Learning Representations for Hyper-Relational Knowledge Graphs

1st Harry Shomer
Computer Science and Engineering
Michigan State University
East Lansing, USA
shomerha@msu.edu

2st Wei Jin
Department of Computer Scienc
Emory University
Atlanta, USA
wei.jin@emory.edu

3st Juanhui Li
Computer Science and Engineering
Michigan State University
East Lansing, USA
lijuanh1@msu.edu

4st Yao Ma
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, USA
may13@rpi.edu

5st Hui Liu
Computer Science and Engineering
Michigan State University
East Lansing, USA
liuhui7@msu.edu

***Abstract*—Knowledge graphs (KGs) have gained prominence for their ability to learn representations for uni-relational facts. Recently, research has focused on modeling hyper-relational facts, which move beyond the restriction of uni-relational facts and allow us to represent more complex and real-world information. However, existing approaches for learning representations on hyper-relational KGs majorly focus on enhancing the communication from qualifiers to base triples while overlooking the flow of information from base triple to qualifiers. This can lead to suboptimal qualifier representations, especially when a large amount of qualifiers are presented. It motivates us to design a framework that utilizes multiple aggregators to learn representations for hyper-relational facts: one from the perspective of the base triple and the other one from the perspective of the qualifiers. Experiments demonstrate the effectiveness of our framework for hyper-relational knowledge graph completion across multiple datasets. Furthermore, we conduct an ablation study that validates the importance of the various components in our framework.**

***Index Terms*—Knowledge graphs, link prediction.**

I. Introduction

Knowledge graphs (KGs) are a collection of facts represented in a structured and graphical format. In traditional triple-based KGs, facts are represented as binary relations between entities, which often fall short in representing the complex nature of the data. To address this shortcoming, hyper-relational KGs are introduced by moving from representing uni-relational facts to representing facts with N-ary relations. In hyper-relational KGs, triples are often associated with relation-entity pairs, which are known as *qualifiers*. Qualifiers help qualify a given fact by providing more supporting information and are defined as an (entity, relation) pair that belongs to a triple. An example is shown in Figure 1. In Figure 1 the triple *(Stephen King, Author Of, The Running Man)* contains a single qualifier pair *(Under Pseudonym, Richard Bachman)*. This qualifier pair helps provide more context to the base triple by telling us that Stephen King published the novel under the pseudonym Richard Bachman. Research on hyper-relational KGs [1]–[4] have focused on learning representations for such graphs and examining how the addition of qualifier pairs can help boost the performance of KG completion.

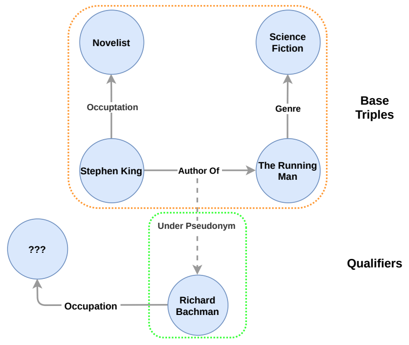

Fig. 1. An example of a hyper-relational KG. The blue circles represent entities and the arrows represent relations. The dashed arrows represent qualifier relations while the solid arrows are base relations. The ??? entity represents the potential occupation of Richard Bachman (Novelist).

However, the majority of existing approaches only consider the impact from the qualifiers on base triples while overlooking the flow of information from the base triples to the qualifiers. This can lead to suboptimal performance especially when a large amount of qualifiers are presented. To illustrate its importance, we use Figure 1 as an example. In (*Richard Bachman*, *Occupation*, *Novelist*), suppose the occupation of *Richard Bachman* is missing and we are trying to predict (*Richard Bachman*, *Occupation*, ?). If there is no information

ASONAM '23, November 6-9, 2023, Kusadasi, Turkey
© 2023 Association for Computing Machinery.
ACM ISBN 979-8-4007-0409-3/23/11…$15.00
https://doi.org/10.1145/3625007.3627591

spreading from the base triple to the qualifier entity *Richard Bachman*, predicting the missing link would be very hard as no knowledge of Stephen King is transferred to the qualifier entity. Prior work such as NaLP [1] and HINGE [5] both struggle to achieve this transfer of information due to the simplicity of their convolutional-based frameworks. StarE [2] ignores such flow of information from the base triples to the qualifiers. Although transformer-based frameworks [3], [4] model the mutual influence between base triples and qualifiers via self attention, they inevitably ignore the structured nature of KGs.

