RotateCT: Knowledge Graph Embedding by Rotation and Coordinate Transformation in Complex Space

Yao Dong\(^1,2\), Lei Wang\(^1\), Ji Xiang\(^1\), Xiaobo Guo\(^1\), Yuqiang Xie\(^1,2\)  
\(^1\)Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China  
\(^2\)School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China  
{dongyao,wanglei,xiangji,guoxiaobo,xieyuqiang}@iie.ac.cn

Abstract

Knowledge graph embedding, which aims to learn representations of entities and relations in knowledge graphs, finds applications in various downstream tasks. The key to success of knowledge graph embedding models are the ability to model relation patterns including symmetry/antisymmetry, inversion, commutative composition and non-commutative composition. Although existing methods fail in modeling the non-commutative composition patterns, several approaches support this pattern by modeling beyond Euclidean space and complex space. Nevertheless, expanding to complicated spaces such as quaternion can easily lead to a substantial increase in the amount of parameters, which greatly reduces the computational efficiency. In this paper, we propose a new knowledge graph embedding method called RotateCT, which first transforms the coordinates of each entity, and then represents each relation as a rotation from head entity to tail entity in complex space. By design, RotateCT can infer the non-commutative composition patterns and improve the computational efficiency. Experiments on multiple datasets empirically show that RotateCT outperforms most state-of-the-art methods on link prediction and path query answering.

1 Introduction

Knowledge graphs (KGs) contain structured facts of the real world. Real-world large-scale KGs such as WordNet (Miller, 1995), YAGO (Suchanek et al., 2007), Freebase (Bollacker et al., 2008) and Nell (Mitchell et al., 2018) have been applied to strengthen the performance of several downstream tasks including recommender systems (Zhang et al., 2016), question answering (Hao et al., 2017), conversation generation (Zhou et al., 2018), relation extraction (Vashishth et al., 2018) and machine translation (Zhao et al., 2020). Generally, KGs have the following features: large-scale and incomplete. Therefore, to exploit the semantic information in KGs, predicting missing links based on the existing facts has gained growing interest in recent years. This task can be divided into two categories according to the length of the path: link prediction and path query answering. Link prediction focuses on single-hop reasoning (e.g., answering the path \( s \rightarrow r \rightarrow ? \), where \( r \) is a relation), while path query answering (PQA) focuses on multi-hop reasoning (e.g., answering the path query \( s \rightarrow \text{path} \rightarrow ? \), where \( \text{path} \) contains multiple relations).

A popular approach for link prediction and PQA is knowledge graph embedding (KGE), which encodes each element in KG into a continuous low-dimensional vector space. The performance of KGE methods greatly relies on the ability of modeling and inferring relation patterns including symmetry/antisymmetry, inversion and composition. In particular, the composition patterns can be further divided into the commutative composition patterns and non-commutative composition patterns. For example, "spouse" is a symmetric relation and "mother" is an antisymmetric relation. Relations such as "has_part" and "part_of" forms an inversion pattern. The meaning of the composition of "mother" and "spouse" depends on the relative order. By contrast, the composition of "mother" and "mother" has a definite meaning, i.e., "grandmother" (Fig. 1 provides an ex-
ample of the non-commutative composition patterns in real world). Existing methods can capture one or more relation patterns. TransE (Bordes et al., 2013) models antisymmetry, inversion and commutative composition patterns by translating relations from head entities to tail entities. RotatE (Sun et al., 2019), which regards relations as rotations from head entities to tail entities in complex space, can model the symmetry/antisymmetry, inversion and commutative composition patterns. However, most approaches fail to model the non-commutative composition patterns, which is essential for learning more meaningful embeddings. Therefore, quaternion-valued methods such as QuatE (Zhang et al., 2019), Rotate3D (Gao et al., 2020) and DualE (Cao et al., 2021) emerge in sight. By the non-commutativity of quaternion, these methods successfully model the non-commutative composition patterns. Nevertheless, a quaternion has two more dimensions than a complex number, which increases the space cost. Seeking out a balanced solution that can model the non-commutative composition patterns and maintain a relatively low space cost is pressing.

To address this challenge, we revisit complex space and notice an interesting case: the combination of rotation and coordinate transformation in complex plane is non-commutative, which can empower a KGE method the ability of modeling the non-commutative composition patterns. In addition, the complex number has two dimensions less than the quaternion, which reduces the space cost.

In this paper, we propose a novel method called RotateCT for knowledge graph embedding. Our method first translates the origin of complex plane by a relation-specific displacement. Further, each relation is regarded as a rotation about the new origin of complex plane from head entity to tail entity. By combining rotation and coordinate transformation, RotateCT obtains non-commutativity, which enables RotateCT to effectively model the non-commutative composition patterns.

In summary, our contributions are listed as follows: (1) RotateCT provides an elegant way to model the non-commutative composition patterns and improve the parameter efficiency. To the best of our knowledge, this paper is the first to introduce coordinate transformation into the complex plane for modeling non-commutative composition patterns. (2) We provide comprehensive theoretical analyses on the non-commutative property of combining rotation and coordinate transformation, and discuss the inference patterns, parameter efficiency of RotateCT. (3) Experimental results demonstrate that RotateCT outperforms most baseline approaches on link prediction and path query answering.

2 Related Work

In this section, unlike most papers, we roughly divide the existing KGE methods into three categories according to the non-commutativity and discuss their connections to our approach.

2.1 Models without Non-commutativity

TransE (Bordes et al., 2013) is the most representative KGE model, which embeds both entities and relations as vectors in the same embedding space based on the principle $h + r \approx t$, where $h$, $r$, $t$ denote head entity, relation and tail entity, respectively. Several variants (Wang et al., 2014; Lin et al., 2015; Ji et al., 2015; Xiao et al., 2016) are proposed to remedy the limitations of TransE when modeling 1-N, N-1 and N-N relations. In addition, TorusE (Ebisu and Ichise, 2018) models triplets on a torus, which is a Non-Euclidean space. Inspired by Euler’s identity $e^{i\theta} = \cos\theta + i\sin\theta$, RotatE (Sun et al., 2019) regards translations as rotations from head entities to tail entities in complex space. Moreover, TransC (Lv et al., 2018) and BoxE (Abboud et al., 2020) encode elements by explicitly defining the regions such as hyperspheres or boxes.

