

HYPERMONO: A Monotonicity-aware Approach to Hyper-Relational Knowledge Representation

Zhiwei Hu
School of Computer and Information
Technology
Shanxi University
Taiyuan, China
zhiwei.hu@whu.edu.cn

Víctor Gutiérrez-Basulto
School of Computer Science and
Informatics
Cardiff University
Cardiff, UK
gutierrezbasultov@cardiff.ac.uk

Zhiliang Xiang
School of Computer Science and
Informatics
Cardiff University
Cardiff, UK
xiangz6@cardiff.ac.uk

Ru Li*
School of Computer and Information
Technology
Shanxi University
Taiyuan, China
liru@sxu.edu.cn

Jeff Z. Pan*
ILCC, School of Informatics
University of Edinburgh
Edinburgh, UK
<http://knowledge-representation.org/j.z.pan/>

Abstract

In a hyper-relational knowledge graph (HKG), each fact is composed of a main triple associated with attribute-value qualifiers, which express additional factual knowledge. The hyper-relational knowledge graph completion (HKGC) task aims at inferring plausible missing links in a HKG. Most existing approaches to HKGC focus on enhancing the communication between qualifier pairs and main triples, while overlooking two important properties that emerge from the monotonicity of the hyper-relational graphs representation regime. *Stage Reasoning* allows for a two-step reasoning process, facilitating the integration of coarse-grained inference results derived solely from main triples and fine-grained inference results obtained from hyper-relational facts with qualifiers. In the initial stage, coarse-grained results provide an upper bound for correct predictions, which are subsequently refined in the fine-grained step. More generally, *Qualifier Monotonicity* implies that by attaching more qualifier pairs to a main triple, may only narrow down the answer set, but never enlarges it. This paper proposes the **HyperMono** model for hyper-relational knowledge graph completion, which considers an initial triple-level reasoning phase to be subsequently refined by a triple+qualifiers reasoning step to infer missing entities. To realize qualifier monotonicity, **HyperMono** represents main triples as cones and each qualifier is modeled as a cone that shrinks the cone of the main triple to which the qualifier has been added. Moreover, **HyperMono** leverages the neighborhood context of an entity to semantically strengthen its representation. Experiments on three real-world datasets with three different scenario conditions demonstrate the strong performance of **HyperMono** when compared to the SoTA.

Keywords

Knowledge Graph, Hyper-relational Graph, Knowledge Graph Completion

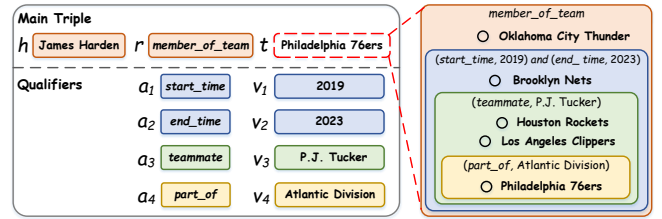


Figure 1: Qualifier Monotonicity: The box on the right presents all five answers to the query $q = (\text{James Harden}, \text{member_of_team}, ?)$. If qualifiers are added to the query, the number of possible answers is reduced, e.g., if we add the qualifier pairs $(\text{start_time}, 2019)$ and $(\text{end_time}, 2023)$ to q , *Oklahoma City Thunder* is not a possible answer anymore.

1 Introduction

Knowledge graphs (KGs) store graph-like knowledge as a collection of factual triples [3, 35]. Each triple (h, r, t) in a KG represents a connection between the head entity h and the tail entity t through the relation type r , e.g., $(\text{James Harden}, \text{member_of_team}, \text{Philadelphia 76ers})$ in Figure 1. Knowledge graphs provide the backbone of various applications like query answering [15, 53], search [30], and recommendation systems [48]. In hyper-relational knowledge graphs (HKGs), hyper-relational facts enhance binary relational facts by augmenting triples with supplementary knowledge expressed in the form of attribute-value qualifier pairs $(a : v)$. For example, in Figure 1, the triple $(\text{James Harden}, \text{member_of_team}, \text{Philadelphia 76ers})$ is contextualized by the qualifier pairs $\{(\text{start_time}: 2019), (\text{end_time}: 2023), (\text{teammate}: \text{P.J. Tucker}), (\text{part_of}: \text{Atlantic Division})\}$, describing teams of the *Atlantic Division* in which *James Harden* and *P.J. Tucker* played together from 2019 to 2023. Like standard KGs, HKGs also suffer from incompleteness. The hyper-relational knowledge graph completion (HKGC) task, which aims at inferring missing links in HKGs, has therefore become an important research topic.

*Contact Authors.

A wide variety of approaches to HKGC have been already proposed, including embedding-based [33, 43, 45, 52], transformer-based with graph neural networks (GNNs) [11, 34] and transformer-based without GNNs methods [7, 16, 27, 41, 50]. Despite the substantial progress achieved so far, each of these approaches has its own disadvantages: embedding-based methods can be categorized into n-ary, key-value pairs and main triple+key-value pairs depending on how facts are defined. While hyper-relational knowledge is effectively represented, the modeling of distinct auxiliary information is done independently. This independence imposes constraints on the mutual interaction between key-value pairs within a given fact. Transformer-based with GNNs methods mainly rely on multi-layer GNNs operations to encode qualifier-pairs information into the main triple, inevitably introducing noise that propagates as the number of GNN layers increases. In addition, the use of GNNs introduces a considerable computational burden. Transformer-based without GNNs methods capture the structure of HKGs within the transformer architecture, thereby avoiding the increase in computational complexity. However, although these methods are in a dominant position on the HKGC task, they overlook two important properties: *Two-Stage Reasoning* and *Qualifier Monotonicity*. When inferring missing entities, two-stage reasoning permits to first reason about binary relational knowledge and then, guided by the results from the previous step, reason about hyper-relational knowledge, while qualifier monotonicity implies that adding qualifiers information to triples might allow for additional inferences, without invalidating those derived solely from considering main triples. Specifically, these properties can be described as follows.

- (1) **Two-Stage Reasoning** allows for a two-step reasoning process, facilitating the integration of coarse-grained inference results derived solely from main triples and fine-grained inference results obtained from hyper-relational facts with qualifiers. In the initial stage, coarse-grained results provide an upper bound for correct predictions, which are subsequently refined in the fine-grained step. For example, in Figure 1, the initial stage derives the following coarse-grained answer set from the main triple: $Ans = \{Oklahoma\ City\ Thunder, Houston\ Rockets, Brooklyn\ Nets, Philadelphia\ 76ers, Los\ Angeles\ Clippers\}$. Ans provides an upper bound to be refined in a subsequent step: after adding qualifiers to the main triple, Ans is narrowed down to $\{Philadelphia\ 76ers\} \subseteq Ans$. Hence, in practice, the outcomes of coarse-grained inference can serve to delimit the decision boundaries of fine-grained predictions, mitigating the occurrence of answer drift phenomena.
- (2) **Qualifier Monotonicity** generally implies that for a given hyper-relational query q , as the number of qualifier pairs in q increases, the answer set of q over a HKG might shrink but it never expands, while conversely removing qualifiers from q could only lead to more possible answers. For example, for the query (*James Harden, member_of_team, ?*) in Figure 1, $?$ has five possible answers $\{Oklahoma\ City\ Thunder, Houston\ Rockets, Brooklyn\ Nets, Philadelphia\ 76ers, Los\ Angeles\ Clippers\}$. If the qualifier pairs (*start_time: 2019*) and (*end_time: 2023*) are added, then *Oklahoma City Thunder* is not a possible answer anymore. The main challenge for qualifier monotonicity lies in devising a representation that accurately captures its essence. It should

possess the capability to faithfully capture the shrinking-like dynamics induced by the incorporation of qualifiers. In the context of embedding-based methods, ShrinkE [45] has explored qualifier monotonicity through box embeddings. However, the inherent limitations of the embedding-based methods lead to a large gap between ShrinkE’s accuracy and that of SoTA models, as shown in Table 1 of § 4.2.

To address the above limitations of existing methods, we introduce a monotonicity-aware framework for hyper-relational knowledge representation (**HyperMono**). To infer the missing entities in hyper-relational queries, **HyperMono** first introduces the **Head Neighborhood Encoder** module to encode the content of the neighbors of the head entity. It considers coarse-grained and fine-grained neighborhood aggregators, characterized by the use of either main triples or complete hyper-relational facts to determine the neighbors of an entity. **HyperMono** further implements two-stage reasoning through a **Missing Entity Predictor** module, which contains two types of predictors: *Triple-based Predictor* and *Qualifier-Monotonicity-aware Predictor*. They differ on whether only main triples or triples+qualifiers are used for prediction. Additionally, **HyperMono** employs cone embeddings to achieve qualifier monotonicity. We utilize cones instead of boxes (as done in ShrinkE) since the main objective is to obtain an answer set qualified by hyper-relational knowledge in a universal subset space. However, the offsets of boxes are unbounded, and how to find boxes to represent a universal set is thus unclear. Cone embeddings naturally represent any finite universal set and its subsets with a proper aperture, as the angle and aperture of cones are bounded to $[-\pi, \pi]$ and $[0, 2\pi]$ [29, 53]. Our contributions can be summarized as follows:

- We propose **HyperMono**, a competitive monotonicity-aware model for hyper-relational knowledge graph completion, which simultaneously considers the two-stage reasoning and qualifier monotonicity properties.
- We model each qualifier as a cone space shrink transformation that narrows down the answer set.
- We conduct experiments on three real-world datasets under three different conditions, and various ablation studies on parameter sensitivity, complexity analysis, and model transferability. The obtained results demonstrate the strong performance when compared with SoTA models.

2 Related Work

Embedding-based. Depending on how the facts are represented, embedding-based methods can be classified as n-ary, key-value pairs and main triple+key-value pairs. (i) In n-ary methods, each fact consists of a pre-defined relation r with n corresponding values, denoted as $r(v_1, v_2, \dots, v_n)$. For instance, m-TransH [43] and RAE [52] generalize TransH [42] by projecting all entities onto a relation-specific hyperplane, while m-CP [10] generalizes Canonical Polyadic (CP) decomposition [36], n-Tucker [21] and GETD [21] generalize Tucker [1] to the higher-arity case. However, these methods convert n-ary facts using one abstract relation, ignoring the semantic distinction of different values for each relation. (ii) In key-value pairs methods, each fact is viewed as a set of attribute-value pairs, denoted as $\{(a_i : v_i)\}_{i=1}^m$. For instance, NaLP [33] employs a

convolutional neural network followed by a multi-layer perceptron to measure the validity of a fact, RAM [22] further encourages the relatedness among different attributes and between an attribute and all corresponding values. However, these methods treat all attribute-value pairs equally and do not distinguish the main triple from the complete fact. (iii) Main triple+key-value pairs methods regard each fact as a main triple with attribute-value pairs, denoted as $((h, r, t), \{(a_i : v_i)\}_{i=1}^m)$. For example, HINGE [33] and NeuInfer [12] independently consider each auxiliary pair in the main triple. However, as different auxiliary information is modeled independently, these methods do not properly encode the interactions between attribute-value pairs within fact.

Transformer-based with GNNs. StarE [11] adopts a message passing based graph encoder CompGCN [37] to obtain the entity and relation embeddings, which are further passed through a transformer decoder to obtain a probability distribution over all entities. QUAD [34] simultaneously takes into account the communication from qualifiers to the main triple and the flow of information from the main triple to qualifiers. It also introduces an auxiliary task to further improve performance. However, these methods primarily employ multi-layer GNNs to incorporate hyper-relational knowledge into the main triple, which inevitably introduces two problems. On the one hand, although multi-layers GNNs operations can encode entity-neighborhood information, they may introduce noise, which can be propagated in a cascading fashion as the number of layers increases. On the other hand, GNNs operations incur in an additional time and space overhead [5, 44, 49].

Transformer-based without GNNs. Many methods abandon the use of GNNs operations and instead incorporate hyper-relational knowledge through various transformer structures. For example, GRAN [41] implements edge-biased fully-connected attention into transformers to encode the graph structure and its heterogeneity. HyNT [7] defines a specialized context transformer to exchange information between the main triple and its qualifiers, HAHE [27] proposes a hierarchical attention structure with global-level and local-level attention to simultaneously model graph-like and sequential content. HyperFormer [16] introduces a mixture-of-experts strategy into the feed-forward layers of a transformer to strengthen its representation capabilities. However, these methods have two shortcomings: on the one hand, they only consider a fine-grained inference steps with hyper-relational knowledge and overlook coarse-grained inferences solely based on the main triples. On the other hand, they do not explicitly embody qualifier monotonicity. We note in passing that although the embedding-based method ShrinkE [45] takes qualifier monotonicity into account, its embedding-based nature results in a large gap in performance when compared with the state-of-the-art, as shown in Table 1 of § 4.2.

