HyperFormer: Enhancing Entity and Relation Interaction for Hyper-Relational Knowledge Graph Completion

Zhiwei Hu
School of Computer and Information Technology
Shanxi University
Taiyuan, China
zhiweihu@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
j.z.pan@ed.ac.uk

ABSTRACT

Hyper-relational knowledge graphs (HKGs) extend standard knowledge graphs by associating attribute-value qualifiers to triples, which effectively represent additional fine-grained information about its associated triple. Hyper-relational knowledge graph completion (HKGC) aims at inferring unknown triples while considering its qualifiers. Most existing approaches to HKGC exploit a global-level graph structure to encode hyper-relational knowledge into the graph convolution message passing process. However, the addition of multi-hop information might bring noise into the triple prediction process. To address this problem, we propose HyperFormer, a model that considers local-level sequential information, which encodes the content of the entities, relations and qualifiers of a triple. More precisely, HyperFormer is composed of three different modules: an ***entity neighbor aggregator*** module allowing to integrate the information of the neighbors of an entity to capture different perspectives of it; a ***relation qualifier aggregator*** module to integrate hyper-relational knowledge into the corresponding relation to refine the representation of relational content; a ***convolution-based bidirectional interaction*** module based on a convolutional operation, capturing pairwise bidirectional interactions of entity-relation, entity-qualifier, and relation-qualifier. Furthermore, we introduce a Mixture-of-Experts strategy into the feed-forward layers of HyperFormer to strengthen its representation capabilities while reducing the amount of model parameters and computation. Extensive experiments on three well-known datasets with four different conditions demonstrate HyperFormer's effectiveness. Datasets and code are available at https://github.com/zhiweihu1103/HKGC-HyperFormer.

CCS CONCEPTS

• **Computing methodologies → Artificial intelligence**; *Knowledge representation and reasoning*; **Semantic networks**;

*Contact Authors.

KEYWORDS

knowledge graphs, hyper-relational knowledge graphs, knowledge graph completion

1 INTRODUCTION

Knowledge Graphs (KGs) [23] store and organize factual knowledge of the world using triples of the form (h, r, t) [22], capturing that entities h, t are connected via relation r. Popular KGs, such as WordNet [21], Freebase [3], and Wikidata [35], are widely used in several tasks, ranging from question answering [8, 13, 25, 27] to recommendation systems [24, 38]. However, relational KGs have no means for representing additional information of facts. For example, for the triple (*Joe Biden*, *educated at*, *University of Delaware*) represented in Figure 1, it is non-trivial representing the major studied by Joe Biden at *the University of Delaware*. To address this shortcoming, *hyper-relational KGs* have been proposed, extending binary relational KGs by associating with each triple additional attributes in the form of relation-entity pairs, known as *qualifiers*. Thus in this case a relational fact is composed by the main triple and its qualifiers. For example, for the triple in Figure 1, the qualifier pairs (*academic major*, *political science*), (*academic degree*, *Bachelor of Arts*), (*start time*, *1961*), and (*end time*, *1965*) describe the major and degree information of *Joe Biden* education at *the University of Delaware* from *1961* to *1965*. Like standard KGs, hyper-relational KGs are inevitably incomplete. To tackle this problem, several hyper-relational knowledge graph completion (HKGC) approaches have been recently proposed [9, 26, 29, 37], to examine the impact of the addition of qualifier pairs on the knowledge graph completion task.

Most existing methods for HKGC [9, 29, 37] employ graph convolutional networks (GCNs) to incorporate qualifier pairs information into entity and relation embeddings. In particular, when encoding the content of the graph structure, these approaches use multiple layers of graph convolution operations to incorporate multi-hop information into the representation of entities. Although these methods enrich the representation of entities, they inevitably introduce additional noise by considering information that might not be relevant for an entity. More precisely, the first source of

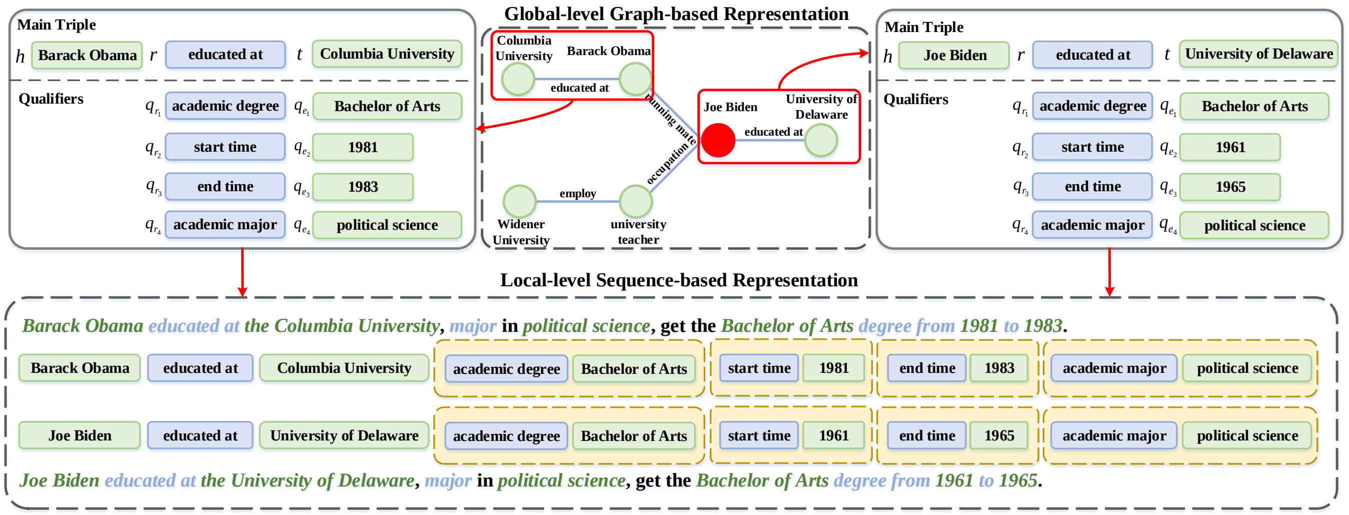


Figure 1: The global-level graph-based representation and local-level sequence-based representation based on two triples with qualifier pairs.

noise comes from entities in the standard KG, i.e., entities occurring in main triples. For instance, in Figure 1 in the *Global-level Graph-based Representation* part, when trying to predict (*Joe Biden*, *educated at*, *?*) using two layers of graph convolution operations, information about the enities *Columbia University*, *Barack Obama*, *Widener University*, *university teacher* will be incorporated into the representation of *Joe Biden*. However, the confidence of the true answer *University of Delaware* will be affected by the information from the entities *Columbia University* and *Widener University*, as the three schools share a high degree of similarity. A second source of noise comes from the introduction of hyper-relational knowledge. Going back to the previous example, to predict the triple (*Joe Biden*, *educated at*, *?*) with qualifiers (*start time*, *1961*), and (*end time*, *1965*), the neighbor (*Barack Obama*, *educated at*, *Columbia University*) includes the qualifiers (*start time*, *1981*) and (*end time*, *1983*), which will affect the representation of the relation *educated at* and the prediction of where *Joe Biden* was educated at.