RESCAL (Nickel et al., 2011) is the first bilinear model that can perform collective learning via matching the latent semantics between entities and relations. DistMult (Yang et al., 2015) and ComplEx (Trouillon et al., 2016) are proposed to solve the overfitting problem of RESCAL. In addition, HolE (Nickel et al., 2016) absorbs the quintessence from DistMult and ComplEx. Recently, approaches such as SimplE (Kazemi and Poole, 2018) and TuckER (Balazević et al., 2019b) turn to different forms of decomposition.

Although some of these methods claim to enable the inference of composition patterns, in practice they cannot infer the non-commutative composition patterns, only support the inference of the commutative composition patterns.

2.2 Models with Non-commutativity

Recently, several approaches explore the usage of more sophisticated spaces to obtain the non-
commutativity, which is important for multi-hop reasoning. Specifically, Rotate3D (Gao et al., 2020) and QuatE (Zhang et al., 2019) model relations as rotations in quaternion space with different score functions. DualE (Cao et al., 2021) makes the first attempt to combine rotation and translation by expanding the embedding space to dual quaternion space. DihEdral (Xu and Li, 2019) limits relation matrices to be block diagonal and represents each block with an element in a dihedral group.

Although such approaches take into account the non-commutative composition patterns, they are parameter inefficient due to the complicated vector spaces such as quaternion space. Our method RotateCT models the non-commutative composition patterns in complex space by introducing the coordinate transformation. Compared to the above three sophisticated spaces, modeling in complex space can significantly improve the parameter efficiency.

2.3 Other Models

Recently, a number of approaches focus on utilizing neural networks. However, the neural networks lack interpretability, and it is difficult to give theoretical analyses from the perspective of inference patterns. Generally, such approaches verify the performance by empirical experiments. ConvE (Dettmers et al., 2018), ConvKB (Nguyen et al., 2018) and InteractE (Vashishth et al., 2020) model the interactions between entities and relations by convolutional neural networks. Additionally, R-GCN (Schlichtkrull et al., 2018) and KBGAT (Nathani et al., 2019) redesign a graph convolutional network and a graph attention network, respectively. Several works tried to exploit more global graph structures like multi-hop paths. Path-RNN (Das et al., 2017) and ROP (Yin et al., 2018) employ RNNs to explicitly model paths. CoKE (Wang et al., 2019) uses a stack of Transformer blocks to model paths.

Compared to the other models, RotateCT is more interpretable, since we can provide comprehensive theoretical analyses of it.

3 Methodology

3.1 RotateCT

Formally, let $E$ denote the set of entities and $R$ denote the set of relations. Then a knowledge graph $G$ is a collection of factual triplets $\{(h, r, t)\}$, where $h, t \in E$ and $r \in R$. Lowercase letters $h, r$ and $t$ denote the head entity, relation and tail entity, respectively; the corresponding boldface letters $h, r$ and $t$ denote the embeddings of them. Note that the $i$-th element of $h$ is $h_i$. Let $k$ denote the dimension of entity and relation embeddings.

In this paper, we propose RotateCT to model non-commutative composition patterns in complex space. Inspired by Euler’s identity $e^{i\theta} = \cos \theta + is\sin \theta$ and coordinate transformation, our model first projects entities to the complex space, i.e., $h, t \in \mathbb{C}^k$. Then we translate the origin of complex planes $O$ to $O'$ by a relation-specific displacement $b$, where $b \in \mathbb{C}^k$. Further, we define each relation as an element-wise rotation about $O'$ from head entity $h$ to tail entity $t$. In other words, given a golden triplet $(h, r, t)$, we expect that:

$$t - b = (h - b) \circ r$$

where $\circ$ is the Hadamard (or element-wise) product. Specifically, we have $t_i - b_i = (h_i - b_i) r_i$ for each element of $h, r, t$ and $b$. Here, we constrain the modulus of each element of $r \in \mathbb{C}^k$, i.e., $r_i \in \mathbb{C}$, to be $|r_i| = 1$. Then $r_i$ is of the form $e^{i\theta}$, which corresponds a counterclockwise rotation by $\theta_i$ radians about the new origin of the complex plane. We define the distance-based score function as follows:

$$d_r(h, t) = \sum_{i=1}^{k} \| (h_i - b_i) r_i - (t_i - b_i) \|$$

Optimization. Following Sun et al. (2019), we use a loss function similar to the negative sampling
loss (Mikolov et al., 2013) for optimizing:

\[
L = - \sum_{i=1}^{m} \frac{1}{m} \log \sigma (d_r(h'_i, t'_i) - \gamma) \\
- \log \sigma (\gamma - d_r(h, t))
\]

where \(\gamma\) is a fixed margin, \(\sigma\) is the sigmoid function, \(m\) is the negative sampling size, and \(d_r(h'_i, t'_i)\) represents the score of \(i\)-th negative triplet. We also use the self-adversarial negative sampling (Sun et al., 2019) for drawing negative samples. Specifically, the negative triplets are generated from the following distribution:

\[
p(h'_j, r, t'_j | (h_i, r, t_i)) = \frac{\exp \alpha f_r(h'_j, t'_j)}{\sum_i \exp \alpha f_r(h'_i, t'_i)}
\]

where \(f_r(h'_i, t'_i) = - d_r(h'_i, t'_i)\), \(\alpha\) is the temperature of sampling, and \((h'_i, r, t'_i)\) denotes the \(i\)-th negative triplet. The modified loss function with self-adversarial negative sampling is as follows:

\[
L = - \sum_{i=1}^{m} p(h'_i, r, t'_i) \log \sigma (d_r(h'_i, t'_i) - \gamma) \\
- \log \sigma (\gamma - d_r(h, t))
\]

In addition, we perform regularization on \(h\) and \(t\) to avoid overfitting. Therefore, the final loss function takes the following form:

\[
L = - \sum_{i=1}^{m} p(h'_i, r, t'_i) \log \sigma (d_r(h'_i, t'_i) - \gamma) \\
- \log \sigma (\gamma - d_r(h, t)) + \lambda (\|h\|^2 + \|t\|^2)
\]

where \(\lambda\) is the regularization rate. We utilize Adam (Kingma and Ba, 2014) as the optimizer.

### Properties of RotateCT.

RotateCT has the non-commutativity and can infer the most common patterns in KG, as stated next.

**Theorem 1 (Non-commutativity)** The combination of the operations (rotation and coordinate transformation) in RotateCT is non-commutative. (See proof in Appendix B)

**Theorem 2 (Inference ability)** RotateCT can infer the symmetry/antisymmetry, inversion and composition patterns. (See proof in Appendix C. We provide illustrations of RotateCT modeling symmetry and non-commutative composition patterns in Appendix D.)

