# Multiple Run Ensemble Learning with Low Dimensional Knowledge Graph Embeddings

Chengjin Xu[1], Mojtaba Nayyeri[1], and Sahar Vahdati[2] Jens Lehmann[1,3]

[1] University of Bonn, Bonn, Germany
[2] University of Oxford, Oxford, UK
[3] Fraunhofer IAIS, Dresden, Germany
{Xu, Nayyeri}@cs.uni-bonn.de
sahar.vahdati@cs.ox.ac.uk
jens.lehmann@iais.fraunhofer.de

**Abstract.** Knowledge graphs (KGs) represent facts about a domain in a structured form. Although KGs can be quantitatively huge and consist of millions of triples, their coverage is usually still only a small fraction of the available knowledge. Among the most promising recent approaches for tackling this incompleteness problem is link prediction using knowledge graph embedding models. Various embedding models have been proposed so far, among which, the RotatE model is reported to obtain state-of-the-art performance in such link prediction tasks. However, RotatE mainly outperforms other models when using a high embedding dimension (e.g. 1000). In this paper, we simulate such scenarios by studying the performance of different models using multiple low dimensions in different repetition rounds of the same model. For example, our studies show better results when instead of training a model one time with a high dimension of 1200, we repeat the training of the model 6 times in parallel with dimension of 200 and then combine the 6 models, This can improve results while maintaining the overall number of adjustable parameters is the same. In order to justify our findings, we perform experiments on various models including TransE, DistMult, RotatE and ComplEx. Experimental results on standard benchmark dataset show that multiple low-dimensional models outperform a single high dimensional model while the overall number parameters is same.

**Keywords:** Graph Embedding · Ensemble Learning · Link Prediction.

## 1 Introduction

The advent of knowledge graph (KG) technology has propelled knowledge representation to the next level and influenced the untimate results of several AI-based applications such as question answering and recommendation systems [14,9]. Several KGs such as WordNet [8], FreeBase [2], and YAGO [10], have been published with different utilization purposes. In recent years, KG technology combined with

the recent advances of hardware technologies such as GPU processing. Therefore, a new horizon for using machine learning approaches on structured data at scale has been opened up for leading science and industry.

Despite the advantages of KGs in down stream tasks, one of the main challenges of existing KGs is their incompleteness [11]. KG completion using link prediction approaches aims at addressing this problem. Among various link prediction approaches, KG embedding (KGE) has gained significant attention recently. A KGE model takes a KG in the form of triple facts $(h, r, t)$ where $h, t$ are entities (nodes) and $r$ is a relation (link) between the entities. A $d$ dimensional vector is then assigned to each element of a triple $(\mathbf{h}, \mathbf{r}, \mathbf{t})$ in a KG and adjusts the vectors by optimizing a loss function. The likelihood of a triple is then measured by using a score function over the embedding vectors $(\mathbf{h}, \mathbf{r}, \mathbf{t})$.

The score functions of models play an important role in the performance of the KGEs. After early proposals of novel KGE models, the research field has continued by designing and publishing new models with a focus on score functions. Among those, TransE is one of the primary models which computes the score of a triple by measuring the distance between the tail vector and the relation-specific translated head (i.e. $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$). Several variants of TransE such as TransR [7], TransH [15], and TransD [4] have been proposed later to address the limitations of the original model such as the problem of not being able to encode one to N, symmetric and reflexive relations.

One of the recent state-of-the-art models is RotatE [12] which utilizes rotation in complex space to compute the score of triples. The rotation is performed from each element of the head vector by using relation specific angle to each element of the tail vector i.e. $\mathbf{h} \circ \mathbf{r} \approx \mathbf{t}$ ($\circ$ is element-wise complex multiplication which induces rotation in complex space). The evaluations reported in the initial work introducing RotatE [12] as well as the ones studied afterwards [1], shows that RotatE obtains state-of-the-art in link prediction tasks by using a relatively high dimension (1000 on freebase datasets). QuatE [17] is another KGE model that outperforms other models by taking the advantage of quaternion space that contains four elements. Similar to RotatE, the high performance of QuatE is achieved by using an embedding dimension of 1000. Due to the quaternion design of ths model, the best performing setting with this high dimension results in 4000 adjustable parameters. A similar fact is observable in the ComplEx model which gets a high accuracy with dimension 1000 [6].