The main objective of this paper is to introduce an alternative to global graph operations for the HKGC task. Note that, as shown in Figure 1 in the *Local-level Sequence-based Representation* part, the local sequential content does not introduce redundant entity and relation content present in the global graph structure, because the entities and relations involved in a local sequence are directly related to the content to be predicted. With this in mind, we introduce **HyperFormer**, a framework for HKGC that considers local-level sequential information and abandons the global-level structural content. Specifically HyperFormer integrates hyper-relational information into the entity and relation embeddings of a fact by using three modules: an ***entity neighbor aggregator*** module allowing to integrate the information of one-hop local neighbors of an entity into its representation to capture different perspectives of it; a ***relation qualifier aggregator*** module to integrate hyper-relational knowledge into the corresponding relation representation, so that relations occurring in different facts are contextualized by the qualifier pairs information; a ***convolution-based bidirectional interaction*** module based on a convolutional operation, capturing pairwise bidirectional interactions of entity-relation, entity-qualifier, and relation-qualifier. Furthermore, to increase HyperFormer's capacity while reducing the amount of parameters and calculations, we introduce a *Mixture-of-Experts (MoE)* strategy to leverage the sparse activation nature in the feed-forward layers of transformers. Our contributions can be summarized as follows:

- We propose a framework for HKGC that fully exploits local-level sequential information, while preserving the structural information of qualifiers. Further, we integrate the information of one-hop neighbors of an entity to capture different perspectives of it. In addition, the adoption of a bidirectional interaction mechanism strengthens the awareness between entities, relations, and qualifiers.
- We introduce a MoE strategy to enhance the representation capabilities of HyperFormer, while reducing the number of parameters and calculations of the model.
- We conduct extensive experiments with four different conditions: mixed-percentage mixed-qualifier, fixed-percentage mixed-qualifier, fixed-percentage fixed-qualifier, and different numbers of entity's neighbors. Our results show that HyperFormer achieves SoTA performance for HKGC. We also conducted various ablation studies.

2 RELATED WORK

Knowledge Graph Completion. There are mainly two kinds of existing KGC methods: structure-based methods and description-based methods. Depending on the type of embedding space, structure-based methods can be divided into three categories: (i) point-wise space methods, e.g., TransE [4], TransR [14], HAKE [46]; (ii) complex vector space methods, e.g. ComplEx [31], RotatE [30], QuatE [45];

(iii) manifold space methods, e.g., MuRP [2], AttH [5]. Description-based methods leverage text descriptions of entities and relations, e.g., KG-BERT [42], StAR [36], CoLE [16]. There are also KGC approaches [10, 40] considering schema information.

Hyper-relational Knowledge Graph Completion. Earlier works on HKGC proposed embedding-based methods to learn and reason with hyper-relational knowledge [11, 18, 39, 44]. However, they assume that hyper-relational knowledge is equally important for all relations, which is often not the case. To encode the contribution of different qualifier pairs, HINGE [26] adopts a convolutional framework to iteratively convolve every qualifier pair into the main triple, which can naturally discriminate the importance of different hyper-relational facts. StarE [9] extends CompGCN [33] by encoding the qualifier pairs of a triple and further combining it with the relation representation, and then uses a transformer [34] encoder to model the interaction between qualifiers and the main triple. Hy-Transformer [43] replaces the computation-heavy GCN aggregation module with a layer normalization operation [1], significantly improving the computational efficiency. QUAD [29], based on the StarE model, proposes a framework that utilizes multiple aggregators to learn better representations for hyper-relational facts. However, the global-level graph structure adopted by both StarE and QUAD integrates the multi-hop neighbor content into the corresponding entity through the graph convolution process, which inevitably introduces noise, because the node content far away from an entity will affect the real representation of such entity. HAHE [20] introduces the global-level and local-level attention to model the graphical structure and sequential structure, however, the introduction of graph structure information brings a large burden to model computation. GRAN [37] represents hyper-relational facts as a heterogeneous graph, representing it with edge-aware attentive biases to capture both local and global dependencies within the given facts. In particular, GRAN also considers a local sequential representation structure and captures the semantic information inside hyper-relation facts by using a transformer encoder [34]. However, GRAN has three shortcomings. First, it only considers the knowledge directly related to the current statement, fully ignoring any type of information from the neighbors of an entity. Second, the constraining process of the hyper-relational knowledge onto the main triple is simply handed over to a transformer, without capturing the pair-like structure of qualifiers. Third, the transformer uses full connection attention to realize the interaction between each token in the sequence, ignoring the explicit interaction between entities, relations and hyper-relational knowledge.

3 METHOD

In this section, we describe the architecture of HyperFormer (cf. Figure 2). We start by introducing necessary background (§ 3.1), and then present in detail its modules. (§ 3.2).

3.1 Background

Hyper-relational Knowledge Graph. Let $\mathcal{V}$ and $\mathcal{R}$ be finite sets of *entities* and *relation types*, respectively. Furthermore, let $Q = 2^{(\mathcal{R}\times\mathcal{V})}$. A *hyper-relational knowledge graph* $\mathcal{G}$ is a tuple $(\mathcal{V}, \mathcal{R}, \mathcal{T})$, where $\mathcal{T}$ is a finite set of *(qualified) relational facts.* The relational facts in $\mathcal{T}$ are of the form (h, r, t, qp) where $h, t \in \mathcal{V}$ are the *head* and *tail* entities, $r \in \mathcal{R}$ is the *relation* connecting h and t, and $qp = \{(q_{r_1}, q_{e_1}), \ldots, (q_{r_n}, q_{e_n})\} \in Q$ is a set of *qualifier pairs*, with *qualifier relations* $q_{r_i} \in \mathcal{R}$ and *qualifier entities* $q_{e_i} \in \mathcal{V}$. We will refer to (h, r, t) as the *main triple* of the relational fact. Under this representation regime, we can enrich the main triple (*Joe Biden*, *educated at*, *University of Delaware*) in Figure 1 with the additional semantic information provided by the qualifiers as follows: (*Joe Biden*, *educated at*, *University of Delaware*, (*academic degree*, *Bachelor of Arts*), (*academic major*, *political science*), (*start time*, *1961*), (*end time*, *1965*)). Crucial to our approach is the information provided by the neighbors of an entity: For an entity h, we define its *neighbors* $\mathcal{N}_h = \{(r, t, qp) \mid (h, r, t, qp) \in \mathcal{G}\}$.

Hyper-relational Knowledge Graph Completion. Following previous work, similar to the KG completion task, the HKGC task aims at predicting the correct head or tail entity in a fact. More precisely, given a relational fact $(h, r, ?, qp)$ or $(?, r, t, qp)$ with the tail or head entity of the main triple missing, the aim is to infer the missing entity ? from $\mathcal{V}$.

3.2 Model Architecture

We present in this section the three modules of HyperFormer: (*i*) *Entity Neighbor Aggregator*, §3.2.1; (*ii*) *Relation Qualifier Aggregator*, §3.2.2; (*iii*) *Convolution-based Bidirectional Interaction*, §3.2.3. Furthermore, we introduce a Mixture-of-Experts strategy, §3.2.4.