3 Method

3.1 Preliminaries

A *hyper-relational knowledge graph (HKG)* $\mathcal{G}$ is defined as $\{\mathcal{E}, \mathcal{R}, \mathcal{H}\}$, where $\mathcal{E}$ is a set of entities, $\mathcal{R}$ is a set of relation types, and $\mathcal{H} = \{(h, r, t, Q) \mid h \in \mathcal{E}, r \in \mathcal{R}, t \in \mathcal{E}\}$ is a set of *hyper-relational facts*, such that (h, r, t) denotes the *main triple* and $Q = \{(a_1 : v_1), \dots, (a_n : v_n) \mid a_i \in \mathcal{R}, v_i \in \mathcal{E}\}$ is a set of auxiliary attribute-value *qualifier pairs*. We use $\mathcal{G}_{triple}$ to denote the set of main triples

occurring in $\mathcal{G}$. Note that when $n = 0$, $\mathcal{G}$ is a standard knowledge graph, i.e., only containing main triples. Under this representation schema, the main triple $(James\ Harden, member_of_team, Philadelphia\ 76ers)$ in Figure 1 with its qualifiers can be represented as: $\{(James\ Harden: member_of_team, Philadelphia\ 76ers), (start_time: 2019), (end_time: 2023), (teammate: P.J.\ Tucker), (part_of: Atlantic\ Division)\}$. Building upon the definition of the knowledge graph completion task [3, 35], the *hyper-relational knowledge graph completion (HKGC)* task aims at predicting entities within hyper-relational facts $\mathcal{H}$. Based on the entities to be predicted, HKGC can be classified into two distinct categories:

- (1) *Head / Tail*: the entities to be predicted are exclusively located within the main triple, i.e., predict a missing entity ? in $(?, r, t, Q)$ or $(h, r, ?, Q)$.
- (2) *All Entities*: the entities to be predicted may not only reside within the main triple but could also be present in the qualifier pairs Q , i.e., predict a missing entity ? in $(?, r, t, Q)$ or $(h, r, ?, Q)$ or $\{(a_1 : v_1), \dots, (a_i : ?), \dots, (a_n : v_n)\}$.

Without loss of generality, in what follows we concentrate on predicting a tail entity ? in $(h, r, ?, Q)$. The other settings can be handled in a similar manner.

Qualifier Monotonicity. Let $q_1 = (h, r, ?, Q_1)$ and $q_2 = (h, r, ?, Q_2)$ be two queries that share the same main triple $(h, r, ?)$ and Q_1, Q_2 satisfy $Q_1 \subseteq Q_2$. We say that the *qualifier monotonicity property* holds over a hyper-relational knowledge graph $\mathcal{G}$ iff $Ans(q_2, \mathcal{G}) \subseteq Ans(q_1, \mathcal{G})$, where $Ans(q_i, \mathcal{G})$ denotes the set of answers of the query q_i over $\mathcal{G}$. Intuitively, this property implies that adding qualifier pairs to a query does not enlarge its answer set, and conversely, removing qualifier pairs can only result in more answers. For example, consider the queries $q_1 = (James\ Harden, member_of_team, ?)$ with $Q_1 = \{(start_time: 2019), (end_time: 2023)\}$ and $q_2 = (James\ Harden, member_of_team, ?)$ with $Q_2 = \{(start_time: 2019), (end_time: 2023), (teammate: P.J.\ Tucker)\}$ in Figure 1. It is clear that the main triple of q_1 and q_2 are the same and $Q_1 \subseteq Q_2$, and the corresponding answer sets $Ans(q_1, \mathcal{G}) = \{Brooklyn\ Nets, Houston\ Rockets, Los\ Angeles\ Clippers, Philadelphia\ 76ers\}$ and $Ans(q_2, \mathcal{G}) = \{Houston\ Rockets, Los\ Angeles\ Clippers, Philadelphia\ 76ers\}$ satisfy $Ans(q_2, \mathcal{G}) \subseteq Ans(q_1, \mathcal{G})$.

Cone Embedding. We present here only necessary background on cone embeddings, for a full discussion we refer the interested reader to [28, 29, 53]. A d -dimensional sector-cone can be parameterized by $C^d(\alpha, \beta) = ((\alpha^1, \beta^1), \dots, (\alpha^d, \beta^d))$, where $\alpha^i \in [-\pi, \pi)$ represents the angle between the symmetry axis of the i -th-dimension sector-cone and the positive x axis, and $\beta^i \in [0, 2\pi)$ represents the aperture of the i -th-dimension sector-cone. We omit the superscript of C^d if it is clear from context. A variety of operations on cone embeddings can be defined, here we introduce the projection $\mathcal{P}$ and intersection $\mathcal{I}$ operations:

- (1) *Projection Operator $\mathcal{P}$* . $C_2 = \mathcal{P}(C_1, C_r)$ computes the *projection* from an input cone C_1 as a *head* entity set to the set of *tail* entities C_2 via the specific relation cone C_r , see Figure 3(b).
- (2) *Intersection Operator $\mathcal{I}$* . $\mathcal{I}(C_1, C_2)$ computes the *intersection* of the set of entities C_1 and the set of entities C_2 , see Figure 3(c).

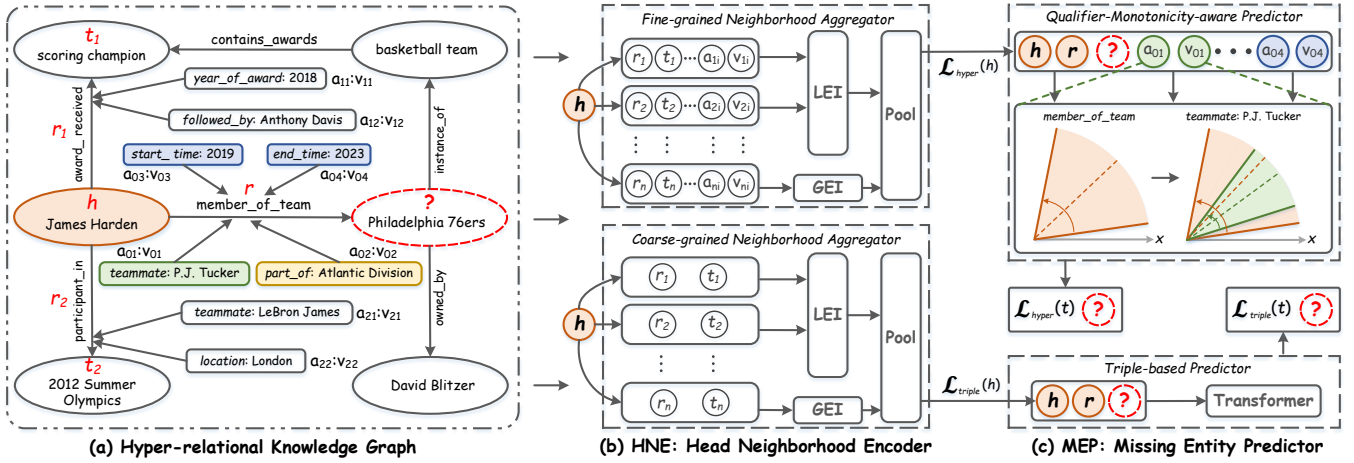


Figure 2: **HyperMono**’s model structure, containing two modules: Head Neighborhood Encoder (HNE) and Missing Entity Predictor (MEP). HNE module contains two sub-modules: Coarse-grained Neighborhood Aggregator (CNA) and Fine-grained Neighborhood Aggregator (FNA). MEP module contains two sub-modules: Triple-based Predictor (TP) and Qualifier-Monotonicity-aware Predictor (QMP).

3.2 Model Architecture

We present the two main components of **HyperMono**’s architecture: Head Neighborhood Encoder (§ 3.2.1) and Missing Entity Predictor (§ 3.2.2).

3.2.1 HNE: Head Neighborhood Encoder. To infer the missing tail entity in the triple $(h, r, ?)$, it is important to not only consider the input embeddings of h and r , but also look at how h relates to its neighboring entities. For instance, in Figure 2(a), for the triple $(\text{James Harden}, \text{member_of_team}, ?)$, the relational neighbors (*award_received*, *scoring_champion*) and (*participant_in*, *2012 Summer Olympics*) of the head entity *James Harden* can help us infer that *James Harden* is an athlete and not a movie star. The same phenomenon is witnessed for hyper-relational facts $(h, r, ?, Q)$, i.e., we can exploit the information about how h relates to its neighbors (including or not including its qualifiers) to obtain an enhanced head entity embedding representation.

To adequately encode the content of the neighbors of a head entity, as a first step, we introduce the **Head Neighborhood Encoder (HNE)** module, cf. Figure 2(b). The HNE module considers two types of relational contexts, characterized by the use of either main triples or complete hyper-relational facts to ascertain the neighbors of a given entity: **Coarse-grained Neighbor Aggregator (CNA)**, only considering main triples in $\mathcal{G}_{\text{triple}}$; **Fine-grained Neighbor Aggregator (FNA)**, considering hyper-relational facts in $\mathcal{G}$. The resulting neighbor-aware embeddings from the CNA and FNA modules will be used in the subsequent module (§3.2.2) to provide further evidence, based on the neighbors of h , for the prediction of the $?$ entity.

Coarse-grained Neighborhood Aggregator. For a main triple $(h, r, ?)$ from $\mathcal{G}_{\text{triple}}$, we use the following three steps to encode information about the neighbors of h into its representation:

- (1) *Neighbor-aware Embeddings.* Let $N_h = \{(h, r', t') \mid (h, r', t') \in \mathcal{G}_{\text{triple}}\}$ be the set of neighbors of h . For a triple $tr = (h, r', t') \in$

N_h , we use the placeholder $[\text{mask}]$ in place of h to obtain the neighbor sequence $\{[\text{mask}], r', t'\}$. Then, we apply a standard transformer encoder [38] to obtain the $[\text{mask}]$ token embedding $N_{tr}^h \in \mathbb{R}^{1 \times d}$, d represents the embedding dimension. Clearly, if $n = |N_h|$, n different $[\text{mask}]$ representations will be obtained. So, we sum up and average all $[\text{mask}]$ representations to obtain an aggregated embedding N^h considering all neighbors of h : $N^h = \text{mean}(\sum_{tr \in N_h} N_{tr}^h)$.

- (2) *Local and Global Entity Inference.* Different neighbors of h might provide information about h from different perspectives. We thus propose the **Local Entity Inference (LEI)** mechanism that independently considers each neighbor to infer the entity corresponding to h . LEI focuses on a single neighbor during inference, reducing the interference from irrelevant information. In practice, for $tr \in N_h$, we perform a matrix multiplication operation on N_{tr}^h and V_{ent} to obtain the entity score vector $P_{tr}^h \in \mathbb{R}^{1 \times N} = N_{tr}^h \cdot V_{ent}^T$, where $V_{ent} \in \mathbb{R}^{N \times d}$ denotes the embedding matrix for all entities in the dataset, V_{ent}^T is the transpose of V_{ent} , and $N = |\mathcal{E}|$. However, in some cases, it is difficult to infer a complex entity from a single neighbor. Therefore, we additionally propose a **Global Entity Inference (GEI)** mechanism which averages the entity score vector predicted by different relational neighbors to aggregate them: $P_G^h = \text{mean}(\sum_{tr \in N_h} P_{tr}^h)$.
- (3) *Aggregating Inference Results.* The LEI and GEI mechanisms will generate multiple entity inference results, to unify them, inspired by [15, 31], we adopt an exponentially weighted pooling method to generate the final inference result:

$$P^h = \text{pool}(\{P_{tr_1}^h, P_{tr_2}^h, \dots, P_{tr_n}^h, P_G^h\}) = \sum_{i=1}^{n,G} \omega_i P_{tr_i}^h \quad (1)$$

where $\omega_i = \frac{\exp \gamma P_{tr_i}^h}{\sum_{j=1}^{n,G} \exp \gamma P_{tr_j}^h}$, γ is a hyperparameter that controls the temperature of the pooling process. We further apply a

softmax function θ to $\mathbf{p}^h$: $\mathbf{p}^h = \theta(\mathbf{P}^h)$ to map the scores between 0 and 1. Finally, we calculate the cross-entropy loss between the prediction logit $\mathbf{p}^h$ and the corresponding label $\mathbf{L}_1^h$:

$$\mathcal{L}_{\text{triple}}(h) = \text{CrossEntropy}(\mathbf{p}^h, \mathbf{L}_1^h).$$

Fine-grained Neighborhood Aggregator. In this module, we follow similar steps as in the CNA module to obtain an aggregated representation of h , but with hyper-relational knowledge. For a hyper-relational fact $(h, r, ?, Q) \in \mathcal{H}$, in the *neighbor-aware embeddings step*, after masking out h for all its neighbors, we obtain $\mathbf{M}^h$ defined as $= \text{mean}(\sum_{hrf \in N_{h,H}} \mathbf{M}_{hrf}^h)$, where $N_{h,H} = \{(h, r', t', Q') \mid (h, r', t', Q') \in \mathcal{H}\}$. In the *local and global entity inference step*, we obtain entity inference results for all elements in $N_{h,H}$: $\{Q_{hrf_1}^h, Q_{hrf_2}^h, \dots, Q_{hrf_m}^h, Q_G^h\}$, such that $|N_{h,H}| = m$. In the *aggregating inference results step*, we also employ a pooling strategy to aggregate multiple inference results into a single prediction $\mathbf{Q}^h$. Finally, we calculate the cross-entropy loss between the $\mathbf{q}^h = \theta(\mathbf{Q}^h)$ and the true label $\mathbf{L}_2^h$ as: $\mathcal{L}_{\text{hyper}}(h) = \text{CrossEntropy}(\mathbf{q}^h, \mathbf{L}_2^h)$.

3.2.2 MEP: Missing Entity Predictor. The CNA and FNA modules introduced the supervision signals $\mathcal{L}_{\text{triple}}(h)$ and $\mathcal{L}_{\text{hyper}}(h)$ for the possible values of h . However, in the hyper-relational knowledge graph completion task, the neighborhood information of the head entity h can be used as prior knowledge to infer the missing placeholder $?$ results. To effectively make use of the neighborhood information, while faithfully capturing the two-stage reasoning and qualifier monotonicity properties, we introduce the *Missing Entity Predictor (MEP)* module. The *Triple-based Predictor (TP)* component of MEP accomplishes coarse-grained inferences by solely considering main triples. Based on the results from the TP sub-module, the *Qualifier-Monotonicity-aware Predictor (QMP)* subsequently obtains fine-grained inferences by considering hyper-relational facts with qualifiers, while properly representing the qualifier monotonicity property through cone embeddings. We note that the interaction between the coarse-grained and the fine-grained stages is implicitly captured through the loss function calculation. This naturally models the interaction between these stages, without having to look for ‘correct’ restrictions to explicitly capture it. We next describe the details of the TP and QMP components of MEP.

