X., Mostafa, H., Tang, J., and Zhu, Z. Towards foundation models for knowledge graph reasoning. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.

Geng, Y., Chen, J., Pan, J. Z., Chen, M., Jiang, S., Zhang, W., and Chen, H. Relational message passing for fully inductive knowledge graph completion. In *Proceedings of the 2023 IEEE 39th International Conference on Data Engineering*, pp. 1221–1233, 2023.

Gesese, G. A., Sack, H., and Alam, M. RAILD: Towards leveraging relation features for inductive link prediction in knowledge graphs. In *Proceedings of the 11th International Joint Conference on Knowledge Graphs*, pp. 82–90, 2022.

Guan, S., Jin, X., Wang, Y., and Cheng, X. Link prediction on N-ary relational data. In *Proceedings of The Web Conference 2019*, pp. 583–593, 2019.

Guan, S., Jin, X., Guo, J., Wang, Y., and Cheng, X. NeuInfer: Knowledge inference on N-ary facts. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6141–6151, 2020.

Guan, S., Jin, X., Guo, J., Wang, Y., and Cheng, X. Link prediction on N-ary relational data based on relatedness evaluation. *IEEE Transactions on Knowledge and Data Engineering*, 35(01):672–685, 2023.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the 2015 IEEE International Conference on Computer Vision*, pp. 1026–1034, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Hu, Z., Gutiérrez-Basulto, V., Xiang, Z., Li, R., and Pan, J. Z. 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*, pp. 803–812, 2023.

Huang, X., Orth, M. R., Barceló, P., Bronstein, M. M., and İsmail İlkan Ceylan. Link prediction with relational hypergraphs. In *arXiv preprint*, 2024.

Ji, S., Pan, S., Cambria, E., Marttinen, P., and Yu, P. S. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 33(2):494–514, 2022.

Kingma, D. P. and Ba, J. L. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

Kwak, J. H., Lee, J., Whang, J. J., and Jo, S. Semantic grasping via a knowledge graph of robotic manipulation: A graph representation learning approach. *IEEE Robotics and Automation Letters*, 7(4):9397–9404, 2022.

Lee, J., Chung, C., Lee, H., Jo, S., and Whang, J. J. VISTA: Visual-textual knowledge graph representation learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 7314–7328, 2023a.

Lee, J., Chung, C., and Whang, J. J. InGram: Inductive knowledge graph embedding via relation graphs. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 18796–18809, 2023b.

Lee, J., Hwang, M., and Whang, J. J. PAC-Bayesian generalization bounds for knowledge graph representation learning. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 26589–26620, 2024.

Li, Z., Wang, C., Wang, X., Chen, Z., and Li, J. HJE: Joint convolutional representation learning for knowledge hypergraph completion. *IEEE Transactions on Knowledge and Data Engineering*, 36(8):3879–3892, 2024a.

Li, Z., Wang, X., Li, J., Guo, W., and Zhao, J. HyCubE: Efficient knowledge hypergraph 3d circular convolutional embedding. In *arXiv preprint*, 2024b.

Liu, S., Grau, B., Horrocks, I., and Kostylev, E. INDIGO: Gnn-based inductive knowledge graph completion using pair-wise encoding. In *Proceedings of the 35th Conference on Neural Information Processing Systems*, pp. 2034–2045, 2021a.

Liu, Y., Yao, Q., and Li, Y. Role-aware modeling for N-ary relational knowledge bases. In *Proceedings of the Web Conference 2021*, pp. 2660–2671, 2021b.

Luo, H., E, H., Yang, Y., Guo, Y., Sun, M., Yao, T., Tang, Z., Wan, K., Song, M., and Lin, W. HAHE: Hierarchical attention for hyper-relational knowledge graphs in global and local level. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8095–8107, 2023.

Luong, T., Pham, H., and Manning, C. D. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421, 2015.

Mai, S., Zheng, S., Yang, Y., and Hu, H. Communicative message passing for inductive relation reasoning. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, pp. 4294–4302, 2021.

Mu, C., Zhang, L., Zhang, J., Huang, Q., and Wang, Z. Inductive knowledge graph embedding via exploring interaction patterns of relations. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 1722–1731, 2024.

Rosso, P., Yang, D., and Cudré-Mauroux, P. Beyond triplets: Hyper-relational knowledge graph embedding for link prediction. In *Proceedings of The Web Conference 2020*, pp. 1885–1896, 2020.

Sadeghian, A., Armandpour, M., Ding, P., and Wang, D. Z. DRUM: End-to-end differentiable rule mining on knowledge graphs. In *Proceedings of the 33rd Conference on Neural Information Processing Systems*, pp. 15347–15357, 2019.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

Suchanek, F. M., Kasneci, G., and Weikum, G. Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, pp. 697–706, 2007.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

Teru, K., Denis, E., and Hamilton, W. Inductive relation prediction by subgraph reasoning. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 9448–9457, 2020.

Toutanova, K. and Chen, D. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pp. 57–66, 2015.

Vashishth, S., Sanyal, S., Nitin, V., and Talukdar, P. Composition-based multi-relational graph convolutional networks. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Proceedings of the 31st Conference on Neural Information Processing Systems*, pp. 5998–6008, 2017.

Vrandečić, D. and Krötzsch, M. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57 (10):78–85, 2014.

Wang, C., Wang, X., Li, Z., Chen, Z., and Li, J. HyConvE: A novel embedding model for knowledge hypergraph link prediction with convolutional neural networks. In *Proceedings of the ACM Web Conference 2023*, pp. 188–198, 2023.