3.2.1 ENA: Entity Neighbor Aggregator. We explore a transformer mechanism (previously used for other KG-reated tasks [6, 12, 16]) for the HKGC task. Given a relational fact (h, r, t, qp), for predicting the tail entity t, similar to the input representation of BERT [7], we build an input sequence $S = (h, r, \texttt{[MASK]}, qp)$, where `[MASK]` is a special token used in place of entity t. We randomly initialize each token input vector to feed it into a transformer and get the output representation $\boldsymbol{E}^{[mask]}$ of the `[MASK]` token:

$$(\boldsymbol{E}^h, \boldsymbol{E}^r, \boldsymbol{E}^{[mask]}, \boldsymbol{E}^{qp}) = \texttt{Trm}(h, r, \texttt{[MASK]}, qp) \qquad (1)$$

The representation $\boldsymbol{E}^{[mask]}$ captures the information interaction between h, r and qp. We will then use $\boldsymbol{E}^{[mask]}$ to score all candidate entities and infer the most likely tail entities.

Enhanced Representation of Entity Neighbors. The one-hop neighbors of an entity help to describe it from different perspectives. In many cases, simultaneously considering the information from multiple neighbors of an entity is necessary to correctly infer the entities that are related to it. For instance, in Figure 2, using the connections (*position held*, *President*) and (*work location*, *Washington, D.C.*) from *Joe Biden*, we can infer that *Joe Biden* is the president of *the United States*, but not the prime minister of *United Kingdom*. To adequately encode the content of the neighbors of an entity, we introduce the **E**ntity **N**eighbor **A**ggregator (**ENA**) module. Given a masked tuple $(h, r, \texttt{[MASK]}, qp)$, besides considering the embeddings of h, r and qp, we also introduce information about the neighbors of h as follows:

(1) For all relational facts in $\mathcal{G}$ having h as the head, we generate masked 4-tuples by using the placeholder `[MASK]` in place of h. More precisely, fix entity h, we define the following set of neighbors of h: $\mathcal{N}_h = \{(\texttt{[MASK]}, r', t', qp') \mid (h, r', t', qp') \in \mathcal{G}\}$.

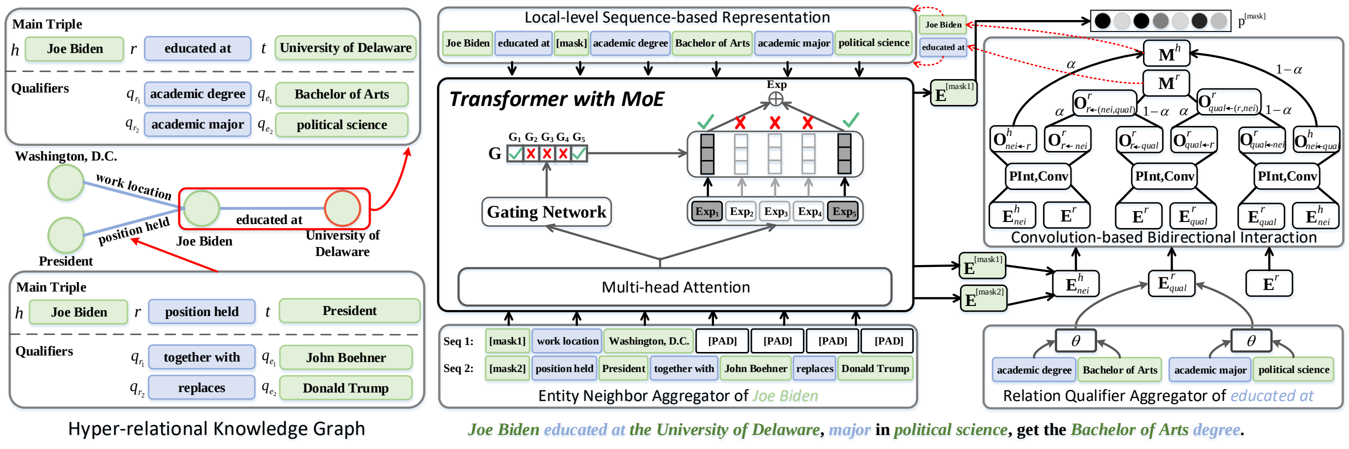

Main Triple
h Joe Biden r educated at t University of Delaware
Qualifiers
academic degree
Bachelor of Arts
academic major
political science
Washington, D.C.
work location
position held
educated at
Joe Biden
University of Delaware
President
Main Triple
h Joe Biden r position held t President
Qualifiers
together with
John Boehner
replaces
Donald Trump
Hyper-relational Knowledge Graph
Local-level Sequence-based Representation
Transformer with MoE
Gating Network
Exp
Multi-head Attention
Seq 1: [mask1] work location Washington, D.C. [PAD] [PAD] [PAD] [PAD]
Seq 2: [mask2] position held President together with John Boehner replaces Donald Trump
Entity Neighbor Aggregator of Joe Biden
Convolution-based Bidirectional Interaction
PInt,Conv
Relation Qualifier Aggregator of educated at
Joe Biden educated at the University of Delaware, major in political science, get the Bachelor of Arts degree.

Figure 2: An overview of our HyperFormer model, containing three modules: Entity Neighbor Aggregator (§ 3.2.1), Relation Qualifier Aggregator (§ 3.2.2), and Convolution-based Bidirectional Interaction (§ 3.2.3).

Note that for different tuples of the form (h, r', t', qp') with h as the head, different [MASK] representations will be obtained. For each tuple $N \in \mathcal{N}_h$, the masked token representation of N is denoted as $\boldsymbol{N}_h^{[mask]}$.

(2) We then sum up all [MASK] representations of the elements in $\mathcal{N}_h$ and further average the results to obtain the aggregated representation of the neighbors of h as $\boldsymbol{E}_{nei}^h = mean(\sum_{N \in \mathcal{N}_h} \boldsymbol{N}_h^{[mask]})$.

3.2.2 RQA: Relation Qualifier Aggregator. As discussed, qualifiers allow to describe relational content in a fine-grained manner. For example, in Figure 2 the qualifier pairs (*academic major*, *political science*) and (*academic degree*, *Bachelor of Arts*) can respectively describe the major and degree information of the relation *educated at* in the main triple (*Joe Biden*, *educated at*, *University of Delaware*). Directly serializing the qualifier content qp into the main triple $(h, r, \texttt{[MASK]})$ can form a tuple representation $(h, r, \texttt{[MASK]}, qp)$, but this representation destroys the structural information of the qualifiers content. For example, *political science* can limit the major that *Joe Biden* obtained at the *University of Delaware* when it appears at the same time as *academic major*. To incorporate the sequence $(h, r, \texttt{[MASK]}, qp)$ along with the hyper-relational knowledge into the message passing process without damaging the qualifier's structural knowledge, we introduce the **R**elation **Q**ualifier **A**ggregator (**RQA**) module. Specifically, we use the following three steps to obtain the aggregated representation of qualifier pairs for the relation occurring in a main triple:

(1) For a relational fact $(h, r, \texttt{[MASK]}, qp)$ with $qp = \{(q_{r_1}, q_{e_1}), \ldots, (q_{r_n}, q_{e_n})\}$, we randomly initialize the embedding of qualifier relations and qualifier entities for each qualifier pair, getting the input embedding: $\boldsymbol{qp} = \{(\boldsymbol{q}_{r_1}, \boldsymbol{q}_{e_1}), \ldots, (\boldsymbol{q}_{r_n}, \boldsymbol{q}_{e_n})\}$.