Triple-based Predictor. For an incomplete triple $(h, r, ?)$, we employ the following three steps to predict entities at the placeholder $?$ location: (i) We replace the placeholder $?$ with the $[\text{mask}]$ token to obtain input sequences $\{h, r, [\text{mask}]\}$. Further, we use the $\mathbf{N}^h$ embedding obtained at Step (1) of the CNA module as the input embedding of h , and randomly initialize the input embedding of r . (ii) We use a transformer encoder to obtain token embeddings $\{\mathbf{F}^h, \mathbf{F}^r, \mathbf{F}^{[\text{mask}]}\}$. (iii) Finally, we use $\mathbf{p}^t = \theta(\mathbf{F}^{[\text{mask}]} \cdot \mathbf{V}_{\text{ent}}^T)$ to infer the most likely probability for entities at the mask position. Additionally, we calculate the cross-entropy loss between the prediction logit $\mathbf{p}^t$ and ground truth label $\mathbf{L}_1^t$ as $\mathcal{L}_{\text{triple}}(t) = \text{CrossEntropy}(\mathbf{p}^t, \mathbf{L}_1^t)$.

Qualifier-Monotonicity-aware Predictor. The previous module only considers the main triple $(h, r, ?)$ in $(h, r, ?, Q = \{(a_1 : v_1), \dots, (a_s : v_s)\})$, disregarding qualifier pairs. To fully leverage the qualifier pairs information in the hyper-relational fact, building

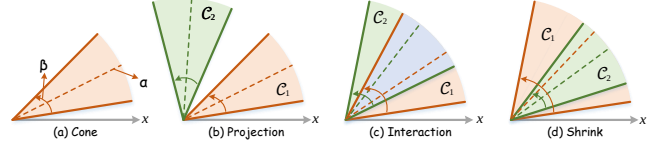


Figure 3: An overview of cone embedding with projection $\mathcal{P}$, intersection $\mathcal{I}$ and shrink $\mathcal{S}$, the embedding dimension $d=1$.

upon the qualifier monotonicity property, we propose the *Qualifier-Monotonicity-aware Predictor* module, composed of the following three sub-modules:

Token Embeddings. Like in the *triple-based predictor* module, we first replace the placeholder $?$ with the $[\text{mask}]$ token to obtain the input sequence $\{h, r, [\text{mask}], \{(a_1, v_1), \dots, (a_s, v_s)\}\}$. We initialize the input embedding of h using the $\mathbf{M}^h$ embedding obtained in the FNA module, and the remaining input embeddings of r , a_i , and v_i are randomly initialized. Then, we use a transformer encoder to obtain the output embedding representation of each token: $\mathbf{S} = \{\mathbf{E}^h, \mathbf{E}^r, \mathbf{E}^{[\text{mask}]}, \{(\mathbf{E}^{a_1}, \mathbf{E}^{v_1}), \dots, (\mathbf{E}^{a_s}, \mathbf{E}^{v_s})\}\}$.

Cone Shrinking. To faithfully capture the monotonicity property, we will resort to a geometric representation based on cone embeddings. We represent main triples as cones and each qualifier is modeled as a cone that shrinks the cone of the main triple to which the qualifier has been added. This shrinkage will be implemented through a reduction of the axis and aperture of main triples cones. Thus, our cone-based representation effectively captures that attaching qualifiers to a main triple might narrow down the answer set, but never enlarge it. Technically, our goal is to predict the most likely entity category of $\mathbf{E}^{[\text{mask}]}$ using within the cone space the different types of token embeddings in $\mathbf{S} = \{\mathbf{E}^h, \mathbf{E}^r, \mathbf{E}^{[\text{mask}]}, \{(\mathbf{E}^{a_1}, \mathbf{E}^{v_1}), \dots, (\mathbf{E}^{a_s}, \mathbf{E}^{v_s})\}\}$. To this aim, we introduce the *Cone Shrink Block (CSB)* module, comprising the following three steps. We also prove that cones respect the qualifier monotonicity property in [Appendix I](#).

- (1) We first convert the different types of token embeddings in $\mathbf{S}$ into the cone space. Specifically, the head entity token embedding $\mathbf{E}^h$ is converted to the cone $C_h = C(\alpha_h, \beta_h)$, and the relation to the cone $C_r = C(\alpha_r, \beta_r)$, with $\alpha_h, \beta_h, \alpha_r$, and β_r defined as follows:

$$\begin{cases} \alpha_h = f(\text{MLP}(\mathbf{E}^h)), \beta_h = g(\text{MLP}(\mathbf{E}^h)) \\ \alpha_r = f(\text{MLP}(\mathbf{E}^r)), \beta_r = g(\text{MLP}(\mathbf{E}^r)) \\ f(x_i) = \pi \tanh(\lambda_1 x_i), g(y_i) = \pi \tanh(\lambda_2 y_i) + \pi \end{cases} \quad (2)$$

where $\text{MLP}: \mathbb{R}^d \rightarrow \mathbb{R}^d$ is a multi-layer perceptron network, the functions f and g are used to scale the semantic center axis and aperture into their normal ranges: $\alpha_h = f(\mathbf{x}) = [\alpha_h^1, \dots, \alpha_h^d]$, $\alpha_h^i = f(x_i)$, α_h^i and $f(x_i)$ denote the i -th element of α_h and $f(\mathbf{x})$; $\beta_h = g(\mathbf{y}) = [\beta_h^1, \dots, \beta_h^d]$, $\beta_h^i = g(y_i)$, β_h^i and $g(y_i)$ denote the i -th element of β_h and $g(\mathbf{y})$, λ_1 and λ_2 are two scaling hyper-parameters. α_r and β_r are defined similarly.

We then map the embedding region of the head entity set C_h to the entity set $C_{h,r}$ through a relation-dependent projection operation $\mathcal{P}$: $C_{h,r} = \mathcal{P}(C_h, C_r) = \mathcal{P}(C_h, C_r)$.

- (2) As discussed, the monotonicity property can be intuitively captured by reducing the axis and aperture of main triple cones to which qualifiers are added. More precisely, to geometrically represent this property in the cone embedding space, we model each qualifier ($a_i : v_i$) as a shrinking of a cone. The spatial representation is such that the cone obtained after shrinking is a sub-cone of the original one. For example, in Figure 3(d), the shrink cone C_2 is a subset of the original cone C_1 . Given a cone $C(\alpha_{h,r}, \beta_{h,r})$ obtained in the previous step and the token embedding (E^{a_i}, E^{v_i}) of the i -th qualifier pair, we formally define a cone-to-cone shrink transformation from $C(\alpha_{h,r}, \beta_{h,r})$ to $C(\alpha_{S_i}, \beta_{S_i})$ using Equation (3) below. Intuitively, Equation (3) adjusts the source angle of axis $\alpha_{h,r}$ and the aperture $\beta_{h,r}$ to shrink them to α_{S_i} and β_{S_i} , respectively, ensuring that the resulting cone is contained in the original cone.

$$\begin{cases} \alpha_{r,a_i,v_i} = f(\text{MLP}_1(\Theta_{r,a_i,v_i})), \beta_{r,a_i,v_i} = g(\text{MLP}_1(\Theta_{r,a_i,v_i})) \\ \text{offset} = (\beta_{h,r} - \beta_{S_i}) \odot \sigma(\alpha_{r,a_i,v_i}), \beta_{S_i} = \sigma(\beta_{r,a_i,v_i}) \odot \beta_{h,r} \\ \alpha_{S_i} = \alpha_{h,r} - 0.5 \odot \beta_{h,r} + 0.5 \odot \beta_{S_i} + 0.5 \odot \text{offset} \end{cases} \quad (3)$$

where $\Theta_{r,a_i,v_i} = \text{MLP}_2(\text{Concat}(E^r, E^{a_i}, E^{v_i}))$, which leverages an MLP layer which takes the primary relation and attribute-value qualifier as input, and outputs the combined representation of relational and hyper-relational knowledge. $\text{MLP}_1: \mathbb{R}^d \rightarrow \mathbb{R}^d$ and $\text{MLP}_2: \mathbb{R}^{3d} \rightarrow \mathbb{R}^d$ are multi-layer perceptron networks, σ is a *sigmoid* function and $\odot$ is the element-wise vector multiplication. From Equation (3), we obtain a sector-cone $C_{S_i} = C(\alpha_{S_i}, \beta_{S_i})$ that has been shrunk by the qualifier ($a_i : v_i$).

- (3) After obtaining a scaled down sector-cone C_{S_i} , for each i -th qualifier, we obtain s independent cones: $\{C_{S_1}, \dots, C_{S_s}\}$. Finally, when considering multiple qualifier pairs at the same time, the answer set can be computed by the cone intersection operation $I: C_S = (\alpha_S, \beta_S) = I(C_{S_1}, \dots, C_{S_s})$. Note that when C_S is empty, the intersection of all C_{S_i} must form a cone with an aperture of 0. However, this scenario is highly improbable. The key reason is that each cone exists in a high-dimensional space (with a dimensionality of 200 for our HyperMono), making it virtually impossible for all β values in C_S to be 0. Even in extreme cases where the aperture is 0, the cone embedding remains effective in handling this situation, as discussed in ConE [53].

Inference Results. We apply $q^t = f(\text{MLP}(\text{Concat}(\alpha_S, \beta_S)), \mathbf{V}_{ent}^T)$, where $f(\cdot) = \mathcal{D}(\mathbf{V}_{ent}, C_S)$ denotes the score function to calculate the distance between $\mathbf{V}_{ent}$ and the scaled cone C_S ; the distance function is consistent with ConE [53], $q^t = \theta(q^t)$ to change the answer cone C_S to the entity probability at the ? position, θ is the *softmax* function. Additionally, we calculate the cross-entropy loss between the prediction score q^t and the label L_2^t as $\mathcal{L}_{hyper}(t) = \text{CrossEntropy}(q^t, L_2^t)$.

3.2.3 Joint Training. We combine the loss functions obtained above into a joint loss function:

$$\mathcal{L}_{joint} = \mathcal{L}_{triple}(h) + \mathcal{L}_{hyper}(h) + \mathcal{L}_{triple}(t) + \mathcal{L}_{hyper}(t) \quad (4)$$

4 Experiments

To evaluate the effectiveness of our model, we aim to explore the following five research questions:

- **RQ1 (Effectiveness):** How our model performs compared to the state-of-the-art under different conditions? We also present the Case study in [Appendix H](#).
- **RQ2 (Ablation studies):** How different components contribute to its performance?
- **RQ3 (Parameter sensitivity):** How hyper-parameters influence its performance? Additional results can be found in [Appendix D](#) and [Appendix F](#).
- **RQ4 (Complexity analysis):** What is the amount of computation and parameters used by HyperMono? Results can be found in [Appendix E](#).
- **RQ5 (Model transferability):** Can HyperMono be used to complete *hypergraphs*? Results can be found in [Appendix G](#).

4.1 Experiment Setup

4.1.1 Datasets. We conduct experiments on three hyper-relational KGs: WD50K [11], WikiPeople [14], and JF17K [43], where WD50K and WikiPeople have been extracted from Wikidata [39] and JF17K from Freebase [2]. We also construct three different sub-datasets, namely: *Mixed-percentage Mixed-qualifier*, *Fixed-percentage Fixed-qualifier*, and *Fixed-percentage Mixed-qualifier*. Statistics and details on the construction can be found in [Appendix A](#).

4.1.2 Baselines. We compare HyperMono with fourteen state-of-the-art baselines for HKGC, including *Embedding-based Methods*: mTransH [43], RAE [52], NaLP-Fix [33], NeuInfer [12], HINGE [33], and ShrinkE [45]; *Transformer-based with GNNs based Methods*: StarE [11] and QUAD [34]; *Transformer-based without GNNs based Methods*: Hy-Transformer [50], GRAN [41], HyNT [7], HAHE [27], HyperFormer [16], and HyperCL [26]. We note that although HELIOS [24], sHINGE [25] and NYLON [51] also tackle the HKGC task, there are some variations in terms of experimental design and the selection of evaluation metrics, e.g., HELIOS [24] selects MAP and NDCG as evaluation metrics, sHINGE [25] heavily relies on additional schema knowledge and NYLON [51] employs data augmentation techniques. Thus, we do not consider them as baselines. Detailed baseline descriptions, evaluation metrics and implementation details can respectively be found in [Appendix B](#) and [Appendix C](#).

4.2 Main Results

To address RQ1, we conduct experiments on the three types of datasets described above. The corresponding results are shown in Table 1, Table 2 and Table 3. As shown in Table 1, the improvement in predicting head and tail entities by HyperMono is less pronounced compared to the improvement on all entities. Consequently, Tables 2 and 3 focus more on the prediction of head and tail entities, which pose greater challenges for HyperMono.