Therefore, in this work, we aim to investigate the novel problem of learning representations for hyper-relational KGs by encouraging the mutual influence between base triples and qualifiers. Essentially, we are faced with the following challenge: how to properly enhance the influence from base triples to qualifiers while maintaining effective impact from qualifiers to base triples. To address it, we propose a novel framework - **QU**alifier Aggregated Hyper-Relation**A**l Knowle**D**ge Graphs (QUAD), which encourages influence in both directions. Specifically, QUAD utilizes two aggregators from different perspectives - a base aggregator and a qualifier aggregator. The base aggregator aggregates information for the base entities from the qualifiers while the qualifier aggregator performs the aggregation from the qualifier perspective. Inside the qualifier aggregator, we further propose the concept of a "qualifier triple" that allows us to easily aggregate information from a base triple through the qualifier relation. Following both aggregations, the entity and relation representations are passed to the decoder to perform knowledge base completion. Extensive experiments demonstrate the effectiveness of our framework on multiple benchmark datasets.

II. Related Work

A. Knowledge Graph Embedding (KGE)

Many different KGE frameworks have been proposed in the literature for . [6] proposed using a translational scoring function to model triples. [7] use a bilinear scoring function to score triples utilizing a diagonal matrix to model relations. RotatE [8] scores a triple as a translation in a complex space and ConvE [9] does so using a convolutional neural network. Other works have focused on modeling KGs using graph neural networks. RGCN [10] extends GCN [11] by using a relation specific weight matrix when aggregating an entity's neighbors. However, RGCN suffers from over-parameterization when there are many relations. To alleviate this concern, CompGCN [12] proposes to use direction specific weight matrices when aggregating instead of relational weight matrices.

B. Hyper-Relational KGs

[5] propose to model Hyper-Relational KGs using a convolutional framework. For a given hyper-relational fact, the base triple is convolved by itself and with each specific qualifier pair. The resulting feature vectors are then combined and used for prediction. StarE [2] extends CompGCN [12] by encoding the qualifiers for a specific triple and combining it with the base relation of the triple. Using StarE [2] as their foundation, [3] replace the GNN aggregation with layer normalization layers [13] to improve performance. Additionally they mask the qualifier entities when training as a form of self-supervised learning (SSL). Lastly, [4] propose GRAN, a transformer based architecture that employs edge-specific attention biases and masks all entities and relations in the sequence. A missing element in previous work is the lack of a clear flow of information from the base triples to the qualifiers. To address this, we propose a graph encoder to aggregate information from the perspective of the qualifiers, thereby creating better qualifier encodings.

III. Preliminaries

A. Knowledge Graphs (KGs)

Let $\mathcal{G} = \{\mathcal{V}, \mathcal{R}, \mathcal{E}\}$ be a KG with nodes (i.e. entities) $\mathcal{V}$, edges $\mathcal{E}$, and relations $\mathcal{R}$. For $e \in \mathcal{E}$, it represents a directed edge where two entities $u \in \mathcal{V}$ and $v \in \mathcal{V}$ are connected by a relation $r \in \mathcal{R}$. We also denote the edge as a triple (v, r, u). Further, we use $\mathcal{N}_v$ to denote the neighboring entities and relations of a node v.