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

Wen, J., Li, J., Mao, Y., Chen, S., and Zhang, R. On the representation and embedding of knowledge bases beyond binary relations. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pp. 1300–1307, 2016.

Xiong, B., Nayyeri, M., Pan, S., and Staab, S. Shrinking embeddings for hyper-relational knowledge graphs. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13306–13320, 2023.

Xiong, W., Hoang, T., and Wang, W. Y. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 564–573, 2017.

Xu, X., Zhang, P., He, Y., Chao, C., and Yan, C. Subgraph neighboring relations infomax for inductive link prediction on knowledge graphs. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence*, pp. 2341–2347, 2022.

Yadati, N. Neural message passing for multi-relational ordered and recursive hypergraphs. In *Proceedings of the 34th Conference on Neural Information Processing Systems*, pp. 3275–3289, 2020.

Yadati, N., Nimishakavi, M., Yadav, P., Nitin, V., Louis, A., and Talukdar, P. HyperGCN: A new method for training graph convolutional networks on hypergraphs. In *Proceedings of the 33rd Conference on Neural Information Processing Systems*, pp. 1511–1522, 2019.

Yan, S., Zhang, Z., Sun, X., Xu, G., Li, S., Liu, Q., Liu, N., and Wang, S. PolygonE: Modeling n-ary relational data as gyro-polygons in hyperbolic space. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence*, pp. 4308–4317, 2022.

Yang, F., Yang, Z., and Cohen, W. W. Differentiable learning of logical rules for knowledge base reasoning. In *Proceedings of the 31st Conference on Neural Information Processing Systems*, pp. 2319–2328, 2017.

Yang, J., Jiang, X., Gao, Y., Yang, L. T., and Yang, J. Generalize to fully unseen graphs: Learn transferable hyper-relation structures for inductive link prediction. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 1274–1282, 2024.

Yu, D. and Yang, Y. Improving hyper-relational knowledge graph completion. In *arXiv preprint*, 2021.

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

Zhang, Y. and Yao, Q. Knowledge graph reasoning with relational digraph. In *Proceedings of The ACM Web Conference 2022*, pp. 912–924, 2022.

Zhu, Z., Zhang, Z., Xhonneux, L.-P., and Tang, J. Neural Bellman-Ford networks: A general graph neural network framework for link prediction. In *Proceedings of the 35th Conference on Neural Information Processing Systems*, pp. 29476–29490, 2021.

A. Comparison of MAYPL with Existing Methods

Table 10: Comparison of MAYPL and all baseline methods regarding their ways of representing knowledge and their abilities to handle new entities and relations in inductive inference.

	Knowledge Representation				Inductive Inference	
	KG	KHG	NRR	HKG	new entities	new relations
CompGCN (Vashishth et al., 2020)	✓					
GraIL (Teru et al., 2020)	✓				✓	
CoMPILE (Mai et al., 2021)	✓				✓	
SNRI (Xu et al., 2022)	✓				✓	
INDIGO (Liu et al., 2021a)	✓				✓	
NeuralLP (Yang et al., 2017)	✓				✓	
DRUM (Sadeghian et al., 2019)	✓				✓	
BLP (Daza et al., 2021)	✓				✓	
NodePiece (Galkin et al., 2022)	✓				✓	
RED-GNN (Zhang & Yao, 2022)	✓				✓	
NBFNet (Zhu et al., 2021)	✓				✓	
RAILD (Gesese et al., 2022)	✓				✓	✓
RMPI (Geng et al., 2023)	✓				✓	✓
InGram (Lee et al., 2023b)	✓				✓	✓
HyRel (Yang et al., 2024)	✓				✓	✓
InKGE-EIPR (Mu et al., 2024)	✓				✓	✓
HGNN (Feng et al., 2019)		△				
HyperGCN (Yadati et al., 2019)		△				
HypE (Fatemi et al., 2020)		✓				
G-MPNN (Yadati, 2020)		✓			△	
S2S (Di et al., 2021)		✓				
HyConvE (Wang et al., 2023)		✓				
MSeaKG (Di & Chen, 2023)		✓		✓		
HJE (Li et al., 2024a)		✓				
HCNet (Huang et al., 2024)		✓			✓	
HyCubE (Li et al., 2024b)		✓				
RAE (Zhang et al., 2018)			✓			
NaLP (Guan et al., 2019)			✓			
RAM (Liu et al., 2021b)			✓			
tNaLP (Guan et al., 2023)			✓			
NeuInfer (Guan et al., 2020)				✓		
HINGE (Rosso et al., 2020)				✓		
StarE (Galkin et al., 2020)				✓		
GRAN (Wang et al., 2021)				✓		
Hy-Transformer (Yu & Yang, 2021)				✓		
PolygonE (Yan et al., 2022)				✓		
HyperFormer (Hu et al., 2023)				✓		
HAHE (Luo et al., 2023)				✓		
ShrinkE (Xiong et al., 2023)				✓		
HyNT (Chung et al., 2023)				✓		
QBLP (Ali et al., 2021)				✓	✓	
MAYPL				✓	✓	✓

Table 10 compares MAYPL and other existing methods in terms of (i) the form of knowledge representations and (ii) the capability of inductive inference. While KGs only contain triplets, KHGs, NRRs, and HKGs provide more enriched information by providing additional details. Comparisons between KHGs, NRRs, and HKGs are described in Section 2, where we show that HKGs are better forms than KHGs and NRRs in preserving the original information of real-world knowledge bases. For the inductive inference, we consider whether a method can handle new entities and new relations.