(2) Qualifier pairs provide additional complementary relational knowledge, each of them capturing different aspects of it. So, we aim to acquire a representation of r given the knowledge of a qualifier pair q_{r_i} and q_{e_i}. To this end, we consider both a relation r and a qualifier pair (q_{r_i}, q_{e_i}) as a form of pseudo-triple (r, q_{r_i}, q_{e_i}). We can then use several knowledge representation functions based on embedding methods to get a representation of r: We compose the representations of the qualifier relations $\boldsymbol{q}_{r_i}$ and qualifier entities $\boldsymbol{q}_{e_i}$ using an entity-relation function θ, such as TransE [4], DistMult [41], ComplEx [31] or RotatE [30]. We denote this composition as $\Theta_i = \theta(\boldsymbol{q}_{r_i}, \boldsymbol{q}_{e_i})$.

(3) The representations of different qualifier pairs are aggregated via a position-invariant summation function, which is then averaged: $\boldsymbol{E}_{qual}^r = mean(\sum_{i=1}^n \Theta_i)$.

3.2.3 CBI: Convolution-based Bidirectional Interaction. To obtain enhanced entity and relation representations, we could directly combine $\boldsymbol{E}^h$ and $\boldsymbol{E}_{nei}^h$, $\boldsymbol{E}^r$ and $\boldsymbol{E}_{qual}^r$ using the addition operation. However, this simple fusion method cannot fully realize the deep interaction between entity, relation and qualifier pairs. Indeed, it is not enough to simply pass the message between the three of them to the transformer, because its fully-connected attention layer only captures universal inter-token associations. To address this shortcoming, we propose a novel **C**onvolution-based **B**idirectional **I**nteraction (**CBI**) module to explicitly integrate each state of pair-wise representation pairs: *entity-relation*, *entity-qualifiers* and *relation-qualifiers*. For instance, to obtain the new relational embedding, the information of the entity neighbor embedding $\boldsymbol{E}_{nei}^h$ and qualifier embeddings $\boldsymbol{E}_{qual}^r$ can be integrated into the relation embedding $\boldsymbol{E}^r$ using the following four steps:

(1) We combine $\boldsymbol{E}^r$ and $\boldsymbol{E}_{nei}^h$ based on the convolution operation $\mathbf{Conv}(\circ, \circ)$, and then pass the obtained joint representation to a *perceptron interaction layer* $\mathbf{PInt}(\circ)$ to obtain $\boldsymbol{O}_{r\leftarrow nei}^r$ and $\boldsymbol{O}_{nei\leftarrow r}^h$ as follows:

$$[\boldsymbol{O}_{r\leftarrow nei}^r; \boldsymbol{O}_{nei\leftarrow r}^h] = \mathbf{PInt}(\mathbf{Conv}(\boldsymbol{E}^r, \boldsymbol{E}_{nei}^h)) \tag{2}$$

We analogously fusion $\boldsymbol{E}^r$ with $\boldsymbol{E}_{qual}^r$ and $\boldsymbol{E}_{nei}^h$ with $\boldsymbol{E}_{qual}^r$ to respectively get $\boldsymbol{O}_{r\leftarrow qual}^r$ and $\boldsymbol{O}_{qual\leftarrow r}^r$, $\boldsymbol{O}_{nei\leftarrow qual}^h$ and $\boldsymbol{O}_{qual\leftarrow nei}^r$:

$$[\boldsymbol{O}_{r\leftarrow qual}^r; \boldsymbol{O}_{qual\leftarrow r}^r] = \mathbf{PInt}(\mathbf{Conv}(\boldsymbol{E}^r, \boldsymbol{E}_{qual}^r)) \tag{3}$$

$$[\boldsymbol{O}^h_{nei\leftarrow qual}; \boldsymbol{O}^r_{qual\leftarrow nei}] = \mathbf{PInt}(\mathbf{Conv}(\boldsymbol{E}^h_{nei}, \boldsymbol{E}^r_{qual})) \quad (4)$$

The used $\mathbf{Conv}(\circ, \circ)$ operation, for the initial fusion of two vectors, is similar to the one proposed by InteractE [32], but in principle any other vector technique could be used instead. We use a one-layer MLP as our $\mathbf{PInt}(\circ)$ operation. Note that the result of **PInt** is divided into two parts to obtain two enhanced vector representations. For example, for the integration of $\boldsymbol{E}^r$ and $\boldsymbol{E}^r_{qual}$, after the two above operations are performed, we obtain the qualifier-aware relation representation $\boldsymbol{O}^r_{r\leftarrow qual}$ and relation-aware qualifier representation $\boldsymbol{O}^r_{qual\leftarrow r}$. These representations are defined in a bidirectional way, so each of them contributes to the definition of the other.

(2) We then employ a gating mechanism to combine both the entity's neighbor-aware relation representation $\boldsymbol{O}^r_{r\leftarrow nei}$ and the qualifier-aware relation representation $\boldsymbol{O}^r_{r\leftarrow qual}$. The final representation of relation r is denoted as $\boldsymbol{O}^r_{r\leftarrow (nei,qual)}$:

$$\boldsymbol{O}^r_{r\leftarrow (nei,qual)} = \alpha \odot \boldsymbol{O}^r_{r\leftarrow nei} + (1-\alpha) \odot \boldsymbol{O}^r_{r\leftarrow qual} \quad (5)$$

$$\alpha = \sigma(W_1 \boldsymbol{O}^r_{r\leftarrow nei} + W_2 \boldsymbol{O}^r_{r\leftarrow qual} + b_1 + b_2) \quad (6)$$

where α is the reset gate that controls the flow of information from $\boldsymbol{O}^r_{r\leftarrow nei}$ to $\boldsymbol{O}^r_{r\leftarrow qual}$, σ is the sigmoid function and W_1, W_2, b_1, b_2 are the parameters to be learned.

(3) Similarly, we get the relation-aware and qualifier-aware entity's neighbor representation $\boldsymbol{O}^h_{nei\leftarrow (r,qual)}$, and the relation-aware and entity's neighbor-aware qualifier representation $\boldsymbol{O}^r_{qual\leftarrow (r,nei)}$. Then we add up $\boldsymbol{O}^r_{r\leftarrow (nei,qual)}$ and $\boldsymbol{O}^r_{qual\leftarrow (r,nei)}$ to obtain the final relational representation $\boldsymbol{M}^r$. We use $\boldsymbol{O}^h_{nei\leftarrow (r,qual)}$ as the final entity representation, denoted as $\boldsymbol{M}^h$, which is the result of combining $\boldsymbol{O}^h_{nei\leftarrow r}$ and $\boldsymbol{O}^h_{nei\leftarrow qual}$ using Step (2).

(4) For the input masked fact $(h, r, \texttt{[MASK]}, qp)$, after performing the above three steps, we get enhanced representations $\boldsymbol{M}^h, \boldsymbol{M}^r$ of the entity h and relation r, which can respectively be used as the input initialization (together with the randomly initialized [MASK] and qp) to the transformer encoder. The output $\boldsymbol{E}^{[mask]}$ is then used to score all candidate entities. More precisely, we use a standard softmax classification layer to predict the target entity, and use cross-entropy between the one-hot label and the prediction as training loss, defined as:

$$p^{[mask]} = softmax(\boldsymbol{E}^{ent}\boldsymbol{E}^{[mask]}) \quad (7)$$

$$\mathcal{L} = -\sum_{mask} y^{[mask]} \log p^{[mask]} \quad (8)$$

$\boldsymbol{E}^{ent}$ represents the embedding matrix of all entities in the dataset, $p^{[mask]}$ is the predicted distribution of the [MASK] position over all entities, $y^{[mask]}$ is the corresponding one-hot label of the [MASK] position.

