A Quaternion-Embedded Capsule Network Model for Knowledge Graph Completion

HENG CHEN\(^1,2\), WEIMEI WANG\(^1\), GUANYU LI\(^1\), AND YIMIN SHI\(^1\)

\(^1\)Faculty of Information Science and Technology, Dalian Maritime University, Dalian 116026, China
\(^2\)Research Center for Language Intelligence, Dalian University of Foreign Languages, Dalian 116044, China

Corresponding author: Guanyu Li (rabitlee@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 61976032, Grant 61806038, Grant 61702072, and Grant 61602076, in part by the National Social Science Foundation of China under Grant 15BYY028, and in part by the Natural Science Foundation of Liaoning Province under Grant 20170540232.

ABSTRACT Knowledge graphs are collections of factual triples. Link prediction aims to predict lost factual triples in knowledge graphs. In this paper, we present a novel capsule network method for link prediction taking advantages of quaternion. More specifically, we explore two methods, including a relational rotation model called QuaR and a deep capsule neural model called CapS-QuaR to encode semantics of factual triples. QuaR model defines each relation as a rotation from the head entity to the tail entity in the hyper-complex vector space, which could be used to infer and model diverse relation patterns, including: symmetry/anti-symmetry, reversal and combination. Based on these characteristics of quaternions, we use the embeddings of entities and relations trained from QuaR as the input to CapS-QuaR model. Experimental results on multiple benchmark knowledge graphs show that the proposed method is not only scalable, but also able to predict the correctness of triples in knowledge graphs and significantly outperform the existing state-of-the-art models for link prediction. Finally, the evaluation of a real dataset for search personalization task is conducted to prove the effectiveness of our model.

INDEX TERMS Capsule network, knowledge graph, knowledge graph completion, link prediction, quaternion.

I. INTRODUCTION

A typical knowledge graph (KG) is usually expressed as a multi-relationship graph, which is a collection of fact triples. A great deal of the facts available in the world can be represented simply as entities and relations between them. YAGO [1], Freebase [2] and DBpedia [3] are usually KGs of triples representing the relations between entities in the form of fact (head entity, relation, tail entity) denoted as \((h, r, t)\), e.g., \((\text{Alaska}, \text{cityOf}, \text{America})\). These KGs are potentially useful to a variety of applications such as recommender systems [4], question answering [5], information retrieval [6], and natural language processing [7]. However, large knowledge graphs, even containing billions of triples, are still incomplete, i.e., missing a lot of valid triples [8], [9]. Therefore, many research efforts have focused on the knowledge completion task which aims to predict missing triples in KGs, i.e., predicting whether a triple not in KGs is likely to be valid or not [10], [11]. Finally, many embedding models have been proposed to learn the low-dimensional representations of entities and relations in KG. These methods have been shown to be scalable and effective [12], [13]. In addition, the accuracy of knowledge graph prediction can be improved using some optimization methods [14]–[17].

Compared with the energy-based models SE and SME, TransE [18] is relatively simple and effective, and achieves better prediction performance on both FB15k and WN18. TransE, inspired by [19], learns low-dimensional representations of entities and relations. TransE views the relation as a translation from the head entity to the tail entity, that is, \(h + r \approx t\) when \((h, r, t)\) holds. Other transition-based models extend TransE to additionally use projection vectors or matrices to translate embeddings of head entity and tail entity into the relation vector space, such as: TransH [20], TransR [21], TransD [22]. These models typically model and infer the connectivity patterns in the knowledge graphs based on the observed facts. For example, some relations
are symmetric (e.g., friend) while others are antisymmetric (e.g., filiation); some relations are the reversal of other relations (e.g., hypernym and hyponym); and some relations may be combined by others (e.g., my father’s wife is my mother). It is important to find methods to model and infer these patterns, i.e., symmetry/anti-symmetry, reversal and combination from the real facts in order to predict missing triples. TransE views relationships as translations in order to model the reversal and combination patterns. DisMult [23] can model the three-way interactions between head entities, relationships, and tail entities, in order to infer the symmetry pattern. ComplEx [24] introduces complex-valued embeddings to better model asymmetric relations, but it cannot infer the combination pattern. RotatE [13] models these patterns by using complex spaces. Specifically, the RotatE model defines each relationship as a rotation from the head entity to the tail entity in the complex vector space. In this paper, we use the quaternion to model and infer all the three types of relation patterns. A complex number traditionally consists of a real part and an imaginary part, while a quaternion has a real part and three imaginary parts. This hyper-complex embedding is used to represent entities, and relations are modeled as rotations in the hyper-complex vector space. Quaternions enable expressive rotation in the four-dimensional space and have more degrees of freedom than rotation in complex plane [25].