Some methods in Table 10 are explained in Section 1 and Section 2. While HGNN (Feng et al., 2019) and HyperGCN (Yadati et al., 2019) are not originally designed for KGs, they are considered in (Yadati, 2020) since they handle hyperedges. Also, G-MPNN (Yadati, 2020) can handle new entities only one-hop distant from observed entities, having limited inductive capability. Thus, these methods are denoted by △ in Table 10.

Table 11: Datasets for Transductive Link Prediction on HKGs.

	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{H}_{\text{tr}}\rvert$	$\lvert\mathcal{H}_{\text{val}}\rvert$	$\lvert\mathcal{H}_{\text{tst}}\rvert$
WD50K (Galkin et al., 2020)	47,155	531	166,435	23,913	46,159
WikiPeople$^-$ (Wang et al., 2021)	34,825	178	294,439	37,715	37,712
WikiPeople (Guan et al., 2019)	47,765	193	305,725	38,223	38,281

Table 12: Datasets for Inductive Link Prediction on KGs.

	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{H}\rvert$	$\lvert\acute{\mathcal{V}}\rvert$	$\lvert\acute{\mathcal{R}}\rvert$	$\lvert\acute{\mathcal{H}}_{\text{fct}}\rvert$	$\lvert\acute{\mathcal{H}}_{\text{val}}\rvert$	$\lvert\acute{\mathcal{H}}_{\text{tst}}\rvert$
NL-100 (Lee et al., 2023b)	1,258	55	7,832	1,709	53	2,378	793	793
NL-75 (Lee et al., 2023b)	2,607	96	11,058	1,578	116	1,818	606	607
NL-50 (Lee et al., 2023b)	4,396	106	17,578	2,335	119	2,576	859	859
NL-25 (Lee et al., 2023b)	4,396	106	17,578	2,146	120	2,230	743	744
WK-100 (Lee et al., 2023b)	9,784	67	49,875	12,136	37	13,487	4,496	4,496
WK-75 (Lee et al., 2023b)	6,853	52	28,741	2,722	65	3,430	1,143	1,144
WK-50 (Lee et al., 2023b)	12,022	72	82,481	9,328	93	9,672	3,224	3,225
WK-25 (Lee et al., 2023b)	12,659	47	41,873	3,228	74	3,391	1,130	1,131
FB-100 (Lee et al., 2023b)	4,659	134	62,809	2,624	77	6,987	2,329	2,329
FB-75 (Lee et al., 2023b)	4,659	134	62,809	2,792	186	9,316	3,106	3,106
FB-50 (Lee et al., 2023b)	5,190	153	85,375	4,445	205	11,636	3,879	3,879
FB-25 (Lee et al., 2023b)	5,190	163	91,571	4,097	216	17,147	5,716	5,716

B. Details of Datasets

Table 11 shows the statistics of datasets for transductive link prediction on HKGs: WD50K (Galkin et al., 2020), WikiPeople$^-$ (Wang et al., 2021), and WikiPeople (Guan et al., 2019), where WikiPeople$^-$ was generated by removing the literals from WikiPeople. Although WikiPeople$^-$ was first used by (Rosso et al., 2020), we refer to (Wang et al., 2021) because it was named WikiPeople$^-$ by (Wang et al., 2021). We noticed that WikiPeople$^-$ was denoted as WikiPeople in some baselines' papers, including (Rosso et al., 2020; Galkin et al., 2020; Luo et al., 2023); by examining the statistics of the datasets used in their official GitHub or reported in the paper, we separate the existing experimental results for WikiPeople$^-$ and WikiPeople. We used both datasets because these datasets are used in different literatures. WikiPeople$^-$ is used to compare methods for HKGs such as StarE (Galkin et al., 2020) or HAHE (Luo et al., 2023), while WikiPeople is used to compare methods for NRRs or KHGs, such as NaLP (Guan et al., 2019) or HCNet (Huang et al., 2024).

For WD50K and WikiPeople$^-$, we adopt the widely-used experimental setting of (Galkin et al., 2020), where a model is tuned using a validation set and re-trained using both training and validation sets. Since there is no official experimental results for HINGE (Rosso et al., 2020) using this setting on WikiPeople$^-$, we do not report its results; the results in (Rosso et al., 2020) are the ones reported after training only using the training set.

The statistics of WD50K in Table 11 differs from that of (Galkin et al., 2020) because (Galkin et al., 2020) also counts some extra tokens. Also, the number of relations of WikiPeople differs from (Guan et al., 2019) due to the different forms of knowledge representations. We note that the JF17K dataset (Wen et al., 2016) is another well-known dataset for HKGs, but the authors who released it (Wen et al., 2016) suggest not using it due to redundant entries. Therefore, we did not use it. Refer to `https://www.site.uottawa.ca/~yymao/JF17K/`.

Table 12 shows the statistics of all 12 datasets for inductive link prediction on KGs: NL-100 (Lee et al., 2023b), NL-75 (Lee et al., 2023b), NL-50 (Lee et al., 2023b), NL-25 (Lee et al., 2023b), WK-100 (Lee et al., 2023b), WK-75 (Lee et al., 2023b), WK-50 (Lee et al., 2023b), WK-25 (Lee et al., 2023b), FB-100 (Lee et al., 2023b), FB-75 (Lee et al., 2023b), FB-50 (Lee et al., 2023b), and FB-25 (Lee et al., 2023b). The full experimental results on these datasets are presented in Appendix E.