3.2.4 Transformer with Mixture-of-Experts. Although transformers can achieve good results in many fields, a recognized challenge for them is that the model parameters grow quadratically as the embedding dimension increases. However, it has been noted that two-thirds of the parameters of a transformer are concentrated in the feed-forward layers (FFN), and that not all of them are necessary [15]. To limit the training burden while increasing the model size, we introduce a Mixture-of-Experts (MoE) [28] strategy into the transformer, which will help selecting the necessary parameters through a gating mechanism. More precisely, given an input x, MoE includes n expert networks $\{\mathbf{Exp}_1(x), \mathbf{Exp}_2(x), ..., \mathbf{Exp}_n(x)\}$, with $\mathbf{Exp}_i(x) \in \mathbb{R}^{d\times d}$, for all $1 \le i \le n$ and a gating network $\mathbf{G}(x) \in \mathbb{R}^{d\times n}$ used to select specific experts: the *i-th* element $\mathbf{G}_i(x)$ in $\mathbf{G}(x)$ specifies whether the expert $\mathbf{Exp}_i(x)$ should be selected. The output of the MoE module is calculated as follows:

$$\begin{aligned} \mathbf{Exp} &= \textstyle\sum_{i=1}^n \mathbf{G}_i(x) \cdot \mathbf{Exp}_i(x) \\ \mathbf{G}(x) &= softmax(x \cdot \boldsymbol{W}_{gate}) \\ \mathbf{Exp}_i(x) &= x(\boldsymbol{W}_i + b_i)\boldsymbol{W}'_i + b'_i \end{aligned} \quad (9)$$

$\boldsymbol{W}_i, \boldsymbol{W}'_i \in \mathbb{R}^{d\times d}$ are the learnable parameters, $\boldsymbol{W}_{gate} \in \mathbb{R}^{d\times n}$ is a trainable matrix with n the number of experts, and each expert $\mathbf{Exp}_i(x)$ corresponds to a FFN. $\mathbf{G}_i(x) \in \mathbb{R}^{d\times 1}$ is the value of $\mathbf{G}$ at the *i-th* position on the *2nd* dimension. In practice, we set it to 0 or 1 depending on whether the value of $\mathbf{G}_i(x)$ exceeds a certain threshold, so $\mathbf{G}(x)$ is sparse. We may set a large number of experts, but for each sample only k of them are selected, called top experts. In the experimental part, we set *n=64* and *k=2*.

4 EXPERIMENTS

Table 1: Statistics of datasets under mixed-percentage mixed-qualifier and fixed-percentage mixed-qualifier scenarios. The values in parentheses indicate that the corresponding percentage in corresponding dataset has hyper-relational facts.

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

In this section, we present the results of the conducted experiments. We first describe the datasets, evaluation protocol and implementation details (§ 4.1), and describe the baseline models (§ 4.1.2). We then discuss the main experimental results (§. 4.2). Finally, we present results of our ablation studies (§ 4.3).

4.1 Experiment Setup

4.1.1 Datasets. We evaluate HyperFormer on three well-known datasets: WD50K [9], WikiPeople [11], and JF17K [39]. WD50K and WikiPeople are derived from Wikidata, and JF17K is collected from Freebase. These datasets have the following two characteristics: i) only certain percentage of main triples contain qualifiers, 13.6% in WD50K, 2.6% in WikiPeople and 45.9% in JF17K; ii) each triple contains a different number of qualifiers, 0~7 for WikiPeople, and 0~4 for JF17K, where the qualifier number means that the main triple does not contain hyper-relational knowledge. We refine these datasets from two perspectives, based on the percentage of triples containing hyper-relational knowledge and on the number of qualifiers associated to triples. So, we construct three datasets with different conditions: *Mixed-percentage Mixed-qualifier, Fixed-percentage Mixed-qualifier, Fixed-percentage Fixed-qualifier*, where Mixed-percentage and Mixed-qualifier respectively indicate that the number of triples with qualifiers is arbitrary (not fixed) and that the number of qualifiers per triple is not fixed. The Fixed condition is defined as expected, and clearly, there is no Mixed-percentange in the Fixed qualifier scenario. In addition, we also construct the datasets in which all entities have low degree. The four scenarios are specifically described as follows:

(1) **Mixed-percentage Mixed-qualifier.** These datasets are directly taken from [9]. They aim at verifying the generalization performance in the scenario where the percentage of triples with qualfiers and the number of qualifiers associated with each triple is arbitrary.
(2) **Fixed-percentage Mixed-qualifier.** We construct subsets of existing datasets in which the percentage of triples with qualifiers is fixed. For example, for WD50K we construct: WD50K (33), WD50K (66) and WD50K (100), with the number in parentheses representing the percentage of triples with qualifiers. We construct similar subsets for the WikiPeople and JF17K datasets. The corresponding datasets statistics are presented in Table 1.
(3) **Fixed-percentage Fixed-qualifier.** Due to the sparsity of higher qualifier facts in WikiPeople and JF17K datasets, we follow GETD [17] to filter out the triples with 3 and 4 associated qualifiers, obtaining WikiPeople-3, WikiPeople-4, JF17K-3, and JF17K-4, respectively. The corresponding datasets statistics are presented in Table 2.
(4) **Entities with Low Degree.** To evaluate the performance of the tested models depending on the node degrees (number of neighbors), we construct subsets of existing datasets in which all nodes have the low degree. In this case, we select as the basic datasets data in which *all* triples have qualifiers. For example, for WD50K (100), we construct four subsets with node degrees from one to four, denoted as WD50K (100) #1, WD50K (100) #2, WD50K (100) #3, and WD50K (100) #4. The corresponding datasets statistics are presented in Table 3.

Table 2: Statistics of datasets under fixed-percentage fixed-qualifier scenarios.

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

Table 3: Statistics of datasets with different number of node degrees. The value behind # indicates that the entity in the training set only contains the number of neighbors with the corresponding value.

Datasets	Train	Valid	Test	Entity	Relation
WD50K (100) #1	2191	3279	5297	10375	189
WD50K (100) #2	4382	3279	5297	11241	200
WD50K (100) #3	6547	3279	5297	11985	207
WD50K (100) #4	8506	3279	5297	12649	210
WikiPeople (100) #1	1253	1181	1173	4212	83
WikiPeople (100) #2	2498	1181	1173	4711	85
WikiPeople (100) #3	3647	1181	1173	5040	87
WikiPeople (100) #4	4515	1181	1173	5338	89
JF17K (100) #1	2492	3152	4142	7320	253
JF17K (100) #2	4984	3152	4142	7930	255
JF17K (100) #3	7294	3152	4142	8367	257
JF17K (100) #4	9219	3152	4142	8688	259

4.1.2 Baselines. We compare HyperFormer with various state-of-the-art methods for hyper-relational knowledge graph completion: m-TransH [39], RAE [44], NaLP-Fix [26], HINGE [26], StarE [9], Hy-Transformer [43], GRAN [37], and QUAD [29]. Note that GRAN contains three variants, i.e., GRAN-hete, GRAN-homo and GRAN-complete. If there is no special suffix, GRAN denotes GRAN-hete. There are two variants of QUAD: QUAD and QUAD (Parallel). If there is no special suffix, QUAD denotes QUAD (Parallel).