The traditional embedding models mentioned above, such as TransE, DisMult and ComplEx, use addition, subtraction or simple multiplication operators, thus only capture the linear relationships between entities [11]. Recently, deep neural networks have been applied to knowledge graph completion tasks. For example, ConvKB [26] is a KG completion model based on convolutional neural network (CNN) and obtains better experimental results. Most knowledge graph embedding models can model entries of the same dimensions and relationships, capturing some specific information about entities and relationships. However, existing embedding models do not employ a “deep” architecture to model attributes of triples of the same dimension [11]. CapSNet [27] introduces capsule networks that employ capsules to capture entities in the image and then use a routing process to specify the connection from the upper capsule to the next capsule. Therefore, the capsule network can not only extract features based on statistical information, but also interpret features, thereby getting rid of the constraint that the convolutional neural network does not have the powerful ability to discern the position and direction of entities and relationships [27]. The capsule network requires the model to learn the feature variables in the capsule and retain valuable information to the maximum. Therefore, it can infer the final variables using a small amount of training data and achieve the expected effect of the experiment faster [28]. Compared with convolutional neural networks, the advantages of capsule networks are: a) In traditional convolutional neural networks, each value of the convolutional layer is the result of a linear weighted summation. But for a capsule network, each value is a vector including direction, state, and other characteristics of the object [27]. Therefore, compared to neurons, capsules can encode many characteristics in the embedding triples to represent the entries at the corresponding dimension. b) Since each layer in a convolutional neural network requires the same convolution operation, it requires a large amount of network data to learn features. The capsule network can learn the feature variables in the capsule and retain valuable information to the maximum [28]. Therefore, it can infer the feature variables with less training data and achieve the expected effect of CNN. c) Convolutional neural network loses a lot of information during the pooling process, so it cannot deal with the blurriness of images well. Unlike CNN, each capsule in the capsule network will carry a large amount of information, which will be saved throughout the network [27]. Therefore, the capsule network can handle the blurriness of images well. To this end, we introduce CapS-QuaR to explore a novel application of CapsNet on triple-based data for two problems: KG completion and search personalization. Different from the traditional CapsNet model, we use capsules to model the entries at the same dimension in the entity and relationship embeddings.

Recently, the combination of quaternions and neural networks has received more and more attention because quaternion numbers allow neural network-based models to code latent inter-dependencies between groups of input features during the learning process with fewer parameters than CNNs. In particular, the deep quaternion network [29], [30], the deep quaternion convolutional network [31], [32], or the deep quaternion recurrent neural network [33] has been employed for challenging tasks such as images and language processing. However, there is still a lack of investigations on the combination of the quaternions and the capsule networks, so this is a topic worth studying. In this paper, we propose CapS-QuaR, a knowledge graph completion model that combines quaternions and capsule networks. The quaternion embeddings of entity and relationship viewed as input in our CapS-QuaR, are more expressive than the traditional real number embedding.

