Article

Modeling Noncommutative Composition of Relations for Knowledge Graph Embedding

Chao Xiang 1,*, Cong Fu 1, Deng Cai 1 and Xiaofei He 1,2

1 State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310058, China
2 Fabu Inc., Hangzhou 310030, China
* Correspondence: chaoxiang@zju.edu.cn

Abstract: Knowledge Graph Embedding (KGE) is a powerful way to express Knowledge Graphs (KGs), which can help machines learn patterns hidden in the KGs. Relation patterns are useful hidden patterns, and they usually assist machines to predict unseen facts. Many existing KGE approaches can model some common relation patterns like symmetry/antisymmetry, inversion, and commutative composition patterns. However, most of them are weak in modeling noncommutative composition patterns. It means these approaches can not distinguish a lot of composite relations like “father’s mother” and “mother’s father”. In this work, we propose a new KGE method called QuatRotatScalE (QRSE) to overcome this weakness, since it utilizes rotation and scaling transformations of quaternions to design the relation embedding. Specifically, we embed the relations and entities into a quaternion vector space under the difference norm KGE framework. Since the multiplication of quaternions does not satisfy the commutative law, QRSE can model noncommutative composition patterns naturally. The experimental results on the synthetic dataset also support that QRSE has this ability. In addition, the experimental results on real-world datasets show that QRSE reaches state-of-the-art in link prediction problem.

Keywords: knowledge graph; knowledge graph embedding; link prediction; quaternion; relation pattern

1. Introduction

Knowledge Graph (KG) is composed by structured, objective facts. The facts are usually expressed in the form of triples as \((h, r, t)\), where \(h\), \(r\), and \(t\) express the head entity, the relation, and the tail entity, respectively. For example, \((\text{China}, \text{located_in}, \text{Asia})\).

Knowledge graphs have successfully supported many applications in various fields, such as recommender systems [1], question answering [2], information retrieval [3], and natural language processing [4]. KGs have also attracted increased attention from both industry and academic communities. However, real-world knowledge graphs, such as Dbpedia [5], Freebase [6], Yago [7], and WordNet [8], are usually incomplete, which restricts their applications. Thus, knowledge graph completion has become a widely studied subject. This subject is usually formulated as a link prediction problem, i.e., predicting the missing links that should be in the knowledge graph. Generally speaking, it asks us to design an agent that takes the query as input and outputs some entities. The query may contain a head entity and a relation or a tail entity and a relation. Every outputted entity should be able to form a plausible triple together with the query.

So far, the fundamental way to deal with the link prediction problem is Knowledge Graph Embedding (KGE) in industry and academia [9–13]. In this way, the agent needs to learn a low-dimensional vector representation, also called embedding, for each entity and relation. We have to design a scorer that can grade any triple in the embedding form for its plausibility. When predicting the unknown entity, the agent only needs to grade all possible triples (composed by the query with each candidate entity) and then take the candidate entities of high score triples as the predicted result.

Citation: Xiang, C.; Fu, C.; Cai, D.; He, X. Modeling Noncommutative Composition of Relations for Knowledge Graph Embedding. Electronics 2023, 12, 1348. https://doi.org/10.3390/electronics12061348

Academic Editor: Stefanos Kollias
Received: 7 February 2023
Revised: 6 March 2023
Accepted: 7 March 2023
Published: 12 March 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).
There are two reasons why these knowledge-graph embedding methods can tackle the link prediction problem effectively. On the one hand, there are many utilisable relation patterns in real-world KGs, such as symmetry/antisymmetry, inversion, and composition. These relation patterns are generally presented as natural redundancies in KGs, such as triple (China, located_in, Asia) and triple (Asia, includes, China) may exist in some KGs simultaneously, and they are describing the same fact. Here, located_in and includes are inverse relations for each other. On the other hand, existing KGE models have been able to model most relation patterns, i.e., evaluate the plausibility of triples by utilizing the relation patterns. For example, TransE [10] can model inversion patterns. When there are many natural redundancies relevant to located_in and includes in training KG, even if it has only seen triple (China, located_in, Asia) but not seen triple (Asia, includes, China), the TransE model can still mark a high plausibility score for triple (Asia, includes, China).

However, as far as we know, almost none of the existing KGE models can perfectly model the aforementioned relation patterns. For example, RotatE [13] declares that it can model all of the relation patterns, but it still has a fatal defect in modeling the composition patterns: It can only model commutative composition patterns, but can not model noncommutative composition patterns. This defect is also existing in some other KGE models which claim themselves can model composition patterns, such as TransE. Briefly speaking, a composition pattern implies the relation pattern that in the shape of \( r_1 \oplus r_2 = r_3 \), where \( \oplus \) means the ordered composition of \( r_1 \) and \( r_2 \). If composition pattern \( r_1 \oplus r_2 = r_3 \) exists in some KG, it means that KG has frequent natural redundancies in the form of \( \{(c_1, r_1, c_2), (c_2, r_2, c_3), (c_1, r_3, c_3)\}, c_i(i = 1, 2, 3) \) can be any entity. If \( r_1 \oplus r_2 = r_3 \) and \( r_2 \oplus r_1 = r_3 \) are both in a KG, we call \( r_1 \oplus r_2 = r_3 \) a commutative composition pattern, such as is_on_the_east_of \( \oplus \) is_on_the_north_of = is_on_the_northeast_of and is_on_the_north_of \( \oplus \) is_on_the_east_of = is_on_the_northeast_of. Otherwise, if only \( r_1 \oplus r_2 = r_3 \) is in the KG, it is a noncommutative composition pattern, such as is_the_husband_of \( \oplus \) is_the_mother_of = is_the_father_of.

Both of RotatE and TransE model the noncommutative composition pattern as the commutative composition pattern by mistake. This mistake will bring severely ridiculous inferences. For example, they will infer (i.e., mark a high score for) triple (Mary, is_the_mother_of, Barbara) based on existing triples (Mary, is_the_mother_of, James) and (James, is_the_husband_of, Barbara). The primary cause of this mistake is that they have not taken the design inspiration of their models carefully. They both expect to express a fact triple (\( h, r, t \)) through an equation relevant to the embeddings of \( h, r, \) and \( t \) (noted as boldface letters \( h, r, \) and \( t \)): \( h \odot r = t \), where \( \odot \) is some binary operation. Thus, they design the score function in the form of \( -||h \odot r - t|| \), and we can see that the closer the equation is to hold, the higher the plausibility score is. TransE embeds entities and relations into the real number vector space and takes the addition in that space as \( \odot \), while RotatE replaces the real numbers with the complex numbers and takes the element-wise multiplication as \( \odot \). Since these two operations both satisfy the commutative law, the corresponding two models can only model commutative composition patterns.