4.1.3 Evaluation Protocol. We evaluate the model performance using two common metrics: MRR and Hits@N (abbreviated as H@N). MRR is the average of reciprocal ranking, and Hits@N is the proportion of top N (we use N={1,3,10}). For both metrics, the larger the value, the better the performance of a model.

4.1.4 Implementation Details. All experiments are conducted on six 32G Tesla V100 GPUs. Our method is implemented with PyTorch. We employ AdamW [19] as the optimizer and a cosine decay scheduler with linear warm-up is used for optimization. We determine the hyperparameter values by using a grid search based on the MRR performance on the validation dataset. We select the neighbors of an entity in {2, **3**, 4}, the qualifier pairs of a relation in {5, **6**, 7}, the learning rate in {3e-4, 4e-4, 5e-4, **6e-4**, 7e-4}, the label smoothing factor in {0.3, 0.5, 0.7, **0.9**}, the number of layer in a Transformer in {2, 4, **8**, 16}, the head number in {1, **2**, 4, 8}, the Transformer input dropout rate in {0.6, **0.7**, 0.8}, the Transformer hidden dropout rate in {**0.1**, 0.2, 0.3}, the dimensions of the embedding size in {80, 200, 320, **400**}, the number of convolution channels in {64, **96**, 128}, the convolutional kernel size is **9**, the convolutional input dropout rate in {0.1, **0.2**, 0.3}, the convolutional hidden dropout rate in {0.4, **0.5**, 0.6}, the number of experts in the MoE module in {8, 16, 32, **64**}, the number of top experts in the MoE module in {**2**, 4, 6, 8}.

4.2 Main Results

Mixed-percentage Mixed-qualifier. Table 4 reports the results on the Hyper-relational KGC task with mixed-percentage mixed-qualifier on the WD50K, WikiPeople, and JF17K datasets. We can

Table 4: Evaluation of different models with mixed-percentage mixed-qualifier on the WD50K, WikiPeople and JF17K datasets. All baseline results are collected from the original literature. Best scores are highlighted in bold, the second best scores are underlined, and '–' indicates the results are not reported in previous work.

Methods	WD50K (13.6)			WikiPeople (2.6)			JF17K (45.9)		
	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
m-TransH [39]	–	–	–	0.063	0.063	0.300	0.206	0.206	0.463
RAE [44]	–	–	–	0.059	0.059	0.306	0.215	0.215	0.469
NaLP-Fix [26]	0.177	0.131	0.264	0.420	0.343	0.556	0.245	0.185	0.358
HINGE [26]	0.243	0.176	0.377	0.476	0.415	0.585	0.449	0.361	0.624
StarE [9]	0.349	0.271	0.496	0.491	0.398	**0.648**	0.574	0.496	0.725
Hy-Transformer [43]	0.356	0.281	0.498	0.501	0.426	0.634	0.582	0.501	0.742
GRAN-homo [37]	–	–	–	0.487	0.410	0.618	0.611	0.533	0.767
GRAN-complete [37]	–	–	–	0.489	0.413	0.617	0.591	0.510	0.753
GRAN-hete [37]	–	–	–	**0.503**	**0.438**	0.620	0.617	0.539	0.770
QUAD [29]	0.348	0.270	0.497	0.466	0.365	0.624	0.582	0.502	0.740
QUAD (Parallel) [29]	0.349	0.275	0.489	0.497	0.431	0.617	0.596	0.519	0.751
HyperFormer	**0.366**	**0.288**	**0.514**	0.473	0.361	0.646	**0.664**	**0.601**	**0.787**

Table 5: Evaluation of different models with fixed-percentage mixed-qualifier on the WD50K, WikiPeople and JF17K datasets. Best scores are highlighted in bold.

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
StarE [9]	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
Hy-Transformer [43]	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 [37]	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
QUAD [29]	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
HyperFormer	**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**

observe that HyperFormer significantly outperforms all baselines on the WD50K and JF17K datasets. Specifically, HyperFormer respectively achieves performance improvements of 1.0% / 0.7% / 1.6% in MRR / Hits@1 / Hits@10 on WD50K, compared to the best performing baseline, Hy-Transformer. It gets analogous improvements of 4.7% / 6.2% / 1.7% on JF17K. On WikiPeople its performance is slightly below the SoTA. These results can be explained by the fact that both WD50K and JF17K contain a relatively high percentage of triples with qualifier pairs: 13.6% and 45.9%, respectively. However, WikiPeople has a much lower percentage of triples with qualifiers, 2.6%, so the triple-only facts dominate the overall score. Hyperformer successfully exploits the interaction between entities, relations and qualifiers to improve the performance on the HKGC task, especially on datasets with a rich amount of hyper-relational knowledge.

Fixed-percentage Mixed-qualifier. We also investigate the effectiveness of HyperFormer and the baselines under different ratios of relational facts with qualifiers. For each of the used datasets, we obtained three subsets (as described in Point 2 in Section 4.1) containing approximately 33%, 66%, and 100% of facts with qualifiers. Table 5 presents an overview of the obtained results. We observe that HyperFormer gets larger improvements over the baselines when the percentage of available facts with qualifiers is higher. Specifically, on WikiPeople, HyperFormer respectively achieves improvements over QUAD of 0.9% / 1.6% / 4.1% in the 33% / 66% / 100% variants. This shows that an important reason of why Hyperformer could not surpass GRAN-hete in the mixed-percentage mixed-qualifier HKGC task on the WikiPeople dataset (cf. Table 4) is that it only contains a very small amount of triples with hyper-relational knowledge. Indeed, the main strength of Hyperformer is in the integration of hyper-relational knowledge by capturing the interaction of entities, relations and qualifiers.

Fixed-percentage Fixed-qualifier. We investigate the performance on hyper-relational data with fixed number of qualifiers in Table 6. We observe that HyperFormer consistently achieves state-of-the-art performance on all datasets in Table 6. At the same time, we find out that the performance of all models is significantly lower in the mixed-percentage mixed-qualifier datasets than in scenarios with a fixed number of hyper-relational knowledge, which is consistent for both WikiPeople and JF17K. This might be explained by the fact that uneven distributions among different quantities of hyper-relational facts may affect the stability of model training.

Different Numbers of Neighbors. We also investigate the performance of the models on entities with few neighbors in Table 7.

Table 6: Evaluation of different models with fixed-percentage fixed-qualifier on WikiPeople and JF17K datasets. 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
StarE [9]	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
Hy-Transformer [43]	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 [37]	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
QUAD [29]	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
HyperFormer	**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**

Table 7: MRR results of different node degrees on the WD50K(100), WikiPeople(100) and JF17K(100) datasets. The last line shows the difference between best scores and the second best scores.