B. Hyper-Relational KGs

A hyper-relational KG can be seen as an extension of a standard KG where there is a set of qualifier pairs $\{(qv_i, qr_i)\}$, where $qv \in \mathcal{V}$ and $qr \in \mathcal{R}$, associated with each triple (v, r, u). For simplicity we use q to represent the set of neighboring qualifier pairs $(qv_i, qr_i) \in Q_{(v,r,u)}$ associated with a triple (v, r, u). We can therefore represent a hyper-relational fact as (v, r, u, q). We refer to hyper-relational facts as statements. Furthermore, we can represent the neighborhood for a qualifier entity qv as $(v, r, u, qr) \in \mathcal{N}_{qv}$.

A representative example for modeling hyper-relational KGs is StarE [2]. It proposes a formulation based on CompGCN [12] to incorporates an embedding $\mathbf{h}_q$ that is an encoded representation of the qualifier pairs for the base triple (v, r, u),

$$\mathbf{h}_v = f\left(\sum_{(u,r)\in\mathcal{N}_v} W_{\lambda(r)}\, \phi\left(\mathbf{h}_u, \gamma(\mathbf{h}_r, \mathbf{h}_q)\right)\right), \qquad (1)$$

where f is a non-linear function and $\lambda(r)$ represents the direction of relation that can be one of: a standard, inverse, or self-loop relation. Several function are proposed for modeling the interaction of the embedding in ϕ including subtraction, multiplication, and the cross correlation [12]. The embedding $\mathbf{h}_q$ is combined with the relation embedding $\mathbf{h}_r$ through a function γ that performs a weighted sum. [3] show that replacing the graph aggregation with layer normalization can achieve comparable if not better performance than other frameworks. Furthermore they show that masking the qualifier entities during training can raise the test performance on KG completion.

IV. The Proposed Framework

In this section, we propose a new framework QUAD for learning representations of hyper-relational KGs. An overview of QUAD is shown in Figure 2 that consists of two main

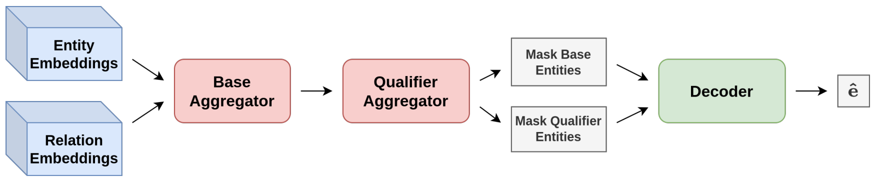

Fig. 2. An overview of QUAD. It takes the set of entity and relation embeddings and encodes them using two aggregators. We then mask both the base and qualifier separately and pass the the statement to the decoder to predict the most likely entity $\hat{e}$ for the mask.

components. The first component, detailed in Section IV-A, encodes the graph using two separate aggregations while the second component detailed in Section IV-B decodes a given hyper-relational fact for a particular downstream task.

A. The Encoder

In this subsection, we define the encoder used in QUAD. The encoder is composed of two neighborhood aggregations: one that aggregates information for the base entities and one that does so for the qualifier entities. We refer to these two aggregators as the base aggregator and qualifier aggregator, respectively. The initial entity E and relation R embeddings are first passed to the base aggregator and then the qualifier aggregator. The encoded entity embedding $\hat{E}$ and relation embedding $\hat{R}$ can be expressed as follows:

$$E', R' = \text{Base-Agg}(E, R, \mathcal{G}), \tag{2}$$

$$\hat{E}, \hat{R} = \text{Qual-Agg}(E', R', \mathcal{G}), \tag{3}$$

where Base-Agg$(\cdot)$ and Qual-Agg$(\cdot)$ are the aggregation functions in base aggregator and qualifier aggregator, respectively. In the following, we introduce the details of both aggregators.