Such observations led us to a systematic evaluation of the state-of-the-art models that shows in most of these works, first a high dimension and, second multiple vectors for each entity/relation due to using either complex (with two elements of real and imaginary) or quaternion (with four elements) space. In other words, all the above-mentioned models use **single** model with a multi-part **high dimensional** embeddings (number of parameters is $1 \times d_h$). In contrast to those models, we use the same model **multiple** ($k$) times in parallel trainings with **low dimension** (number of parameters is $k \times d_l$). In order to have a fair comparison, we enforce ($1 \times d_h = k \times d_l$). The experimental results show that the ensemble of the same model several times trained with low-dimensions, results in

a better performance than training that model once with a high dimension while the overall numbers of adjustable parameters are same.

## 2 Related Work

Here we review the existing KGE models including TransE, RotatE, ComplEx, and DistMult. Each model defines a score function $f(h, r, t)$ which takes the triple embeddings $(\mathbf{h}, \mathbf{r}, \mathbf{t})$ and returns the degree of correctness of the triple.
**TransE** [3] Given a triple $(h, r, t)$, the TransE model computes the score by measuring the distance between relation-specific translated head $(\mathbf{h} + \mathbf{r})$ and the tail $\mathbf{t}$ as $f(h, r, t) = -\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|$ to enforce $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ $(\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{R}^d)$ for each positive triple $(h, r, t)$ in the vector space.
**RotatE** [12] RotatE aims at mapping each element of the head embedding $(\mathbf{h}_i)$ to the corresponding tail embedding $\mathbf{t}_i$ by using relation-specific rotation $\mathbf{r}_i = e^{i\theta}$. The score of each triple $(h, r, t)$ is computed as $f(h, r, t) = -\|\mathbf{h} \circ \mathbf{r} - \mathbf{t}\|$ where $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{C}^d$. This enforces $\mathbf{h} \circ \mathbf{r} \approx \mathbf{t}$ for each positive triple $(h, r, t)$.
**DistMult** [16] captures the interaction between elements that has the same index in $\mathbf{h}$ and $\mathbf{t}$. The formulation of score function is $f_r(h, t) = \mathbf{h}^\top \mathbf{diag}(\mathbf{r})\mathbf{t} = \sum_{i=0}^{d} \mathbf{r}_i \cdot \mathbf{h}_i \cdot \mathbf{t}_i$ The model captures symmetric relation, but not antisymmetric.
**CompEx** [13] was proposed as a more elegant way to solve the shortcoming of DistMult in modeling antisymmetric relation. Its main contribution is to embed KGs in complex space. The score function is defined as $f(h, r, t) = \mathrm{Re}(\langle \mathbf{r}, \mathbf{h}, \bar{\mathbf{t}} \rangle)$ where $\mathbf{r}, \mathbf{h}, \mathbf{t} \in \mathbb{C}^d$. Although effective, ComplEx is not expressive enough to model composition relations [12].

Several surveys have been reviewed the existing KGEs and reported about their performance from different aspects and settings. In [5], a set of KGEs combined evaluations have been done for multiple models and settings. It lifts the models intro one score function and combines them during the training phase, however, we focus on stretching and squeezing the dimensions of the same models which are trained separately and combined for testing.