Inspired by QuatE [14], which will be discussed in Section 2.1.2, we propose a new KGE model called QuatRotatScaLE (or QRSE, for short) in this paper. The main difference from TransE and RotatE is it embeds entities and relations into the quaternion [15] vector space and takes the element-wise multiplication in that space as \( \odot \). Because the quaternion multiplication generally does not satisfy the commutative law (but there are special cases where the law holds), QRSE can model both composition patterns. Furthermore, we can prove that QRSE can also model the rest relation patterns. Thus it has become one of the KGE models that can model most relation patterns up to now. We evaluated QRSE and compared it with many baselines in two well-established and widely used real-world datasets FB15k-237 [16] and WN18RR [17]. The results indicate our method has reached the state-of-the-art in link prediction problem.
2. Related Work

At present, there are two classes of methods to solve the knowledge graph completion or link prediction problem. One class of methods is the KGE methods, and the other is the path-finding methods. Both of them are introduced below:

2.1. KGE Method

Embedding methods are widely used in many fields of machine learning since the embeddings of sentences, graphs, and many other data types can be easily transferred to various downstream tasks with only a little task-specific fine-tuning. For example, studies [18,19] first learn the embeddings of sentences and then use these embeddings to perform sentiment classification. Study [20] first learns an embedding for each graph, then use these embeddings to predict the missing labels of graphs. In addition, some other studies learn an embedding vector for each object (i.e., the node in a graph) of a given Heterogeneous Information Network (HIN) in a (semi-)supervised [21] or self-supervised [22] manner. Taking advantage of the learned embeddings of objects, they can fulfill many tasks, e.g., object classification, clustering, and visualization.

In knowledge graph completion or link prediction problem, Knowledge Graph Embedding methods are also the most studied methods. Let us use $E$ to represent the set of all entities and use $R$ to represent the set of all relations in KG. KGE methods need to assign a vector representation to every entity $e \in E$ and relation $r \in R$, noted in boldface letters $e$ and $r$, respectively. $e$ or $r$ is also called the embedding of $e$ or $r$. In addition to this, KGE methods still need to design a score function $f_r(h, t)$ to mark the plausibility of the triple $(h, r, t)$. The objective of optimization is to mark high scores for the true triples and low scores for the false triples. Based on the type of score function, we can further divide the KGE methods into two sorts, KGE based on difference norm and KGE based on semantic matching:

2.1.1. KGE Based on Difference Norm

The common motivation of this sort of method is they want to use a triple approximate equation $f_1(h, r) \approx f_2(t, r)$ to describe any triple $(h, r, t)$, and the strict equation should hold for fact triples. As for the unknown triples, they think the proximity of the two sides can reflect the plausibility of the triple. Thus, the score functions of these methods are always in the form of $f_r(h, t) = -\|f_1(h, r) - f_2(t, r)\|$.

Among them, there is a kind of method that is widely studied, called translational methods. We call them “translational” because the origin of this kind of method, TransE, uses the translation transformation to design the triple approximate equation. Precisely, it chooses the real number vector space $\mathbb{R}^k$ as the embedding space and regards the relation embedding $r$ as a translation transformation from head entity embedding $h$ to tail entity embedding $t$. So it designs the triple approximate equation as $h + r \approx t$. Following TransE, many improvements have emerged. TransH [23] claims it is better to assign a hyperplane in embedding space for every relation (the hyperplane’s normal vector noted as $r_p$), and only regards $r$ as a translation from the projection of $h$ to the projection of $t$ on that hyperplane. Hence the triple approximate equation of TransH is $(I - r_p r_p^\top)h + r \approx (I - r_p r_p^\top)t$, where $I$ is the identity matrix. TransR [24] generalizes TransH, it assigns a linear map to every relation $r$, noted as transfer matrix $W_r$. This linear map maps $h$ and $t$ into the relation space. Then TransR utilizes the images of $h$ and $t$ in the relation space with $r$ to design the triple approximate equation in TransE’s style: $W_r h + r \approx W_r t$. Further, Stranse [25] assigns each relation $r$ two different transfer matrices $W_{r,1}$ and $W_{r,2}$. Similarly, the triple approximate equation is designed as $W_{r,1} h + r \approx W_{r,2} t$. These derivative methods of TransE are collectively known as TransX. Their score functions can be written in the form of $f_r(h, t) = -\|g_{r,1}(h) + r - g_{r,2}(t)\|$, where $g_{r,\cdot}(\cdot)$ denotes a matrix multiplication concerning relation $r$.

Since the large number of the derivative methods of TransE, some literature uses the translational methods to refer to KGE based on difference norm in general. But this is
not accurate enough. Some other methods do not turn to translation transformation to design their triple approximate equations, such as TorusE [26] and RotatE. TorusE chooses a compact Lie group as its embedding space and can be regarded as a special case of RotatE when the embedding modulus are fixed [13]. RotatE embeds entities and relations into the complex number vector space \( \mathbb{C}^k \). It wants to replace the translation in \( \mathbb{R}^3 \) with the rotation in \( \mathbb{C}^k \). Specifically, for each element \( r_i \) (\( 1 \leq i \leq k \)) of \( r \), RotatE fixes it as a unitary complex number (i.e., \(|r_i| = 1\)). Hence the complex multiplication between the \( i \)-th element of \( h \) (i.e., \( h_i \)) and \( r_i \) means \( h_i \) rotates in its complex plane with angle \( \text{Arg}(r_i) \) (i.e., the argument of complex \( r_i \)). Let us use \( \circ \) to denote the Hadamard (element-wise) product between two complex vectors, the triple approximate equation of RotatE is \( h \circ r \approx t \).

There are some KGE methods with score functions belonging to a special case of difference norm, which is in the form of \(-\| h \circ r - t \|\), where \( \circ \) is some binary operation. When the ideal optimization is achieved, the triple approximate equations of these methods hold: \( h \circ r = t \). This property is useful to explain some abilities to model relation patterns. For example, TransE and RotatE are two of these methods, and because their binary operations are both associative and commutative, they can only model commutative composition patterns. For more details, please see Section 5.

2.1.2. KGE Based on Semantic Matching

The intuition of this sort of method is to measure the plausibility of a triple by inspecting the matching degree of the latent semantics of the two entities and the relation.

There is a family of methods called bilinear models that design score functions as bilinear maps of head and tail entities. RESCAL [9] may be the first bilinear model. It selects real vector space \( \mathbb{R}^k \) as the embedding space of entities and assigns a \( k \times k \) real matrix \( W_r \) to each relation \( r \). Then it directly applies \( W_r \) to define a bilinear map as the score function. To reduce the complexity of \( W_r \), DistMult [11] restricts \( W_r \) to be a diagonal matrix. So DistMult can express \( W_r \) as a vector \( r \) in \( \mathbb{R}^k \) and rewrite the score function in the form of the multi-linear dot product of \( h, r, \) and \( t \). To overcome DistMult’s weakness in modeling antisymmetry relation pattern, ComplEx [12] extends the embedding space into the complex vector space \( \mathbb{C}^k \), and modifies the score function. QuatE [14] further develops ComplEx, it extends the embedding space into the quaternion vector space to obtain better expression ability. DualE [27] uses the dual quaternion vectors to design the embeddings of entities and relations, and chooses the dual quaternion inner product as the score function. DihEdral [28] designs entity embeddings with real vectors, and designs relation embeddings with dihedral group vectors, where each dihedral group is expressed as a second-order discrete real matrix. Although its score function is a bilinear form, which belongs to the type of semantic matching, it is theoretically proven that this score function is equivalent to a difference norm function in the form of \(-\| h \circ r - t \|\) for optimizing relation embeddings. So DihEdral has the ability to model composition patterns like TransE and RotatE. Furthermore, since the multiplication of dihedral groups generally does not satisfy the commutative law, DihEdral can model noncommutative composition patterns. However, because the relation embeddings take discrete values, DihEdral has to use special treatments of the relation embeddings during the training process, and the actual performance is easily affected by special treatments. As for QuatE and DualE, their relation embeddings have the potential to model noncommutative composition patterns for the (dual) quaternion multiplication generally does not satisfy the commutative law. Nevertheless, because their score functions belong to the type of semantic matching and lack the theoretical equivalence to a difference norm function in the form of \(-\| h \circ r - t \|\) like DihEdral at present, their abilities to model the composition patterns have no strict theoretical guarantees. More precisely, their triple approximate equations, if any, do not necessarily hold when the ideal optimization is achieved, which is a crucial but easily overlooked step for a rigorous proof.
Apart from bilinear models, some models based on neural networks emerged recently. Such as ConvE \cite{Dettmers2018} and ConvKB \cite{Dettmers2018a} take the convolutional neural networks to construct the score functions.