### Connections to RotatE.

Combining rotation and coordinate transformation enables RotateCT to model the non-commutative composition patterns, which RotatE cannot. Fig. 2 provides illustrations of RotatE and RotateCT with only 1-dimensional embedding. RotatE can be viewed as a special case of RotateCT. Specifically, RotateCT will degenerate into RotatE when the displacement \(b = 0\).

### 4 Experiments

We evaluate RotateCT on two common tasks: link prediction (Bordes et al., 2013) and path query answering (Guu et al., 2015). In addition, we give analyses about the computational efficiency between RotatE and Rotate3D.

#### 4.1 Link Prediction

Link prediction aims to predict the missing \(h\) or \(t\) for a triplet \((h, r, t)\), i.e., predicting the head query \(h \rightarrow r \rightarrow t\) or the tail query \(h \rightarrow r \rightarrow ?\). Thus, link prediction is a single-hop reasoning task.

**Datasets.** We evaluate RotateCT on four well-established benchmarks: WN18 (Bordes et al., 2013), FB15k (Bordes et al., 2013), WN18RR (Dettmers et al., 2018) and FB15k-237 (Toutanova and Chen, 2015). Please refer to Appendix F for the details of the four benchmarks.

**Evaluation Protocol.** Similar to most previous models, the link prediction performance of RotateCT is reported on three standard evaluation metrics: Mean Rank (MR), Mean Reciprocal Rank (MRR) and Hits@N, where \(N = 1, 3, 10\). MR is the average rank of all correct entities. MRR is the mean reciprocal rank of all correct entities. Hits@N represents the proportion of correct entities whose rank is not larger than \(N\). A lower MR,
Table 1: Link prediction results on WN18 and FB15k.

| Model                      | WN18 | FB15k |
|---------------------------|------|-------|
|                           | MR   | MRR   | Hits@1 | Hits@3 | Hits@10 | MR   | MRR   | Hits@1 | Hits@3 | Hits@10 |
| Models without non-commutativity |      |       |        |        |         |      |       |        |        |         |
| TransE (2013)             | - 0.495 | - | 0.113 | 0.888 | 0.943 | - 0.463 | - | 0.297 | 0.578 | 0.749 |
| DistMult (2015)           | 655 | 0.797 | - | - | 0.946 | 42 | 0.798 | - | - | **0.893** |
| ComplEx (2016)            | - 0.941 | 0.936 | 0.945 | 0.947 | - | - 0.692 | 0.599 | 0.759 | 0.840 |
| SimpIIE (2018)            | - 0.942 | 0.939 | 0.944 | 0.947 | - | - 0.727 | 0.660 | 0.773 | 0.838 |
| TorusE (2018)             | - 0.947 | 0.943 | 0.950 | 0.954 | - | - 0.733 | 0.674 | 0.771 | 0.832 |
| RotatE (2019)             | 309 | 0.949 | 0.944 | 0.952 | 0.959 | 40 | **0.797** | **0.746** | 0.830 | 0.884 |
| Models with non-commutativity |       |       |        |        |         |      |       |        |        |         |
| DihEdral (2019)           | - 0.946 | 0.942 | 0.949 | 0.954 | - | - 0.733 | 0.641 | 0.803 | 0.877 |
| QuatE (2019)              | 338 | 0.949 | 0.941 | 0.954 | 0.960 | 41 | 0.770 | 0.700 | 0.821 | 0.878 |
| Rotate3D (2020)           | 214 | **0.951** | **0.945** | 0.953 | 0.961 | 39 | 0.789 | 0.728 | 0.832 | 0.887 |
| DualE (2021)              | - | **0.951** | **0.945** | **0.956** | 0.961 | - | 0.790 | 0.734 | 0.829 | 0.881 |
| RotateCT (ours)           | 201 | **0.951** | 0.944 | **0.956** | 0.963 | 34 | **0.794** | 0.737 | **0.834** | **0.888** |

a higher MRR and a higher Hits@N indicate the better performance. Filtered results are reported to avoid possibly flawed evaluation.

**Implementation Details.** We use PyTorch to implement our model and test it on a Tesla V100 GPU. Hyperparameters of RotateCT are determined via grid search according to the MRR on the validation set. In general, the embedding dimension $k$ is selected in $\{500, 1000\}$; batch size $n$ is searched in $\{512, 1024\}$; the negative sampling size $m$ is picked from $\{256, 512\}$; the fixed margin $\gamma$ is tuned among $\{6, 9, 12, 24\}$; self-adversarial temperature $\alpha$ is selected from $\{0.5, 1.0\}$. The regularization rate $\lambda$ is adjusted in $\{0, 0.1\}$. Entity embeddings are uniformly initialized, and the phases of relation embeddings are uniformly initialized between $-\pi$ and $\pi$. Both the real and imaginary parts of each dimension in displacement $b$ are initialized to zero. The best hyperparameters settings and are provided in the Appendix E.

**Main Results.** We select competitive baselines from the most recent publications with good results reported. Our baselines are categorized into three groups: models without non-commutativity, models with non-commutativity and other models, which is consistent with the Section 2.

The empirical results on four benchmarks are reported in Table 1 and Table 2. The best results are in bold and second best results are underlined. We can see that RotateCT outperforms all the baselines on WN18 and WN18RR, and achieves extremely competitive performance on FB15k and FB15k-237. On WN18RR and FB15k-237, the main relation patterns are symmetry/antisymmetry and composition, which validates the effectiveness of combining rotation and coordinate transformation for inferring the non-commutative composition patterns. On WN18 and FB15k, the main relation patterns are symmetry/antisymmetry and inversion. Since RotateCT has no obvious superiority over other state-of-the-art baselines in modeling symmetry/antisymmetry and inversion patterns, the performance improvement is not significant on WN18 and FB15k. Although DihEdral, QuatE, Rotate3D and DualE can model the non-commutative composition patterns, the link prediction performance of these models is still inferior to RotateCT, which indicates that parameter efficiency is crucial. Note that RotateCT, as a complex-valued method, outperforms the quaternion-valued method Rotate3D on all datasets across most metrics, which fully demonstrates that the combination of rotation and coordinate transformation can effectively model the non-commutative composition patterns.