Mixed-percentage Mixed-qualifier. Table 1 summarizes the results on WD50K, WikiPeople, and JF17K under the mixed-percentage mixed-qualifier conditions. We observe that HyperMono outperforms existing SoTA baselines by a large margin across most metrics

Table 1: Evaluation of different models under the mixed-percentage mixed-qualifier scenario on the WD50K, WikiPeople and JF17K datasets. All baseline results are collected from the original papers. Best scores are highlighted in **bold, the second best scores are highlighted in **normal**, the Δ represents the performance improvement over the state-of-the-art modes, and ‘-’ indicates the results are not reported in previous work. The number in parentheses represents the proportion of triples with hyper-relational knowledge.**

Methods	WD50K (13.6)						WikiPeople (2.6)						JF17K (45.9)					
	Head / Tail			All Entities			Head / Tail			All Entities			Head / Tail			All Entities		
	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 [43]	-	-	-	-	-	-	0.063	0.063	0.300	-	-	-	0.206	0.206	0.462	0.102	0.069	0.168
RAE [52]	-	-	-	-	-	-	0.058	0.058	0.306	0.172	0.102	0.320	0.215	0.215	0.466	0.310	0.219	0.504
NaLP-Fix [33]	0.177	0.131	0.264	0.224	0.158	0.330	0.408	0.331	0.546	0.338	0.272	0.466	0.221	0.165	0.331	0.366	0.290	0.516
NeuInfer [12]	0.243	0.176	0.377	0.228	0.162	0.341	0.476	0.415	0.585	0.333	0.259	0.477	0.449	0.361	0.624	0.473	0.397	0.618
HINGE [33]	0.243	0.176	0.377	0.232	0.164	0.343	0.342	0.272	0.463	0.350	0.282	0.467	0.431	0.342	0.611	0.517	0.436	0.675
ShrinkE [45]	0.345	0.275	0.482	-	-	-	0.485	0.431	0.601	-	-	-	0.589	0.506	0.749	-	-	-
StarE [11]	0.349	0.271	0.496	-	-	-	0.491	0.398	0.592	0.378	0.265	0.542	0.574	0.496	0.725	0.542	0.454	0.685
QUAD [34]	0.349	0.275	0.489	-	-	-	0.497	0.431	0.617	-	-	-	0.596	0.519	0.751	-	-	-
Hy-Transformer [50]	0.356	0.281	0.498	-	-	-	0.501	0.426	0.634	-	-	-	0.582	0.501	0.742	-	-	-
GRAN [41]	-	-	-	0.309	0.240	0.441	0.503	0.438	0.620	0.479	0.410	0.604	0.617	0.539	0.770	0.656	0.582	0.799
HyNT [7]	0.314	0.241	0.454	-	-	-	0.464	0.380	0.601	-	-	-	0.554	0.475	0.711	-	-	-
HAHE [27]	0.368	0.291	0.516	0.402	0.327	0.546	0.509	0.447	0.639	0.495	0.420	0.631	0.623	0.554	0.806	0.668	0.597	0.816
HyperFormer [16]	0.366	0.288	0.514	0.417	0.344	0.555	0.491	0.386	0.654	0.479	0.373	0.644	0.664	0.601	0.787	0.662	0.596	0.790
HyperCL [26]	0.302	0.231	0.441	0.395	0.321	0.539	0.455	0.365	0.596	0.509	0.437	0.644	0.513	0.429	0.684	0.673	0.604	0.818
HyperMono	0.375	0.298	0.522	0.447	0.376	0.582	0.494	0.390	0.657	0.498	0.394	0.659	0.685	0.630	0.796	0.766	0.717	0.861
Δ (%)	+0.7	+0.7	+0.6	+3.0	+3.2	+2.7	-1.5	-5.7	+0.3	-1.1	-4.3	+1.5	+2.1	+2.9	-1.0	+9.3	+11.3	+4.3

Table 2: Evaluation of different models under the fixed-percentage fixed-qualifier scenario on the WikiPeople and JF17K datasets. Best scores are highlighted in **bold, the second best scores are highlighted in **normal**.**

Methods	WikiPeople-3				WikiPeople-4				JF17K-3				JF17K-4			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
ShrinkE [45]	0.366	0.291	0.397	0.515	0.222	0.155	0.252	0.353	0.712	0.648	0.745	0.837	0.779	0.726	0.809	0.886
StarE [11]	0.401	0.310	0.434	0.592	0.243	0.156	0.269	0.430	0.707	0.635	0.744	0.847	0.723	0.669	0.753	0.839
QUAD [41]	0.403	0.321	0.438	0.563	0.251	0.167	0.280	0.425	0.730	0.660	0.767	0.870	0.787	0.730	0.823	0.895
Hy-Transformer [50]	0.403	0.323	0.436	0.569	0.248	0.165	0.275	0.422	0.690	0.617	0.725	0.837	0.773	0.717	0.806	0.875
GRAN [41]	0.397	0.328	0.429	0.533	0.239	0.178	0.261	0.364	0.779	0.724	0.811	0.893	0.798	0.744	0.830	0.904
HyNT [7]	0.383	0.306	0.415	0.533	0.233	0.166	0.257	0.369	0.733	0.664	0.771	0.873	0.779	0.725	0.809	0.881
HAHE [27]	0.397	0.323	0.432	0.537	0.251	0.179	0.286	0.392	0.762	0.701	0.791	0.892	0.736	0.666	0.773	0.869
HyperFormer [16]	0.573	0.511	0.603	0.693	0.393	0.336	0.415	0.496	0.832	0.790	0.855	0.914	0.857	0.811	0.884	0.937
HyperCL [26]	0.397	0.327	0.425	0.538	0.247	0.175	0.279	0.387	0.766	0.707	0.796	0.890	0.733	0.654	0.779	0.874
HyperMono	0.586	0.531	0.611	0.690	0.398	0.359	0.408	0.474	0.867	0.839	0.880	0.927	0.881	0.847	0.901	0.945

on WD50K and JF17K in the head/tail entities and all entities scenarios. In particular, in the prediction of head and tail entities, HyperMono respectively achieves 2.1% and 2.9% improvements over HyperFormer (the best baseline) in MRR and Hits@1 on the JF17K dataset. A similar phenomenon occurred on the WD50K dataset. However, for the WikiPeople dataset, even though we can achieve optimal performance on the Hits@10 metric, we cannot surpass the best performing baseline on MRR and Hits@1 in the head/tail and all entities settings. The intuitive reason for this is that, in comparison to the WD50K and JF17K datasets where 13.6% and 45.9% of the main triples respectively contain hyper-relational knowledge, only 2.6% of triples in WikiPeople have qualifier pairs. This hinders HyperMono’s ability to exploit stage reasoning and

qualifier monotonicity. Specifically, we employ a scaling transformation in the cone space to capture qualifier monotonicity, but the lack of qualifier pairs constrains this functionality. Furthermore, the proportion of triples with hyper-relational knowledge in JF17K is higher than that in WD50K. We can observe that the performance improvement in JF17K is more significant, which further confirms the importance of the richness of hyper-relational knowledge.

Fixed-percentage Fixed-qualifier. To better understand the impact of hyper-knowledge on the performance, we consider the WikiPeople and JF17K datasets with a fixed number of qualifiers, cf. Table 2. We can observe that HyperMono consistently achieves optimal performance on almost all datasets. Specifically, in terms of the MRR metric, HyperMono respectively shows improvements of 3.5% and 2.4% on the JF17K-3 and JF17K-4 subsets when compared

Table 3: Evaluation of different models under the fixed-percentage mixed-qualifier scenario on the WD50K, WikiPeople and JF17K datasets. Best scores are highlighted in **bold, the second best scores are highlighted in **normal**.**

Methods	WD50K						WikiPeople						JF17K					
	33%		66%		100%		33%		66%		100%		33%		66%		100%	
	MRR	H@1	MRR	H@1	MRR	H@1	MRR	H@1	MRR	H@1	MRR	H@1	MRR	H@1	MRR	H@1	MRR	H@1
ShrinkE [45]	0.246	0.190	0.356	0.294	0.470	0.404	0.176	0.129	0.244	0.198	0.332	0.276	0.259	0.176	0.263	0.185	0.299	0.220
StarE [11]	0.308	0.247	0.449	0.388	0.610	0.543	0.192	0.143	0.259	0.205	0.343	0.279	0.290	0.197	0.302	0.214	0.321	0.223
QUAD [41]	0.329	0.266	0.479	0.416	0.646	0.572	0.204	0.155	0.282	0.228	0.385	0.318	0.307	0.210	0.334	0.241	0.379	0.277
Hy-Transformer [50]	0.313	0.255	0.458	0.397	0.621	0.557	0.192	0.140	0.268	0.215	0.372	0.316	0.298	0.204	0.325	0.234	0.361	0.266
GRAN [41]	0.322	0.269	0.472	0.419	0.647	0.593	0.201	0.156	0.287	0.244	0.403	0.349	0.307	0.212	0.326	0.237	0.382	0.290
HyNT [7]	0.295	0.238	0.438	0.377	0.595	0.525	0.175	0.130	0.243	0.192	0.329	0.269	0.285	0.195	0.298	0.216	0.352	0.261
HAHE [27]	0.301	0.239	0.432	0.368	0.607	0.537	0.188	0.142	0.263	0.213	0.367	0.304	0.311	0.221	0.330	0.248	0.387	0.293
HyperFormer [16]	0.338	0.280	0.492	0.434	0.666	0.611	0.213	0.161	0.298	0.255	0.426	0.373	0.352	0.254	0.411	0.325	0.478	0.396
HyperCL [26]	0.263	0.208	0.388	0.334	0.632	0.566	0.155	0.109	0.221	0.177	0.363	0.303	0.284	0.191	0.300	0.218	0.386	0.290
HyperMono	0.341	0.280	0.490	0.433	0.667	0.612	0.221	0.173	0.309	0.264	0.427	0.379	0.357	0.267	0.423	0.340	0.493	0.408

Table 4: Ablation studies about different components. Best scores are highlighted in bold.

Methods	WikiPeople-3				WikiPeople-4				JF17K-3				JF17K-4			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
w/o CNA+TP	0.559	0.493	0.591	0.686	0.367	0.303	0.382	0.477	0.832	0.792	0.855	0.912	0.860	0.820	0.885	0.940
w/o FNA+QMP	0.560	0.510	0.578	0.663	0.377	0.340	0.383	0.445	0.839	0.811	0.848	0.900	0.842	0.809	0.859	0.909
w/o LEI	0.576	0.523	0.599	0.678	0.394	0.359	0.400	0.465	0.860	0.834	0.871	0.918	0.878	0.843	0.898	0.946
w/o GEI	0.584	0.527	0.606	0.687	0.396	0.355	0.404	0.471	0.866	0.838	0.878	0.926	0.877	0.842	0.898	0.941
w/o CSB	0.565	0.502	0.595	0.687	0.355	0.296	0.380	0.464	0.831	0.789	0.853	0.915	0.854	0.815	0.881	0.933
HyperMono	0.586	0.531	0.611	0.690	0.398	0.359	0.408	0.474	0.867	0.839	0.880	0.927	0.881	0.847	0.901	0.945

with HyperFormer. Furthermore, we find that for all models, their performance in the fixed-percentage fixed-qualifier setting is generally higher than in the fixed-percentage mixed-qualifier one. The main reason is that in the fixed-percentage fixed-qualifier scenario, the number of qualifier pairs for each triple is fixed, and there is no imbalance in the distribution of hyper-relational knowledge.

Fixed-percentage Mixed-qualifier. Table 3 presents the results on WD50K, WikiPeople, and JF17K under different fixed ratios of relational facts with qualifiers. For each dataset, we construct three subsets (cf. § 4.1.1, Point 2) containing approximately 33%, 66%, and 100% of facts with qualifiers. We observe that under different proportions of hyper-relational knowledge HyperMono also achieves optimal performance. Particularly, for the WikiPeople dataset, in which HyperMono performs poorly in the mixed-percentage mixed-qualifier scenario, adding a certain percentage of qualifiers information to triples results in HyperMono outperforming all baselines. For example, for the Hits@1 metric, HyperMono respectively achieves improvements over the best performing baseline, HyperFormer, of 1.2% / 0.9% / 0.6% in the 33% / 66% / 100% variants. This confirms that the proportion of hyper-relational knowledge is a major factor influencing the performance over WikiPeople under the mixed-percentage mixed-qualifier condition. In contrast, on the WD50K dataset, no significant additional improvement was obtained. The most likely reason is the insufficient richness of hyper-relational knowledge in WD50K. For example, in WD50K_100 65.23% of the triples only contain *one* hyper-relation pair, which limits the functionality of qualifier monotonicity module.

4.3 Ablation Studies

To address RQ2, we conduct ablation experiments on WikiPeople-3, WikiPeople-4, JF17K-3, and JF17K-4 subsets to verify the contribution of each component of HyperMono. These include the following three aspects: a) whether the knowledge of qualifiers is considered to infer missing entities, see rows “w/o CNA+TP” and “w/o FNA+QMP” in Table 4; b) different neighbor aggregation strategies, see rows “w/o LEI” and “w/o GEI” in Table 4; c) without the cone shrink operation in QMP, see row “w/o CSB” in Table 4.

Different Variants of HyperMono. In Table 4, we observe that under most metrics removing any functional module will bring some performance degradation, which justifies the necessity of each component. Specifically, a) removing either CNA+TP, which are related to coarse-grained answer prediction, or FNA+QMP, which are related to with fine-grained answer prediction, will result in a significant loss of accuracy. The intuitive reason is that the coarse-grained answer set can be used to constrain the decision space of fine-grained answers, preventing the occurrence of answer drift. Meanwhile, the fine-grained answer set allows to refine the answer set, ensuring the reliability of the predicted results. b) The neighborhood information of known entities has a positive effect on predicting missing entities, and the use of LEI for integrating the neighbors of known entities has a greater impact on inference than GEI. c) Removing the CSB module will bring a significant decrease in performance. The main reason is that the CSB module captures the qualifier monotonicity of hyper-relational KGs. The main characteristic of the HKGC task is the application of qualifier knowledge in triples to infer answer sets. Through the natural

scaling transformation of the cone space, it can be captured that attaching qualifiers to a triple can only narrow down the answer set, but never enlarge it.