1) The Base Aggregator: The base aggregator aims to aggregate information for the base entities. Concretely, it takes E and R as input and aggregates the neighborhood information for a given base entity v using a function ψ_v:

$$\mathbf{h}_v = \text{Aggregate}(\psi_v(\mathbf{h}_u, \mathbf{h}_r, \mathbf{h}_q), \forall (u, r, q) \in \mathcal{N}_v), \tag{4}$$

where $\mathbf{h}_q$ is the encoded representation for all the qualifier pairs belonging to the base triple (v, r, u). We utilize CompGCN [12] as the Aggregate$(\cdot)$ function. We can then rewrite Eq. (4) as follows,

$$\mathbf{h}_v = f\left(\sum_{(u,r)\in\mathcal{N}(v)} \mathbf{W}_{\lambda(r)} \psi_v\left(\mathbf{h}_u, \mathbf{h}_r, \mathbf{h}_q\right)\right), \tag{5}$$

where the encoded qualifier representation $\mathbf{h}_q$ is defined as in StarE [2] as:

$$\mathbf{h}_q = \mathbf{W}_q \sum_{(qr,qv)\in Q_{(v,r,u)}} \phi\left(\mathbf{h}_{qr}, \mathbf{h}_{qv}\right), \tag{6}$$

with $\mathbf{W}_q$ as the projection matrix, $\mathbf{h}_{qr}$ as the qualifier relation embedding, and $\mathbf{h}_{qv}$ as the qualifier entity embedding. Note that the relation $\mathbf{h}_r$ is updated through a linear transformation. StarE implements ψ_v in Eq. (5) by combining the encoded qualifier information $\mathbf{h}_q$ with the triple's relation, i.e., $\psi_v(\mathbf{h}_u, \mathbf{h}_r, \mathbf{h}_q) = \phi\left(\mathbf{h}_u, \gamma(\mathbf{h}_r, \mathbf{h}_q)\right)$ as in Eq. (1). However, it is restricted by the assumption that qualifier information should be incorporated into the base relation embedding. Instead, we remove this restriction and combine the qualifier information with the output of the composition function ϕ as follows:

$$\psi_v(\mathbf{h}_u, \mathbf{h}_r, \mathbf{h}_q) = \alpha \odot \phi(\mathbf{h}_u, \mathbf{h}_r) + (1 - \alpha) \odot \mathbf{h}_q, \tag{7}$$

where $\alpha \in [0, 1]$ is a hyperparameter that balances the contribution of the base triple information and the encoded qualifiers. In this way, the qualifier information encoded in $\mathbf{h}_q$ can directly interact with both the base entity and relation.

2) The Qualifier Aggregator: The previously introduced base aggregator only considers the aggregation of information from the qualifiers to the base triple but not vice versa. Encoding base triple information in the qualifiers is advantageous as it can help learn better representations for the qualifiers. For example, in Figure 1 the base aggregation doesn't consider the flow of information from the triple *(Stephen King, Author Of, The Running Man)* to the the qualifier entity *Richard Bachman* resulting in limited information about the author being encoded in the qualifier entity. This makes it difficult to infer new facts where *Richard Bachman* is a base entity. To encode more information for a qualifier entity, we can aggregate information from its neighbors, i.e., the base triples it belongs to and the qualifier relation connecting them to those triples. Using our previous example, the qualifier entity *Richard Bachman* would aggregate information from the base triple *(Stephen King, Author Of, The Running Man)* and the qualifier relation *Under Pseudonym.*

Hence, the Qual-Agg$(\cdot)$ function is designed to aggregate the neighborhood information for the qualifier entity qv as follows:

$$\begin{aligned}\mathbf{h}_{qv} = \; & \text{Aggregate}(\psi_q(\mathbf{h}_v, \mathbf{h}_r, \mathbf{h}_u, \mathbf{h}_{qr}), \\ & \forall (v, r, u, qr) \in \mathcal{N}_{qv}),\end{aligned} \tag{8}$$

where $\mathbf{h}_{qr}$ is the qualifier relation embedding and the function ψ_q is used to combine the neighboring embeddings. For ψ_q, we hope that the base triple embeddings $(\mathbf{h}_v, \mathbf{h}_r, \mathbf{h}_u)$ should be considered as a whole. Since the qualifiers serve as additional context for explaining the whole triple, we should aggregate information from the whole triple instead of treating $\mathbf{h}_v, \mathbf{h}_r, \mathbf{h}_u, \mathbf{h}_{qr}$ individually. To achieve this goal, we first consider that a qualifier entity qv is linked to some base triple $t = (v, r, u)$ by the qualifier relation qr. This can be seen as analogous to a standard triple where the base triple is the head, the qualifier relation is the relation, and the qualifier entity is the tail entity. For convenience, we can write this in a triple notation as (t, qr, qv), which we refer to as a *qualifier triple*.