### 4.2 Path Query Answering

This task is to answer path queries on KGs (Guu et al., 2015). Given a path query $q$ consisting of a start entity $s$ and a path $p$, the answer of $q$ is the entities that can be reached from $s$ via $p$. A path $p$ is a sequence of relations, i.e., $r_1 \rightarrow \ldots \rightarrow r_j$, where $j$ is the length of $p$. Link prediction can be viewed as a special case of path query answering when the
path length is 1. By contrast, path query answering requires more ability of multi-hop reasoning, in which inferring the composition patterns is crucial.

Datasets. We conduct experiments on two datasets released by Guu et al. (2015), which are extracted from WordNet (Miller, 1995) and Freebase (Bollacker et al., 2008). Both datasets contain triplets and paths. Paths are generated from triplets by random walks. The maximum of path length is 5. Paths used for training are only generated from training triplets, while paths used for test are generated from both training triplets and test triplets. Test paths which appear in training paths are removed. Note that paths of length 1 are not sampled, but created by directly adding triplets. Details of generating paths can be found in (Guu et al., 2015). See details of these two datasets in Appendix G.

Evaluation Protocol. We use the same evaluation protocol as in (Guu et al., 2015). Specifically, for each test path \( p_t = s \rightarrow r_1 \rightarrow \ldots \rightarrow r_j \rightarrow o \), the corresponding query \( q = s \rightarrow r_1 \rightarrow \ldots \rightarrow r_j \rightarrow ? \). Details of evaluation protocol for PQA can be found in Appendix H. We report the mean quantile (MQ) and Hits@10. MQ is the average quantile of all test paths. Hits@10 is the percentage of target answers whose rank is not larger than 10. An excellent model should achieve a higher MQ and a higher Hits@10.

Implementation Details. We train RotateCT with all paths in the training set, which is denoted as “Comp” in (Guu et al., 2015). Note that triplets are the paths of length 1. To make the results directly comparable, we follow Gao et al. (2020) to train RotateCT on paths of length 1 to 5 in turn, i.e., we train our model on paths of length \( i \) until convergence before training on paths of length \( i + 1 \). Hyperparameters are determined via grid search according to the MQ on the validation set. In general, the embedding dimension \( k \) is selected in \{500, 1000\}; batch size \( n \) is searched in \{512, 1024\}; the negative sampling size \( m \) is picked from \{256, 512\}; the fixed margin \( \gamma \) is tuned among \{6, 9, 12, 24\}; self-adversarial temperature \( \alpha \) is selected from \{1.0, 2.0, 3.0\}. The regularization rate \( \lambda \) is adjusted in \{0, 0.1\}. The initialization methods and experimental environment are the same as in link

### Table 2: Link prediction results on WN18RR and FB15k-237.

| Model           | WN18RR               | FB15k-237             |
|-----------------|----------------------|-----------------------|
| MR MRR Hits@1   | Hits@3 | Hits@10          | MR MRR Hits@1 | Hits@3 | Hits@10 |
| TransE (2013)   | 33.84 | 0.226 | - | - | 0.501 | 35.7 | 0.294 | - | - | 0.465 |
| DistMult (2015) | 51.00 | 0.430 | 0.390 | 0.440 | 0.490 | 25.4 | 0.241 | 0.155 | 0.263 | 0.419 |
| ComplEx (2016)  | 52.61 | 0.440 | 0.410 | 0.460 | 0.510 | 33.9 | 0.247 | 0.158 | 0.275 | 0.428 |
| MuRP (2019a)    | -    | 0.475 | 0.436 | 0.487 | 0.554 | -   | 0.336 | 0.245 | 0.370 | 0.521 |
| RotatE (2019)   | 33.40 | 0.476 | 0.428 | 0.492 | 0.571 | 17.7 | 0.338 | 0.241 | 0.375 | 0.533 |
| BoxE (2020)     | 32.07 | 0.451 | 0.400 | 0.472 | 0.541 | 16.3 | 0.337 | 0.238 | 0.374 | 0.538 |
| Models without non-commutativity |          |          |          |          |          |          |          |          |          |
| Models with non-commutativity |          |          |          |          |          |          |          |          |          |
| DihEdral (2019) | -    | 0.486 | 0.442 | 0.505 | 0.557 | -   | 0.320 | 0.230 | 0.353 | 0.502 |
| QuatE (2019)    | 34.72 | 0.481 | 0.436 | 0.500 | 0.564 | 17.6 | 0.311 | 0.221 | 0.342 | 0.495 |
| Rotate3D (2020) | 33.28 | 0.489 | 0.442 | 0.505 | 0.579 | 16.5 | 0.347 | 0.250 | 0.385 | 0.543 |
| DualE (2021)    | -    | 0.482 | 0.440 | 0.500 | 0.561 | -   | 0.330 | 0.237 | 0.363 | 0.518 |
| RotateCT (ours) | 32.85 | 0.492 | 0.448 | 0.507 | 0.579 | 17.1 | 0.347 | 0.251 | 0.382 | 0.537 |
| Other models    |          |          |          |          |          |          |          |          |          |
| ConvE (2018)    | 41.87 | 0.430 | 0.400 | 0.440 | 0.520 | 24.4 | 0.325 | 0.237 | 0.356 | 0.501 |
| R-GCN+ (2018)   | -     | -     | -     | -     | -     | -   | -     | -     | -     | -     |
| NKGE (2018)     | 41.70 | 0.450 | 0.421 | 0.465 | 0.526 | 23.7 | 0.330 | 0.241 | 0.365 | 0.510 |
| Models with non-commutativity |          |          |          |          |          |          |          |          |          |
| DihEdral (2019) | -    | 0.486 | 0.442 | 0.505 | 0.557 | -   | 0.320 | 0.230 | 0.353 | 0.502 |
| QuatE (2019)    | 34.72 | 0.481 | 0.436 | 0.500 | 0.564 | 17.6 | 0.311 | 0.221 | 0.342 | 0.495 |
| Rotate3D (2020) | 33.28 | 0.489 | 0.442 | 0.505 | 0.579 | 16.5 | 0.347 | 0.250 | 0.385 | 0.543 |
| DualE (2021)    | -    | 0.482 | 0.440 | 0.500 | 0.561 | -   | 0.330 | 0.237 | 0.363 | 0.518 |
| RotateCT (ours) | 32.85 | 0.492 | 0.448 | 0.507 | 0.579 | 17.1 | 0.347 | 0.251 | 0.382 | 0.537 |