Specifically, our contributions are as follows:

- We introduce a simple, capsule link prediction model embedded in quaternion called CapS-QuaR. To the best of our knowledge, CapS-QuaR model uses the capsule network to replace the traditional neural network and uses the quaternion as the input of the model, which effectively improves the accuracy of the knowledge graph completion.
- In addition, we also propose a knowledge graph embedding model QuaR that uses the quaternion as a relational rotation. A quaternion is an extension of a complex number in a higher dimension. A quaternion represented as a vector not only has no redundant information but can represent all numbers.
- We evaluate our models and several proposed models for knowledge graph completion on two benchmark datasets WN18RR and FB15k-237. To prove the effectiveness of our model on real dataset, we also evaluated...
II. RELATED WORK AND OUR MOTIVATION

A. RELATED WORK

Inspired by the idea of word2vec translation invariant, TransE [18], a well-known knowledge graph embedding model that represents relations as translations, aims to model the reversal and combination patterns. In addition, TransE cannot model the symmetry relation pattern because it would yield \( r = 0 \) for symmetric relations. TransE hopes that \( h + r \approx t \) when \((h, r, t)\) holds. Therefore, the tail embedding \( t \) should be the nearest neighbor of \( h + r \). The score function of TransE is defined as follows:

\[
f_r(h, t) = \|h + r - t\|^2_2
\]

The lower the score function, the more correct the triple. However, TransE encounters issues when modeling one-to-many and many-to-one relations. TransH [20] is proposed to solve the problem of TransE by modeling relations as hyperplanes and projecting \( h \) and \( t \) to the relational-specific hyperplane, allowing entities playing different roles in different relationships [34]. The score function of TransH is defined as follows:

\[
f_r(h, t) = \| (h - w^T_h h_r) + r - (t - w^T_t t_w) \|^2_2
\]

Both TransE and TransH assume that entity and relation are in the same vector space. In order to address the issue, TransR [21] defines the corresponding mapping matrix for each relationship to map the entity embeddings into relation vector space. The score function of TransR is defined as follows:

\[
f_r(h, t) = \| h M_r + r - t M_r \|^2_2
\]

The above models have been trying to either implicitly or explicitly model one or a few of the three relation patterns. For example, TransE can implicitly model reversal and combination of relationships, but it cannot model symmetric relationships. TransX can model the symmetry/antisymmetry pattern when \( M_r = M'_r \), e.g. in TransH, but cannot infer reversal and combination as \( M_r' \) are invertible matrix multiplications. Due to the symmetrical nature of DistMult, it cannot model and infer asymmetric and reverse pattern. ComplEx addresses the problem that DistMult can’t infer symmetry and asymmetric relation patterns by using complex embeddings. However, ComplEx cannot infer combination pattern, since it does not model a bijection mapping from \( h \) to \( t \) via relation \( r \). RotatE [13] defines relations as rotations in the complex vector space. It can effectively model all the three relation patterns: symmetry/anti-symmetry, reversal and combination. The reasoning work of RotatE modeling three relationship patterns can be found in the literature [13]. Then the score function of RotatE is defined as follows:

\[
f_r(h, t) = -\|h \circ r - t\|^2_2
\]

Different from the above embedding models, ConvE [35] is the first model to apply CNN for the KG completion task. In ConvE model, the head entity and relation embeddings are reshaped into an input matrix which is fed to the convolution layer. The interactions between entities and relationships are modeled through convolutional layers and fully connected layers. In addition, ConvE can use CNNs to efficiently train triples to obtain embedding representations of entities and relationships, and it can also learn more features of triples. The score function of ConvE is defined as follows:

\[
f_r(h, t) = g(vec(g(concat(h, r))W) \cdot t
\]

ConvE focuses solely on the local relationships among different dimensional entries in each of \( h \) or \( r \), i.e., ConvE does not observe the global relationships among same dimensional entries of an embedding triple \((h, r, t)\), so that ConvE ignores the transitional characteristic in transition-based models, which is one of the most useful intuitions for the task. To solve this problem, ConvKB [26] takes the embedding triple \((h, r, t)\) as the input and captures the global characteristic of the triple. The score function of ConvKB is defined as follows:

\[
f_r(h, t) = concat(g([h, r, t] \ast \Omega)) \cdot w
\]