Some mentioned KGE methods are listed in Table 1 with their score functions. Their abilities to model the relation patterns are shown in Table 2. We can see that our QRSE can model all relation patterns, which is a rare ability.

### Table 1. Score functions and embedding spaces of several KGE models.

<table>
<thead>
<tr>
<th>Model</th>
<th>Score Function</th>
<th>Embedding Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransE</td>
<td>$-|h + r - t|$</td>
<td>$h, r, t \in \mathbb{R}^k$</td>
</tr>
<tr>
<td>TransX</td>
<td>$-|g_{r,1}(h) + r - g_{r,2}(t)|$</td>
<td>$h, r, t \in \mathbb{R}^k$</td>
</tr>
<tr>
<td>RotatE</td>
<td>$-|h \circ r - t|$</td>
<td>$h, r, t \in \mathbb{C}^k,</td>
</tr>
<tr>
<td>RESCAL</td>
<td>$h^\top W_r t$</td>
<td>$h, t \in \mathbb{R}^k, W_r \in \mathbb{R}^{k \times k}$</td>
</tr>
<tr>
<td>DistMult</td>
<td>$\langle h, r, t \rangle$</td>
<td>$h, r, t \in \mathbb{R}^k$</td>
</tr>
<tr>
<td>ComplEx</td>
<td>$\text{Re}(\langle h, r, t \rangle)$</td>
<td>$h, r, t \in \mathbb{C}^k$</td>
</tr>
<tr>
<td>QuatE</td>
<td>$\langle h, r, \bar{t} \rangle$</td>
<td>$h, r, t \in \mathbb{H}^k,</td>
</tr>
<tr>
<td>QRSE</td>
<td>$-|h \otimes r - t|$</td>
<td>$h, r, t \in \mathbb{H}^k$</td>
</tr>
</tbody>
</table>

### Table 2. The modeling ability comparison for various relation patterns among different models (partial reference from \cite{Jiang2019}).

<table>
<thead>
<tr>
<th>Model</th>
<th>Symmetry</th>
<th>Anti-Symmetry</th>
<th>Inversion</th>
<th>Commutative Composition</th>
<th>Noncommutative Composition</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransE</td>
<td>(\times)</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>(\times)</td>
</tr>
<tr>
<td>TransX</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>(\times)</td>
<td>(\times)</td>
<td>(\times)</td>
</tr>
<tr>
<td>RotatE</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>(\times)</td>
</tr>
<tr>
<td>RESCAL</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>(\times)</td>
<td>(\times)</td>
</tr>
<tr>
<td>DistMult</td>
<td>(\checkmark)</td>
<td>(\times)</td>
<td>(\times)</td>
<td>(\times)</td>
<td>(\times)</td>
</tr>
<tr>
<td>ComplEx</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>(\times)</td>
<td>(\times)</td>
</tr>
<tr>
<td>QuatE</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
</tr>
<tr>
<td>QRSE</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
<td>(\checkmark)</td>
</tr>
</tbody>
</table>

Additionally, “supervised relation composition” \cite{Sun2018} is a method that can model composition patterns under supervision. But it is not a KGE method. Its goal is to design and train a function model that can take the embeddings of two relations as input and output the embedding of the composite relation of these two relations. The relation embeddings used are provided by an existing KGE model and are fixed once obtained. The supervisory information used for training is mined from the original KGs by another method. This method and the KGE models mentioned before belong to different research directions. The direction of KGE models studies how to directly model relation patterns (including composition patterns) by training entity and relation embeddings from the original KGs.
2.2. Path-Finding Method

This class of methods does not need score function to predict unknown entities, such as MINERVA [31], MultiHopKG [32], and DeepPath [33]. Instead, they should start from the query entity node and follow the direction implied by the query relation to search the KG for the unknown entity. Compared with KGE methods, their results are explainable to some extent since they can provide the inference paths as evidence, but the lack of precision is their weakness at present.

3. Preliminaries

Before introducing our proposed method, let us briefly explain the related concepts and geometric meaning of quaternions.

3.1. A Brief Introduction of Quaternion

As an extended number system from the complex numbers \( \mathbb{C} \), quaternions \( \mathbb{H} \) [15] have to import three fundamental quaternion units \( i, j, \) and \( k \), which are not existing in the real numbers. Each quaternion \( q \) can be expressed as \( q = a + bi + cj + dk \), where \( a, b, c, \) and \( d \) are all real numbers. The addition of quaternions is defined as \( (q_1 + q_2) = (a_1 + a_2) + (b_1 + b_2)i + (c_1 + c_2)j + (d_1 + d_2)k \). The multiplication between any two fundamental quaternion units are defined as \( i^2 = j^2 = k^2 = −1 \) and \( ij = −ji = k, jk = −kj = i, ki = −ik = j \). Obviously, this multiplication is associative but not commutative. For completeness, we also confirm the multiplication between any one in \( \{i, j, k\} \), and a real number is commutative and associative. To obey the distributive law, we consequently get the multiplication between two arbitrary quaternions as:

\[
\begin{align*}
(a_1 + b_1i + c_1j + d_1k)(a_2 + b_2i + c_2j + d_2k) & = a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2 \\
& + (a_1b_2 + b_1a_2 + c_1d_2 - d_1c_2)i \\
& + (a_1c_2 + c_1a_2 + d_1b_2 - b_1d_2)j \\
& + (a_1d_2 + d_1a_2 + b_1c_2 - c_1b_2)k.
\end{align*}
\]

We can conclude that the multiplication of quaternions (also known as the Hamilton product) holds the associative and distributive law, but does not hold the commutative law in general. Nevertheless, there are some special cases where the commutative law holds.

Some useful concepts of quaternions are listed as follows (let \( q = a + bi + cj + dk \)):

**Modulus:** The modulus of \( q \) is written as \( |q| \) and is defined as \( |q| = \sqrt{a^2 + b^2 + c^2 + d^2} \). Since the set of quaternions \( \mathbb{H} \) is a linear space isomorphic to \( \mathbb{R}^4 \) with basis \( (1, i, j, k) \), modulus means the length of \( q \) intuitively. In addition, if \( |q| = 1 \), \( q \) is called a unit quaternion.