An example found in Figure 1 would be $t =$ *(Stephen King, Author Of, The Running Man)*, $qr =$ *Under Pseudonym* and $qv =$ *Richard Bachman*. Using these ideas, we can view ψ_q as a function of the base triple t and the qualifier relation qr:

$$\psi_q(\mathbf{h}_v, \mathbf{h}_r, \mathbf{h}_u, \mathbf{h}_{qr}) = \phi(\mathbf{h}_t, \mathbf{h}_{qr}), \tag{9}$$

where the embedding $\mathbf{h}_t$ is the encoded representation of the base triple t and ϕ is defined similarly to the composition function used in CompGCN [12]. The base triple representation $\mathbf{h}_t$ is formulated as the linear projection of the concatenated embeddings of the base triple:

$$\mathbf{h}_t = \text{Linear}\left(\text{Concat}\left(\mathbf{h}_v, \mathbf{h}_r, \mathbf{h}_u\right)\right). \tag{10}$$

B. The Decoder

Using the encoded representations of the entities and relations, i.e., $\hat{E}$ and $\hat{R}$, each hyper-relational fact is passed to a decoder to make the final prediction. To decode each fact we utilize a transformer [14] that employs an architecture similar to CoKE [15] extended to include qualifier information. For a given input sample S, we mask the entity token we are trying to predict. Eq. (11) is an example where we mask the object entity:

$$S = (\mathbf{h}_v,\ \mathbf{h}_r,\ [Mask],\ \mathbf{h}_{qr1},\ \mathbf{h}_{qv1},\ \cdots). \tag{11}$$

After being passed through the transformer, we extract the masked embedding and pass it through a fully-connected layer and score it against all possible entities. This is then passed through a sigmoid function with the highest scoring entity being chosen as our prediction.

1) Model Training: As the downstream task is to perform KG completion for entities in the base triples, we mask the head and tail entities for the fact $(\mathbf{h}_v, \mathbf{h}_r, \mathbf{h}_u, q)$ where q is the set of qualifier pairs associated with the triple. Then we try to predict the masked entities and minimize the loss $\mathcal{L}_{\text{base}}$ using binary cross-entropy loss.

To further enhance the learned representations of the embeddings and exploit the qualifier information, we include an auxiliary task that masks and attempts to predict the qualifier entities for each fact as proposed in Hy-Transformer [3]. We refer to this loss as $\mathcal{L}_{\text{qual}}$ and minimize it using the binary cross-entropy loss. Since the downstream task is solely to predict the missing base entities, we introduce a hyperparameter $\beta \in [0, 1]$ that balances the contribution of the qualifier entity loss. The loss can now be written as $\mathcal{L} = \mathcal{L}_{\text{base}} + \beta\mathcal{L}_{\text{qual}}$.

C. Parallel Architecture

We also consider a version of QUAD that combines the entity representations encoded by the two aggregation schemes in parallel instead of sequentially. Under this setting, both aggregate functions take the initial entity embedding matrix $\mathcal{E}$ as input. The encoded entity representations outputted by the two encoders are then combined via a weight matrix $\mathbf{W}_p$ and passed to the decoder. The relation embeddings are still passed sequentially as we found that this performed best. We believe that this version of our framework may be better at balancing the contribution of the base and qualifier aggregation for some datasets. We formulate the parallel encoding scheme as follows where E_{base} and R_{base} are the output of Base-Agg$(\cdot)$ and E_{qual} the output of Qual-Agg$(\cdot)$:

$$E_{base}, R_{base} = \text{Base-Agg}(E, R, \mathcal{G}), \tag{12}$$

$$E_{qual}, \hat{R} = \text{Qual-Agg}(E, R_{base},\ \mathcal{G}), \tag{13}$$

$$\hat{E} = \mathbf{W}_p\ \text{Concat}\left(E_{base},\ E_{qual}\right). \tag{14}$$