ConvKB uses CNN to encode the triples in the knowledge graph, but the linear operations between neurons are too simple to characterize the deep information of entities and relationships. Inspired by [27], CapsE [11] applies a capsule network to knowledge graph to learn the vector embeddings of entities and relationships. Hinton et al. [27] proposed CapsNet, which employs capsules to capture entities in an image and then uses a routing operation to specify the connection from the previous capsule layer to the next capsule layer. Therefore, CapsNet could be used for encoding the intrinsic spatial relationship between the parts and the whole, which constitute the viewpoint. In the process, knowledge is automatically generalized into novel viewpoints. Each capsule accounts for capturing variations of an object or object part in the image, which can be efficiently visualized. Our high-level hypothesis is that embedding entries at the same dimension of the triple also have these variations, so we introduce capsules into the knowledge graph. The CapsNet architecture is shown in Fig. 1. According to [11], we introduce Caps-QuaR to explore a novel application of CapsNet on triple-based data for two problems: KG completion and search personalization.

Unlike CapsNet’s traditional modeling design (CapsNet constructs capsules by splitting feature maps), CapsE uses capsules to model the entries of the same dimensions in entity and relationship embeddings. Therefore, CapsE introduces a
capsule neural network-based knowledge graph completion model, which uses capsules instead of neurons to operate triples, and has achieved good results in experiments. The score function of CapsE is defined as follows:

\[ f_r(h, t) = \|capsnet(g([h, r, t] \ast \Omega))\| \]  

(7)

CapsE employs the embeddings of entities and relationships trained from the TransE, but there are still differences in representing external dependencies between different entities. Therefore, Caps-QuaR uses quaternions to encode the sequence of triples to capture the structural information of the triples as much as possible, so as to complete the missing triples in the knowledge graph. In addition, our experimental results show that Caps-QuaR is better than CapsE in search personalization tasks.

Table 1 summarizes several state-of-the-art knowledge graph completion models, including the score functions and models’ parameters. In summary, TransE, DisMult and ComplEx cannot effectively model all three types of relationship patterns. Zhang et al. [25] shows that quaternions enable expressive rotation in four-dimensional space and have more degree of freedom than rotation in complex plane. These facts show the model concerning the quaternions is advantageous with its capability in modelling several key relation patterns. So the QuaR model proposed in this paper uses quaternions instead of traditional complex numbers to model three relation patterns: symmetry/anti-symmetry, reversal and combination.

### B. OUR MOTIVATION

In this paper, we propose QuaR to express entities and relationships with quaternions. We make full use of the expressive rotational ability of quaternions. The RotatE model has only one plane of rotation (complex plane), as shown in Fig. 2(a), while QuaR has two planes of rotation. In Fig. 2(c), QuaR models \( r \) as a rotation in the hyper-complex vector space. Flexibility to represent entities and relations with quaternions is higher than complex planes. In addition, quaternions represent entities and relationships more efficiently than rotation matrices, and values are more stable. Our motivation is from quaternion extension of Euler’s Formula:

\[ e^{\theta/2(u_i,i+u_j+j+u_k)} = \cos(\theta/2) + (u_i,i + u_j+j + u_k,k) \sin(\theta/2), \]

unit vector \( u = (u_i, u_j, u_k) \), which indicates that a unitary hyper-complex number can be regarded as a rotation in the hyper-complex space. Specifically, the QuaR model maps the entities and relations to the hyper-complex vector space \( H \) and defines each relation as a rotation from the head entity to the tail entity. Given a triple \((h, r, t)\), we expect that \( t \approx h \circ r \), where \( h, r, t \in H \) are the quaternion embeddings of entity and relation, the modulus \(|r_1| = 1\) and \( \circ \) denotes the Hadamard (or element-wise) product. 

To verify whether the three relation patterns are implicitly represented by QuaR relationship embeddings, we state as follows. Symmetry pattern requires the symmetric relations to have property \( r \circ r = 1 \), and the solution is \( r_1 = \pm 1 \). We investigate the relation embeddings from a 500-dimensional QuaR trained on WN18RR. The study found that the embedding phases were either \( \pi(r_1) = -1 \) or \( \pi(r_1) = 0 \) or \( 2\pi(r_1) = 1 \). Reversal pattern requires the embeddings of a pair of reverse relations to be conjugate. We use the same QuaR model trained on WN18 for an analysis. It indicates that the additive embedding phases are 0 or \( 2\pi \), which represents that \( r_1 = r_2^{-1} \). Combination pattern requires the embedding phases of the combined relation to be the addition of the other two relations. We investigate the relation embeddings from a 1000-dimensional QuaR trained on FB15k-237. The result shows such a \( r_1 = r_2 \circ r_3 \) case, where \( \theta_1 = \theta_2 + \theta_3 \). 