5 Conclusions

In this paper, we propose `HyperMono`, a framework for hyper-relational graph completion that properly implements two-stage reasoning and faithfully captures qualifier monotonicity. Specifically, we model each qualifier as a shrinking of the main triple cone to a qualifier cone to realize qualifier monotonicity. Empirical experiments under various settings and datasets demonstrate `HyperMono`'s robust performance, when compared to competitive baselines. Additionally, we present various ablation studies on parameter sensitivity, complexity analysis and model transferability. For future work, we believe that it is necessary to pay special attention to numerical attribute knowledge in hyper-relational knowledge. How to elegantly integrate schema knowledge into the representation process is also an interesting line of research.

References

- [1] Ivana Balazevic, Carl Allen, and Timothy M. Hospedales. 2019. TuckER: Tensor Factorization for Knowledge Graph Completion. In *EMNLP. ACL*, Hong Kong, China, 5184–5193.
- [2] Kurt D. Bollacker, Colin Evans, Praveen K. Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a Collaboratively Created Graph Database for Structuring Human Knowledge. In *SIGMOD. ACM*, Vancouver, BC, Canada, 1247–1250.
- [3] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *NeurIPS. Curran Associates*, Lake Tahoe, Nevada, United States, 2787–2795.
- [4] Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, and Christopher Ré. 2020. Low-Dimensional Hyperbolic Knowledge Graph Embeddings. In *ACL. ACL*, Online, 6901–6914.
- [5] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In *ICLR. OpenReview.net*, Vancouver, Canada, 1–15.
- [6] Zirui Chen, Xin Wang, Chenxu Wang, and Jianxin Li. 2022. Explainable Link Prediction in Knowledge Hypergraphs. In *CIKM. ACM*, Atlanta, GA, USA, 262–271.
- [7] Chanyoung Chung, Jaehun Lee, and Joyce Jiyoung Whang. 2023. Representation Learning on Hyper-Relational and Numeric Knowledge Graphs with Transformers. In *KDD. ACM*, Long Beach, CA, USA, 310–322.
- [8] Shimin Di and Lei Chen. 2023. Message Function Search for Knowledge Graph Embedding. In *WWW. ACM*, Austin, TX, USA, 2633–2644.
- [9] Shimin Di, Quanming Yao, and Lei Chen. 2021. Searching to Sparsify Tensor Decomposition for N-ary Relational Data. In *WWW. ACM*, online, 4043–4054.
- [10] Bahare Fatemi, Perouz Taslakian, David Vázquez, and David Poole. 2020. Knowledge Hypergraphs: Prediction Beyond Binary Relations. In *IJCAI. ijcai.org*, online, 2191–2197.
- [11] Mikhail Galkin, Priyansh Trivedi, Gaurav Maheshwari, Ricardo Usbeck, and Jens Lehmann. 2020. Message Passing for Hyper-Relational Knowledge Graphs. In *EMNLP. ACL*, online, 7346–7359.
- [12] Saiping Guan, Xiaolong Jin, Jiafeng Guo, Yuanzhuo Wang, and Xueqi Cheng. 2020. NeuInfer: Knowledge Inference on N-ary Facts. In *ACL. ACL*, online, 6141–6151.
- [13] Saiping Guan, Xiaolong Jin, Jiafeng Guo, Yuanzhuo Wang, and Xueqi Cheng. 2023. Link Prediction on N-ary Relational Data Based on Relatedness Evaluation. *TKDE* 35 (2023), 672–685.
- [14] Saiping Guan, Xiaolong Jin, Yuanzhuo Wang, and Xueqi Cheng. 2019. Link Prediction on N-ary Relational Data. In *WWW. ACM*, San Francisco, CA, USA, 583–593.
- [15] Zhiwei Hu, Víctor Gutiérrez-Basulto, Zhiliang Xiang, Ru Li, and Jeff Z. Pan. 2022. Transformer-based Entity Typing in Knowledge Graphs. In *EMNLP. ACL*, Abu Dhabi, United Arab Emirates, 5988–6001.
- [16] 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 *CIKM. ACM*, Birmingham, UK, 803–812.
- [17] Zhao Li, Chenxu Wang, Xin Wang, Zirui Chen, and Jianxin Li. 2024. HJE: Joint Convolutional Representation Learning for Knowledge Hypergraph Completion. *TKDE* (2024).
- [18] Zhao Li, Xin Wang, Jianxin Li, Wenbin Guo, and Jun Zhao. 2024. HyCubE: Efficient Knowledge Hypergraph 3D Circular Convolutional Embedding. *CoRR abs/2402.08961* (2024), 1–11.
- [19] Xiao Liu, Shiyu Zhao, Kai Su, Yukuo Cen, Jiezhong Qiu, Mengdi Zhang, Wei Wu, Yuxiao Dong, and Jie Tang. 2022. Mask and Reason: Pre-Training Knowledge Graph Transformers for Complex Logical Queries. In *KDD. ACM*, Washington, DC, USA, 1120–1130.
- [20] Yang Liu, Zequn Sun, Guangyao Li, and Wei Hu. 2022. I Know What You Do Not Know: Knowledge Graph Embedding via Co-distillation Learning. In *CIKM. ACM*, Atlanta, GA, USA, 1329–1338.
- [21] Yu Liu, Quanming Yao, and Yong Li. 2020. Generalizing Tensor Decomposition for N-ary Relational Knowledge Bases. In *WWW. ACM*, Taipei, China, 1104–1114.
- [22] Yu Liu, Quanming Yao, and Yong Li. 2021. Role-Aware Modeling for N-ary Relational Knowledge Bases. In *WWW. ACM*, Ljubljana, Slovenia, 2660–2671.
- [23] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *ICLR. OpenReview.net*, New Orleans, LA, USA, 1–11.
- [24] Yuhuan Lu, Bangchao Deng, Weijian Yu, and Dingqi Yang. 2023. HELIOS: Hyper-Relational Schema Modeling from Knowledge Graphs. In *MM. ACM*, Ottawa, ON, Canada, 4053–4064.
- [25] Yuhuan Lu, Dingqi Yang, Pengyang Wang, Paolo Rosso, and Philippe Cudre-Mauroux. 2023. Schema-Aware Hyper-Relational Knowledge Graph Embeddings for Link Prediction. *TKDE* (2023).
- [26] Yuhuan Lu, Weijian Yu, Xin Jing, and Dingqi Yang. 2024. HyperCL: A Contrastive Learning Framework for Hyper-Relational Knowledge Graph Embedding with Hierarchical Ontology. In *ACL. ACL*, Bangkok, Thailand, 2918–2929.
- [27] Haoran Luo, Haihong E, Yuhao Yang, Yikai Guo, Mingzhi Sun, Tianyu Yao, Zichen Tang, Kaiyang Wan, Meina Song, and Wei Lin. 2023. HAHE: Hierarchical Attention for Hyper-Relational Knowledge Graphs in Global and Local Level. In *ACL. ACL*, Toronto, Canada, 8095–8107.
- [28] Chau Nguyen, Tim French, Wei Liu, and Michael Stewart. 2023. CylE: Cylinder Embeddings for Multi-hop Reasoning over Knowledge Graphs. In *EACL. ACL*, Dubrovnik, Croatia, 1728–1743.
- [29] Chau Nguyen, Tim French, Wei Liu, and Michael Stewart. 2023. SConE: Simplified Cone Embeddings with Symbolic Operators for Complex Logical Queries. In *ACL. ACL*, Toronto, Canada, 11931–11946.
- [30] Dai Quoc Nguyen, Thanh Vu, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Q. Phung. 2019. A Capsule Network-based Embedding Model for Knowledge Graph Completion and Search Personalization. In *NAACL. NAACL-HLT*, Minneapolis, the United States, 2180–2189.
- [31] Weiran Pan, Wei Wei, and Xian-Ling Mao. 2021. Context-aware Entity Typing in Knowledge Graphs. In *EMNLP. ACL*, online, 2240–2250.
- [32] Hongyu Ren and Jure Leskovec. 2020. Beta Embeddings for Multi-Hop Logical Reasoning in Knowledge Graphs. In *NeurIPS*. Online.
- [33] Paolo Rosso, Dingqi Yang, and Philippe Cudré-Mauroux. 2020. Beyond Triplets: Hyper-Relational Knowledge Graph Embedding for Link Prediction. In *WWW. ACM*, Taipei, China, 1885–1896.
- [34] Harry Shomer, Wei Jin, Juan-Hui Li, Yao Ma, and Jiliang Tang. 2022. Learning Representations for Hyper-Relational Knowledge Graphs. *CoRR abs/2208.14322* (2022), 1–10.
- [35] Zhiqing Sun, ZhiHong Deng, JianYun Nie, and Jian Tang. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *ICLR. OpenReview.net*, New Orleans, LA, USA, 1–18.
- [36] Théo Trouillon, Christopher R. Dance, Éric Gaussier, Johannes Welbl, Sebastian Riedel, and Guillaume Bouchard. 2017. Knowledge Graph Completion via Complex Tensor Factorization. *J. Mach. Learn. Res.* 18 (2017), 1–38.
- [37] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha P. Talukdar. 2020. Composition-based Multi-Relational Graph Convolutional Networks. In *ICLR. OpenReview.net*, Addis Ababa, Ethiopia, 1–15.
- [38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NeurIPS. Curran Associates*, Long Beach, CA, USA, 5998–6008.
- [39] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a Free Collaborative Knowledge Base. *Commun. ACM* 57, 10 (2014), 78–85.
- [40] Chenxu Wang, Xin Wang, Zhao Li, Zirui Chen, and Jianxin Li. 2023. HyConvE: A Novel Embedding Model for Knowledge Hypergraph Link Prediction with Convolutional Neural Networks. In *WWW. ACM*, Austin, TX, USA, 188–198.
- [41] Quan Wang, Haifeng Wang, Yajuan Lyu, and Yong Zhu. 2021. Link Prediction on N-ary Relational Facts: A Graph-based Approach. In *ACL. ACL*, online, 396–407.
- [42] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *AAAI. AAAI Press*, Quebec, Canada, 1112–1119.
- [43] Jianfeng Wen, Jianxin Li, Yongyi Mao, Shini Chen, and Richong Zhang. 2016. On the Representation and Embedding of Knowledge Bases beyond Binary Relations. In *IJCAI. ijcai.org*, New York, NY, USA, 1300–1307.
- [44] Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying Graph Convolutional Networks. In *ICML. PMLR*, Long Beach, California, USA, 6861–6871.
- [45] Bo Xiong, Mojtaba Nayyeri, Shirui Pan, and Steffen Staab. 2023. Shrinking Embeddings for Hyper-Relational Knowledge Graphs. In *ACL. ACL*, Toronto, Canada, 13306–13320.
- [46] Shiyao Yan, Zequn Zhang, Xian Sun, Guangluan Xu, Shuchao Li, Qing Liu, Nayu Liu, and Shensi Wang. 2022. PolygonE: Modeling N-ary Relational Data as Gyro-Polygons in Hyperbolic Space. In *AAAI. AAAI Press*, online, 4308–4317.
- [47] Dong Yang, Peijun Qing, Yang Li, Haonan Lu, and Xiaodong Lin. 2022. GammaE: Gamma Embeddings for Logical Queries on Knowledge Graphs. In *EMNLP. ACL*, Abu Dhabi, United Arab Emirates, 745–760.
- [48] Yuhao Yang, Chao Huang, Lianghao Xia, and Chenliang Li. 2022. Knowledge Graph Contrastive Learning for Recommendation. In *SIGIR. ACM*, Madrid, Spain, 1434–1443.
- [49] Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. 2020. L2-GCN: Layer-Wise and Learned Efficient Training of Graph Convolutional Networks. In *CVPR. IEEE*, Seattle, WA, USA, 2124–2132.
- [50] Donghan Yu and Yiming Yang. 2021. Improving Hyper-Relational Knowledge Graph Completion. *CoRR abs/2104.08167* (2021), 1–5.
- [51] Weijian Yu, Jie Yang, and Dingqi Yang. 2024. Robust Link Prediction over Noisy Hyper-Relational Knowledge Graphs via Active Learning. In *WWW. ACM*, Singapore, 2282–2293.
- [52] Richong Zhang, Junpeng Li, Jiajie Mei, and Yongyi Mao. 2018. Scalable Instance Reconstruction in Knowledge Bases via Relatedness Affiliated Embedding. In *WWW. ACM*, Lyon, France, 1185–1194.
- [53] Zhanqiu Zhang, Jie Wang, Jiajun Chen, Shuiwang Ji, and Feng Wu. 2021. ConE: Cone Embeddings for Multi-Hop Reasoning over Knowledge Graphs. In *NeurIPS. Curran Associates*, online, 19172–19183.

Appendix

Table 5: Statistics of datasets under three different scenarios. The values in parentheses indicate the percentage of triples in the corresponding dataset with hyper-relational facts. The hyphen -n indicates that each main triple in the corresponding dataset contains n qualifier pairs.