Table 3: Path query answering results on WordNet and Freebase. | Implicitly model paths | Explicitly model paths |
|------------------------|------------------------|
| MQ Hits@10             | MQ Hits@10             |
| Bilinear\(\uparrow\)   | 0.894 | 0.543 | 0.835 | 0.421 |
| DistMult\(\uparrow\)   | 0.904 | 0.311 | 0.848 | 0.386 |
| TransE\(\uparrow\)     | 0.933 | 0.435 | 0.880 | 0.505 |
| RotatE\(\uparrow\)     | 0.947 | 0.653 | 0.901 | 0.601 |
| Rotate3D\(\uparrow\)    | 0.949 | 0.671 | 0.905 | 0.621 |
| RotateCT\(\uparrow\)    | 0.949 | 0.673 | 0.907 | 0.630 |

Table 3: Path query answering results on WordNet and Freebase. \[\uparrow\]: Models do not use extra structures to model paths; \[\uparrow\]: Model uses RNN to model paths; \[\uparrow\]: Model uses Transformer to model paths. Best results are in **bold** and second best results are underlined.
 prediction. The best hyperparameters settings of RotateCT are provided in the Appendix E.

**Main Results.** Baselines are divided into two categories according to whether they use extra structures to model paths. The first category including Bilinear (Nickel et al., 2011), DistMult (Yang et al., 2015), TransE (Bordes et al., 2013), RotatE (Sun et al., 2019) and Rotate3D (Gao et al., 2020) has no extra structure to model paths; another category including ROP (Yin et al., 2018) and CoKE (Wang et al., 2019) uses RNN or Transformer to model paths.

Table 3 shows experimental results on the two datasets. Compared with methods that implicitly model paths, RotateCT achieves better performance on both WordNet and Freebase, which demonstrates the ability of RotateCT to model the composition patterns. Overall, RotateCT significantly outperforms Bilinear, DistMult, TransE and RotatE. The reason why RotateCT slightly surpasses Rotate3D is that both RotateCT and Rotate3D have the ability of inferring the non-commutative composition patterns. Compared with methods that explicitly model paths, RotateCT outperforms ROP and still obtains results comparable to CoKE (Wang et al., 2019) uses RNN or Transformer to model paths.

Table 5: Training time per 1000 steps of Rotate3D and RotateCT on all datasets.

| Space   | Rotate3D | RotateCT |
|---------|----------|----------|
| FB15k   | 147s     | 65s      |
| FB15k-237| 153s    | 65s      |
| WN18    | 85s      | 57s      |
| Freebase| 103s     | 68s      |

Table 4: Number of free parameters.

| Model    | Rotate3D | RotateCT |
|----------|----------|----------|
| Space    | $\mathbb{H}^k$ | $\mathbb{C}^k$ |
| Dimension| 1000     | 1000     |
| FB15k    | 48.89M   | 33.94M (30.6%) |
| FB15k-237| 44.33M   | 32.84M (31.8%) |
| WN18     | 122.89M  | 81.94M (33.3%) |
| WN18RR   | 122.86M  | 81.92M (33.3%) |
| Freebase | 225.17M  | 150.13M (33.3%) |
| WordNet  | 115.69M  | 77.14M (33.3%) |

4.3 Analyses

**Space Efficiency.** To compare the space cost of Rotate3D and RotateCT, we calculate the number of free parameters over different datasets. Results are shown in Table 4, from which we observe that RotateCT with same dimension reduces up to 30% parameters on all datasets. Intuitively, Rotate3D utilizes quaternions to model relations as rotations in 3D space. However, a quaternion has two more dimensions than a complex number, which sharply increases the space cost of Rotate3D.

**Time Efficiency.** To further compare the training cost of Rotate3D and RotateCT, we report the training time per 1000 steps on all datasets with embedding dimension $k = 1000$. From Table 5, we can find that RotateCT takes much less training time than Rotate3D on all datasets, which confirms the time efficiency superiority of RotateCT. Experiments of training time are performed on an Intel(R) Xeon(R) Silver 4114 CPU at 2.20GHz and a single NVIDIA Tesla V100 32GB GPU.

5 Case Studies

To verify that RotateCT can effectively model all the three types of relation patterns, we provide case studies via histograms on symmetry/antisymmetry and inversion patterns, and some intuitive examples on composition patterns.

5.1 Symmetry/Antisymmetry

**Corollary 1** Based on Theorem 2, if $r$ is a symmetric relation, then $\theta_i = 0 \lor \pm \pi$; if $r$ is an antisymmetric relation, then $\theta_i \neq 0, \pm \pi$. (See proof in Appendix I)

According to the Corollary 1, the phase of each element in the embeddings of a symmetric relation $r$ should be $0$ or $\pm \pi$. Otherwise, the phases should not be $0$ and $\pm \pi$. We investigate the phases of elements in relation embeddings from a RotateCT trained on WN18 and a RotateCT trained on FB15k-237 with their best hyperparameters in link prediction. The two models are trained with the setting $k = 1000$. Results are shown in Fig. 3. Specifically, Fig. 3(a)-3(b) give the histograms of the two symmetric relations in WN18: derivationally_related_form and verb_group, in which most phases of the two relations are either
we find that RotateCT predicts the query Maria → parents → spouse correctly, which indicates that RotateCT can effectively model the non-commutative composition patterns. However, we observe that RotatE gives unsatisfactory answers in predicting the query Maria → parents → ? and the query Maria → spouse → parents → ?, which verifies that RotatE lacks the ability of inferring the non-commutative composition patterns.

### 5.2 Inversion

**Corollary 2** Based on Theorem 2, if \( r_1 \) is the inverse of \( r_2 \), then \( \theta_{r_1} + \theta_{r_2} = 0 \) or \( \pm 2\pi \). (See proof in Appendix J)

According to the Corollary 2, if \( r_1 \) is inverse to \( r_2 \), the additive embedding phases, i.e., \( \theta_{r_1} + \theta_{r_2} \), should be 0 or \( \pm 2\pi \). Same RotateCT model trained on WN18 in Section 5.1 is used for investigating. Fig. 3(e)-3(f) show the element-wise addition of the embedding phases of two pairs of inverse relations: has_part ∘ part_of (e) and hyponym ◦ hypernym (f), where \( \circ \) is the Hadmard (or element-wise) product.

### 5.3 Composition

To illustrate the superiority of RotateCT in modeling the composition patterns, we use a RotateCT and a RotatE, both trained with their best hyper-parameters on the Freebase dataset in path query answering. Fig. 4 shows a subgraph extracted from the Freebase dataset. This subgraph includes two non-commutative composition patterns: "parents’ spouse" and "spouse’s parents", and one commutative composition pattern: "parents’ parents".