It turns out that QuaR model can effectively model all the three relation patterns: symmetry/anti-symmetry, reversal and combination. For example, a relation \( r \) is symmetric if and only if each element of its embedding \( r \), i.e., \( r_i \) satisfies \( r_i = e^{\theta/2(u_i,i+u_j+j+u_k,k)} = \pm 1 \); two relations \( r_1 \) and \( r_2 \) are reverse if and only if their embeddings are conjugates: \( r_2 = r_1^{-1} \); a relation \( r_3 = e^{\theta_3/2(u_i,i+u_j+j+u_k,k)} \) is a combination of other two relations \( r_1 = e^{\theta_1/2(u_i,i+u_j+j+u_k,k)} \) and \( r_2 = e^{\theta_2/2(u_i,i+u_j+j+u_k,k)} \) if and only if \( r_3 = r_1 \circ r_2 \) (i.e. \( \theta_3 = \theta_1 + \theta_2 \)). Moreover, the QuaR model is scalable to large knowledge graphs as it remains linear in both time and memory [13].

### TABLE 1. Score functions of state-of-the-art knowledge graph embedding models.

![FIGURE 1. A simple CapsNet with 3 layers. A 28 x 28 image is introduced into the model and we obtained a vector containing 10 neurons, i.e., a capsule.](image)

| Models | Score Functions | Parameters |
|--------|-----------------|------------|
| TransE | \( \|h + r - t\|^2 \) | \( h, r, t \in R^k \) |
| TransX | \( \|M_r(h) + r - M_r(t)\|^2 \) | \( h, r, t \in R^k \) |
| DistMult | \( \langle h, r, t \rangle \) | \( h, r, t \in R^k \) |
| ComplEx | \( Re(h, r, t) \) | \( h, r, t \in C^k \) |
| RotatE | \( \|h \circ r - t\|^2 \) | \( h, r, t \in C^k \), \( r \in \{1, -1\} \) |
| ConvE | \( g(vec(g(comcat(h, r) \ast \Omega))W) \ast t \) | \( h, r, t \in R^k \) |
| QuaR | \( h \circ r - t \) | \( h, r, t \in H \) |

\( \circ \) denotes the Hadamard (or element-wise) product. \(+\) denotes 2D convolution.

TransX represents a wide range of TransE's variants, such as TransH [20], TransR [21] and STTransE [36], where \( M_r(.) \) and \( M_r(.) \) denote a matrix multiplication with respect to relation \( r \).
We use the quaternion embeddings of entities and relationships trained from the QuaR model as the input to CapS-QuaR. Inspired by the CapsNet model, we employ a capsule network instead of the traditional convolutional neural network to capture the semantic information of entities and relationships in triples. To this end, we introduce CapS-QuaR to explore a novel application of CapsNet on triple-based data to solve the problem of knowledge graph completion. Unlike CapsNet’s traditional modeling design, we use capsules to model the same dimension entries in the entity and relationship embeddings. In our CapS-QuaR, \( h, r \) and \( t \) are unique \( K \)-dimensional quaternion embeddings of \( h, r \) and \( t \) respectively. The embedding matrix \([h, r, t]\) of the triple \((h, r, t)\) is fed to the convolution layer for convolution. In the convolutional layer, multiple filters are repeatedly operated over every row of the matrix to generate \( K \)-dimensional feature maps. Then, entries of the same dimension in all feature maps are encapsulated into capsules. In this way, each capsule can encode multiple features in the embedding triple to represent the entries of the corresponding dimension. These capsules are routed to obtain another capsule of smaller dimension, which outputs a continuous vector, and the value after the dot product of the weight vector is used as the score of the triple. Finally, the score is used to determine whether the triple \((h, r, t)\) is correct.