Also, Table 13 shows the statistics of datasets for inductive link prediction on HKGs: WD20K(100)v1 (Ali et al., 2021), WD20K(100)v2 (Ali et al., 2021), WP-IND (Yadati, 2020), and MFB-IND (Yadati, 2020). The statistics of WD20K(100)v1 and WD20K(100)v2 differ from the original paper (Ali et al., 2021) because (Ali et al., 2021) reported the statistics by only considering at most six qualifiers per fact, whereas we removed this constraint. In addition, (Ali et al., 2021) also counts some extra tokens, as in (Galkin et al., 2020). On the other hand, in (Yadati, 2020), unseen entities are assumed to be predefined in WP-IND and MFB-IND, and all other entities are considered as seen entities, regardless of whether they

Table 13: Datasets for Inductive Link Prediction on HKGs.

	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{H}\rvert$	$\lvert\acute{\mathcal{V}}\rvert$	$\lvert\acute{\mathcal{R}}\rvert$	$\lvert\acute{\mathcal{H}}_{\text{fct}}\rvert$	$\lvert\acute{\mathcal{H}}_{\text{val}}\rvert$	$\lvert\acute{\mathcal{H}}_{\text{tst}}\rvert$
WD20K(100)v1 (Ali et al., 2021)	5,785	91	7,785	4,286	74	2,667	295	365
WD20K(100)v2 (Ali et al., 2021)	3,226	56	4,146	5,593	53	4,274	539	678
WP-IND (Yadati, 2020)	4,017	35	4,139	4,463	35	4,690	275	275
MFB-IND (Yadati, 2020)	3,223	22	336,733	3,783	22	344,260	3,763	3,764

actually appear in a training set. However, in our definition, unseen entities are those that do not appear in the training set, which reduces the number of seen entities $|\mathcal{V}|$. Additionally, we transform hyperedges into hyper-relational facts, which increases the number of relation labels $|\mathcal{R}|$. Therefore, the statistics we report for WP-IND and MFB-IND differ from those reported in (Yadati, 2020). Note that although WD20K(100)v1, WD20K(100)v2, WP-IND, and MFB-IND include node features, MAYPL was trained and evaluated purely based on the graph structure, without utilizing the node features.

C. Coverage of Methods Across Datasets

In Section 5, we compare MAYPL with 40 different baseline methods using 10 benchmark datasets. Also, Appendix E shows additional experimental results of inductive link prediction on KGs, which add 9 more datasets (NL-100, NL-75, NL-25, WK-100, WK-75, WK-25, FB-100, FB-75, FB-25) and one more baseline (RAILD ran on NL-100, WK-100, and FB-100). Thus, MAYPL is compared with 41 baseline methods on 19 datasets in total. Table 14 shows which methods are ran on which datasets. The baselines' results are obtained from the baseline's original paper (marked as ○) or the paper that proposed the dataset (marked as ●). For brevity, NL-75, NL-50, and NL-25 are denoted by NLs, WK-75, WK-50, and WK-25 are denoted by WKs, and FB-75, FB-50, and FB-25 are denoted by FBs.

D. Implementation and Experimental Details of MAYPL

We provide implementation details of MAYPL. Also, we describe how we ran MAYPL and present the runtime and memory usage of MAYPL. Our codes are available at `https://github.com/bdi-lab/MAYPL`.

D.1. Implementation Details of MAYPL

In our implementation of MAYPL, we use the Adam optimizer (Kingma & Ba, 2015), PReLU (He et al., 2015) activation function, dropout (Srivastava et al., 2014), residual connection (He et al., 2016), label smoothing (Szegedy et al., 2016), and layer normalization (Ba et al., 2016). All linear projections are followed by a bias term omitted for brevity in the formulas of Section 4. In the attentive neural message passing, we use multi-heads (Vaswani et al., 2017; Brody et al., 2022). We use the dynamic split strategy (Lee et al., 2023b) with modifications: instead of creating a spanning tree, we simply use some portion of a training graph as a message set even if it is not connected and use the remaining facts consisting of entities and relations in the chosen message set for loss calculation. We use Python 3.9, and PyTorch 2.0.1 with cuda version 11.7.

D.2. Experimental Details, Runtime, and Memory Usage of MAYPL

We provide experimental details of each experiment along with MAYPL's runtime and memory usage. For runtime, we report the time required to reproduce the reported results from scratch, including validation and test time. We report the GPU memory using `torch.cuda.max_memory_allocated()` function in PyTorch. The time and memory are rounded for brevity. Note that d is the dimension of all layers ($d = d_0 = d_1 = \cdots = d_L$), ϵ is the amount of label smoothing, e_{best} is the best epoch, lr is the learning rate, n_{head} is the number of attention heads, n_{batch} is the number of batches, δ_{tr} is the train graph ratio, and δ_{drop} is the dropout ratio.

Table 15 shows the best hyperparameters, runtime, and memory usage of MAYPL for WD50K, WikiPeople$^-$, and WikiPeople. We trained MAYPL on the training set of WikiPeople$^-$ to find the best hyperparameters for both WikiPeople and WikiPeople$^-$. We chose the best hyperparameters based on the validation performance. After finding the best hyperparameters, we fixed the best epoch for WikiPeople$^-$ and re-trained MAYPL on WikiPeople$^-$ with training+validation sets. For WikiPeople, we used the training set of WikiPeople to re-train MAYPL, and tuned the epoch while fixing all other parameters. For WD50K, we tuned MAYPL on the training set of WD50K, fixed all hyperparameters, and re-trained

Table 14: Coverage of Methods Across Datasets and the Sources of Baselines' Results. The baselines' results are obtained from the baseline's original paper (marked as ○) or the paper that proposed the dataset (marked as ●). NLs denote NL-75, NL-50, and NL-25, WKs denote WK-75, WK-50, and WK-25, and FBs denote FB-75, FB-50, and FB-25. In Section 5 and Appendix E, MAYPL is compared with 41 baseline methods on 19 datasets in total.