**Non-commutative Composition.** From Fig. 5, we find that RotateCT predicts the query Maria → parents → spouse correctly, and the query Maria → spouse → parents ? incorrectly, which indicates that RotateCT can effectively model the non-commutative composition patterns. However, we observe that RotatE gives unsatisfactory answers in predicting the query Maria → parents → ? and the query Maria → spouse → parents → ?, which verifies that RotatE lacks the ability of inferring the non-commutative composition patterns.
on embeddings’ learning. If RotatE encodes "parents’ spouse" and "spouse’s parents" in the same way, some semantic information will be lost. This flaw is harmful for RotatE effectively learning the semantics of parents and spouse, which reduces RotatE’s performance on modeling the commutative composition pattern "parents’ parents".

6 Conclusion

In this paper, we propose a novel knowledge graph embedding method called RotateCT to model the non-commutative composition patterns while improving the parameter efficiency against quaternion-valued methods. RotateCT transforms the coordinates of each entity by translating the origin of coordinates. Further, relations are represented as rotations from head entities to tail entities in complex space. Combining rotation and coordinate transformation empowers RotateCT to model not only commutative composition patterns but also non-commutative composition patterns. As a complex-valued method, the space cost of RotateCT is lower than quaternion-valued methods. Experimental results on link prediction and path query answering show that RotateCT achieves significant performance and demonstrate the superiority of RotateCT for inferring composition patterns. The parameter efficiency analysis proves that RotateCT can reduce the space cost.

In future work, we will explore different ways to model the non-commutative composition patterns and plan to study the combination of quaternion and coordinate transformation.

Acknowledgements

This work was supported by the National Key Research and Development Program of China.

References

Ralph Abboud, Ismail Ceylan, Thomas Lukasiewicz, and Tommaso Salvadori. 2020. BoxE: A box embedding model for knowledge base completion. In NeurIPS, volume 33.

Ivana Balazevic, Carl Allen, and Timothy Hospedales. 2019a. Multi-relational poincaré graph embeddings. NeurIPS, 32:4463–4473.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In SIGMOD, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In NeurIPS, volume 26, pages 2787–2795.

Zongsheng Cao, Qianqian Xu, Zhiyong Yang, Xiaochun Cao, and Qingming Huang. 2021. Dual quaternion knowledge graph embeddings. In AAAI, volume 35, pages 6894–6902.

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2017. Chains of reasoning over entities, relations, and text using recurrent neural networks. In EACL, pages 132–141.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In AAAI, volume 32.

Takuma Ebisu and Ryutaro Ichise. 2018. Toruse: Knowledge graph embedding on a lie group. In AAAI, volume 32.

Chang Gao, Chengjie Sun, Lili Shan, Lei Lin, and Mingjiang Wang. 2020. Rotate3d: Representing relations as rotations in three-dimensional space for knowledge graph embedding. In CIKM, pages 385–394.
Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In EMNLP, pages 318–327.

Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In ACL, pages 221–231.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In ACL, pages 687–696.

Seyed Mehran Kazemi and David Poole. 2018. Simple embedding for link prediction in knowledge graphs. In NeurIPS, pages 4284–4295.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In AAAI, volume 29.

Xin Lv, Lei Hou, Juanzi Li, and Zhiyuan Liu. 2018. Differentiating concepts and instances for knowledge graph embedding. In EMNLP, pages 1971–1979.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In NeurIPS, pages 3111–3119.

George A Miller. 1995. Wordnet: a lexical database for english. Communications of the ACM, 38(11):39–41.

Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Bishan Yang, Justin Betteridge, Andrew Carlson, Bhavana Dalvi, Matt Gardner, Bryan Kiesiel, et al. 2018. Never-ending learning. Communications of the ACM, 61(5):103–115.

Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. 2019. Learning attention-based embeddings for relation prediction in knowledge graphs. In ACL, pages 4710–4723.

Tu Dinh Nguyen, Dat Quoc Nguyen, Dinh Phung, et al. 2018. A novel embedding model for knowledge base completion based on convolutional neural network. In NAACL, pages 327–333.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In AAAI, volume 30.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In ICML, volume 11, pages 809–816.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In ESWC, pages 593–607. Springer.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In WWW, pages 697–706.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In ICLR.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality, pages 57–66.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In ICML.

Shikhari Vashishth, Risabh Joshi, Sai Suman Prayaga, Chiranjib Bhattacharyya, and Partha Talukdar. 2018. Reside: Improving distantly-supervised neural relation extraction using side information. In EMNLP, pages 1257–1266.

Shikhari Vashishth, Soumya Sanyal, Vikram Nitin, Nilesch Agrawal, and Partha P Talukdar. 2020. Interacte: Improving convolution-based knowledge graph embeddings by increasing feature interactions. In AAAI, pages 3009–3016.

Kai Wang, Yu Liu, Xiujuan Xu, and Dan Lin. 2018. Knowledge graph embedding with entity neighbors and deep memory network. arXiv preprint arXiv:1808.03752.

Quan Wang, Pingping Huang, Haifeng Wang, Songtai Dai, Wenbin Jiang, Jing Liu, Yajuan Lyu, Yong Zhu, and Hua Wu. 2019. Coke: Contextualized knowledge graph embedding. arXiv preprint arXiv:1911.02168.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In AAAI, volume 28.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016. From one point to a manifold: knowledge graph embedding for precise link prediction. In IJCAI, pages 1315–1321.

Canran Xu and Ruijiang Li. 2019. Relation embedding with dihedral group in knowledge graph. In ACL, pages 263–272.

Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases.
Wenpeng Yin, Yadollah Yaghoobzadeh, and Hinrich Schütze. 2018. Recurrent one-hop predictions for reasoning over knowledge graphs. In COLING, pages 2369–2378.

Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In SIGKDD, pages 353–362.

Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu. 2019. Quaternion knowledge graph embeddings. In NeurIPS, pages 2735–2745.

Yang Zhao, Jiajun Zhang, Yu Zhou, and Chengqing Zong. 2020. Knowledge graphs enhanced neural machine translation. In IJCAI, pages 4039–4045.

Hao Zhou, Tom Young, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018. Commonsense knowledge aware conversation generation with graph attention. In IJCAI, pages 4623–4629.
A Definitions of Inference Patterns

We give the formal definitions of inference patterns as follows:

Definition 1 A relation \( r \) is symmetric (antisymmetric) if

\[
\forall x, y \in \mathcal{E}, r(x, y) \Rightarrow r(y, x) \quad (r(x, y) \Rightarrow \neg r(y, x))
\]

A relation with such form is a symmetry (antisymmetry) pattern.