III. METHODOLOGY

A. QuaR: RELATION AS ROTATION IN HYPER-COMPLEX VECTOR SPACE

In this section, we introduce the QuaR model proposed in this paper. We first introduce three important relation patterns that the QuaR model can handle. Afterwards, we introduce Hamilton’s Quaternions and quaternion representations for knowledge graph completion. Finally, we introduce our proposed QuaR model, which defines relations as rotations in the hyper-complex vector space.

To better express the three relation patterns in knowledge graph completion with formulas, we define them as follows:

**Definition 1:** A relation \( r \) is symmetric (antisymmetric) if \( \forall e_i, e_j \)

\[
\begin{align*}
r(e_i, e_j) & \Rightarrow r(e_j, e_i) \quad (9) \\
r(e_i, e_j) & \Rightarrow -r(e_j, e_i) \quad (10)
\end{align*}
\]

**Definition 2:** Relation \( r_1 \) is reverse to relation \( r_2 \) if \( \forall e_i, e_j \)

\[
\begin{align*}
r_1(e_i, e_j) & \Rightarrow r_2(e_j, e_i) \quad (11) \\
r_2(e_i, e_j) & \Rightarrow r_1(e_j, e_i) \quad (12)
\end{align*}
\]

**Definition 3:** Relation \( r_1 \) is composed of relation \( r_2 \) and relation \( r_3 \) if \( \forall e_i, e_j, e_k \)

\[
r_2(e_i, e_j) \land r_3(e_j, e_k) \Rightarrow r_1(e_i, e_k) \quad (13)
\]

where \( e_i, e_j \) and \( e_k \) denote different entities in the knowledge graph.

The above is the definition of three relation patterns. We use quaternions instead of traditional complex numbers to model entities and relationships to complete the missing triples in the knowledge graph. In addition, the introduction of Hamilton’s Quaternions is shown below. A quaternion is a simple super complex number. The complex number consists of a real number plus an imaginary unit \( i \), where \( i^2 = -1 \). Similarly, quaternions are composed of real numbers plus three imaginary units \( i, j, k \), and they have the following relations: \( i^2 = j^2 = k^2 = -1, i^0 = j^0 = k^0 = 1 \), each quaternion is a linear combination of \( 1, i, j, k \), that is, a quaternion can generally be expressed as \( Q = a + bi + cj + dk \), where \( a, b, c, d \) are real numbers and \( i, j, k \) are imaginary units. In addition, \( ij = k, ji = -k, jk = i, ki = j, kj = -i, ik = -j \). Fig. 2(b) shows the quaternion imaginary units product.

Other important operational rules for quaternions are described below:

**Conjugate:** The conjugate of a quaternion \( \bar{Q} \) is defined as \( \bar{Q} = a - bi - cj - dk \).

**Quaternion Norm:** The norm of a quaternion is defined as \( |Q| = |Q| = \sqrt{a^2 + b^2 + c^2 + d^2} \).

**Quaternion Addition:** The quaternion addition between \( Q_1 = a + bi + cj + dk \) and \( Q_2 = a' + b'i + c'j + d'k \) is obtained:

\[
\begin{align*}
Q_1 + Q_2 & = (a + a') + (b + b')i + (c + c')j + (d + d')k
\end{align*}
\]

**Quaternion Product:** The quaternion product between \( Q_1 = a + bi + cj + dk \) and \( Q_2 = a' + b'i + c'j + d'k \) is obtained:

\[
\begin{align*}
Q_1Q_2 & = (aa' - bb' - cc' - dd') + (ab' + ba' + cd' - dc')i + (ac' - bd' + ca' + db')j + (ad' + bc' - cb' + dd')k
\end{align*}
\]

**Quaternion Division:** The quaternion division between \( Q_1 = a + bi + cj + dk \) and \( Q_2 = a' + b'i + c'j + d'k \) is
obtained: \( Q_1/Q_2 = (aa' - bb' - cc' - dd') + (ab' + ba' - cd' - dc')i + (ac' + ad' + ca - db')j + (ad' + be' + cb' - da')k. \)

