Improving Hyper-Relational Knowledge Graph Completion

Donghan Yu
Carnegie Mellon University
Pittsburgh, USA
dyu2@cs.cmu.edu

Yiming Yang
Carnegie Mellon University
Pittsburgh, USA
yiming@cs.cmu.edu

ABSTRACT

Different from traditional knowledge graphs (KGs) where facts are represented as entity-relation-entity triplets, hyper-relational KGs (HKGs) allow triplets to be associated with additional relation-entity pairs (a.k.a. qualifiers) to convey more complex information. How to effectively and efficiently model the triplet-qualifier relationship for prediction tasks such as HKG completion is an open challenge for research. This paper proposes to improve the best-performing method in HKG completion, namely STARE, by introducing two novel revisions: (1) Replacing the computation-heavy graph neural network module with light-weight entity/relation embedding processing techniques for efficiency improvement without sacrificing effectiveness; (2) Adding a qualifier-oriented auxiliary training task for boosting the prediction power of our approach on HKG completion. The proposed approach consistently outperforms STARE in our experiments on three benchmark datasets, with significantly improved computational efficiency.

CCS CONCEPTS

• Computing methodologies → Neural networks; Reasoning about belief and knowledge.

KEYWORDS

knowledge graph completion, hyper relation, neural network

1 INTRODUCTION

Knowledge graphs (KGs) have received increasing attention in recent machine learning community due to their broad range of important applications including natural language processing [11, 14], recommender systems [10], computer vision [4], and more. KGs typically store factual information in the form of (head entity, relation, tail entity) triplet, such as (Alan Turing, educated at, Princeton University).

Hyper-relational KGs (HKGs) go beyond conventional KGs by representing facts with more complex semantic information, e.g., using relation-entity pairs as the qualifiers of triplets. The combination of a triplet and its qualifiers together is called a statement. For example, one statement can be (Alan Turing, educated at, Princeton University, (academic degree, Doctorate), (academic major, Mathematics)) where (academic degree, Doctorate) and (academic major, Mathematics) are qualifiers. This statement conveys the information that Alan Turing studied at Princeton University and received a doctorate degree major in math from there. Such hyper-relational data is ubiquitous in KGs. For example in Freebase [2], more than 30% of its entities are involved in such hyper-relational facts [19].

HKG completion task, which aims to predict semantically valid but unobserved triplets based on the observed ones, is an important task and has been intensively studied for triplet-based KGs in recent years [12]. However, this task remains underexplored for hyper-relational KGs. Existing works often lose semantic information in statement representation learning as a results of ignoring the paired relationship between qualifier entity and qualifier relation [7, 15, 19, 20], not differentiating triplets from qualifiers [9], or failing to model the interaction among multiple qualifiers [16]. The most successful approach so far is STARE [8], which uses a graph neural network (GNN) module to improve entity and relation embeddings and a Transformer [18] encoder network to model the interactions among qualifiers and the triplet being qualified in one statement. Although STARE achieved the state-of-the-art HKG completion performance, its GNN module causes a large computation overhead, which would limit its practical success to very large applications.

In this paper we aim to address the computational efficiency issue in STARE as well as to improve its prediction performance, with the following strategies: (1) Replacing the computation-heavy GNN module with light-weight entity/relation embedding processing techniques; (2) Adding an qualifier-oriented auxiliary training task for boosting the prediction power on HKG completion. Experiment results on three benchmark datasets show that our model achieves the new state-of-the-art results and is significantly more efficient than STARE.

2 METHOD

We begin by introducing mathematical notations. In a hyper-relational KG \(G_H\), we denote the set of entities and relations as \(V\) and \(R\) respectively. The total number of entities is \(N\) and the number of relations is \(M\). The edge connecting them, which we call a statement (or fact), is expressed in the domain \(\mathcal{V} \times \mathcal{R} \times \mathcal{V} \times \mathcal{P} (\mathcal{R} \times \mathcal{V})\) where \(\mathcal{P}\) denotes the power set. It’s usually written as \((m_h, m_r, m_t, Q)\) where \((m_h, m_r, m_t)\) is the main triplet of the statement containing head entity \(m_h \in \mathcal{V}\), relation \(m_r \in \mathcal{R}\) and tail entity \(m_t \in \mathcal{V}\) respectively. \(Q\) is the set of qualifiers consisting of relation-entity pairs \(\{(q_{ri}, q_{ei})\}_{i=1}^{n}\) where \(q_{ri} \in \mathcal{R}\) and \(q_{ei} \in \mathcal{V}\). Note that the number...
of qualifier pairs \( n \) can be different for different statements. \( n \) can also be 0 when one statement only has the main triplet \((Q = \emptyset)\) is an empty set in this case).