## 3 Proposed Approach

Recent models such as RotatE, ComplEx, and QuatE obtain state-of-the-art performances in link prediction. They go beyond the real space and embed a KG into a Complex or Quaternion space. Therefore, the embedding vectors contain two (complex) or four (quaternion) parts. Consequently, designing KGE models with specific embeddings containing multiple parts (complex vectors with real and imaginary parts) has been shown to be effective in performance boosting. Based on this, we initiated the methodology of this research as follows:

**Hypothesis 1.** *Multiple combined slices of a KGE model in low-dimensions perform better than a single version of that model in high dimensions.*

Our aim is to provide evidences in order to evaluate the hypothesis.

In contrast to the mentioned model which uses a **single** model with **high dimension**, in this part we propose a new approach which combines **multiple** models of the same type in which each model contains **low-dimensional** embeddings. Therefore, the overall number of adjustable parameters remains the same compared to the approaches using a single model with high dimensions. In order to formulate this scenario, let us have a model $\mathcal{M}$ (e.g. RotatE) with embedding dimension $d_l$. We follow the steps below in our approach:

(a) We first generate $k$ times copies of an underlying model $\mathcal{M}$. The $j$th slice of the model is denoted by $\mathcal{M}_j, j = 1, \ldots, k$, and the corresponding $d_l$ dimensional embeddings of $(h, r, t)$ are denoted by $(\mathbf{h}^j, \mathbf{r}^j, \mathbf{t}^j)$. The vectors are randomly initialized in the beginning of learning process.
(b) We then train each of the models $\mathcal{M}_j, j = 1, \ldots, k$ separately.
(c) Finally, the testing is performed by using the following score function

$$f(h, r, t) = \sum_{j=1}^{k} f_{\mathcal{M}_j}(\mathbf{h}_j, \mathbf{r}_j, \mathbf{t}_j), \tag{1}$$

where $f_{\mathbf{M}_j}(\mathbf{h}_j, \mathbf{r}_j, \mathbf{t}_j)$ is the score of a triple $(h, r, t)$ computed by the $j$th copy of the model.

Note that all models are trained on a same KG and the only difference between the models is the initialization of the embedding vectors. To have a fair comparison with original models, we keep the overall number of adjustable parameters (embeddings) equal when comparing with the original single model, i.e. $d_h = k \times d_l$. We will later show in the evaluation part that such a simple approach improves the performance of KGEs without additional cost.

## 4 Experiments

**Dataset** We use FB15k and FB15k-237 for evaluation. FB15k contains 483,142 triples in training set, 50,000 and 59,071 in validation and test set respectively. FB15k-237 contains 272,115, 17,535 and 20,466 triples in the training, validation and testing set respectively.
**Evaluation Metric** We use Mean Reciprocal Rank (MRR) and Hist@n (n=1,3,10) for evaluation. The detail process of computing these metrics can be found in [3].
**Experimental Setup** we evaluate our proposed approach by training TransE, RotatE, DistMult and ComplEx in both single model with high dimension and multiple models with low dimension. For single models with high dimensions, we set embedding dimension to $d_h = \{2, 4, 6, 8, 10, 12\} \times 100$. For models sliced in multiple versions with low dimensions, we set the number of models to $k = 2, \ldots, 6$ and dimension $d_l = 200$. Note that each single mode e.g. RotatE with dimension $d_h$ is compared with multiple models e.g. CRotatE (combination of $k = 6$ RotatE models) with dimension $d_l$. Therefore the number of parameters in both models are the same i.e. $k \times d_l = 1 \times d_h$ e.g. $6 \times 200 = 1 \times 1200$. The implementation has been done by using Pytorch on GPU servers. We use RotatE loss for the models in the table 1. We use uniform negative sampling with only one negative sample.