	NL-100	NLs	WK-100	WKs	FB-100	FBs	WP	WP$^-$	WD50K	WP-IND	MFB-IND	WD20K(100)v1	WD20K(100)v2
CompGCN (Vashishth et al., 2020)	●	●	●	●	●	●						●	●
GraIL (Teru et al., 2020)	●	●											
CoMPILE (Mai et al., 2021)	●	●											
SNRI (Xu et al., 2022)	●	●											
INDIGO (Liu et al., 2021a)	●	●											
NeuralLP (Yang et al., 2017)	●	●	●	●	●	●							
DRUM (Sadeghian et al., 2019)	●	●	●	●	●	●							
BLP (Daza et al., 2021)	●	●	●	●	●	●						●	●
NodePiece (Galkin et al., 2022)	●	●	●	●	●	●							
RED-GNN (Zhang & Yao, 2022)	●	●	●	●	●	●							
NBFNet (Zhu et al., 2021)	●	●	●	●	●	●							
RAILD (Gesese et al., 2022)	●		●		●								
RMPI (Geng et al., 2023)	●	●											
InGram (Lee et al., 2023b)	○	○	○	○	○	○							
HyRel (Yang et al., 2024)	○	○	○	○	○	○							
InKGE-EIPR (Mu et al., 2024)	○	○	○	○	○	○							
HGNN (Feng et al., 2019)										●	●		
HyperGCN (Yadati et al., 2019)										●	●		
G-MPNN (Yadati, 2020)										○	○		
RAE (Zhang et al., 2018)							●						
HypE (Fatemi et al., 2020)							●						
S2S (Di et al., 2021)							○						
HyConvE (Wang et al., 2023)							○						
MSeaKG (Di & Chen, 2023)							○						
HJE (Li et al., 2024a)							○						
HCNet (Huang et al., 2024)							○			○	○		
HyCubE (Li et al., 2024b)							○						
NaLP (Guan et al., 2019)							○		●				
RAM (Liu et al., 2021b)							○						
tNaLP (Guan et al., 2023)							○						
NeuInfer (Guan et al., 2020)							○						
HINGE (Rosso et al., 2020)									●				
StarE (Galkin et al., 2020)								○	○			●	●
GRAN (Wang et al., 2021)								○	○				
Hy-Transformer (Yu & Yang, 2021)								○	○				
PolygonE (Yan et al., 2022)							○						
HyperFormer (Hu et al., 2023)								○	○				
HAHE (Luo et al., 2023)								○	○				
ShrinkE (Xiong et al., 2023)								○	○				
HyNT (Chung et al., 2023)									○				
QBLP (Ali et al., 2021)	●	●	●	●	●	●						○	○
MAYPL	○	○	○	○	○	○	○	○	○	○	○	○	○

MAYPL on WD50K with training+validation sets. We performed validation every 50 epoch, and stopped training if the validation performance did not increase for 250 epochs. We ran MAYPL on NVIDIA RTX A6000 with $d = 256$.

Table 16 shows the best hyperparameters, runtime, and memory usage of MAYPL for NL-100, NL-75, NL-50, NL-25, WK-100, WK-75, WK-50, WK-25, FB-100, FB-75, FB-50, and FB-25. MAYPL was trained on a training graph, validated and tested on an inference graph without re-training. We chose the best hyperparameters based on the validation performance. We did not adopt the early stop strategy and ran MAYPL for 500 epochs, performing validation every 10 epochs. One exception is NL-75, which we ran MAYPL for 1000 epochs. We ran MAYPL on NVIDIA RTX 2080 Ti with $d = 32$.

Table 17 shows best hyperparameters, runtime, and memory usage of MAYPL for WD20K(100)v1, WD20K(100)v2, WP-IND, and MFB-IND. MAYPL was trained on a training graph, validated and tested on an inference graph without re-training. We chose the best hyperparameters based on the validation performance. We did not adopt the early stop strategy for WD20K(100)v1 and WD20K(100)v2, and we ran MAYPL for 1000 epochs and 500 epochs, respectively, while performing validation every 10 epochs. For WP-IND, we did not adopt the early stop strategy and ran MAYPL for 2000 epochs while performing validation every 20 epochs. For MFB-IND, we performed validation every 50 epochs, and stopped training if the validation performance did not increase for 250 epochs. We ran MAYPL on NVIDIA RTX 2080 Ti with $d = 256$ for WD20K(100)v1 and WD20K(100)v2, and $d = 128$ for WP-IND. For MFB-IND, we ran MAYPL on NVIDIA RTX A6000 with $d = 128$.

Table 15: Best hyperparameters, runtime, and memory usage of MAYPL for WD50K, WikiPeople$^-$, and WikiPeople.