Datasets	Train	Valid	Test	Entity	Relation
WikiPeople-3	20656	2582	2582	12270	66
WikiPeople-4	12150	1519	1519	9528	50
JF17K-3	27635	3454	3455	11541	104
JF17K-4	7607	951	951	6536	23
WD50K (33)	73406	10568	18133	38123	474
WD50K (66)	35968	5154	8045	27346	403
WD50K (100)	22738	3279	5297	18791	278
WikiPeople (33)	28280	3550	3542	20921	145
WikiPeople (66)	14130	1782	1774	13651	133
WikiPeople (100)	9319	1181	1173	8068	105
JF17K (33)	56959	8122	9112	24081	490
JF17K (66)	27280	4413	5403	19288	469
JF17K (100)	17190	3152	4142	12656	307
WD50K	166435	23913	46159	47155	531
WikiPeople	294439	37715	37712	34825	178
JF17K	76379	-	24568	28645	501

A Datasets

We conduct experiments on three hyper-relational KGs: WD50K [11], WikiPeople [14], and JF17K [43], with WD50K and WikiPeople being extracted from Wikidata [39] and JF17K from Freebase [2]. These datasets have the following two characteristics. On the one hand, *not* all main triples contain hyper-relational knowledge: only 13.6% in WD50K, 2.6% in WikiPeople and 45.9% in JF17K. On the other hand, each main triple has a different number of qualifier pairs: ranging from 0 to 20 in WD50K, 0 to 7 in WikiPeople and 0 to 4 in JF17K. To validate the models’ robustness across multiple scenarios, these datasets have been refined taking into account the two described characteristics [16], i.e., the percentage of triples containing hyper-relational knowledge (mixed-percentage or fixed-percentage) and the number of qualifiers associated to main triples (mixed-qualifier or fixed-qualifier). We can thus construct three different sub-datasets, namely: *Fixed-percentage Fixed-qualifier*, *Fixed-percentage Mixed-qualifier*, and *Mixed-percentage Mixed-qualifier*. Note that the by definition *Mixed-percentage Fixed-qualifier* scenario *does not exist*. The statistics of the considered datasets are as shown in the Table 5.

- (1) **Mixed-percentage Mixed-qualifier**. This is the original setting introduced in [11], used to test the performance under variable number of main triples with qualifiers and with different number of qualifiers associated to those triples.
- (2) **Fixed-percentage Fixed-qualifier**. Considering the monotonicity of this representational regime, we filter out the triples with 3 and 4 associated qualifiers to obtain WikiPeople-3, WikiPeople-4, JF17K-3, and JF17K-4.
- (3) **Fixed-percentage Mixed-qualifier**. Following [11, 16], we construct subsets where the number of qualifiers for each triple

varies, but the proportion of triples with hyper-relational knowledge is fixed. For example, in the subsets WD50K (33), WD50K (66), and WD50K (100), the number in parentheses indicates the corresponding proportion of main triples with hyper-relational knowledge. We consider similar subsets for the WikiPeople and JF17K datasets.

It is noteworthy that the construction process of the dataset inherently satisfies the qualifier monotonicity property, i.e., for a given main triple, when new qualifier pairs are added to it, the corresponding answer space will not expand. Specifically, the qualifier monotonicity includes two aspects: *from nothing to something* and *from little to more*. Taking the WD50K dataset as an example (the same applies to any other dataset), the content prefix with P and Q below corresponds to the ID value from the WikiData knowledge graph.

- **From nothing to something**: This phrase encapsulates the process of extending a main triple into a hyper-relational fact by incorporating qualifier pairs. For instance, given the main triple $(Q200405, P1411, ?)$, the possible answers to $?$ are $Q4834546$, $Q593098$, and $Q103916$. By introducing a qualifier pair $Q_1 = (P1868: Q272860)$, the resulting hyper-relational fact is defined as $q_1 = \{(Q200405, P1411, ?), (P1868: Q272860)\}$. Consequently, the set of valid answers to $?$ is refined to $Q593098$ and $Q103916$, effectively eliminating $Q4834546$ as a possibility. This demonstrates how the incorporation of qualifiers progressively constrains the answer space. During dataset construction, we begin by extracting the main triple and iteratively introduce relevant qualifier pairs, gradually refining the possible answer set. The concept of *from nothing to something* signifies a fundamental transition from conventional triple-based representations to hyper-relational facts enriched with hyper-relational knowledge, marking a significant conceptual shift from 0 to 1.
- **From little to more** refers to the progressive enrichment of hyper-relational facts through the incorporation of additional qualifier pairs. For instance, by appending the qualifier pair $Q_2 = (P805: Q767355)$ to q_1 , we obtain $q_2 = \{(Q200405, P1411, ?), (P1868: Q272860), (P805: Q767355)\}$. Consequently, the corresponding answer to $?$ is refined to $Q103916$, thereby further narrowing the answer set by excluding $Q593098$. This process exemplifies the transition from an incomplete hyper-relational knowledge representation to a more comprehensive one, enhancing the specificity and informativeness of the captured knowledge.

B Baselines

We compare HyperMono with fourteen state-of-the-art baselines for HKGC, including **Embedding-based Methods**, **Transformer-based with GNNs Methods**, and **Transformer-based without GNNs Methods**:

The **Embedding-based Methods** include:

- *m-TransH* [43] builds on TransH by transforming each hyper-relational fact using a star-to-clique conversion.
- *RAE* [52] integrates m-TransH with a multi-layer perceptron by considering the relatedness of entities. However, since the distance-based scoring function of TransH enforces constraints on relations, it fails to represent some binary relations in KGs.

- *NaLP-Fix* [33] uses a convolutional-based framework to compute a relatedness vector for a triple. However, the model does not distinguish between the main triple and relation-specific qualifiers.
- *NeuInfer* [12] distinguishes the information in the same n-ary fact and represented each n-ary fact as a primary triple coupled with a set of its auxiliary descriptions.
- *HINGE* [33] adopts a convolutional framework for modeling hyper-relational facts. However, it only operates on a triple-quintuple level that lacks the necessary granularity for representing a relation instance with its qualifiers.
- *ShrinkE* [45] models the primary triplets as a spatial-functional transformation from the head into a relation-specific box.

The **Transformer-based with GNNs Methods** include:

- *StarE* [11] employs a message passing based graph encoder to obtain entity/relation embeddings, and feeds these embeddings to a transformer decoder to score hyper-relational facts.
- *QUAD* [34] introduces a graph encoder that aggregates information from the perspective of the qualifier entities.

The **Transformer-based without GNNs Methods** include:

- *Hy-Transformer* [50] replaces the graph encoder with light-weight embedding modules, achieving higher efficiency without sacrificing effectiveness.
- *GRAN* [41] represents a fact as a heterogeneous graph, and employs edge-specific attentive bias to capture both local and global dependencies within the given fact.
- *HyNT* [7] proposes a unified framework that learns representations of a hyper-relational knowledge graph containing numeric literals in either triples or qualifiers.
- *HAHE* [27] presents a model with hierarchical attention in global and local level to encode the graphical structure and sequential structure via self-attention layers.
- *HyperFormer* [16] considers local-level sequential information to encode the content of entities, relations and qualifiers. It includes an entity neighbor aggregator, relation qualifier aggregator and convolution-based bidirectional interaction modules.
- *HyperCL* [26] designs relation-aware graph attention networks to capture the hierarchical ontology and a concept-aware contrastive loss to alleviate the dominance issue.

C Evaluation Protocol and Implements Details

Evaluation Protocol. Following existing work [7, 16, 27, 45], we rank all entities to predict the head entity in query $(?, r, t, Q)$ or the tail entity in query $(h, r, ?, Q)$. Specifically, we employ MRR and Hits@ k (abbreviated sometimes as $H@k$, $k \in \{1, 3, 10\}$) as evaluation metrics. MRR defines the inverse of the rank for the first correct answer, while Hits@ k calculates the percentage of correct types ranked among the top- k answers. For all metrics, the larger the value, the better. Following the standard evaluation protocol in the filtered setting [3], all true triples in the HKGs are filtered out during evaluation, since predicting a low rank for these triples should not be penalized. All metrics are computed by averaging over two directions: head entity prediction and tail entity prediction.

Implementation Details. We conduct all experiments on six 32G Tesla V100 GPUs, use the Pytorch 1.8.1 framework (it should be mentioned that using Pytorch 2.1.2 will result in incorrect running results), and employ the AdamW [23] optimizer and a cosine decay scheduler with a linear warm-up for optimization. The hyperparameters are tuned according to the MRR on the validation set. We use grid search to select the optimal hyperparameters, mainly including: the learning rate $lr \in \{3e-4, 4e-4, 5e-4, \mathbf{6e-4}, 7e-4\}$, the label smooth values $ls \in \{0.3, 0.4, 0.5, 0.6, 0.7, \mathbf{0.8}, 0.9\}$, the embedding dimension $d \in \{50, 100, 150, \mathbf{200}, 400, 600\}$, the temperature of Equation (1) $\gamma \in \{1, 2, 3, 4, 5, 6\}$, the initialization range of embeddings $ir \in \{0.01, \mathbf{0.02}, 0.05, 0.1, 0.2, 0.35, 0.5\}$, the neighbors of the head entity $\{2, 3, 4\}$, the number of qualifiers for a relation $\{5, 6, 7\}$, the number of layers in the transformer $\{2, 4, \mathbf{8}\}$, the number of heads in transformer $\{1, 2, 4, \mathbf{8}\}$, the input dropout rate in the transformer $\{0.6, \mathbf{0.7}, 0.8\}$. For the reproducibility of the model, we also provide the running logs of HyperMono in different scenarios in the anonymous code link: <https://anonymous.4open.science/r/HKGC-HyperMono-AE61>.

D Parameter Sensitivity

To address RQ3, we carry out parameter sensitivity experiments on the WikiPeople-3, WikiPeople-4, JF17K-3, and JF17K-4 datasets, including: a) effect of label smoothing; b) effect of embedding dimension d ; c) effect of temperature γ , and d) effect of initialization range. The corresponding results are shown in Figure 4.

Effect of Label Smoothing. Label smoothing is widely used in tasks related to knowledge graphs [19, 20]. It introduces a degree of uncertainty, making the model more adaptable to different data distributions during the training process. From Figure 4(a), we observe that the values of label smoothing have an impact on the performance, but there is no evident pattern. We need to make appropriate adjustments to its values based on the dataset. For instance, we select 0.8 and 0.5 as the most suitable label smoothing values for the WikiPeople-4 and JF17K-3 datasets, respectively.

Effect of Embedding Dimension d . In principle, more semantic knowledge could be captured by having a higher number of dimensions in entity, relation, and qualifiers embeddings. However, as shown in Figure 4(b), there is not a strict proportional relationship between embedding dimensions and the actual performance. After reaching a certain value of embedding dimensions, the accuracy stabilizes. In practice, one chooses the minimum value that achieves stability. For example, for the WikiPeople-4 and JF17K-3 datasets, the embedding dimension d is set at 200.

Effect of Temperature γ . The temperature coefficient γ in Equation (1) can be used to adjust the contribution of each neighbor of an entity. Figure 4(c) illustrates the performance difference with values γ ranging from 1 to 6. One can observe that the choice of the γ value has a minimal impact on the final performance. The performance curve only exhibits local oscillations, indicating that the HyperMono model is robust to the choice of the γ value.

Effect of the Initialization Range. We observe that the initialization value range of embeddings has a significant impact on HyperMono’s performance. As shown in Figure 4(d), when the initialization values are relatively small, the performance of HyperMono is more stable. However, the model becomes sensitive to larger initialization values. In practice, it is important to carefully

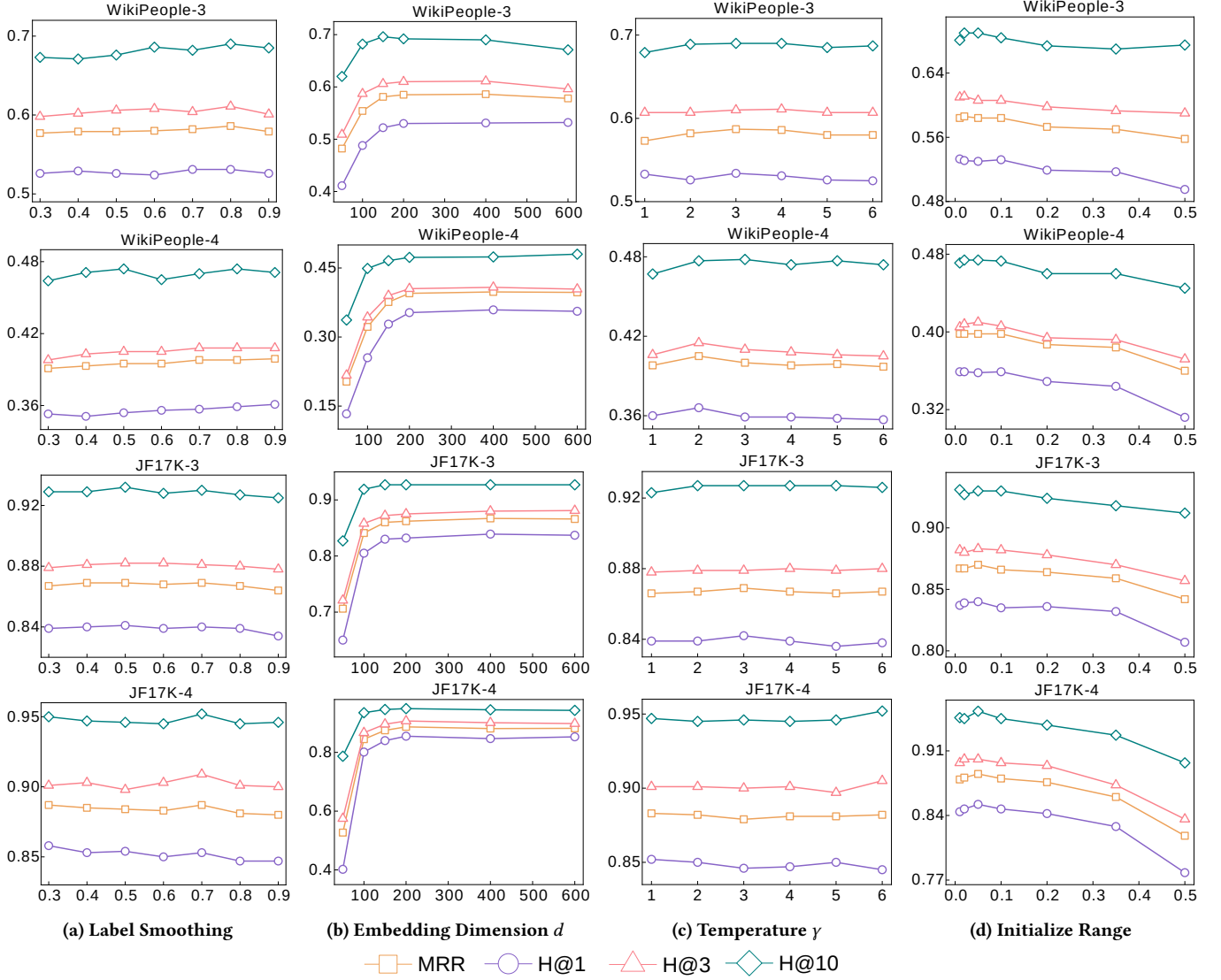


Figure 4: Parameter sensitivity experiments under different conditions on Wikipeople-3, Wikipeople-4, JF17K-3 and JF17K-4 datasets.

set the initialization range to prevent the model from experiencing sharp performance declines.