Table 1: Link prediction results on FB15K and FB15K-237.

| Model | FB15K | | | | FB15K-237 | | | |
|---|---|---|---|---|---|---|---|---|
| | MRR | Hits@1 | Hits@3 | Hits@10 | MRR | Hits@1 | Hits@3 | Hits@10 |
| TransE (d=200) | 0.698 | 0.597 | 0.777 | 0.859 | 0.278 | 0.187 | 0.305 | 0.465 |
| TransE (d=400) | 0.718 | **0.623** | 0.790 | 0.866 | 0.285 | **0.192** | **0.315** | **0.476** |
| TransE (d=600) | **0.718** | 0.623 | **0.791** | **0.867** | **0.285** | 0.191 | 0.315 | 0.475 |
| TransE (d=800) | 0.716 | 0.620 | 0.790 | 0.866 | 0.283 | 0.191 | 0.310 | 0.473 |
| TransE (d=1000) | 0.712 | 0.616 | 0.786 | 0.864 | 0.280 | 0.188 | 0.308 | 0.470 |
| TransE (d=1200) | 0.704 | 0.604 | 0.781 | 0.862 | 0.277 | 0.186 | 0.303 | 0.464 |
| CTransE (d=2*200) | 0.720 | 0.624 | 0.794 | 0.870 | 0.288 | 0.193 | 0.318 | 0.479 |
| CTransE (d=3*200) | 0.726 | 0.632 | 0.799 | 0.873 | 0.292 | 0.198 | 0.324 | 0.485 |
| CTransE (d=4*200) | 0.729 | 0.635 | 0.800 | 0.874 | 0.295 | 0.200 | 0.324 | 0.490 |
| CTransE (d=5*200) | 0.731 | 0.639 | 0.802 | 0.875 | 0.297 | 0.202 | 0.327 | 0.490 |
| CTransE (d=6*200) | **0.732** | **0.640** | **0.802** | **0.876** | **0.298** | **0.202** | **0.329** | **0.491** |
| DitMult (d=200) | 0.653 | 0.540 | 0.744 | 0.827 | 0.222 | **0.148** | 0.241 | 0.372 |
| DitMult (d=400) | 0.675 | 0.564 | 0.764 | 0.839 | 0.227 | 0.148 | 0.248 | 0.388 |
| DitMult (d=600) | 0.682 | 0.568 | 0.774 | 0.852 | 0.228 | 0.147 | 0.248 | 0.392 |
| DitMult (d=800) | 0.685 | 0.570 | 0.777 | 0.854 | 0.227 | 0.144 | 0.250 | 0.397 |
| DitMult (d=1000) | **0.690** | **0.577** | **0.781** | 0.854 | **0.228** | 0.143 | **0.251** | 0.398 |
| DitMult (d=1200) | 0.688 | 0.573 | 0.781 | **0.855** | 0.227 | 0.142 | 0.249 | **0.399** |
| CDitMult (d=2*200) | 0.693 | 0.576 | 0.787 | 0.867 | 0.229 | 0.153 | 0.250 | 0.379 |
| CDitMult (d=3*200) | 0.703 | 0.587 | 0.799 | 0.874 | 0.232 | 0.156 | 0.255 | 0.384 |
| CDitMult (d=4*200) | 0.709 | 0.593 | 0.806 | 0.880 | 0.237 | **0.161** | 0.259 | 0.386 |
| CDitMult (d=5*200) | 0.716 | 0.601 | 0.810 | 0.882 | 0.237 | 0.160 | **0.260** | 0.389 |