Definition 2 Relation \( r_1 \) is inverse to \( r_2 \) if

\[
\forall x, y \in \mathcal{E}, r_2(x, y) \Rightarrow r_1(y, x)
\]

Relations with such form is an inversion pattern.

Definition 3 Relation \( r_1 \) is composed of relation \( r_2 \) and relation \( r_3 \) if

\[
\forall x, y, z \in \mathcal{E}, r_2(x, y) \land r_3(y, z) \Rightarrow r_1(x, z)
\]

\( r_2 \) and \( r_3 \) are commutative if

\[
\forall x, y \in \mathcal{E}, r_2(x, y) \land r_3(x, y) \Rightarrow r_3 \circ r_2(x, y)
\]

\( r_2 \) and \( r_3 \) are non-commutative if

\[
\forall x, y \in \mathcal{E}, r_2 \circ r_3(x, y) \neq r_3 \circ r_2(x, y)
\]

where \( \circ \) is the composition operator.

Relations with such form is a composition pattern.

B Proof of Theorem 1

Proof 1 Given \( r_1, r_2 \in \mathcal{R} \) and \( x \in \mathcal{E} \), we have

\[
\begin{align*}
  r_{1i} \circ r_{2i}(x_i) &= (x_i - b_{1i})r_{1i} + b_{1i} - b_{2i})r_{2i} \\
  &= x_ir_{1i}r_{2i} - b_{1i}r_{1i}r_{2i} + b_{1i}r_{2i} - b_{2i}r_{2i}, \\
  &\quad \text{(1)} \\
  r_{2i} \circ r_{1i}(x_i) &= (x_i - b_{2i})r_{2i} + b_{2i} - b_{1i})r_{1i} \\
  &= x_ir_{1i}r_{2i} - b_{2i}r_{1i}r_{2i} + b_{2i}r_{1i} - b_{1i}r_{1i}, \\
  &\quad \text{(2)}
\end{align*}
\]

Apparentley, Equation (1) is not equal to (2), which means that the combination of operations in RotateCT is non-commutative. If and only if \( b_{1i} = b_{2i} \) or \( r_{1i}r_{2i} = r_{1i} + r_{2i}, r_{1i} \circ r_{2i}(x_i) = r_{2i} \circ r_{1i}(x_i) \).

C Proof of Theorem 2

Proof 2 For the symmetry/antisymmetry pattern, given \( r \in \mathcal{R} \) and \( x, y \in \mathcal{E} \), if \( r(x, y) \) and \( r(y, x) \) hold, we have

\[
\begin{align*}
  \{ y_i - b_i = (x_i - b_i)r_i \\
  x_i - b_i = (y_i - b_i)r_i \} &\Rightarrow r_i = \pm 1
\end{align*}
\]

Otherwise, if \( r(x, y) \) and \( \neg r(y, x) \) hold, we have

\[
\{ y_i - b_i = (x_i - b_i)r_i \\
  x_i - b_i \neq (y_i - b_i)r_i \} \Rightarrow r_i \neq \pm 1
\]

For the inversion pattern, given \( r_1, r_2 \in \mathcal{R} \) and \( x, y \in \mathcal{E} \), if \( r_1(x, y) \) and \( r_2(x, y) \) hold, we have

\[
\begin{align*}
  \{ x_i - b_{1i} = (y_i - b_{1i})r_{1i} \\
  y_i - b_{2i} = (x_i - b_{2i})r_{2i} \} &\Rightarrow r_{1i}r_{2i} = 1 \text{ or } r_{1i} = r_{2i} = 1
\end{align*}
\]

For the composition pattern, given \( r_1, r_2, r_3 \in \mathcal{R} \) and \( x, y, z \in \mathcal{E} \), if \( r_1(x, z) \), \( r_2(x, y) \) and \( r_3(y, z) \) hold, we have

\[
\begin{align*}
  \{ z_i - b_{1i} = (x_i - b_{1i})r_{1i} \\
  y_i - b_{2i} = (x_i - b_{2i})r_{2i} \\
  z_i - b_{3i} = (x_i - b_{3i})r_{3i} \} &\Rightarrow r_{1i}r_{2i} = r_{3i} \\
  b_{1i}r_{1i}r_{2i} &= b_{1i}r_{2i} - b_{2i}r_{2i} + b_{3i}r_{3i}
\end{align*}
\]

if \( r_2 \) and \( r_3 \) are commutative, then

\[
b_{2i} = b_{3i} \text{ or } r_{2i}r_{3i} = r_{2i} + r_{3i},
\]

if \( r_2 \) and \( r_3 \) are non-commutative, then

\[
\{ b_{2i} \neq b_{3i} \\
  r_{2i}r_{3i} \neq r_{2i} + r_{3i}
\}
\]

In conclusion, RotateCT can infer symmetry/antisymmetry, inversion and composition patterns.

D Illustrations of RotateCT

Fig. 6 shows illustrations of RotateCT modeling symmetry and non-commutative composition patterns.
### Table 6: Hyperparameters setting of RotateCT over different datasets.

| Dataset       | embedding dimension | batch size | self-adversarial temperature | margin | negative sample size | distance | regularization/rate | learning rate |
|---------------|---------------------|------------|-------------------------------|--------|----------------------|----------|--------------------|---------------|
| WN18          | 1000                | 512        | 0.5                           | 12     | 256                  | L1       | L1/0.1             | $1 \times 10^{-4}$ |
| WN18RR        | 1000                | 512        | 0.5                           | 24     | 256                  | L2       | -                  | $2 \times 10^{-4}$ |
| FB15k         | 1024                | 1024       | 1.0                           | 6      | 512                  | L1       | L1/0.1             | $5 \times 10^{-5}$ |
| FB15k-237     | 1024                | 1024       | 1.0                           | 12     | 256                  | L2       | -                  | $2 \times 10^{-4}$ |
| WordNet       | 1000                | 512        | 1.0                           | 6      | 256                  | L2       | L2/0.1             | $5 \times 10^{-5}$ |
| Freebase      | 1000                | 1024       | 2.0                           | 12     | 512                  | L2       | -                  | $2 \times 10^{-5}$ |

Figure 6: Illustrations of RotateCT over two relation patterns. Fig. 6(a) shows how RotateCT models the symmetry pattern, i.e., $r = \pm 1$. In 6(b), blue solid lines represent the path $h \rightarrow r_1 \rightarrow r_2 \rightarrow t$ and red dashed lines represent the path $h \rightarrow r_2 \rightarrow r_1 \rightarrow t'$. $r_1$ and $r_2$ form a non-commutative composition pattern. When the relative order of $r_1$ and $r_2$ changes, RotateCT can get two different entities: $t$ and $t'$.