V. Experiment

A. Experimental Settings

1) Datasets: We consider three datasets for our experiments including JF17K [16], Wikipeople [1], and WD50K [2]. An issue with WD50K and Wikipeople is that only a small percentage of triples contain qualifiers, being 13.6% for WD50K and 2.6% for Wikipeople. We therefore also measure the performance on the WD50K splits introduced by Galkin et al. [2] that contain a higher percentage of triples with qualifiers. The three splits are WD50K (33), WD50K (66), and WD50K (100) with the number in parentheses representing the percentage of triples with qualifiers.

2) Baselines: We compare the results of our framework with other prominent hyper-relational baselines including NaLP-Fix [5], HINGE [5], StarE [2], and Hy-Transformer [3]. Note that we do not include GRAN [4] as one baseline since (1) similar to Hy-Transformer, it is also a transformer-based method; and (2) in addition to the auxiliary task, GRAN also masks the relations and we can incorporate such component to the proposed framework that we leave as one future work.

B. Performance Comparison on Benchmarks

1) Performance on WD50K, Wikipeople and JF17K: In this subsection, we evaluate QUAD on the benchmark datasets and compare its performance to the aforementioned baselines. We first evaluate performance on the WD50K, Wikipeople, and JF17K datasets. The method *QUAD* is the original formulation of our framework while the method *QUAD (Parallel)* is the alternative formulation presented in Section IV-C. The results are shown in Table I. For each dataset we include the percentage of triples with qualifiers in parentheses.

Evaluating the results in Table I we observe that the performance of QUAD varies by dataset. For JF17K it achieves the best performance for all three metrics including a 2.4% increase in MRR over the second best performing model. On Wikipeople its performance is similar Hy-Trasnformer while for WD50K it is slightly below state of the art. This is due to both WD50K and Wikipeople containing a low percentage of triples with qualifier pairs with 13.6% and 2.6%, respectively. On JF17K, which has a much higher percentage of qualifiers at 45.9%, QUAD is able to outperform the baseline models. We therefore believe that datasets with a higher ratio of qualifiers is where QUAD shows its value. To test this hypothesis, we evaluate QUAD on the WD50K subsets introduced in Section V-A1. The percentage of triples with qualifier pairs for the three subsets is approximately 33%, 66%, and 100%, respectively.

TABLE I
KNOWLEDGE GRAPH COMPLETION RESULTS FOR MULTIPLE DATASETS

Method	**WD50K (13.6)**			**Wikipeople (2.6)**			**JF17K (45.9)**		
	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
NaLP-Fix	0.177	0.131	0.264	0.420	0.343	0.556	0.245	0.185	0.358
HINGE	0.243	0.176	0.377	0.476	0.415	0.585	0.449	0.361	0.624
StarE	0.349	0.271	0.496	0.491	0.398	**0.648**	0.574	0.496	0.725
Hy-Transformer	**0.356**	**0.281**	**0.498**	**0.501**	0.426	0.634	0.582	0.501	0.742
QUAD	0.348	0.270	0.497	0.466	0.365	0.624	0.582	0.502	0.740
QUAD (Parallel)	0.349	0.275	0.489	0.497	**0.431**	0.617	**0.596**	**0.519**	**0.751**

TABLE II
KNOWLEDGE GRAPH COMPLETION RESULTS ON WD50K SPLITS

Method	**WD50K (33)**			**WD50K (66)**			**WD50K (100)**		
	MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
NaLP-Fix	0.204	0.164	0.277	0.334	0.284	0.423	0.458	0.398	0.563
HINGE	0.253	0.190	0.372	0.378	0.307	0.512	0.492	0.417	0.636
StarE	0.331	0.268	0.451	0.481	0.420	0.594	0.654	0.588	0.777
Hy-Transformer	0.343	-	-	**0.515**	-	-	0.699	0.637	0.812
QUAD	**0.349**	0.286	**0.470**	**0.515**	**0.456**	**0.623**	**0.703**	**0.638**	**0.820**
QUAD (Parallel)	0.346	**0.287**	0.459	0.510	0.454	0.615	0.693	0.628	0.812

The results are presented in Table II. From the results, we observe SOTA for QUAD across all datasets.