| CDitMult (d=6*200) | **0.718** | **0.603** | **0.815** | **0.883** | **0.237** | 0.160 | 0.260 | **0.390** |
| ComplEx (d=200) | 0.635 | 0.510 | 0.734 | 0.835 | **0.229** | **0.153** | 0.250 | 0.381 |
| ComplEx (d=400) | 0.682 | 0.568 | 0.773 | 0.848 | 0.230 | 0.150 | 0.252 | 0.393 |
| ComplEx (d=600) | 0.687 | 0.572 | 0.780 | 0.860 | 0.229 | 0.145 | 0.252 | 0.398 |
| ComplEx (d=800) | **0.698** | **0.585** | 0.789 | 0.864 | 0.229 | 0.144 | 0.252 | **0.400** |
| ComplEx (d=1000) | 0.697 | 0.582 | **0.791** | **0.864** | 0.228 | 0.141 | **0.254** | 0.400 |
| ComplEx (d=1200) | 0.696 | 0.580 | 0.791 | 0.862 | 0.226 | 0.139 | 0.249 | 0.400 |
| CComplEx (d=2*200) | 0.684 | 0.562 | 0.782 | 0.869 | 0.235 | 0.158 | 0.255 | 0.391 |
| CComplEx (d=3*200) | 0.697 | 0.575 | 0.798 | 0.878 | 0.237 | 0.159 | 0.259 | 0.393 |
| CComplEx (d=4*200) | 0.705 | 0.583 | 0.807 | 0.882 | 0.239 | **0**.161 | 0.261 | 0.396 |
| CComplEx (d=5*200) | **0.710** | **0.590** | 0.809 | 0.884 | 0.240 | 0.162 | 0.264 | 0.395 |
| CComplEx (d=6*200) | 0.710 | 0.590 | **0.810** | **0.886** | **0.240** | **0.162** | **0.264** | **0.398** |
| RotatE (d=200) | 0.680 | 0.563 | 0.773 | 0.859 | 0.282 | 0.190 | 0.309 | 0.469 |
| RotatE (d=400) | 0.719 | 0.617 | 0.801 | 0.871 | **0.295** | **0.203** | **0.324** | 0.482 |
| RotatE (d=600) | 0.729 | 0.631 | 0.807 | **0.873** | 0.293 | 0.200 | 0.323 | **0.485** |
| RotatE (d=800) | **0.735** | **0.639** | **0.810** | **0.873** | 0.294 | 0.200 | 0.323 | 0.482 |
| RotatE (d=1000) | 0.730 | 0.634 | 0.805 | 0.870 | 0.292 | 0.199 | 0.321 | 0.481 |
| RotatE (d=1200) | 0.727 | 0.630 | 0.802 | 0.868 | 0.290 | 0.197 | 0.319 | 0.478 |
| CRotatE (d=2*200) | 0.726 | 0.622 | 0.809 | 0.879 | 0.297 | 0.205 | 0.325 | 0.486 |
| CRotatE (d=3*200) | 0.739 | 0.639 | 0.821 | 0.885 | 0.301 | 0.208 | 0.331 | 0.492 |
| CRotatE (d=4*200) | 0.744 | 0.644 | 0.826 | 0.888 | 0.306 | **0.213** | 0.334 | 0.493 |
| CRotatE (d=5*200) | 0.748 | 0.651 | 0.829 | 0.890 | 0.306 | **0.213** | 0.337 | **0.496** |
| CRotatE (d=6*200) | **0.753** | **0.656** | **0.832** | **0.891** | **0.307** | **0.213** | **0.338** | **0.496** |