**Real and imaginary part:** Similar to complex numbers, real number \( a \) is the real part of \( q \), and real vector \( v = (b, c, d)^T \) is the imaginary part of \( q \). Sometimes we would like to express \( q \) in the form of \( [a, v] \) for convenience. Then, the multiplication of quaternions can be written as \( [a_1, v_1][a_2, v_2] = [a_1a_2 - v_1 \cdot v_2, a_1v_2 + a_2v_1 + v_1 \times v_2] \), where \( \cdot \) is the dot product and \( \times \) is the cross product.

**Conjugate:** The conjugate of \( q \) is the quaternion \( ˘q = a - bi - cj - dk \). It has theses properties: (1) \( ˘q \overline{q} = q \overline{q} = |q|^2 \), and from (1), (2) we get (3) \( |q_1||q_2| = |q_1 \overline{q}_2| \). As a corollary, the product of two unit quaternions is also a unit quaternion.

**Reciprocal:** If \( q \neq 0 \), the reciprocal of \( q \) is the quaternion \( q^{-1} \) such that \( qq^{-1} = q^{-1}q = 1 \), and it is equivalent to define \( q^{-1} = ˘q/|q|^2 \).

3.2. The Geometric Meaning of the Multiplication of Quaternions

To see the geometric meaning of the multiplication of quaternions, we have to view the \( \mathbb{H} \) as a linear space isomorphic to \( \mathbb{R}^4 \) with an orthonormal basis \( (1, i, j, k) \). Any \( q \) in \( \mathbb{H} \) can be expressed in the form as \( \rho[\cos \theta, \sin \theta n] \), where \( \rho \geq 0 \) and \( |n| = 1 \). This is because if \( q \neq 0 \) we could set \( \rho = |q|, \theta = \arccos (a/|q|) \), and \( n = v/|v| \), whereas if \( q = 0 \) we could...
set ρ = 0 and choose θ and n arbitrarily. Note that [cos θ, sin θ n] is a unit quaternion and implies the direction of q, while ρ implies the length of q.

Take another quaternion \( p = [s, u] \) from \( \mathbb{H} \), then the product \( pq = \rho(p[cos θ, sin θ n]) \) means a new quaternion reached via two steps from \( p \): (1) changing the direction of \( p \) according to \([cos θ, sin θ n]\), (2) stretching the length by \( \rho \) times. So we only have to see what is the change implied by \( p[cos θ, sin θ n] \).

Without loss of generality, we suppose \( u \neq 0 \) and \( u \) is not parallel with \( n \). Thus we can find another orthonormal basis of \( \mathbb{H} \): \( ([1, 0], [0, n], [0, n_⊥], [0, n_⊥]) \). Here, \( n_⊥ = (u \cdot n) n / \|u \cdot n\| \) and \( n_⊥ = n_⊥ \times n \). Besides, we confirm the coordinates of \( p \) under this basis is \( (s, l, l⊥, 0) \T \), where \( l = u \cdot n \) and \( l_⊥ = \|u \cdot (u \cdot n)\| \). Thus we can split \( p \) into two parts: \( p = p_1 + p_2 \), where \( p_1 = [s, l n] \) and \( p_2 = [0, l_⊥ n_⊥] \). So we only have to see what do \( p_1[cos θ, sin θ n] \) and \( p_2[cos θ, sin θ n] \) mean.

Since \( p_1[cos θ, sin θ n] = [s cos θ − l sin θ, (s sin θ + l cos θ) n] \), this product and \( p_1 \) are both in the plane with basis \( ([1, 0], [0, n]) \). And we can show the transformation from \( p_1 \) to the product by their coordinates under basis \( ([1, 0], [0, n]) \) as:

\[
p_1 \rightarrow p_1[cos θ, sin θ n] : \begin{pmatrix} s \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} s cos θ − l sin θ \\ s sin θ + l cos θ \end{pmatrix} = \begin{pmatrix} cos θ & −sin θ \\ sin θ & cos θ \end{pmatrix} \begin{pmatrix} s \\ l \end{pmatrix}.
\]

(2)

So \( p_1[cos θ, sin θ n] \) means \( p_1 \) rotates with angle \( θ \) counterclockwise in plane \( span([1, 0], [0, n]) \).

In the same way, since \( p_2[cos θ, sin θ n] = [0, l_⊥ cos θ n_⊥ + l_⊥ sin θ n_⊥] \), this product and \( p_2 \) are both in the plane with basis \( ([0, n_⊥], [0, n_⊥]) \). And we can show the transformation from \( p_2 \) to the product by their coordinates under basis \( ([0, n_⊥], [0, n_⊥]) \) as:

\[
p_2 \rightarrow p_2[cos θ, sin θ n] : \begin{pmatrix} l_⊥ \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} l_⊥ cos θ \\ l_⊥ sin θ \end{pmatrix} = \begin{pmatrix} cos θ & −sin θ \\ sin θ & cos θ \end{pmatrix} \begin{pmatrix} l_⊥ \\ 0 \end{pmatrix}.
\]

(3)

So \( p_2[cos θ, sin θ n] \) means \( p_2 \) rotates with angle \( θ \) counterclockwise in plane \( span([0, n_⊥], [0, n_⊥]) \).

In a word, the change implied by \( p[cos θ, sin θ n] \) is: (1) Split \( p \) into two components \( p_1 \) and \( p_2 \), where \( p_1 \) is in plane \( span([1, 0], [0, n]) \) and \( p_2 \) is in plane \( span([0, n_⊥], [0, n_⊥]) \); (2) Rotate \( p_1 \) and \( p_2 \) with angle \( θ \) counterclockwise in each plane simultaneously, as shown in Figure 1; (3) Add two new components together.

![Figure 1. How do \( p_1 \) and \( p_2 \) rotate when they are multiplied by \([cos θ, sin θ n]\) on the right.](image)

As a special case, when \( u = 0 \) or \( u \) is parallel with \( n \), \( p = p_1 \). Thus at that time, \( p[cos θ, sin θ n] \) only means the rotation in plane \( span([1, 0], [0, n]) \). Moreover, the geometric meaning of \( qp \) is almost the same with \( pq \), except that the rotation for \( p_2 \) is clockwise.

4. Proposed Method

Now we start to introduce our proposed KGE model. The embedding spaces of entities \( E \) and relations \( R \) are both the quaternion vector space \( \mathbb{H}^k \). For any \( e \in E \) and \( r \in R \), their
Electronics 2023, 12, 1348

embeddings are noted as e and r in lower case bold letters, respectively. The i-th elements of e and r are written as e_i (e_i ∈ ℝ) and r_i (r_i ∈ ℝ) for every integer i from 1 to k. Our model is based on difference norm, so we first define its triple approximate equation as h ⊗ r ≈ t for any triple (h, r, t). Here, ⊗ denotes the Hadamard (element-wise) product between two quaternion vectors. So this triple approximate equation is equivalent to asking for h_i r_i ≈ t_i for all i (1 ≤ i ≤ k). In consequence, we get our score function:

\[ f_r(h, t) = -\|h \otimes r - t\| . \]

(4)

Here, \( \|q\| \) is the abbreviation of \( \|q\|_{p,1} = \sum_{i=1}^{k} (|a_i|^p + |b_i|^p + |c_i|^p + |d_i|^p)^{\frac{1}{p}} \) for any quaternion vector \( q_{(a_i, b_i, c_i, d_i)} = a_i + b_i i + c_i j + d_i k, 1 \leq i \leq k \). \( p (p \geq 1) \) is a hyperparameter.