Methods	WD50K (100)				WikiPeople (100)				JF17K (100)			
	#1	#2	#3	#4	#1	#2	#3	#4	#1	#2	#3	#4
StarE [9]	0.104	0.208	0.313	0.369	0.121	0.112	0.193	0.255	0.169	0.249	0.275	0.286
Hy-Transformer [43]	0.071	0.167	0.315	0.374	0.091	0.148	0.186	0.233	0.137	0.241	0.299	0.318
GRAN [37]	0.125	0.235	0.327	0.374	0.119	0.186	0.242	0.273	0.203	0.267	0.284	0.301
QUAD [29]	0.065	0.134	0.284	0.371	0.075	0.140	0.186	0.255	0.228	0.241	0.280	0.306
HyperFormer	**0.193**	**0.303**	**0.374**	**0.410**	**0.194**	**0.252**	**0.303**	**0.328**	**0.305**	**0.338**	**0.350**	**0.374**
Absolute improvement (%)	6.8%	6.8%	4.7%	3.6%	7.3%	6.6%	6.1%	5.5%	7.7%	7.1%	5.1%	5.6%

In this case we look at training datasets in which all entities have one, two, three or four neighbors (see Point 4 in Section 4.1), while the validation and test sets remain unchanged. We found that the baseline models perform very poorly on these subsets. For example, in WD50K_100 (#1), StarE, and QUAD can respectively obtain an MRR metric of 10.4% and 6.5%, while HyperFormer can achieve 19.3%. This is explained by the way that these two models use the global-level structure to encode qualifier knowledge into the relation representation, which is suitable for scenarios in which all nodes have several neighbors. GRAN achieves 12.5%, since it does encode qualifiers into the main triple in a local-level fashion like HyperFormer, but it ignores the structural content of qualifier pairs. Differently, HyperFormer proposes a new integration method that realizes the interaction between entities, relations and qualifiers.

4.3 Ablation Studies

We verify the contribution of each component of HyperFormer and the effect of different hyperparameters on the performance. First, we explore the impact of different translation operations on the performance, cf. Table 8. Then, we look at different variants of the model, and different hidden sizes, number of experts, and values of label smoothing, cf. Figure 3. Finally, we show in Table 9 the amount of parameters and calculations with and without MoE.

Different Translation Methods. Table 8 shows detailed results of selecting different translation methods to compose the qualifier entity and qualifier relation. Specifically, we adopt four translation methods, i.e., TransE [4], DistMult [41], ComplEx [31], and RotatE [30]. We find that the selection of translation methods has little impact on the performance. This may be because different translation methods are only used to convert entities and relations in the qualifier pairs into single vector, while the subsequent CBI module is used to determine the combined representation of qualifier pairs and its impact on entities and relations.

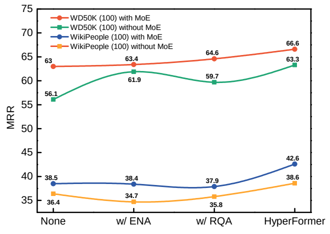
Different Variants of HyperFormer. Figure 3(a) presents the results on the impact of each component of HyperFormer. We consider four variants with/without MoE in the transformer part: (i) without any modules, denoted *None*; (ii) with the ENA module as the only component, denoted *w/ ENA*; (iii) with the RQA module as the only component, denoted *w/ RQA*; (iv) with both ENA, RQA, and CBI, denoted as HyperFormer. We observe that the introduction of the ENA or RQA module in some cases can bring an improvement in the performance compared to *None*. The addition of the CBI module brings a consistent improvement, because CBI realizes the bidirectional interaction between entities, relations, and hyper-relational knowledge. In addition, after adding the MoE mechanism to the transformer, a further improvement is obtained.

Different Hidden Sizes. Figure 3(b) presents results showing the influence of the hidden sizes. We observe that the increase of the hidden size helps to capture a large amount of messages, which can improve the model performance. However, after the hidden size is set to 320, the results show a stable trend. This indicates that it is not the case that the higher the embedding dimension, the better the performance of the model, which is consistent with the finding by [15]. Note that setting a larger hidden size requires more video memory, so in practice, after the performance is stable the minimum value is set as the final one.

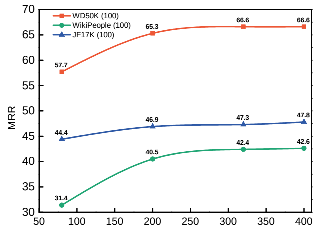
Number of Experts. We also investigate the impact of the number of experts on the performance of HyperFormer, cf. Figure 3(c). The selection of the number of experts is an important factor as it directly affects the size of the model. We observe that by selecting a larger number of experts a slight improvement is obtained. In practice, it is necessary to balance the number of video memory and experts, while ensuring that the number of experts remains as

Table 8: Evaluation of different transaction methods on WD50K(100), WikiPeople(100) and JF17K(100) datasets. Best scores are highlighted in bold.

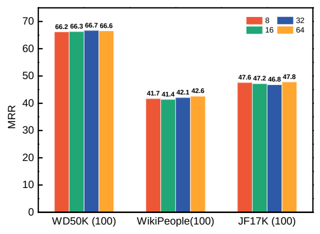
Methods	WD50K (100)				WikiPeople (100)				JF17K (100)			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
HyperFormer-TransE	0.666	**0.611**	0.697	0.768	0.424	**0.374**	0.452	0.524	**0.487**	**0.401**	**0.526**	**0.662**
HyperFormer-DistMult	0.666	**0.611**	**0.698**	0.770	**0.426**	0.373	**0.454**	**0.527**	0.478	0.396	0.515	0.645
HyperFormer-ComplEx	**0.667**	**0.611**	**0.698**	0.769	0.422	0.370	0.445	0.518	0.479	0.399	0.519	0.642
HyperFormer-RotatE	0.655	0.592	0.690	**0.772**	0.415	0.371	0.434	0.496	0.479	**0.401**	0.515	0.644



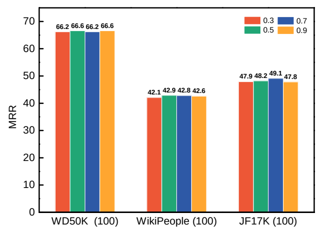
(a) Different variants of model



(b) Different hidden sizes



(c) Number of experts



(d) Label smoothing

Figure 3: The ablation studies results under different experimental conditions.

small as possible without affecting the performance.

Table 9: The amount of parameters and calculation with or without the introduction of MoE. w/ means with, w/o means without.

Metrics	Mode	WD50K	WikiPeople	JF17K
FLOPs	w/ MoE	118.397G	118.167G	118.070G
	w/o MoE	286.851G	286.621G	286.524G
params	w/ MoE	66.956M	66.956M	66.956M
	w/o MoE	79.877M	79.877M	79.877M

Label Smoothing. The label smoothing strategy has been successfully used in the KGC task [30, 46]. It mitigates the bias of the pre-trained data due to random sampling. In Figure 3(d), we observe that setting different label smoothing values brings subtle performance differences, showing that HyperFormer is robust to the label smoothing strategy, and therefore not causing performance gaps due to improper value selection. In addition, we note that there is no unique label smoothing value that works for all datasets.

Parameters and Computational Complexity. The number of parameters can be used to evaluate the trainable parameters of a model, while computational cost refers to the number of floating-point operations (FLOPs) required during training or inference. Table 9 shows that introducing the MoE mechanism can simultaneously reduce the number of parameters and computational cost, with a more significant reduction in computational cost. This is intuitively explained by the fact that MoE replaces the feed-forward layers in the original transformer. So, when obtaining the final result, only the predictions from experts with higher confidence are considered, suppressing the involvement of irrelevant neurons in the entire computation process. As a result, both the number of parameters and computational cost are reduced simultaneously.