The completion task on hyper-relational KGS is that, following previous setting \([8, 16]\), given an incomplete statement \((m_h, m_r, ?, Q)\) or \((?, m_r, m_t, Q)\) where its head or tail entity of the main triplet is missing, the model is required to predict the missing entity from entity set \( \mathcal{V} \).

### 2.1 Recap of STARE

We first introduce the basic of our model: STARE\(^1\). Similar to the masking mechanism in BERT \([6]\), STARE place a special [MASK] entity in the position of unknown entity in the input statement. For example, if the tail entity is unknown, the input is \((m_h, m_r, [MASK], Q)\). Then it's flattened into a sequence \( s = [m_h, m_r, [MASK], q_{r1}, q_{e1}, \ldots, q_{rn}, q_{en}] \) following the order from main triplet to qualifiers. The maximum input length is set as \(T \) and the sequence will be padded if its length is less than \(T \).

Suppose the initial entity embedding matrix is \( E \in \mathbb{R}^{(N+1) \times d} \) where the [MASK] entity takes the last row of embedding matrix and \( d \) is the embedding dimension. The initial relation embedding matrix is \( R \in \mathbb{R}^{M \times d} \). STARE first updates the embedding matrices with a graph neural network (GNN) module:

\[
\hat{E}, \hat{R} = \text{GNN}(E, R, \mathcal{G}_H) \tag{1}
\]

where \( \hat{E} \in \mathbb{R}^{(N+1) \times d} \) and \( \hat{R} \in \mathbb{R}^{M \times d} \) are the updated entity and relation embeddings respectively. By doing this, the local context information of entities has been explicitly encoded into their representations. Then based on the input sequence \( s \), STARE queries the entity and relation embeddings from \( \hat{E} \) and \( \hat{R} \) and concatenate them to form the statement representation \( S \):

\[
S = [\hat{E}_{m_h}; \hat{R}_{m_r}; \hat{E}_{\text{mask}}; \ldots; \hat{E}_{q_{r1}}; \hat{E}_{q_{e1}}; \ldots] \in \mathbb{R}^{T \times d} \tag{2}
\]

Then \( S \) is forwarded to a Transformer encoder module and the output is the updated statement representation \( \hat{S} \):

\[
\hat{S} = \text{Transformer}(S) \in \mathbb{R}^{T \times d} \tag{3}
\]

where, for simplicity, we assume the dimension of hidden representation in the Transformer model is also \( d \). Then the representation of [MASK] entity is selected to pass through a feed-forward neural network \( f: \mathbb{R}^{d} \rightarrow \mathbb{R}^{d} \) and dot product with each entity embedding to calculate the similarity values. Finally an element-wise sigmoid function is applied on the values to output the probability of each entity being the missing entity:

\[
\sigma = \text{Sigmoid}(f(\hat{S}_{\text{ind}_{\text{mask}}})\hat{E}_{0:N-1}^T) \in \mathbb{R}^{1 \times N} \tag{4}
\]

where \( \sigma_i \) is the probability entity \( i \) being the missing entity, \( \text{ind}_{\text{mask}} \) is the index of the [MASK] entity in the input sequence. \( \hat{E}_{0:N-1} \) is the processed entity embedding matrix without the embedding of [MASK] entity, which should be avoided as final output. Next, we'll introduce the two revisions we made on STARE.

\(^{1}\)The original paper proposes different variants of models. Our model is based on STARE+MskTrf, which we refer to STARE for simplicity.

---

**Figure 1**: A simple realization of our proposed model Hy-Transformer compared to STARE. The different parts are marked in red. LN is short for layer normalization \([1]\).

### 2.2 Replacing the GNN Module

We first propose to replace the GNN module for reducing computation cost. We show the inefficiency of GNN model by computing its time complexity of each training step: For each statement \((m_h, m_r, m_t, Q)\), the GNN module computes the combined embedding \( h_Q \) of qualifiers \( Q \) as follows and update the embeddings of entity \( m_h \) and \( m_t \) with it:

\[
h_Q = W \sum_{(q_{r1}, q_{e1}) \in Q} \phi(\hat{R}_{q_{r1}}, \hat{E}_{q_{e1}}) \tag{5}
\]

where \( \phi: \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^d \) is a function to combine qualifier entity embedding and relation embedding and \( W \in \mathbb{R}^{d \times d} \) is a projection matrix. The computation of \( h_Q \) involves matrix-vector multiplication which takes \( O(d^2) \). Even if we ignore the time complexity of other operations, the total complexity is already \( O(L_0Zd^2) \) where \( Z \) is the total number of statements in the HKG and \( L_0 \) is the number of GNN layers. This can be a large computation overhead since \( Z \) can be very large. Note that this complexity can not be reduced by only updating the entities involved in the mini-batch of current training step, since the whole entity embedding matrix \( \hat{E} \) is used in Equation \((4)\) for output entity prediction.