According to the geometric meaning of the quaternion multiplication, we can explain the purpose of this triple approximate equation intuitively: We treat each element of relation embedding \( r_i \) (written in the form of \( \rho_i [\cos \theta_i, \sin \theta_i, n_i] \)) as a two-step transformation from \( h_i \) to \( t_i \): (1) Rotate \( h_i \) in two planes (\( \text{span}([1, 0], [0, n_i]) \) and \( \text{span}([0, n_{i,0}], [0, n_{i,x}]) \)) counterclockwise with angle \( \theta_i \); (2) Stretch \( h_i \) with scaling factor \( \rho_i \). Thus we refer to our model as QuatRotatScalE (or QRSE, for short) due to we use Quaternions with Rotation and Scaling transformations to design the Embedding model.

Optimization

The general objective of KGE models is to return high scores for true triples and low scores for false triples. We adopt negative sampling as our training style to avoid the efficiency loss brought by the huge number of entities like most of the other KGE methods. The training KG usually only contains true triples (positive samples, noted as \( \Omega \)) without false triples (negative samples). Thus we apply a common way (i.e., corrupting the positive samples) to obtain the negative samples. Suppose \( (h, r, t) \) is a positive sample, we can get two sets of negative samples by replacing the head or tail entity with other entities:\n\( \mathcal{N}_h(r, t) = \{ (h', r, t) \mid h' \text{ is uniformly sampled from } \mathcal{E} \text{ s.t. } (h', r, t) \notin \Omega \} \) and \( \mathcal{N}_i(h, r) = \{ (h, r', t) \mid t' \text{ is uniformly sampled from } \mathcal{E} \text{ s.t. } (h, r, t') \notin \Omega \} \). The size of negative samples \( \mathcal{N}_h(r, t) \) and \( \mathcal{N}_i(h, r) \) is fixed and much smaller than \( |\mathcal{E}| \).

Following RotatE [13], we use the loss function on each triple \( (h, r, t) \) in the training KG as

\[ L = -\log \sigma(\gamma + f_r(h, t)) - \sum_{(h', r', t') \in \mathcal{N}'} p(h', r', t') \log \sigma(-\gamma - f_r(h', t')) \]

(5)

where \( \sigma \) is the sigmoid function, \( \gamma \) is a fixed margin, and \( \mathcal{N}' = \mathcal{N}_h(r, t) \cup \mathcal{N}_i(h, r) \). In practice, \( \mathcal{N}' \) is regenerated in the same way (\( \mathcal{N}_h(r, t) \) or \( \mathcal{N}_i(h, r) \)) for every positive sample in one training batch. Once it turns to the next training batch, \( \mathcal{N}' \) should switch the regenerating way. \( p(h', r', t') \) is the distribution of self-adversarial negative sampling proposed by RotatE [13] and is defined as

\[ p(h', r', t') = \frac{\exp(\alpha f_r(h', t'))}{\sum_{(h', r', t') \in \mathcal{N}' \exp(\alpha f_r(h', t'))} } \]

(6)

where \( \alpha \) is the temperature of sampling. The self-adversarial negative sampling can moderate the low efficiency of the uniform negative sampling. We also take Adam as our optimizer. Moreover, \( p(h', r', t') \) plays a role of importance sampling ratio in \( L \), so it need not backpropagate gradients through it.

5. Theoretic Analysis

5.1. Relation Patterns

As mentioned in introduction Section 1, modeling (i.e., identifying and utilizing) the relation patterns in KGs are the fundamental for KGE models to solve the link prediction problem. There are three types of relation patterns, which are very powerful and widely exist in various KGs [12,13,16,34,35]:
Symmetry/antisymmetry: A relation \( r \) is symmetric (antisymmetric) if \( \forall e_1, e_2 \in E, (e_1, r, e_2) \Rightarrow (e_2, r, e_1) \) \( \Leftrightarrow (e_2, r, e_1) \Rightarrow (e_2, r, e_1) \).

Inversion: Relation \( r_1 \) is inverse to relation \( r_2 \) if \( \forall e_1, e_2 \in E, (e_1, r_2, e_2) \Rightarrow (e_2, r_1, e_1) \).

Composition: Relation \( r_1 \) is composed of relation \( r_2 \) and relation \( r_3 \) if \( \forall e_1, e_2, e_3 \in E, (e_1, r_2, e_2) \land (e_2, r_3, e_3) \Rightarrow (e_1, r_1, e_3) \). We adopt form \( r_2 \circ r_3 = r_1 \) to describe this composition pattern for simplicity. Moreover, if both of \( r_2 \circ r_3 = r_1 \) and \( r_3 \circ r_2 = r_1 \) make sense, \( r_2 \circ r_3 = r_1 \) is a commutative composition pattern. Otherwise, if only \( r_2 \circ r_3 = r_1 \) holds, it is a noncommutative composition pattern.

5.2. Abilities to Model Relation Patterns

In this subsection, we prove that QRSE can model symmetry/antisymmetry, inversion, and composition patterns. Additionally, TransE and RotatE are unable to model noncommutative composition patterns. Next, if triple \((h, r, t)\) is in the knowledge graph, we write it in the embedding space as \( h \otimes r = t \) for QRSE because its score function is a special case of difference norm \(-\|h \otimes r - t\|\) and when the ideal optimization is achieved, we can directly get \( h \otimes r = t \) (we can replace \( \otimes \) with + or \( \circ \) for TransE or RotatE for the same reason).

- **QRSE can model symmetry/antisymmetry patterns:**
  Suppose \( e_2 \otimes r = e_1 \) and \( e_1 \otimes r = e_2 \). We can get \( e_1 \otimes r \otimes r = e_1 \). It means for any \( i (1 \leq i \leq k), e_i \in E \). If \( e_1 = 0, r_i \) can be any quaternion. But if \( e_1 \neq 0, r_i \) must satisfies:

  \[
eq \quad e_i \otimes r_i = e_{i1} \quad \Leftrightarrow \quad e_{i1}^{-1}e_i \otimes r_i = e_{i1}^{-1}e_{i1} \quad \Leftrightarrow \quad r_i = r_i^{-1}.
\]

- **QRSE can model inversion patterns:**
  Suppose \( e_2 \otimes r_1 = e_1 \) and \( e_1 \otimes r_2 = e_2 \). We can get \( e_1 \otimes r_2 \otimes r_1 = e_1 \). It means for any \( i (1 \leq i \leq k), e_i \in E \). If \( e_1 = 0, r_{i2} \) and \( r_{i1} \) can be any quaternions. But if \( e_1 \neq 0, r_{i2} \) and \( r_{i1} \) must satisfy:

  \[
eq \quad e_{i1} \otimes r_{i1} = e_{i1} \quad \Leftrightarrow \quad e_{i1}^{-1}e_i \otimes r_{i1} = e_{i1}^{-1}e_{i1} \quad \Leftrightarrow \quad r_{i1} = r_{i1}^{-1}.
\]

Define \( q^{-1} = (q_1^{-1}, q_2^{-1}, ..., q_k^{-1}) \) for all \( q \in H^k \). We can conclude that if \( r_i = r_i^{-1}, r_1 \) and \( r_2 \) model an inversion pattern.