5 CONCLUSION AND FUTURE WORK

In this paper, we proposed HyperFormer, a framework for the HKGC task which strengthens the bidirectional interaction between entities, relations, and qualifiers, while retaining the structural information of qualifiers in a local-level sequence. Experiments under different conditions on the WD50K, WikiPeople, and JF17K datasets show that HyperFormer achieves in most cases better results than existing models. The ablation experimental results demonstrate the effectiveness of each module of HyperFormer. For future work, we will integrate other types of data in a KG, e.g., entities's textual descriptions or literals, for better entity representation, and apply HyperFormer into larger scale HKGs like the full WikiData.

ACKNOWLEDGMENTS

This work has been supported by the National Natural Science Foundation of China (No.61936012, No.62076155), by the Key Research and Development Program of Shanxi Province (No.202102020101008), by the Science and Technology Cooperation and Exchange Special Program of Shanxi Province (No.202204041101016), by the Chang Jiang Scholars Program (J2019032) and by a Leverhulme Trust Research Project Grant (RPG-2021-140).

REFERENCES

[1] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. *CoRR* abs/1607.06450 (2016), 1–14.
[2] Ivana Balazevic, Carl Allen, and Timothy M. Hospedales. 2019. Multi-relational Poincaré Graph Embeddings. In *NeurIPS*. Curran Associates, Vancouver, BC, Canada, 4465–4475.
[3] 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.
[4] 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.

[5] 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.

[6] Sanxing Chen, Xiaodong Liu, Jianfeng Gao, Jian Jiao, Ruofei Zhang, and Yangfeng Ji. 2021. HittER: Hierarchical Transformers for Knowledge Graph Embeddings. In *EMNLP*. ACL, Virtual Event / Punta Cana, Dominican Republic, 10395–10407.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*. ACL, Minneapolis, MN, USA, 4171–4186.

[8] Biralatei Fawei, Jeff Z. Pan, Martin J. Kollingbaum, and Adam Z. Wyner. 2020. A Semi-automated Ontology Construction for Legal Question Answering. *New Generation Computing* 37(4), 1 (2020), 453–478.

[9] 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.

[10] Yuxia Geng, Jiaoyan Chen, Zhuo Chen, Jeff Z Pan, Zhiquan Ye, Zonggang Yuan, Yantao Jia, and Huajun Chen. 2021. OntoZSL: Ontology-enhanced zero-shot learning. In *WWW*. ACM, New York, NY, USA, 3325–3336.

[11] 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.

[12] 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.

[13] Zhiwei Hu, Víctor Gutiérrez-Basulto, Zhiliang Xiang, Xiaoli Li, Ru Li, and Jeff Z. Pan. 2022. Type-aware Embeddings for Multi-Hop Reasoning over Knowledge Graphs. In *IJCAI*. ijcai.org, Vienna, Austria, 3078–3084.

[14] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *AAAI*. AAAI Press, Austin, Texas, USA, 2181–2187.

[15] 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.

[16] 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.

[17] Yu Liu, Quanming Yao, and Yong Li. 2020. Generalizing Tensor Decomposition for N-ary Relational Knowledge Bases. In *WWW*. ACM, Taipei, Taiwan, 1104–1114.

[18] Yu Liu, Quanming Yao, and Yong Li. 2021. Role-Aware Modeling for N-ary Relational Knowledge Bases. In *WWW*. ACM, Virtual Event / Ljubljana, Slovenia, 2660–2671.

[19] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *ICLR*. OpenReview.net, New Orleans, LA, USA, 1–11.

[20] 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.

[21] George A. Miller. 1995. WordNet: A Lexical Database for English. *Commun. ACM* 38, 11 (1995), 39–41.

[22] J. Z. Pan. 2009. Resource Description Framework. In *Handbook of Ontologies*. Springer, Berlin, Germany.

[23] J. Z. Pan, G. Vetere, J.M. Gomez-Perez, and H. Wu (Eds.). 2017. *Exploiting Linked Data and Knowledge Graphs for Large Organisations*. Springer, Berlin, Germany.

[24] Tao Qi, Fangzhao Wu, Chuhan Wu, and Yongfeng Huang. 2021. Personalized News Recommendation with Knowledge-aware Interactive Matching. In *SIGIR*. ACM, Virtual Event, Canada, 61–70.

[25] Hongyu Ren, Hanjun Dai, Bo Dai, Xinyun Chen, Michihiro Yasunaga, Haitian Sun, Dale Schuurmans, Jure Leskovec, and Denny Zhou. 2021. LEGO: Latent Execution-Guided Reasoning for Multi-Hop Question Answering on Knowledge Graphs. In *ICML*. PMLR, Virtual Event, 8959–8970.

[26] Paolo Rosso, Dingqi Yang, and Philippe Cudré-Mauroux. 2020. Beyond Triplets: Hyper-Relational Knowledge Graph Embedding for Link Prediction. In *WWW*. ACM, Taipei, Taiwan, 1885–1896.

[27] Apoorv Saxena, Aditay Tripathi, and Partha P. Talukdar. 2020. Improving Multi-hop Question Answering over Knowledge Graphs using Knowledge Base Embeddings. In *ACL*. ACL, online, 4498–4507.

[28] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. 2017. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. In *ICLR*. OpenReview.net, Toulon, France, 1–19.

[29] 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.

[30] 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.

[31] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. In *ICML*. JMLR.org, New York City, NY, USA, 2071–2080.

[32] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, Nilesh Agrawal, and Partha P. Talukdar. 2020. InteractE: Improving Convolution-Based Knowledge Graph Embeddings by Increasing Feature Interactions. In *AAAI*. AAAI Press, New York, NY, USA, 3009–3016.

[33] 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.

[34] 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.

[35] Denny Vrandecic and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* 57, 10 (2014), 78–85.

[36] Bo Wang, Tao Shen, Guodong Long, Tianyi Zhou, Ying Wang, and Yi Chang. 2021. Structure-Augmented Text Representation Learning for Efficient Knowledge Graph Completion. In *WWW*. ACM, Virtual Event / Ljubljana, Slovenia, 1737–1748.

[37] Quan Wang, Haifeng Wang, Yajuan Lyu, and Yong Zhu. 2021. Link Prediction on N-ary Relational Facts: A Graph-based Approach. In *Findings of the ACL*. ACL, online, 396–407.

[38] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge Graph Attention Network for Recommendation. In *SIGKDD*. ACM, Anchorage, AK, USA, 950–958.

[39] 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.

[40] Kemas Wiharja, Jeff Z. Pan, and Martin J. Kollingbaum. 2020. Schema aware iterative Knowledge Graph completion. *Journal of Web Semantics* 65, 100616 (2020), 100616.

[41] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *ICLR*. OpenReview.net, San Diego, CA, USA, 1–12.

[42] Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. KG-BERT: BERT for Knowledge Graph Completion. *CoRR* abs/1909.03193 (2019), 1–8.

[43] Donghan Yu and Yiming Yang. 2021. Improving Hyper-Relational Knowledge Graph Completion. *CoRR* abs/2104.08167 (2021), 1–5.

[44] 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.

[45] Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019. Quaternion Knowledge Graph Embeddings. In *NeurIPS*. Curran Associates, Vancouver, BC, Canada, 2731–2741.

[46] Zhanqiu Zhang, Jianyu Cai, Yongdong Zhang, and Jie Wang. 2020. Learning Hierarchy-Aware Knowledge Graph Embeddings for Link Prediction. In *AAAI*. AAAI Press, New York, NY, USA, 3065–3072.