Another motivation to remove the GNN module is that intuitively the input statement already contains sufficient information to predict the unknown entity and the Transformer module is expressive enough to capture such information. Thus the encoding of local context of entities by graph convolution network might not be necessary.

To replace the GNN module, we propose to use layer normalization \([1]\) (LN) and dropout \([17]\) for entity and relation embedding processing:

\[
\hat{E} = \text{Dropout}(\text{LN}(E)) \in \mathbb{R}^{(N+1) \times d} \tag{6}
\]

\[
\hat{R} = \text{LN}(R) \in \mathbb{R}^{M \times d} \tag{7}
\]

where LN can introduce inductive bias about the embedding distribution by re-centering and re-scaling. Dropout has been empirically
We further propose an auxiliary training task to boost the model performance on HKG completion: predicting the missing qualifier entity in a statement. For each training statement with qualifiers, and for each \( i \in \{1, 2, \ldots, n\} \), we construct an incomplete statement \((m_1, m_r, m_t, (q_{r1}, q_{e1}), \ldots, (q_{rn}, q_{en}))\) where the qualifier entity \( q_{ei} \) is missing instead of triplet entity. Given the incomplete statement, our model needs to predict the missing qualifier entity. Similar to predicting missing entity in the main triplet introduced in Section 2.1, we place a [MASK] entity in the position of missing qualifier entity and flat it into a sequence \( s = [m_1, m_r, m_t, q_{r1}, q_{e1}, \ldots, q_{rn}, q_{en}] \). Then this sequence is input into our model and the representation of [MASK] entity after Transformer module is used for final entity prediction. The final training data is simply the union of the training data on the primary task and on the auxiliary task.

Although the primary task is to predict the entity of main triplet, we think this auxiliary training task can augment the training data to learn better interaction among the entities and relations in a statement. The effectiveness of this training strategy will be empirically demonstrated in Section 3.3.

Our final model is named as Hy-Transformer where Hy is the abbreviation of Hyper-relational KG. Figure 1 shows the comparison between our model and STARE.

### 3 EXPERIMENT

In this section, we demonstrate both the effectiveness and efficiency of our model by comprehensive experiments on three benchmark datasets.

#### 3.1 Basic Setting

We conduct experiments on the following benchmark datasets: JF17K [19], WikiPeople [9] and WD50K [8], where the dataset statistics is shown in Table 1. We follow the conventional train/valid/test split setting [8, 9, 19]. Following previous work [5, 8], we train the model in 1-N setting using binary cross entropy loss with label smoothing. The optimizer is Adam [13] and the learning rate is 0.0001. The number of training epochs is 400 for JF17K and WD50K and 500 for WikiPeople. For the transformer module, the number of layer is 2 and the hidden dimension is 512 with dropout rate as 0.1. The dimension of initial entity/relation embedding is 200. The dropout rate for entity embedding matrix is 0.3. We perform grid search to find the best performing hyperparameters similar in [8]. All the experiments are conducted on one Nvidia 2080-Ti GPU.

The baseline methods we compare are m-TransH [19], RAE [20], NaLP-Fix [16] (an improved version of NaLP [9]), HINGE [16], vanilla Transformer [8], and STARE [8]. For the evaluation metric, we use the filtered setting [3] for computing mean reciprocal rank (MRR) and hits at 1 and 10 (H@1 and H@10). Higher MRR, H@1, and H@10 scores indicate better performance. The metrics are computed for head entity and tail entity prediction separately and then averaged.

#### 3.2 Main Results

Table 2 shows the HKG completion results on all the three datasets. We see that our model Hy-Transformer consistently outperforms the strongest baseline method STARE except slightly lower H@10 on the WikiPeople dataset. Our model gets 7.0% higher H@1 and 3.6% higher H@1 on the WikiPeople dataset and WD50K dataset respectively. Moreover, our model outperforms other baselines by a large margin: more than 5.3%, 13.6%, and 24.4% in MRR on the three datasets respectively.

We also show the training time comparison between our model and STARE in Figure 2. As shown in the figure, our model takes much less training time to achieve the same test MRR results. For example, in the WikiPeople dataset, our model achieved 0.45 MRR within 2 hours while STARE takes about 10 hours, which is 5 times slower. In the WD50K dataset, our model takes 1 hour to achieve 0.31 MRR while STARE costs about 10 hours, which is 10 times slower. These results demonstrate that our model can be trained much more efficiently than STARE while still achieving on-par or better results. This also shows that the larger the KG is, the more efficiency improvement our model can gain, which is consistent with the analysis in Section 2.2.