	Time	Memory	ϵ	e_{best}	lr	$\widetilde{L}$	L	n_{head}	n_{batch}	δ_{tr}	δ_{drop}
WD50K	21h	42.8GB	0.05	3000	0.0001	4	6	16	20	0.7	0.2
WikiPeople$^-$	20h	43.8GB	0.0	2900	0.0001	3	4	32	20	0.7	0.1
WikiPeople	21h	45.8GB	0.0	2400	0.0001	3	4	32	20	0.7	0.1

Table 16: Best hyperparameters, runtime (m: minutes, h: hours), and memory usage of MAYPL for NL-100, NL-75, NL-50, NL-25, WK-100, WK-75, WK-50, WK-25, FB-100, FB-75, FB-50, and FB-25.

	Time	Memory	ϵ	e_{best}	lr	$\widetilde{L}$	L	n_{head}	n_{batch}	δ_{tr}	δ_{drop}
NL-100	15m	0.1GB	0.0	160	0.0002	4	4	4	30	0.35	0.05
NL-75	30m	0.1GB	0.0	660	0.0002	4	1	2	35	0.15	0.0
NL-50	55m	0.4GB	0.0	390	0.001	5	8	4	30	0.4	0.1
NL-25	40m	0.5GB	0.0	260	0.002	2	10	2	30	0.5	0.05
WK-100	1h	0.6GB	0.0	410	0.0005	3	5	2	30	0.3	0.05
WK-75	25m	0.5GB	0.0	250	0.0005	4	5	4	30	0.5	0.05
WK-50	35m	1.2GB	0.0	230	0.0002	3	5	8	30	0.3	0.1
WK-25	30m	0.8GB	0.0	300	0.001	3	5	8	30	0.5	0.05
FB-100	50m	0.7GB	0.0	310	0.00015	4	5	4	30	0.3	0.1
FB-75	20m	0.7GB	0.0	140	0.0005	3	5	8	30	0.3	0.1
FB-50	45m	1.1GB	0.0	320	0.001	4	5	4	30	0.35	0.1
FB-25	50m	1.6GB	0.0	390	0.001	3	5	8	30	0.6	0.1

Table 17: Best hyperparameters, runtime (m: minutes, h: hours), and memory usage of MAYPL for WD20K(100)v1, WD20K(100)v2, WP-IND, and MFB-IND.

	Time	Memory	ϵ	e_{best}	lr	$\widetilde{L}$	L	n_{head}	n_{batch}	δ_{tr}	δ_{drop}
WD20K(100)v1	2h	3.1GB	0.0	930	0.0001	5	6	32	30	0.5	0.05
WD20K(100)v2	55m	1.2GB	0.0	490	0.0002	3	5	8	30	0.4	0.05
WP-IND	8h	1.4GB	0.0	1800	0.0001	4	10	4	50	0.7	0.1
MFB-IND	7h	27GB	0.0	1550	0.001	3	3	16	20	0.7	0.1

E. Additional Experimental Results of Inductive Link Prediction on KGs

Table 18 shows inductive link prediction performance of MAYPL and other baseline methods on all 12 different datasets provided in (Lee et al., 2023b). The prefixes in the dataset names denote their origins (WK for Wikidata68K (Gesese et al., 2022), FB for FB15K237 (Toutanova & Chen, 2015), and NL for NELL-995 (Xiong et al., 2017)), while the suffixes indicate the ratio of triplets in an inference graph that contains new relations (Lee et al., 2023b). For example, the WK-75 dataset is extracted from Wikidata68K, and its 75% triplets in the inference graph involve new relations. In Table 6, the best result is **boldfaced**, the second-best result is underlined, and the third-best result is *italic*. Among the baseline methods, RAILD only can be run on WK-100, FB-100, and NL-100. Also, GraIL, CoMPILE, SNRI, INDIGO, and RMPI are only run on datasets extracted from NELL-995 due to their scalability issues (Lee et al., 2023b).

On each dataset, we use three standard link prediction metrics, MRR, Hit10, and Hit1. To assess the relative performance of each model in comparison to the best-performing model for each metric on a given dataset, we compute the loss as follows:

$$\text{loss}_{\text{metric,model}} = \frac{\text{score}_{\text{metric,best}} - \text{score}_{\text{metric,model}}}{\text{score}_{\text{metric,best}}}$$

where $\text{score}_{\text{metric,best}}$ denotes the performance of the best-performing model using the given metric, and $\text{score}_{\text{metric,model}}$ denotes the performance of a target model. Since each metric presents a different perspective of a model's performance, and a model's performance varies depending on metrics and ratios of triplets with new relations, we introduce a holistic measure called the total loss, providing a comprehensive evaluation of a model's performance across multiple metrics and varying ratios of triplets with new relations. The total loss of a model is obtained by summing its losses across all metrics for datasets with the same origin. A lower total loss indicates superior performance.

Table 18: Inductive Link Prediction Performance on 12 Different Datasets in (Lee et al., 2023b).