Inspired by quaternion extension of Euler’s formula, we map the head and tail entities \( h, t \) to the hyper-complex embeddings, i.e., \( h, t \in \mathbb{H}^8 \); then each relationship \( r \) is defined as an element-by-element rotation from the head entity \( h \) to the tail entity \( t \). Given a triple \((h, r, t)\), we expect that as follows:

\[
t \approx h \circ r, \quad \text{where } |r| = 1
\]

and \( \circ \) is the Hadamard product. For each element in the embeddings, we have \( t_i = h_i r_i \). According to the above definition, for each triple \((h, r, t)\), we define the objective function of QuaR as follows:

\[
f_r(h, t) = \| h \circ r - t \| \quad \text{(15)}
\]

Negative sampling has been proved quite effective for both learning graph embedding [24] and word embedding [19]. Here we use a loss function similar to the negative sampling loss [19] to effectively optimize distance-based models. The loss function is described as follows:

\[
L = -\log \sigma(\gamma - f_r(h, t)) - \frac{1}{k} \sum_{i=1}^{n} \log \sigma(f_r(h_i^r, t_i^r) - \gamma) \quad \text{(16)}
\]

where \( \gamma \) is a fixed margin hyper-parameter, \( \sigma \) is the sigmoid function, and \((h_i^r, t_i^r)\) is the \( i \)-th negative triple.

By defining each relationship as a rotation in the hyper-complex vector spaces, QuaR can model and infer all the three types of relation patterns, as shown in Table 2. Formally, we have the following results:

**Lemma 1:** QuaR can infer and model the symmetry or anti-symmetry pattern.

**Lemma 2:** QuaR can infer and model the reversal pattern.

**Lemma 3:** QuaR can infer and model the combination pattern.

**Lemma 4:** QuaR can degenerate into TransE.

In the proofs below, \( \circ \) represents the Hadamard (or element-by-element) product, \((\cdot)^{-1}\) indicates the reversal of a relationship.

**Proof of Lemma 1:**

**Proof:** If \( r(e_i, e_j) \) and \( r(e_j, e_i) \) hold, we have

\[
e_j = r \circ e_j \land e_i = r \circ e_j \Rightarrow r \circ r = 1 \quad \text{(17)}
\]

Otherwise, if \( r(e_i, e_j) \) and \(-r(e_j, e_i)\) hold, we have

\[
e_j = r \circ e_j \land e_i \neq r \circ e_j \Rightarrow r \circ r \neq 1 \quad \text{(18)}
\]

**Proof of Lemma 3:**

**Proof:**

If \( r_1(e_i, e_j) \) and \( r_2(e_j, e_i) \) hold, we have \( e_j = r_1 \circ e_i \land e_j = r_2 \circ e_i \Rightarrow r_1 = r_2^{-1} \) \( \text{(19)} \)

**Proof of Lemma 4:**

**Proof:**

By further restricting \(|h_i| = |t_i| = C\), we can rewrite \( h, r, t \) by

\[
h = Ce^{\theta_h (u_iu_j + u_k)} = C \cos \theta_h^2 + (u_iu_j + u_k) \sin \theta_h \quad \text{(21)}
\]

\[
r = e^{\theta_r (u_iu_j + u_k)} = \cos \theta_r^2 + (u_iu_j + u_k) \sin \theta_r \quad \text{(22)}
\]

\[
t = Ce^{\theta_r (u_iu_j + u_k)} = C \cos \theta_r^2 + (u_iu_j + u_k) \sin \theta_r \quad \text{(23)}
\]

Therefore, we have

\[
\| h \circ r - t \| = \| e^{\theta_h} - e^{\theta_r} \| = C\| \cos \theta_h^2 + (u_iu_j + u_k) \sin \theta_h \| \quad \text{(24)}
\]

If the embedding of \((h, r, t)\) in TransE is \( h', r', t' \), let \( \theta_h = ch', \theta_r = cr', \theta_t = ct' \) and \( C = 1/c \), we have

\[
\lim_{c \to 0} \| h \circ r - t \| = \| h' + r' - t' \| \quad \text{(25)}
\]