C. Ablation Study

In this subsection, we conduct an ablation study to determine the importance for each component in QUAD. We do so by considering three versions of QUAD: (1) One that doesn't mask the qualifier entities, (2) one without the qualifier aggregation component and (3) one without (1) and (2). Evaluating the results on the three ablated frameworks will help us ascertain both the individual and cumulative effect those two components have on the performance. Of most importance is the impact of the novel qualifier aggregation introduced in this paper. We report the results of this study on the WD50K (100) dataset under the original (non-parallel) setting. The results can be found in Table III. We observe that each proposed component is able to improve performance, validating their inclusion.

TABLE III
ABLATION STUDY ON WD50K (100)

Method	**MRR**	**H@1**	**H@10**
w/o Qual Agg & Mask	0.658	0.591	0.781
w/o Qual Mask	0.677	0.613	0.794
w/o Qual Agg	0.696	0.628	0.820
QUAD	0.703	0.638	0.820

VI. CONCLUSION

In this paper we introduce our framework QUAD for learning representations for hyper-relational knowledge graphs.To better encode the qualifier information for a given hyper-relational fact, we design a novel qualifier aggregation module to learn better encoded representations for the qualifiers. Experiments show that our framework performs well on several benchmark datasets as compared to competitive baselines. Further experiments validate the importance of the various components in our framework and the need to balance the auxiliary loss.

REFERENCES

[1] S. Guan, X. Jin, Y. Wang, and X. Cheng, "Link prediction on n-ary relational data," in *The World Wide Web Conference*, 2019, pp. 583–593.
[2] M. Galkin, P. Trivedi, G. Maheshwari, R. Usbeck, and J. Lehmann, "Message passing for hyper-relational knowledge graphs," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 7346–7359.
[3] D. Yu and Y. Yang, "Improving hyper-relational knowledge graph completion," *arXiv preprint arXiv:2104.08167*, 2021.
[4] Q. Wang, H. Wang, Y. Lyu, and Y. Zhu, "Link prediction on n-ary relational facts: A graph-based approach," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 2021, pp. 396–407.
[5] P. Rosso, D. Yang, and P. Cudré-Mauroux, "Beyond triplets: hyper-relational knowledge graph embedding for link prediction," in *Proceedings of The Web Conference 2020*, 2020, pp. 1885–1896.
[6] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," *Advances in neural information processing systems*, vol. 26, 2013.
[7] B. Yang, S. W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in *Proceedings of the International Conference on Learning Representations (ICLR) 2015*, 2015.
[8] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang, "Rotate: Knowledge graph embedding by relational rotation in complex space," in *International Conference on Learning Representations*, 2018.
[9] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in *Thirty-second AAAI conference on artificial intelligence*, 2018.
[10] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European semantic web conference*. Springer, 2018, pp. 593–607.
[11] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.
[12] S. Vashishth, S. Sanyal, V. Nitin, and P. Talukdar, "Composition-based multi-relational graph convolutional networks," in *International Conference on Learning Representations*, 2019.
[13] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.
[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.
[15] Q. Wang, P. Huang, H. Wang, S. Dai, W. Jiang, J. Liu, Y. Lyu, Y. Zhu, and H. Wu, "Coke: Contextualized knowledge graph embedding," *arXiv preprint arXiv:1911.02168*, 2019.
[16] J. Wen, J. Li, Y. Mao, S. Chen, and R. Zhang, "On the representation and embedding of knowledge bases beyond binary relations," *arXiv preprint arXiv:1604.08642*, 2016.