- **QRSE can model composition patterns:**
  Suppose \( e_1 \otimes r_2 = e_2 \) and \( e_2 \otimes r_3 = e_3 \), and \( e_1 \otimes r_1 = e_3 \). We can get \( e_1 \otimes r_2 \otimes r_3 = e_1 \otimes r_1 \). It means for any \( i (1 \leq i \leq k), e_i \in E \). If \( e_1 \neq 0, r_{i2}, r_{i3}, \) and \( r_{i1} \) can be any quaternions. But if \( e_1 \neq 0, r_{i2}, r_{i3}, \) and \( r_{i1} \) must satisfy:

  \[
eq \quad e_{i1} \otimes r_{i2} \otimes r_{i3} = e_{i1} \otimes r_{i1} \quad \Leftrightarrow \quad e_{i1}^{-1}e_i \otimes r_{i2} \otimes r_{i3} = e_{i1}^{-1}e_{i1} \quad \Leftrightarrow \quad r_{i2} \otimes r_{i3} = r_{i1}.
\]

Moreover, if we still suppose \( e_4 \otimes r_3 = e_5 \) and \( e_5 \otimes r_2 = e_6 \), and \( e_4 \otimes r_1 = e_6 \). Then if \( e_4 \neq 0 \) for all \( i (1 \leq i \leq k), r_{i2}, r_{i3}, \) and \( r_{i1} \) must satisfy: \( r_{i2} \otimes r_{i3} = r_{i1} \). This means \( r_{i2} \otimes r_{i3} = r_{i1} \). If we note \( r_{i2} = [a_{i2} \cdot v_{i2}] \) and \( r_{i3} = [a_{i3} \cdot v_{i3}] \), then we get:

\[
[a_{i2} \cdot a_{i3} - v_{i2} \cdot v_{i3}, a_{i2} \cdot v_{i3} + a_{i3} \cdot v_{i2} + v_{i2} \times v_{i3}] = [a_{i3} \cdot a_{i2} - v_{i3} \cdot v_{i2}, a_{i3} \cdot v_{i2} + a_{i2} \cdot v_{i3} + v_{i3} \times v_{i2}]
\]

\[
\Leftrightarrow v_{i2} \times v_{i3} = v_{i3} \times v_{i2}
\]

\[
\Leftrightarrow v_{i2} = 0 \text{ or } v_{i3} = \lambda v_{i2} \quad (\lambda \in \mathbb{R}).
\]
We can conclude that if \( r_2 \otimes r_3 = r_1 \), \( r_2, r_3 \), and \( r_1 \) model a composition pattern. Moreover, if \( v_{2,i} \) is parallel with \( v_{3,i} \) for all \( i(1 \leq i \leq k) \), it is a commutative composition pattern, otherwise, it is a noncommutative composition pattern.

- **TransE and RotatE** can not model noncommutative composition patterns, and they can only model commutative composition patterns:

  For TransE, we suppose \( e_1 + r_2 = e_2, e_2 + r_3 = e_3, e_1 + r_1 = e_3, e_4 + r_3 = e_5, e_5 + r_2 = e_6 \) but \( e_4 + r_1 \neq e_6 \), which means the composition of relation \( r_2 \) and \( r_3 \) is noncommutative. From the first three equations we get \( r_2 + r_3 = r_1 \), and from the fourth and fifth equations we get \( e_4 + r_3 + r_2 = e_6 \). Because \( r_2 + r_3 = r_3 + r_2 \), we get \( e_4 + r_1 = e_6 \), which contradicts the condition. Therefore TransE can not model noncommutative composition patterns. If we replace the condition \( e_4 + r_1 \neq e_6 \) with \( e_4 + r_1 = e_6 \), then the composition of relation \( r_2 \) and \( r_3 \) becomes commutative composition. In this case the previous contradiction disappears, which means TransE can model commutative composition patterns.

  As for RotatE, we suppose \( e_1 \circ r_2 = e_2, e_2 \circ r_3 = e_3, e_1 \circ r_1 = e_3, e_4 \circ r_3 = e_5, e_5 \circ r_2 = e_6 \), but \( e_4 \circ r_1 \neq e_6 \), which means the composition of relation \( r_2 \) and \( r_3 \) is noncommutative. Since \( r_2 \circ r_3 = r_3 \circ r_2 \) (the multiplication of complex numbers satisfies the commutative law), we can get \( e_4 \circ r_1 = e_6 \) in the same way as TransE, which contradicts the condition. So RotatE can not model noncommutative composition patterns. If we replace the condition \( e_4 \circ r_1 \neq e_6 \) with \( e_4 \circ r_1 = e_6 \), then the composition of relation \( r_2 \) and \( r_3 \) becomes commutative composition. In this case the previous contradiction disappears, which means RotatE can model commutative composition patterns.

6. **Experiments**

   In this section, we first evaluate QRSE with RotatE on a small knowledge graph made up of two families. This experiment will verify the superiority of QRSE in modeling noncommutative composition relation patterns. Then we evaluate QRSE and compare it with many baselines in two well-established and widely used real-world datasets.

6.1. **Experiment on a KG about Two Families**

   There are 10 entities and 4 relations in the training KG. Each entity is a member of one family, and each relation is a type of kinship. Such as triple \((Am1, son, Am2)\) means \(Am1\) has a son called \(Am2\). All of the triples in the training KG are shown in Figure 2, where each directed edge represents a triple, and its direction is from the head entity to the tail entity. Furthermore, the test set contains two triples: \((Bm1, daughter_of_son, Bw3)\) and \((Bm1, son_of_daughter, Bm3)\). We let models predict the head or tail entity for each test triple, so there are 4 queries during the test process.

   ![Figure 2. The structure of the training KG, where each directed edge represents a triple.](image)

   Since we need 2 and 4 real numbers to determine a complex number and a quaternion respectively, we take \(\mathbb{C}^{10}\) (i.e., embedding dimension \(k = 10\)) and \(\mathbb{H}^{5}\) (i.e., \(k = 5\)) as the entity embedding spaces for RotatE and QRSE. Thus in practice, we can express the entity embeddings of RotatE and QRSE as 20-D real vectors. Except for the embedding
dimension \( k \), we keep other hyperparameters the same for two models: batch size \( b = 10 \), self-adversarial sampling temperature \( a = 0 \), fixed margin \( \gamma = 0 \), learning rate \( \eta = 0.001 \), negative sampling size \( |N| = 2 \), and the order of norm in score function \( p = 2 \).