	WK-100			WK-75			WK-50			WK-25			Total Loss
	MRR	Hit10	Hit1	MRR	Hit10	Hit1	MRR	Hit10	Hit1	MRR	Hit10	Hit1	
CompGCN	0.003	0.009	0.000	0.015	0.028	0.003	0.003	0.002	0.001	0.009	0.020	0.000	11.679
NodePiece	0.007	0.018	0.002	0.021	0.052	0.003	0.008	0.013	0.002	0.053	0.122	0.019	10.917
NeuralLP	0.009	0.016	0.005	0.020	0.054	0.004	0.025	0.054	0.007	0.068	0.104	0.046	10.269
DRUM	0.010	0.019	0.004	0.020	0.043	0.007	0.017	0.046	0.002	0.064	0.116	0.035	10.542
BLP	0.012	0.025	0.003	0.043	0.089	0.016	0.041	0.092	0.013	0.125	0.283	0.055	9.003
QBLP	0.012	0.025	0.003	0.044	0.091	0.016	0.035	0.080	0.011	0.116	0.294	0.042	9.222
NBFNet	0.014	0.026	0.005	0.072	0.172	0.028	0.062	0.105	*0.036*	0.154	0.301	0.092	7.571
RED-GNN	0.096	0.136	0.070	0.172	0.290	0.110	0.058	0.093	0.033	0.170	0.263	0.111	4.864
RAILD	0.026	0.052	0.010	-	-	-	-	-	-	-	-	-	-
InGram	*0.107*	*0.169*	*0.072*	0.247	0.362	0.179	*0.068*	0.135	0.034	0.186	0.309	0.124	*3.342*
HyRel	0.091	0.165	0.059	*0.255*	0.389	*0.187*	*0.068*	*0.138*	*0.036*	*0.191*	*0.316*	*0.125*	3.403
InKGE-EIPR	0.125	0.209	0.086	0.262	*0.380*	0.188	0.076	0.164	0.038	0.203	0.330	0.137	2.300
MAYPL	**0.139**	**0.236**	**0.091**	**0.290**	**0.441**	**0.213**	**0.109**	**0.230**	**0.054**	**0.275**	**0.501**	**0.169**	**0.000**
	FB-100			FB-75			FB-50			FB-25			Total Loss
	MRR	Hit10	Hit1	MRR	Hit10	Hit1	MRR	Hit10	Hit1	MRR	Hit10	Hit1	
CompGCN	0.015	0.025	0.008	0.013	0.026	0.000	0.004	0.006	0.002	0.003	0.004	0.000	11.698
NodePiece	0.006	0.009	0.001	0.016	0.029	0.007	0.021	0.048	0.006	0.044	0.114	0.011	11.007
NeuralLP	0.026	0.057	0.007	0.056	0.099	0.030	0.088	0.184	0.043	0.164	0.309	0.098	7.897
DRUM	0.034	0.077	0.011	0.065	0.121	0.034	0.101	0.191	0.061	0.175	0.320	0.109	7.350
BLP	0.017	0.035	0.004	0.047	0.085	0.024	0.078	0.156	0.037	0.107	0.212	0.053	9.001
QBLP	0.013	0.026	0.003	0.041	0.084	0.017	0.071	0.147	0.030	0.104	0.226	0.043	9.268
NBFNet	0.072	0.154	0.026	0.089	0.166	0.048	0.130	0.259	0.071	0.224	0.410	0.137	5.768
RED-GNN	0.121	0.263	0.053	0.107	0.201	0.057	0.129	0.251	0.072	0.145	0.284	0.077	6.168
RAILD	0.031	0.048	0.016	-	-	-	-	-	-	-	-	-	-
InGram	0.223	0.371	0.146	0.189	0.325	0.119	0.117	0.218	0.067	0.133	0.271	0.067	4.761
HyRel	*0.282*	*0.463*	*0.188*	*0.277*	*0.433*	0.196	*0.178*	*0.333*	*0.101*	0.210	*0.420*	0.114	*1.451*
InKGE-EIPR	0.333	0.509	0.245	**0.285**	**0.462**	*0.195*	0.204	**0.376**	0.123	*0.221*	0.430	*0.125*	0.344
MAYPL	**0.341**	**0.512**	**0.254**	0.282	0.436	**0.197**	**0.205**	0.361	**0.129**	**0.238**	**0.438**	**0.144**	**0.107**
	NL-100			NL-75			NL-50			NL-25			Total Loss
	MRR	Hit10	Hit1	MRR	Hit10	Hit1	MRR	Hit10	Hit1	MRR	Hit10	Hit1	
GraIL	0.135	0.173	0.114	0.096	0.205	0.036	0.162	0.288	0.104	0.216	0.366	0.160	6.830
CoMPILE	0.123	0.209	0.071	0.178	0.361	0.093	0.194	0.330	0.125	0.189	0.324	0.115	6.180
SNRI	0.042	0.064	0.029	0.088	0.177	0.040	0.130	0.187	0.095	0.190	0.270	0.140	8.226
INDIGO	0.160	0.247	0.109	0.121	0.156	0.098	0.167	0.217	0.134	0.166	0.206	0.134	6.907
RMPI	0.220	0.376	0.136	0.138	0.275	0.061	0.185	0.307	0.109	0.213	0.329	0.130	5.870
CompGCN	0.008	0.014	0.001	0.014	0.025	0.003	0.003	0.005	0.000	0.006	0.010	0.000	11.793
NodePiece	0.012	0.018	0.004	0.042	0.081	0.020	0.037	0.079	0.013	0.098	0.166	0.057	10.446
NeuralLP	0.084	0.181	0.035	0.117	0.273	0.048	0.101	0.190	0.064	0.148	0.271	0.101	8.058
DRUM	0.076	0.138	0.044	0.152	0.313	0.072	0.107	0.193	0.070	0.161	0.264	0.119	7.683
BLP	0.019	0.037	0.006	0.051	0.120	0.012	0.041	0.093	0.011	0.049	0.095	0.024	10.689
QBLP	0.004	0.003	0.000	0.040	0.095	0.007	0.048	0.097	0.020	0.073	0.151	0.027	10.666
NBFNet	0.096	0.199	0.032	0.137	0.255	0.077	0.225	0.346	0.161	0.283	0.417	0.224	5.700
RED-GNN	0.212	0.385	0.114	0.203	0.353	0.129	0.179	0.280	0.115	0.214	0.266	0.166	5.292
RAILD	0.018	0.037	0.005	-	-	-	-	-	-	-	-	-	-
InGram	0.309	0.506	0.212	0.261	0.464	0.167	0.281	0.453	0.193	0.334	0.501	0.241	1.965
HyRel	**0.394**	**0.574**	0.299	0.305	0.502	0.205	0.321	0.520	0.222	*0.348*	0.541	0.263	**0.403**
InKGE-EIPR	*0.333*	0.533	*0.241*	**0.318**	**0.525**	**0.211**	*0.315*	**0.529**	*0.217*	0.350	**0.554**	**0.267**	*0.684*
MAYPL	0.374	*0.523*	**0.301**	*0.289*	*0.479*	*0.201*	**0.343**	*0.508*	**0.262**	**0.352**	*0.532*	*0.262*	0.465