Table 2: Link prediction results on FB15K with one to N scoring and N3 regularization.

| Model | FB15K | | | |
|---|---|---|---|---|
| | MRR | Hits@1 | Hits@3 | Hits@10 |
| DitMult (d=200) | 0.816 | 0.776 | 0.841 | 0.889 |
| DitMult (d=400) | 0.831 | 0.792 | 0.854 | 0.898 |
| DitMult (d=600) | 0.835 | 0.799 | 0.858 | 0.901 |
| DitMult (d=800) | 0.837 | **0.800** | 0.860 | 0.903 |
| DitMult (d=1000) | **0.839** | 0.799 | **0.866** | **0.909** |
| DitMult (d=1200) | 0.836 | 0.796 | 0.865 | 0.909 |
| CDitMult (d=2*200) | 0.836 | 0.800 | 0.859 | 0.901 |
| CDitMult (d=3*200) | 0.842 | 0.806 | 0.865 | 0.905 |
| CDitMult (d=4*200) | 0.844 | 0.809 | 0.867 | 0.908 |
| CDitMult (d=5*200) | 0.846 | 0.812 | 0.868 | 0.908 |
| CDitMult (d=6*200) | **0.847** | **0.813** | **0.869** | **0.909** |
| ComplEx (d=200) | 0.826 | 0.790 | 0.849 | 0.890 |
| ComplEx (d=400) | 0.840 | 0.806 | 0.860 | 0.898 |
| ComplEx (d=600) | 0.840 | 0.805 | 0.863 | 0.902 |
| ComplEx (d=800) | 0.842 | 0.802 | 0.870 | 0.908 |
| ComplEx (d=1000) | 0.842 | **0.803** | 0.869 | 0.909 |
| ComplEx (d=1200) | **0.843** | 0.802 | **0.871** | **0.910** |
| CComplEx (d=2*200) | 0.845 | 0.814 | 0.864 | 0.902 |
| CComplEx (d=3*200) | 0.851 | 0.820 | 0.870 | 0.906 |
| CComplEx (d=4*200) | 0.856 | 0.825 | 0.873 | 0.909 |
| CComplEx (d=5*200) | 0.858 | 0.827 | 0.876 | 0.911 |
| CComplEx (d=6*200) | **0.859** | **0.829** | **0.877** | **0.911** |
| QuatE (d=1000) | 0.833 | 0.800 | 0.859 | 0.900 |

**Results** Results are shown in Table 1 and Table 2. Table 1 presents the results on FB15k and FB15k-237. As can be seen, multiple models with low dimension (started with "C" such as CTransE) outperforms single models with high dimensions. For example, on FB15K, the single ComplEx model with dimension 800 obtains 0.698, 0.585, 0.789 and 0.864 respectively on MRR, Hist@1,3,10 respectively. Whereas CComplEx with $k = 4, d_l = 200$ obtains 0.705, 0.583, 0.807, 0.882. Therefore, in all metrics except Hits@1 CComplEx outperforms ComplEx while both of the models use same number of parameters ($4 \times 200 = 1 \times 800$).

## 5 Conclusion

In this paper, we compare performance of a single model using high dimension with multiple combined models using low dimension (for each model). Our experimental evaluation on FB15k and FB15k-237 show that instead of using a single model with $d_h$ dimension, using $k$ models with $d_l = d_h/k$ dimension results in a higher accuracy.

**Acknowledgement** This work is supported by the EC Horizon 2020 grant LAMBDA (GA no. 809965), the CLEOPATRA project (GA no. 812997), and the German national funded BmBF project MLwin.

## References


1. F. Akrami, M. S. Saeef, Q. Zhang, W. Hu, and C. Li. Realistic re-evaluation of knowledge graph completion methods: An experimental study. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1995–2010, 2020.
2. K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM, 2008.
3. A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013.
4. G. Ji, S. He, L. Xu, K. Liu, and J. Zhao. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*, pages 687–696, 2015.
5. D. Krompaß and V. Tresp. Ensemble solutions for link-prediction in knowledge graphs. In *Proceedings of the 2nd Workshop on Linked Data for Knowledge Discovery, Porto, Portugal*, pages 1–10, 2015.
6. T. Lacroix, N. Usunier, and G. Obozinski. Canonical tensor decomposition for knowledge base completion. *arXiv preprint arXiv:1806.07297*, 2018.
7. Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.
8. G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.
9. M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2015.
10. M. Nickel, V. Tresp, and H.-P. Kriegel. Factorizing yago: scalable machine learning for linked data. In *Proceedings of the 21st international conference on World Wide Web*, pages 271–280. ACM, 2012.
11. H. Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8(3):489–508, 2017.
12. Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*, 2019.
13. T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080, 2016.
14. Q. Wang, Z. Mao, B. Wang, and L. Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.
15. Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge graph embedding by translating on hyperplanes. In *Aaai*, volume 14, pages 1112–1119, 2014.
16. B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.
17. S. Zhang, Y. Tay, L. Yao, and Q. Liu. Quaternion knowledge graph embeddings. In *Advances in Neural Information Processing Systems*, pages 2735–2745, 2019.