E Complexity Analysis

To address RQ4, we conduct a complexity analysis of HyperMono and HyperFormer (the best performing baseline in most cases) across the WD50K, Wikipeople, and JF17K datasets from the following five perspectives: a) *time complexity*, which is measured by the number of floating-point operations (#FLOPs) required during training phase; b) *space complexity*, which refers to the amount of a model’s parameters (#Params); c) *model training time* (#TrainTime), measured in hours (h); d) *inference time per sample* (#InferTime), measured in milliseconds (ms); e) *GPU memory consumption* (#GPU), measured in gigabytes (G). All the aforementioned metrics are desired to be as minimal as possible, as lower values indicate reduced

resource expenditure and greater efficiency. The corresponding results are shown in Table 6. We can observe that HyperMono requires significantly less overhead, specifically, on the WD50K dataset, HyperMono only requires nearly half the #FLOPs and one-sixth the #Params of HyperFormer, which demonstrates that HyperMono is more efficient and can achieve better results while occupying less time and space, a similar trend is observed in #TrainTime, #InferTime, and #GPU.

F Additional Results

To demonstrate the robustness of HyperMono, we further conducted incremental experiments, including: a) effect of pooling operation; b) effect of aggregation operation; c) effect of embedding

Table 6: The quantitative results of HyperFormer and HyperMono under different complexity evaluation metrics.

Metrics	Model	WD50K	WikiPeople	JF17K
#FLOPs	HyperFormer	118.397G	118.167G	118.070G
	HyperMono	69.600G	69.371G	69.273G
#Params	HyperFormer	66.956M	66.956M	66.956M
	HyperMono	10.645M	10.645M	10.645M
#TrainTime	HyperFormer	39h	68.7h	15.9h
	HyperMono	31h	50.7h	12.1h
#InferTime	HyperFormer	6.0ms	6.5ms	6.0ms
	HyperMono	4.7ms	5.0ms	4.2ms
#GPU	HyperFormer	23.9G	23.5G	23.6G
	HyperMono	9.5G	8.5G	8.0G

space; d) effect of learning rate; e) effect of neighbor numbers; f) effect of sparse ratio. The corresponding results are shown in Table 7, Table 8, and Table 9.

Effect of Pooling Operation. In Equation 1, we adopt the weighted pooling method to integrate multiple inference results. To investigate the impact of different pooling strategies, we further replace the weighted pooling with mean pooling, max pooling, and the sum operation. The experimental results on the WikiPeople-3, WikiPeople-4, JF17K-3, and JF17K-4 datasets are presented at the top of Table 7. We can observe that the weighted pooling operation generally yields the best performance, which proves the rationality of our choice of weighted pooling operation.

Effect of Aggregation Operation. For P_G^h in the Coarse-grained Neighborhood Aggregator and Q_G^h in the Fine-grained Neighborhood Aggregator, we use the mean operation to average the entity vector predicted by different relational neighbors to aggregate them. To understand the impact of different operations, we replace mean with max, sum, and attention operations, and conduct experiments on WikiPeople-3, WikiPeople-4, JF17K-3 and JF17K-4, the corresponding results are shown in the middle of Table 7. We can observe that replace mean operation in GEI with other methods does not result in significant performance fluctuations for HyperMono across the four datasets. Therefore, this component is not the key factors affecting performance.

Effect of Embedding Space. Our model employs the cone space to encode qualifier monotonicity. To assess the rationale behind the selection of the cone space, we substitute it with Beta [32] and Gamma [47] probability distribution spaces. It is essential to note that the chosen embedding space must support three operations: projection, intersection, and shrinking. While hyperbolic [4] embeddings support projection and shrinking operations, the intersection of two hyperbolic embeddings does not yield a hyperbolic space, thus hyperbolic space does not meet the intersection condition. In contrast, Beta and Gamma spaces accommodate all three aforementioned operations. The experimental results of replacing cone with Beta and Gamma on the WikiPeople-3, WikiPeople-4, JF17K-3, and JF17K-4 datasets are shown at the bottom of Table 7. We also replace the cone operation with an operation similar to the TP module, and the corresponding results are shown in the None row in Table 7. We observe that the cone space demonstrates superior performance, particularly on the JF17K dataset, where it achieves

a 1.8% and 0.9% improvement in the MRR metric on the JF17K-3 and JF17K-4 datasets, respectively, compared to Beta embeddings. The primary reason for this lies in the fact that the cone space exhibits greater closure under scaling operations relative to Beta and Gamma spaces. In contrast, the scaling outcomes of Beta and Gamma operations often necessitate simulation through attention mechanisms.

Effect of Learning Rate. A larger learning rate can accelerate the convergence of the model but may risk bypassing the optimal solution during the optimization process. Conversely, a smaller learning rate facilitates more stable convergence to the optimal solution, albeit at a slower pace. Consequently, the selection of an appropriate learning rate entails a trade-off between the speed of convergence and the attainment of optimal performance. HyperMono employs a learning rate of $6e-4$. We further conduct comparative experiments with learning rates of $5e-4$ and $7e-4$ on the WikiPeople-3, WikiPeople-4, JF17K-3, and JF17K-4 datasets, the corresponding results are shown in the upper section of Table 8, which indicate that the choice of learning rate has minimal impact on HyperMono, thereby demonstrating the robustness and stability of the HyperMono model.

Effect of Neighbor Numbers. For the CNA and FNA modules, we aim to enhance the embedding representation of the central entity by leveraging its surrounding neighbors. To investigate the impact of the number of neighboring entities on entity representation, HyperMono sets the number of neighbors to 3. On the WikiPeople-3, WikiPeople-4, JF17K-3, and JF17K-4 datasets, we also conduct comparative analyses using 2 and 4 neighbors, with the results presented at the bottom of Table 8. Our findings indicate that selecting 2, 3, or 4 neighbors for a given entity does not significantly affect the final performance of HyperMono. Further analysis of the datasets reveals a pronounced long-tail distribution. For instance, in the WikiPeople-3 dataset, entities with only 1 neighbor account for 45.99%, while those with 5 or more neighbors constitute 22.00%. Consequently, the most substantial performance degradation occurs when no neighboring entities are available (as shown in rows w/o CNA+TP and w/o FNA+QMP of Table 4). The comparable performance observed with 2, 3, and 4 neighbors may stem from the fact that a certain number of neighbors already suffice to effectively characterize the entity’s knowledge. Given that the number of neighbors is merely a hyperparameter, it is essential to select an appropriate value based on the distribution characteristics of the dataset in practical applications.

Effect of Sparse Ratio. In Section 3.2.1, the modules CNA and FNA utilize entity neighbor information to enhance the entity representation. To simulate sparse neighbors, we delete a certain proportion neighbors of entity, specifically we perform neighbor pruning on entities with over five neighbors at ratios of 25%, 50%, and 75%. Comparative experimental results for HyperFormer and HyperMono using Wikipeople-3 and JF17K-3 are shown in Table 9. We can observe that removing entity samples with a higher number of neighbors significantly impacts the performance of both HyperFormer and HyperMono, with performance fluctuations becoming more pronounced as the removal ratio increases. This phenomenon primarily stems from the reduced number of neighbors diminishing the effectiveness of the CNA and FNA modules,

Table 7: Additional ablation study experimental results with different settings.

Settings	WikiPeople-3				WikiPeople-4				JF17K-3				JF17K-4			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
Mean	0.579	0.529	0.598	0.675	0.391	0.356	0.404	0.462	0.860	0.831	0.874	0.919	0.871	0.846	0.894	0.941
Max	0.580	0.529	0.607	0.681	0.394	0.354	0.405	0.470	0.863	0.834	0.878	0.923	0.878	0.846	0.900	0.945
Sum	0.578	0.531	0.600	0.670	0.392	0.356	0.398	0.464	0.863	0.834	0.875	0.921	0.869	0.839	0.891	0.941
Max	0.581	0.528	0.606	0.688	0.396	0.356	0.406	0.470	0.864	0.837	0.876	0.919	0.875	0.849	0.898	0.943
Sum	0.583	0.529	0.603	0.685	0.395	0.356	0.402	0.469	0.866	0.836	0.881	0.925	0.872	0.843	0.898	0.942
Attention	0.579	0.526	0.603	0.680	0.395	0.354	0.405	0.469	0.867	0.838	0.878	0.928	0.880	0.849	0.900	0.944
None	0.569	0.518	0.584	0.671	0.382	0.346	0.392	0.458	0.847	0.816	0.852	0.900	0.868	0.823	0.869	0.918
Beta	0.576	0.526	0.603	0.686	0.392	0.350	0.401	0.465	0.849	0.817	0.860	0.905	0.872	0.838	0.892	0.932
Gamma	0.578	0.524	0.601	0.683	0.390	0.351	0.398	0.463	0.855	0.825	0.868	0.916	0.870	0.837	0.884	0.931
HyperMono	0.586	0.531	0.611	0.690	0.398	0.359	0.408	0.474	0.867	0.839	0.880	0.927	0.881	0.847	0.901	0.945

Table 8: Additional parameter sensitivity experimental results with different settings.

Settings	WikiPeople-3				WikiPeople-4				JF17K-3				JF17K-4			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
5e-4	0.585	0.531	0.608	0.688	0.400	0.359	0.410	0.480	0.867	0.840	0.883	0.928	0.884	0.851	0.903	0.951
7e-4	0.582	0.530	0.605	0.681	0.398	0.359	0.409	0.473	0.868	0.841	0.880	0.926	0.883	0.851	0.902	0.946
num=2	0.582	0.531	0.606	0.683	0.397	0.357	0.407	0.473	0.867	0.838	0.882	0.925	0.881	0.849	0.902	0.947
num=4	0.583	0.531	0.606	0.684	0.398	0.358	0.405	0.471	0.865	0.837	0.879	0.928	0.880	0.846	0.898	0.946
HyperMono	0.586	0.531	0.611	0.690	0.398	0.359	0.408	0.474	0.867	0.839	0.880	0.927	0.881	0.847	0.901	0.945

Table 9: Experimental results after removing entity neighbors at different proportions.

Settings	WikiPeople-3									JF17K-3								
	25%			50%			75%			25%			50%			75%		
	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
HyperFormer	0.417	0.346	0.543	0.365	0.297	0.493	0.282	0.224	0.387	0.423	0.360	0.543	0.399	0.336	0.516	0.349	0.283	0.467
HyperMono	0.414	0.344	0.547	0.380	0.305	0.500	0.304	0.247	0.413	0.424	0.365	0.571	0.410	0.341	0.534	0.379	0.311	0.494

as these modules rely on the neighbors of entities to enhance the representation of the central entity. For HyperFormer, which only incorporates the CNA module, and HyperMono, which includes both CNA and FNA modules, we find that HyperMono consistently outperforms HyperFormer across all removal ratios, underscoring the necessity of the FNA module. Furthermore, in the WikiPeople-3 and JF17K-3 datasets, the proportions of samples with more than five entity neighbors are 22.00% and 29.06%, respectively. Combined with the analysis in *effect of neighbor numbers*, it becomes evident that existing HKGC datasets exhibit a pronounced long-tail distribution, where some entities have sparse neighbors while others possess an abundance. We plan to address the performance improvement challenges associated with long-tail distribution in the next phase of our work.

G Model Transferability

To address RQ5, we further validate the transfer ability of our model. To this aim, we also conduct experiments on the *knowledge hypergraph link prediction* task. We start by introducing the task definition, dataset descriptions, and baselines. It should be emphasized that our objective here is solely to evaluate the transferability of HyperMono, rather than to conduct an in-depth investigation into

knowledge hypergraph link prediction task, therefore, we only introduce necessary background, for a detailed discussion, interested readers are directed to the relevant literature [8, 9, 17, 18, 40, 46].