### E Hyperparameters Settings

We list the best hyperparameters setting of RotateCT on all datasets in Table 6.

### F Details of Link Prediction Datasets

WN18 is a subset of WordNet (Miller, 1995), a database consisting of lexical relations between words. FB15k is extracted from Freebase (Bollacker et al., 2008), a large-scale knowledge graph containing general facts. The main relation patterns in WN18 and FB15k are symmetry/antisymmetry and inversion (Sun et al., 2019). However, both WN18 and FB15k suffer from test leakage through inverse relations (Toutanova and Chen, 2015). To avoid this problem, WN18RR and FB15k-237 remove the inverse relations in WN18 and FB15k, respectively. Therefore, the main relation patterns in WN18RR and FB15k-237 are symmetry/antisymmetry and composition (Sun et al., 2019). Notably, semantic information in WN18RR and FB15k-237 is more difficult to capture on account of removing the inverse relations. The statistics of the benchmarks for link prediction are listed in Table 7.

### G Details of Path Query Answering Datasets

The statistics of path query answering datasets are summarized in Table 8.

### H Evaluation Protocol of PQA

For each test path $p_t = s \rightarrow r_1 \rightarrow \ldots \rightarrow r_j \rightarrow o$, the corresponding query $q$ is $s \rightarrow r_1 \rightarrow \ldots \rightarrow r_j \rightarrow ?$. For each query $q$, the candidate answers are entities that “type-match”, i.e., all tail entities of the final relation $r_j$. The correct answers are entities that can be reached from $s$ by traversing the path $p$; the incorrect answers are obtained by filtering out the correct answers from the candidate answers. The set of candidate answers to a query $q$ denotes as $C(q)$; the set of correct answers to a query $q$ denotes as $P(q)$; the set of incorrect
answers to a query $q$ denotes as $\mathcal{N}(q)$. We give
the formal definition of $\mathcal{C}(q)$, $\mathcal{P}(q)$, and $\mathcal{N}(q)$ as follows:

$$
\mathcal{C}(q) \triangleq \{o \mid \exists e \text{ s.t. } (e, r_j, o) \in \mathcal{G}\}
$$

$$
\mathcal{P}(q) \triangleq \{o \mid \exists e_1, \ldots, e_{j-1} \text{ s.t. } (s, r_1, e_1),
\ldots, (e_{j-1}, r_j, o) \in \mathcal{G}\}
$$

$$
\mathcal{N}(q) \triangleq \mathcal{C}(q) \setminus \mathcal{P}(q)
$$

Here $\mathcal{G}$ includes training triplets and test triplets. For each test path $p_t$, we replace entity $o$ with entities in $\mathcal{C}(q)$ and compute the score of each candidate answer. Then we rank the scores of candidates along with the score of $p_t$ in descending order and calculate the quantile, which is the proportion of incorrect answers ranked after the target answer $o$.

Notably, some test paths in both datasets are “type-match trivial”, i.e., all type matching candidate answers are correct. Hence, the quantile of these test paths are undefined, and we exclude them from evaluation.

I Proof of Corollary 1

**Proof 3** As shown in Theorem 2, if $r$ is a symmetric relation, we have

$$
r_i = \pm 1 \Rightarrow \cos \theta_i + isin \theta_i = \pm 1
$$

$$
\Rightarrow \cos \theta_i = \pm 1
$$

$$
\Rightarrow \theta_i = 0 \lor \pm \pi
$$

Otherwise, if $r$ is an antisymmetric relation, we have

$$
r_i \neq \pm 1 \Rightarrow \cos \theta_i + isin \theta_i \neq \pm 1
$$

$$
\Rightarrow \cos \theta_i \neq \pm 1
$$

$$
\Rightarrow \theta_i \neq 0, \pm \pi
$$

J Proof of Corollary 2

**Proof 4** As shown in Theorem 2, if $r_1$ is the inverse of $r_2$, we have

$$
\begin{aligned}
(r_1, r_2) & = 1 \\
(b_{1i}, b_{2i}) & = b_{2i}
\end{aligned}
$$

In the first case, we have

$$
\begin{aligned}
(r_1, r_2) & = 1 \\
(b_{1i}, b_{2i}) & = b_{2i}
\end{aligned}
$$

In the second case, we have

$$
r_{1i} = r_{2i} = 1 \Rightarrow \cos \theta_{1i} + isin \theta_{1i} = \cos \theta_{2i} + isin \theta_{2i} = 1
$$

$$
\Rightarrow \theta_{1i} = \theta_{2i} = 0
$$

Combining the above two cases, we have

$$
\theta_{1i} + \theta_{2i} = 0 \lor \pm 2\pi
$$

K PQA Results of Different Path Lengths

We further report the path query answering results for different path lengths. As shown in Table 9 and Table 10, RotateCT outperforms RotatE and Rotate3D over most metrics, which verifies the superior capability of RotateCT to deal with the multi-hop reasoning.
| Model   | length=1 MQ | length=2 MQ | length=3 MQ | length=4 MQ | length=5 MQ |
|---------|-------------|-------------|-------------|-------------|-------------|
| RotatE  | 0.934       | 0.794       | 0.882       | 0.922       | 0.858       |
| Rotate3D| 0.933       | 0.796       | 0.901       | 0.921       | 0.854       |
| RotateCT| 0.934       | 0.803       | 0.885       | 0.928       | 0.863       |

Table 9: Path query answering results of each path length on Freebase. Best results are in **bold**.

| Model   | length=1 MQ | length=2 MQ | length=3 MQ | length=4 MQ | length=5 MQ |
|---------|-------------|-------------|-------------|-------------|-------------|
| RotatE  | 0.868       | 0.350       | 0.970       | 0.973       | 0.970       |
| Rotate3D| 0.872       | 0.365       | 0.972       | 0.974       | 0.973       |
| RotateCT| 0.865       | 0.338       | 0.972       | 0.975       | 0.973       |

Table 10: Path query answering results of each path length on WordNet. Best results are in **bold**.

4932