We use Hit@1 to measure the performance of models, which means the proportion of the correctly answered queries (i.e., the true answer’s score is ranked first) among all test queries. The test performances of RotatE and QRSE are shown in Figure 3. We can see that QRSE gets the best Hit@1 value 1.00 quickly, and after that, it keeps this Hit@1 value all the time during the training process. RotatE also gets the best Hit@1 value quickly; however, after that, it’s Hit@1 value is always fluctuating between 0.5 and 1.00 randomly. To explain this phenomenon, we inspected the detailed scores and embeddings at step 16,000, which is large enough to ensure the convergence of the two models.

Figure 3. The Hit@1 performance of RotatE and QRSE on the test set along with the training process.

The top 3 scores for all test queries are shown in Table 3. For the two queries to predict the head entity \( Bm1 \), the scores of \( Bm1 \) are much higher than the second candidate entities for both RotatE and QRSE. However for the two queries to predict the tail entities \( Bm3 \) and \( Bw3 \), only QRSE keeps the large gap between the first and the second score, whereas RotatE gives very close scores for the top 2 candidate entities on both of the two queries. This result reveals that, for RotatE, the score ranks for the top 2 candidates are unstable and easily affected by the random noise on the two queries to predict the tail. That is why the Hit@1 of RotatE fluctuates during training. Moreover, for RotatE, the top 2 candidate entities are \( Bm3 \) and \( Bw3 \) for both of the two tail queries. Thus we guess the embeddings of these two entities are also very close.

Table 3. The detailed test results of RotatE and QRSE at training step 16000.

<table>
<thead>
<tr>
<th>Model</th>
<th>Test Triple</th>
<th>Entity to Predict</th>
<th>Ranked Scores for Top 3 Candidate Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>RotatE</td>
<td>(Bm1, son_of_daughter, Bm3)</td>
<td>Bm1</td>
<td>Bm1: −0.0224, Am2: −5.4026, Bm3: −5.6849</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bm3</td>
<td>Bm3: −0.0224, Bw3: −0.0227, Bm1: −5.6840</td>
</tr>
<tr>
<td></td>
<td>(Bm1, daughter_of_son, Bw3)</td>
<td>Bm1</td>
<td>Bm1: −0.0178, Am2: −5.3975, Bm3: −5.6668</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bw3</td>
<td>Bw3: −0.0178, Bm3: −0.0179, Bm1: −5.6654</td>
</tr>
<tr>
<td>QRSE</td>
<td>(Bm1, son_of_daughter, Bm3)</td>
<td>Bm1</td>
<td>Bm1: −0.00087, Am2: −4.8535, Aw3: −4.9947</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bm3</td>
<td>Bm3: −0.00087, Aw3: −5.2716, Bw3: −5.3903</td>
</tr>
<tr>
<td></td>
<td>(Bm1, daughter_of_son, Bw3)</td>
<td>Bm1</td>
<td>Bm1: −0.00094, Aw2: −4.8532, Aw3: −4.9948</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bw3</td>
<td>Bw3: −0.00094, Bm3: −5.3905, Am2: −5.8095</td>
</tr>
</tbody>
</table>

Figure 4 shows the embeddings of \( Bm3 \) and \( Bw3 \) in RotatE and QRSE. As we guessed, the two embeddings are very close in RotatE but different in QRSE. This result verified that RotatE is unable to model noncommutative composition patterns, but QRSE can. Let us use the bold type to indicate the embeddings as before. For RotatE, along with the training process, \( Bw3 \) will close to \( Bm2 \), \( \text{daughter} \), and \( Bm2 \) will close to \( Bm1 \), \( \text{son} \). Hence \( Bw3 \)
will close to $Bm1 \circ son \circ daughter$. Similarly, $Bm3$ will close to $Bm1 \circ daughter \circ son$. Because $daughter \circ son = son \circ daughter$, $Bw3$ will close to $Bm3$. For QRSE, $daughter \otimes son \neq son \otimes daughter$ in general, so $Bw3$ will not close to $Bm3$.

Figure 4. The entity embeddings of RotatE and QRSE on training step 16,000. The 10-D complex or 5-D quaternion vectors are expressed in the corresponding 20-D real vectors.

We can also show this fact by directly inspecting the relation embeddings of the two models in Figure 5. Note that $daughter \oplus son$ and $son \oplus daughter$ are not the relations in KG but the combinations made up of the relations in KG. Their “embeddings” are calculated from the embeddings of some relations (e.g., the “embedding” of $daughter \oplus son$ is $daughter \otimes son$ in QRSE). Obviously, the embeddings of $son \_of\_daughter$ and $daughter\_of\_son$ are almost the same in RotatE, since they are both approaching $daughter \circ son$ during training. However, they are different in QRSE since the embeddings of $son \_of\_daughter$ is approaching $daughter \otimes son$ while the other is approaching $son \otimes daughter$ during training.

Figure 5. The relation embeddings of RotatE and QRSE on training step 16,000. A relation embedding of RotatE has 10 complex numbers with modulus 1, which are determined by their 10 arguments. Thus we express it by its 10 arguments in angle degrees. For QRSE, we continue use the corresponding 20-D real vectors for each relation embedding.

6.2. Experiment on Real-World Datasets

6.2.1. Experimental Setting

We still evaluated our method on two well-established and widely used real-world knowledge graphs, FB15k-237 [16] and WN18RR [17], with several strong baselines.

FB15k-237 is selected from FB15k [10], which is a subset of Freebase and mainly records the facts about movies, actors, and sports. Because FB15k suffers from test leakage through inverse relations: there are too many inversion patterns in KG, which are too easy to model, and even a simple rule-based model can perform well [17]. To make the results more reliable, FB15k-237 removed these inverse patterns. The statistics of FB15k-237 are 14,541 entities, 237 relations, 272,115 training triples, 17,535 validation triples, and 20,466 test triples.
WN18RR is selected from WN18 [10], which is a subset of WordNet and records lexical relations between words. WN18 also suffers from test leakage through inverse relations, so WN18RR removed its inverse patterns too. The statistics of WN18RR are 40,943 entities, 11 relations, 86,835 training triples, 3034 validation triples, and 3134 test triples.

The ranges of the hyperparameters for the grid search are following RotatE as embedding dimension $k \in \{125, 250, 500, 1000\}$, batch size $b \in \{512, 1024, 2048\}$, and fixed margin $\gamma \in \{3, 6, 9, 12, 18, 24, 30\}$. Moreover, we searched self-adversarial sampling temperature $\alpha$ in $\{0.5, 1.0, 1.5\}$, learning rate $\eta$ in $\{0.00005, 0.0001, 0.0002\}$, negative sampling size $|N|$ in $\{16, 32, 64, 128\}$, and the order $p$ of norm in score function in $\{2, 3, 4, 5, 6, 7\}$. The embeddings are also uniformly initialized.

From each test triple $(h, r, t)$, we generate two queries: $(?, r, t)$ and $(h, r, ?)$. Given each query, we can make a candidate triple by placing a candidate entity on the place of the entity to predict. The score of each candidate entity is just the score of its corresponding candidate triple. While ranking all the scores, we omit the scores of those candidate triples that already exist in training, validation, and test set, except the true answer for the query. This process is called “filtered” in some literature and is widely adopted in existing methods to avoid possibly flawed evaluation.