For datasets derived from Wikidata68K (i.e., WK-100, WK-75, WK-50, WK-25), MAYPL achieves the best performance across all metrics. Similarly, for datasets from FB15K237 (i.e., FB-100, FB-75, FB-50, FB-25), MAYPL significantly outperforms the baseline models, showing a considerable gap to the best baseline in the total loss. For datasets extracted from NELL-995 (i.e., NL-100, NL-75, NL-50, NL-25), MAYPL demonstrates competitive performance compared to the best-performing baseline, HyRel, while surpassing all the other baseline methods. Note that HyRel is not as competitive as MAYPL in the other Wikidata68K and FB15K237 datasets. Table 19 shows the sum of total losses, i.e., the sum of losses for all metrics over all 12 datasets provided in (Lee et al., 2023b). This comprehensive evaluation shows that MAYPL substantially outperforms the baseline methods in inductive link prediction on KGs.

Table 19: Sum of total losses for all 12 datasets provided in (Lee et al., 2023b).

	Sum of Total Losses
CompGCN	35.170
NodePiece	32.370
NeuralLP	26.224
DRUM	25.575
BLP	28.692
QBLP	29.155
NBFNet	19.039
RED-GNN	16.323
InGram	10.068
HyRel	*5.257*
InKGE-EIPR	3.328
MAYPL	**0.572**

F. More Detailed Explanations about Ablation Studies

In Table 7, we consider six different variations of MAYPL. **(i)** introduces learnable initial representation vectors for individual entities and relations instead of computing them using the structure-driven initializer described in Section 4.1. When the initializer is removed, performance significantly drops in both transductive and inductive settings, which shows the critical role of the structure-driven initializer. **(ii)** removes the attentive neural message passing module proposed in Section 4.2 and the initial entity and relation representations are directly used for link prediction. The considerable performance drop explains the importance of attentive neural message passing. **(iii)** does not apply the attention-based aggregation and uses a simple mean aggregation when updating entity and relation representations in Eq.(3) & Eq.(5). We see that attentively aggregating representations leads to better performance than the simple mean aggregation. **(iv)** does not apply the relation aggregation process of Eq.(5) and updates a relation representation using a simple learnable projection matrix. The performance drop shows the effectiveness of our relation aggregation module. **(v)** replaces all Hadamard products with concatenations in computing messages of facts, entities, and relations in Section 4.2 and Section 4.3. With these changes, the sizes of the corresponding projection matrices are also appropriately modified. The original design using the Hadamard product is more effective than the concatenation. **(vi)** separates the parameters for computing entity representations in Section 4.1 and Section 4.2 and computing a query's representation in link prediction in Section 4.3, while the original MAYPL uses the parameters computed in Section 4.1 and Section 4.2 to perform link prediction in Section 4.3. The performance drop shows the advantage of the parameter sharing in MAYPL.

G. Additional Case Studies about MAYPL

Table 20 shows MAYPL's top 3 predictions on problems in WikiPeople$^-$, where the ground-truth answers are **boldfaced**. Note that MAYPL correctly predicts the answers to these problems, and MAYPL's top 3 predictions of each problem include entities of the same type as the answer. For example, in Problem #4, when the answer is `Ron Howard`, the second and the third predictions are also directors, `James Cameron` and `Steven Spielberg`. In Problem #3 and Problem #4, their primary triplets are identical: (**?**, `awarded`, `Oscar for Best Director`). However, their answers are different due to their different qualifiers. MAYPL accurately predicts the answers by appropriately considering the qualifiers in facts.

Table 20: Top 3 predictions made by MAYPL for problems in WikiPeople$^-$. The answer is **boldfaced**.

Problem #1	(`Marilyn Monroe`, `born in`, `Los Angeles`), {(`country`, `USA`), (`is located in`, **?**)}
Prediction	**`California`**, `New York`, `New York City`
Problem #2	(`Steven Spielberg`, `nominated for`, `Oscar for Best Director`), {(`subject of`, **?**), (`for work`, `Schindler's List`)}
Prediction	**`66th Oscars`**, `71st Oscars`, `47th Oscars`
Problem #3	(**?**, `awarded`, `Oscar for Best Director`), {(`subject of`, `60th Oscars`), (`for work`, `The Last Emperor`)}
Prediction	**`Bernardo Bertolucci`**, `Miloš Forman`, `David Byrne`
Problem #4	(**?**, `awarded`, `Oscar for Best Director`), {(`for work`, `A Beautiful Mind`)}
Prediction	**`Ron Howard`**, `James Cameron`, `Steven Spielberg`