Task Definition. A *knowledge hypergraph graph* $\mathcal{G}$ is defined as $\{\mathcal{E}, \mathcal{R}, \mathcal{F}\}$, where $\mathcal{E}$ is a set of entities, $\mathcal{R}$ is a set of relation types and $\mathcal{F}$ is a set of n-ary facts of the form $r(e_1, e_2, \dots, e_n)$ with $r \in \mathcal{R}$, $e_i \in \mathcal{E}$, for all $i \in [1, n]$ and n is the non-negative arity of r , representing the number of entities involved within each relation. The *knowledge hypergraph link prediction* task looks at predicting a missing entity in the i -th position of the tuple $r(e_1, e_2, \dots, ?, \dots, e_{n-1}, e_n)$. Consistent with PolygonE [46], an n-ary fact $\mathcal{F}$ can essentially be represented in the form of triples + pairs, i.e., $\mathcal{F} = r(e_1, e_2, \dots, e_n) \rightarrow \mathcal{F}' = (e_1, r, e_2, r_1, e_3, \dots, r_{n-2}, e_n) \rightarrow \mathcal{F}'' = \{(e_1, r, e_2), \{(r_1 : e_3), \dots, (r_{n-2} : e_n)\}\}$. In case $n = 2$, $\mathcal{F}''$ is a binary fact, if $n > 2$, $\mathcal{F}''$ is a beyond-binary fact, then (e_1, r, e_2) is taken as primary triple. Regarding the methodology for transforming $\mathcal{F}$ into $\mathcal{F}''$, we shall herein provide a concrete example to elucidate the process. Given $\mathcal{F} = \text{member_of_team}(\text{James Harden, Philadelphia 76ers, P.J. Tucker})$, by assigning numerical identifiers to the relation *member_of_team*, can effectively transform $\mathcal{F}$ into $\mathcal{F}'' = \{(\text{James Harden, member_of_team, Philadelphia 76ers}), \{(member_of_team_1, P.J. Tucker)\}\}$. More details about this data transformation and the rationale behind it can be found in the PolygonE [46] work. Consequently, knowledge hypergraph link prediction task

Table 10: Evaluation of the *knowledge hypergraph completion* task in the mixed arity setting on the N-WikiPeople, N-JF17K and N-FB-AUTO datasets. Best scores are highlighted in **bold, the second best scores are highlighted in **normal**.**

Methods	N-WikiPeople				N-JF17K				N-FB-AUTO			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
RAE [52]	0.253	0.118	0.343	0.463	0.396	0.312	0.433	0.561	0.703	0.614	0.764	0.854
NaLP [14]	0.338	0.272	0.364	0.466	0.310	0.239	0.334	0.450	0.672	0.611	0.712	0.774
HINGE [33]	0.333	0.259	0.361	0.477	0.473	0.397	0.490	0.618	0.678	0.630	0.706	0.765
NeuInfer [12]	0.350	0.282	0.381	0.467	0.451	0.373	0.484	0.604	0.737	0.700	0.755	0.805
HypE [10]	0.263	0.127	0.355	0.486	0.494	0.399	0.532	0.650	0.804	0.774	0.824	0.856
RAM [22]	0.380	0.279	0.455	0.539	0.539	0.463	0.573	0.690	0.830	0.803	0.851	0.876
HyperMLN [6]	0.351	0.270	0.394	0.497	0.556	0.482	0.597	0.717	0.831	0.803	0.851	0.877
tNaLP+ [13]	0.339	0.269	0.369	0.473	0.449	0.370	0.484	0.598	0.729	0.645	0.748	0.826
S2S [9]	0.372	0.277	0.439	0.533	0.528	0.457	0.570	0.690	-	-	-	-
HyConvE [40]	0.362	0.275	0.388	0.501	0.590	0.478	0.610	0.729	0.847	0.820	0.872	0.901
MSeaKG [8]	0.392	0.290	0.468	0.553	0.561	0.475	0.591	0.705	-	-	-	-
HJE [17]	0.450	0.375	0.487	0.582	0.590	0.507	0.613	0.729	0.872	0.848	0.886	0.903
PolygonE [46]	0.431	0.334	0.454	0.568	0.565	0.485	0.602	0.708	0.858	0.826	0.871	0.921
HyCubE [18]	0.438	0.358	0.482	0.575	0.581	0.507	0.615	0.731	0.884	0.863	0.897	0.920
HyperMono	0.481	0.381	0.545	0.637	0.676	0.612	0.708	0.801	0.884	0.855	0.907	0.937

Table 11: Evaluation of the *knowledge hypergraph completion* task in the fixed arity setting on the N-WikiPeople-3, N-WikiPeople-4, N-JF17K-3 and N-JF17K-4 datasets. Best scores are highlighted in **bold, the second best scores are highlighted in **normal**.**

Methods	N-WikiPeople-3				N-WikiPeople-4				N-JF17K-3				N-JF17K-4			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
RAE [52]	0.239	0.168	0.252	0.379	0.150	0.080	0.149	0.273	0.505	0.430	0.532	0.644	0.707	0.636	0.751	0.835
NaLP [14]	0.301	0.226	0.327	0.445	0.342	0.237	0.400	0.540	0.515	0.431	0.552	0.679	0.719	0.673	0.742	0.805
HINGE [33]	0.338	0.255	0.360	0.508	0.352	0.241	0.419	0.557	0.587	0.509	0.621	0.738	0.745	0.700	0.775	0.842
NeuInfer [12]	0.355	0.262	0.388	0.521	0.361	0.255	0.424	0.566	0.622	0.533	0.658	0.770	0.765	0.722	0.808	0.871
HypE [10]	0.266	0.183	-	0.443	0.304	0.191	-	0.527	0.364	0.255	-	0.573	0.408	0.300	-	0.627
RAM [22]	0.254	0.190	-	0.383	0.226	0.161	-	0.367	0.578	0.505	-	0.722	0.743	0.701	-	0.845
HyperMLN [6]	0.252	0.193	-	0.385	0.224	0.167	-	0.370	0.574	0.501	-	0.711	0.734	0.687	-	0.831
tNaLP+ [13]	0.270	0.185	-	0.444	0.344	0.223	-	0.578	0.411	0.325	-	0.617	0.630	0.531	-	0.722
S2S [9]	0.386	0.299	0.421	0.559	0.391	0.270	0.470	0.600	0.740	0.676	0.770	0.860	0.822	0.761	0.853	0.924
HyConvE [40]	0.318	0.240	-	0.482	0.386	0.271	-	0.607	0.729	0.670	-	0.861	0.827	0.770	-	0.931
MSeaKG [8]	0.403	-	-	0.579	0.409	-	-	0.624	0.754	-	-	0.889	0.833	-	-	0.938
HJE [17]	0.381	0.303	-	0.596	0.388	0.276	-	0.607	0.707	0.637	-	0.851	0.827	0.775	-	0.928
HyCubE [18]	0.322	0.247	-	0.469	0.370	0.252	-	0.583	0.615	0.544	-	0.752	0.792	0.744	-	0.879
HyperMono	0.454	0.375	0.486	0.611	0.486	0.395	0.545	0.650	0.793	0.739	0.824	0.901	0.888	0.855	0.910	0.951

will be transformed into predicting the head and tail entities, e_1 and e_2 , along with the other entities $\{e_3, \dots, e_n\}$, which is equivalent to all entities prediction within HKGC task.

Datasets. We test the performance of HyperMono on the knowledge hypergraph link prediction task and conduct experiments on hypergraphs with *mixed arity*: N-WikiPeople, N-JF17K, and N-FB-AUTO and with *fixed arity*: N-WikiPeople-3, N-WikiPeople-4, N-JF17K-3, and N-JF17K-4. The statistics of the corresponding datasets are consistent with existing literature [8, 9, 17, 18, 40, 46]. In order to differentiate from the datasets for the HKGC task, we have prefixed the dataset for knowledge hypergraph link prediction task with an "N-" designation.

Baselines. For knowledge hypergraph link prediction task, we consider 14 state-of-the-art baselines that are universally acknowledged, including RAE [52], NaLP [14], HINGE [33], NeuInfer [12], HypE [10], RAM [22], HyperMLN [6], tNaLP+ [13], S2S [9], HyConvE [40], MSeaKG [8], HJE [17], PolygonE [46], and HyCubE [18].

Performance Analysis. The results of the knowledge hypergraph completion experiments in the mixed arity and fixed arity settings are respectively shown in Table 10 and Table 11. We observe that our HyperMono consistently achieves state-of-the-art performance on all benchmark datasets. For the mixed arity scenario, compared with the best-performing baseline HyCubE, HyperMono obtains improvements of 4.3% on N-WikiPeople and 9.5% on N-JF17K in the MRR metric. On the N-FB-AUTO dataset, HyperMono virtually matches the results of the best performing baseline. These results demonstrate the capability and effectiveness of HyperMono for capturing and extracting various types of information and knowledge. The same phenomenon occurs in the fixed arity scenario, HyperMono respectively obtains performance improvements of 13.2%, 11.6%, 17.8%, and 9.6% in the MRR metric on the N-WikiPeople-3, N-WikiPeople-4, N-JF17K-3, and N-JF17K-4

datasets, when compared with the best baseline HyCubE. Such a notable performance improvement is mainly due to our explicit treatment of qualifier monotonicity during the encoding process. This feature is more suitable for scenarios where arity is fixed, because introducing qualifier monotonicity can greatly reduce the scope of the optional answer space to obtain the final answer.

H Case Study

To demonstrate HyperMono’s encoding capability for qualifier monotonicity, we conduct a case study, with the corresponding results present in Table 12. For the main triple (*Leonardo DiCaprio*, *cast member*, ?), HyperMono’s predictions for potential entities at the ? position including multiple entities, with the probability values for each prediction showing minimal variance. Upon introducing the qualifier pair (*genre: drama film*), we can observe that entities previously associated with higher probability values, such as *Titanic* and *Inception* are excluded. Conversely, the prediction probabilities for entities *The Revenant*, *The Wolf of Wall Street*, and *Don’t Look Up* significantly increased. Further addition of the qualifier pair (*director: Martin Scorsese*) led to a further narrowing of the answer space. This process exhibits a phenomenon of answer space expansion and contraction as the number of qualifier pairs increased. The cone embedding within HyperMono elegantly accommodates this property through the shrinking operations of the cone.

Table 12: Case study on the importance of the qualifier monotonicity. The probability of – is close to 0, so we do not show it here.

Hyper-relational Fact	Answer	Probability
(Leonardo DiCaprio, cast member, ?)	<i>Titanic</i>	8.0%
	<i>Inception</i>	7.5%
	<i>The Revenant</i>	6.8%
	<i>The Wolf of Wall Street</i>	6.2%
	<i>Catch Me If You Can</i>	5.9%
	<i>Don’t Look Up</i>	5.3%
(Leonardo DiCaprio, cast member, ?) (genre: drama film)	<i>Titanic</i>	–
	<i>Inception</i>	–
	<i>The Revenant</i>	25.6%
	<i>The Wolf of Wall Street</i>	23.4%
	<i>Catch Me If You Can</i>	–
	<i>Don’t Look Up</i>	20.3%
(Leonardo DiCaprio, cast member, ?) (genre: drama film) (director: Martin Scorsese)	<i>Titanic</i>	–
	<i>Inception</i>	–
	<i>The Revenant</i>	–
	<i>The Wolf of Wall Street</i>	75.2%
	<i>Catch Me If You Can</i>	–
	<i>Don’t Look Up</i>	–

I Faithful Encoding of Qualifier Monotonicity

Proposition 1. Given two hyper-relational facts $q_1 = (h, r, ?, Q_1)$ and $q_2 = (h, r, ?, Q_2)$ where $Q_1 \subseteq Q_2$, i.e., q_1 is a partial fact of q_2 , the output of the score function $f(\cdot)$ of the Shrink operation (denoted here as $S(\cdot)$ for simplicity) defined in Equation 3 adheres to the constraint $f(S(q_1)) \geq f(S(q_2))$.

Proof. We begin by demonstrating that the resulting cone of q_1 subsumes that of q_2 . Since both q_1 and q_2 contain the same primary triple $(h, r, ?)$ and satisfy $Q_1 \subseteq Q_2$, it follows that the final shrunken cone of q_2 must be a subset of the shrunken cone of q_1 . That is, we obtain $S(q_1) \supseteq S(q_2)$. Given the tail entity ?, whose embedding is represented as **E** (which effectively consists only of the symmetry axis with an aperture size of 0), we primarily consider three possible cases regarding its position.

- (1) **E** is located inside the smaller cone $S(q_2)$. Since $S(q_1) \supseteq S(q_2)$, it follows that **E** must also be inside $S(q_1)$. The distance function $\mathcal{D}(\cdot)$ for measuring the point-to-cone distance, as proposed in Section 4.3 of ConE [53], is monotonically increasing w.r.t. the distance from the tail point to the symmetry axis of the cone. Consequently, we obtain $\mathcal{D}(\mathbf{E}, S(q_1)) \geq \mathcal{D}(\mathbf{E}, S(q_2))$, which implies that $f(S(q_1)) \geq f(S(q_2))$.
- (2) **E** is located outside the smaller cone $S(q_2)$ but inside the larger cone $S(q_1)$. According to the definition of the point-to-cone distance $\mathcal{D}(\cdot)$, it follows that $\mathcal{D}(\mathbf{E}, S(q_1)) \geq \mathcal{D}(\mathbf{E}, S(q_2))$, which implies $f(S(q_1)) \geq f(S(q_2))$.
- (3) **E** is located outside the larger cone $S(q_1)$. Since $S(q_1) \supseteq S(q_2)$, this implies that **E** is also outside the smaller cone $S(q_2)$. Note that the point-to-cone distance $\mathcal{D}(\cdot)$ is monotonically decreasing w.r.t. the increase in the sector-cone area. Therefore, we obtain $\mathcal{D}(\mathbf{E}, S(q_1)) \leq \mathcal{D}(\mathbf{E}, S(q_2))$, which implies $f(S(q_1)) \geq f(S(q_2))$.