6.2.2. Results

We adopt these standard evaluation measures for both of the datasets: the mean reciprocal rank of the true answers (MRR), the proportion of queries whose true answers are ranked in the top k (Hit@k).

The link prediction results on real-world datasets are shown in Table 4. The result of TransE is taken from [29]. The results of DistMult, ComplEx, and ConvE are taken from [17]. The results of RotatE and DualE are taken from [13,27], respectively. The results of DihEdral(STE) and DihEdral(Gumbel) are taken from [28], where STE and Gumbel are two special treatments of the discrete relation embeddings. The results of QuatE and QuatE(TC) are taken from [14], where TC indicates the corresponding model using type constraints [36]. From this table, we can see that QRSE outperforms RotatE largely on all datasets and evaluation measures. This result supports our analysis of the modeling ability of the composition patterns. Compared with DihEdral(STE) and DihEdral(Gumbel), we find QRSE outperforms both of them on the two real-world datasets, whereas DihEdral(STE) is better than DihEdral(Gumbel) on FB15k-237 and just the opposite on WN18RR. This means the performance of DihEdral is easily affected by special treatments, and DihEdral can not perform well on the two real-world datasets simultaneously. Compared with DualE and QuatE, we find QRSE outperforms both of them too. This means that among all methods using (dual) quaternions so far, QRSE has explored the greatest potential of the (dual) quaternion space in the implementation of knowledge graph embedding. Because type constraints [36] can integrate prior knowledge into various KGE models and can significantly improve their performance in link prediction tasks, QRSE and most baselines display the results without it for fairness except QuatE(TC). Surprisingly, we can even see that QRSE is superior to QuatE with type constraints overall slightly. The success on this unfair comparison further demonstrates the excellence of QRSE. Overall, our QRSE has reached the state-of-the-art in link prediction problem on real-world datasets.
Table 4. Link prediction results on the FB15k-237 and WN18RR datasets. Numbers in boldface are the best, and underlined numbers are the second best.

<table>
<thead>
<tr>
<th>Model</th>
<th>FB15k-237</th>
<th></th>
<th></th>
<th></th>
<th>WN18RR</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MRR</td>
<td>Hit@1</td>
<td>Hit@3</td>
<td>Hit@10</td>
<td>MRR</td>
<td>Hit@1</td>
<td>Hit@3</td>
<td>Hit@10</td>
</tr>
<tr>
<td>TransE</td>
<td>0.294</td>
<td></td>
<td></td>
<td></td>
<td>0.465</td>
<td>0.226</td>
<td></td>
<td>0.501</td>
</tr>
<tr>
<td>DistMult</td>
<td>0.241</td>
<td>0.155</td>
<td>0.263</td>
<td>0.419</td>
<td>0.43</td>
<td>0.39</td>
<td>0.44</td>
<td>0.49</td>
</tr>
<tr>
<td>ComplEx</td>
<td>0.247</td>
<td>0.158</td>
<td>0.275</td>
<td>0.428</td>
<td>0.44</td>
<td>0.41</td>
<td>0.46</td>
<td>0.51</td>
</tr>
<tr>
<td>ConvE</td>
<td>0.325</td>
<td>0.237</td>
<td>0.356</td>
<td>0.501</td>
<td>0.43</td>
<td>0.40</td>
<td>0.44</td>
<td>0.52</td>
</tr>
<tr>
<td>RotatE</td>
<td>0.338</td>
<td>0.241</td>
<td>0.375</td>
<td>0.533</td>
<td>0.476</td>
<td>0.428</td>
<td>0.492</td>
<td>0.571</td>
</tr>
<tr>
<td>DualE</td>
<td>0.330</td>
<td>0.237</td>
<td>0.363</td>
<td>0.518</td>
<td>0.482</td>
<td>0.440</td>
<td>0.500</td>
<td>0.561</td>
</tr>
<tr>
<td>DihEdral(STE)</td>
<td>0.320</td>
<td>0.230</td>
<td>0.353</td>
<td>0.502</td>
<td>0.480</td>
<td></td>
<td>0.452</td>
<td>0.491</td>
</tr>
<tr>
<td>DihEdral(Gumbel)</td>
<td>0.300</td>
<td>0.204</td>
<td>0.332</td>
<td>0.496</td>
<td>0.486</td>
<td>0.442</td>
<td>0.505</td>
<td>0.557</td>
</tr>
<tr>
<td>QuatE</td>
<td>0.311</td>
<td>0.221</td>
<td>0.342</td>
<td>0.495</td>
<td>0.481</td>
<td>0.436</td>
<td>0.500</td>
<td>0.564</td>
</tr>
<tr>
<td>QuatE(TC)</td>
<td>0.348</td>
<td>0.248</td>
<td>0.382</td>
<td>0.550</td>
<td>0.488</td>
<td>0.438</td>
<td>0.508</td>
<td>0.582</td>
</tr>
<tr>
<td>QRSE</td>
<td>0.350</td>
<td>0.252</td>
<td>0.390</td>
<td>0.548</td>
<td>0.491</td>
<td>0.443</td>
<td>0.508</td>
<td>0.581</td>
</tr>
</tbody>
</table>

7. Conclusions and Future Work

We proposed a novel knowledge graph embedding model QRSE based on quaternions. QRSE is a KGE model that can model the noncommutative composition patterns. Besides, it can also model many other relation patterns, such as symmetry/antisymmetry, inversion, and commutative composition patterns. We verified these properties by theoretical proofs and experiments. From the definition of the triple approximate equation of QRSE, we can easily see that QRSE is a generalization of RotatE. Conversely, in some special cases, QRSE will degenerate to RotatE. For example, the case when the coefficients of $j$ and $k$ are fixed as 0 for all quaternions in all embeddings, and the modulus of all quaternions in relation embeddings are fixed as 1. Before QRSE, QuatE has already generalized ComplEx through replacing the complex numbers with quaternions. However, QuatE only takes advantage of that quaternions are more expressive than complex numbers. While our method not only leverages the expression advantage but also exploits the noncommutative property of quaternion multiplication to model the noncommutative composition patterns. The results of experiments on real-world datasets show that QRSE reaches the state-of-the-art on the link prediction problem. For future work, our plan is to combine QRSE with deep models for natural language processing. With its help, we expect deep models to achieve higher accuracy on question answering tasks and make the model’s answers more interpretable.

Author Contributions: Conceptualization, C.X., C.F., D.C. and X.H.; methodology, C.X.; software, C.X.; validation, C.X. and C.F.; formal analysis, C.X.; investigation, C.X.; resources, D.C. and X.H.; data curation, C.X.; writing—original draft preparation, C.X.; writing—review and editing, C.X., C.F., D.C. and X.H.; visualization, C.X.; supervision, D.C. and X.H.; project administration, D.C. and X.H.; funding acquisition, D.C. and X.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by The National Nature Science Foundation of China (Grant Nos. 62273302, 62036009, U1909203, 61936006), in part by Innovation Capability Support Program of Shaanxi (Program No. 2021TD-05).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study and the code are available on request from the corresponding author.
Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References


Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.