#### 3.3 Ratio of Statements with Qualifiers

In this section, we study the model performance under different ratios of hyper-relational statements (statements with qualifiers). In the WD50K dataset, about 13% of statements contain qualifiers while the remaining 87% only have main triplets. We further conduct experiments on three different variants of the WD50K dataset provides by [8]: WD50K (33), WD50K (66), and WD50K (100), which...
Table 2: Experiment results in HKG completion on all the three datasets. The baseline results are directly taken from [8]. The average results over 5 different runs are reported for our model.

| Methods              | WikiPeople | JF17K       | WD50K       |
|----------------------|------------|-------------|-------------|
|                      | MRR        | H@1        | H@10       |
| m-TransH [19]        | -          | 0.063      | 0.300      |
| RAE [20]             | -          | 0.059      | 0.306      |
| NaLP-Fix [16]        | 0.420      | 0.343      | 0.556      |
| HINGE [16]           | 0.476      | 0.415      | 0.585      |
| Transformer [8]      | 0.469      | 0.403      | 0.586      |
| STARE [8]            | 0.491      | 0.398      | 0.648      |
| Hy-Transformer       | **0.501**  | **0.426**  | **0.634**  |

Table 3: MRR results in HKG completion on three variants of the WD50K dataset with different qualifier ratios. The baseline results are directly taken from [8]. The averaged results over 5 different runs are reported for our model. HT w/o aux stands for our model Hy-Transformer without the auxiliary training task.

| Method                  | WD50K (33) | WD50K (66) | WD50K (100) |
|-------------------------|------------|------------|-------------|
| Transformer [8]         | 0.276      | 0.404      | 0.562       |
| STARE [8]               | 0.331      | 0.481      | 0.654       |
| HT w/o aux              | 0.332      | 0.490      | 0.661       |
| Hy-Transformer          | **0.343**  | **0.515**  | **0.699**   |

contain approximately 33%, 66%, and 100% of hyper-relational statements, respectively. For example, on WD50K (100) dataset, all the statements contain one or more qualifiers.

The results are shown in Table 3, where we see that our model outperforms STARE and Transformer even without training with the auxiliary task proposed in Section 2.3. By the auxiliary training task, we see that the MRR results of our model improve by 3.3%, 5.1%, and 5.7% on WD50K (33), WD50K (66), and WD50K (100) respectively. This demonstrates the effectiveness of our proposed training task. Moreover, the improvement becomes larger as the ratio of hyper-relational statements increases, which makes sense because the augmented training signals comes from statements with qualifiers.

### 3.4 Ablation Study

In this section, we illustrate the effectiveness of each component for embedding processing in Hy-Transformer by conducting ablation study on the WD50K (100) dataset mentioned in Section 3.3.

As shown in Table 4, without entity embedding layer normalization, the performance of our model drops significantly by 6.6% of MRR. This shows that controlling the scale of entity embeddings is very important in this task. Entity embedding dropout also plays an essential role by improving 4.0% of MRR since it helps to prevent over-fitting when the number of entity embedding parameters is very large. Layer normalization on relation embeddings can also slightly improve the performance. Overall, all the three embedding processing techniques can improve the HKG completion performance.

Table 4: HKG completion results on the WD50K (100) datasets. LN is short for Layer Normalization.

| Method                  | MRR   | H@1  | H@10 |
|-------------------------|-------|------|------|
| Hy-Transformer          | 0.699 | 0.637| 0.812|
| w/o entity embedding LN | 0.653 | 0.572| 0.803|
| w/o entity embedding dropout | 0.672 | 0.617| 0.772|
| w/o relation embedding LN | 0.693 | 0.632| 0.806|

### 4 CONCLUSION

We tackle the problem of hyper-relational KG completion. Based on the current best-performing model STARE, we propose two novel strategies to improve both its efficiency and effectiveness: (1) replacing the computation-heavy GNN module with light-weight techniques, layer normalization and dropout, on entity/relation embeddings; (2) adding an auxiliary training task which predicts the missing qualifier entity for boosting the prediction power on HKG completion. Our model Hy-Transformer consistently outperforms all the baseline methods over three benchmark datasets and is significantly more efficient than STARE. In the future, we plan to apply our model into larger scale of HKGs like the full Wikidata with millions of entities and extend our model to incorporate additional contextual information like the text descriptions of entities and relations.

### REFERENCES

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. arXiv preprint arXiv:1607.06450 (2016).

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

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Olksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States. Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger (Eds.). 2787–2795. https://proceedings.neurips.cc/paper/2013/hash/1cecc7a7792ca8133fa24680a88d2f9-Abstract.html

[4] Riquan Chen, Tianshui Chen, Xiaolu Hui, Hefeng Wu, Guanbin Li, and Liang Lin. 2020. Knowledge Graph Transfer Network for Few-Shot Recognition. In The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020. AAAI Press, 10575–10582. https://aaai.org/ojs/index.php/AAAI/article/view/6630