**B. CapS-QuaR: Capsule Neural Model for Knowledge Graph Completion**

Since the QuaR model has fewer parameters and lower computational complexity than RotatE, it is relatively simple to train the triple. We employ the entity and relation embeddings trained from QuaR model as the input to the CapS-QuaR model, as shown in Fig. 3. In addition, unlike traditional embedding representations, we use the quaternions to represent the embeddings of entities and relationships that will be more accurate and visual. We define the knowledge graph as: \( KG = (E, R, S) \), \( E \) is a collection of entities, \( R \) is a relation set, and \( S \) is all datasets, including training sets, test sets, and
validation sets. The embedding dimensions of entities and relationships are \( K \).

We define \( h, r \) and \( t \) as \( K \)-dimensional quaternion embeddings of \( h, r \) and \( t \), respectively. In CapS-QuaR, we define the embedding triple \((h, r, t)\) as a matrix \( A = [h, r, t] \in \mathbb{H}^{K \times 3} \). And \( A_i \in \mathbb{H}^{1 \times 3} \) denotes the \( i \)-th row of \( A \). We use the filter \( \omega \in \mathbb{H}^{1 \times 3} \) operated on the convolution layer. \( \omega \) is repeatedly operated over every row of matrix \( A \) to generate the feature map \( q = [q_1, q_2, \ldots, q_k] \in \mathbb{H}^K \) in which \( q_i = g(\omega \cdot A_i + b) \), where \( \cdot \) denotes a dot product, \( b \in \mathbb{H} \) is a bias term and \( g \) is a non-linear activation function such as ReLU or Sigmoid.

In the convolution layer, we use Quaternion Addition to calculate the values of \( K \) capsules.

The convolution layer operation is described above, and a capsule network algorithm is constructed using the capsule layer to simplify the architecture. First, we use the feature maps obtained by convolution to reconstruct \( K \) capsules, wherein entries at the same dimension from all feature maps are encapsulated into a corresponding capsule. A capsule is a set of neurons that represent the instantiation parameters of a specific type of entity in a vector, which can represent various features of a particular entity or a specific relationship in the KG. Thus, each capsule can capture different features of the corresponding dimensions embedded in the triples, resulting in another capsule of smaller dimensions. These capsules are eventually generalized into one capsule in the second layer, which produces a vector for dot product operation with the weight vector \( W \in \mathbb{H}^{d \times 1} \), and the value of the dot product is used as a score of the triple. The lower the score, the more correct the triple. In the capsule layer, we use Quaternion Product, Quaternion Addition and Quaternion Norm to calculate the value of the capsule.

We explain our proposed model in Fig. 3(b) where embedding size: \( K = 5 \), the number of filters: \( N = 5 \). In the first capsule layer, the number of neurons in all capsules is equal to \( N \), and the number of neurons in the second layer of capsules: \( d = 2 \). The first capsule layer contains \( K \) capsules, for which each capsule \( i \in \{1, 2, \ldots, K\} \) has a vector output \( \mathbf{v}_i \in \mathbb{H}^{N \times 1} \).

Algorithm 1 Routing Algorithm

| Output: \( c = (c_1, c_2, \ldots, c_i, \ldots) \) |
|------------------------------------------------|
| \( b_i \leftarrow 0 \) |
| for all capsule \( i \) in the first layer do |
| \( b_i \leftarrow \text{softmax}(b) \) |
| for iteration = 1, 2, \ldots k do |
| \( c \leftarrow \text{softmax}(b) \) |
| \( s \leftarrow \sum_{i=1}^{k} c_i \mathbf{v}_i \) |
| \( e \leftarrow \frac{s}{1 + \|s\|} \) |
| for all capsule \( i \) in the first layer do |
| \( b_i \leftarrow b_i + \mathbf{v}_i \cdot e \) |
| end |
| end |

Vector outputs \( \mathbf{v}_i \) are multiplied by weight matrices \( W_i \in \mathbb{H}^{d \times N} \) to produce vectors \( \mathbf{v}_i \in \mathbb{H}^{1 \times 1} \) which are summed to produce a vector input \( \mathbf{e} \in \mathbb{H}^{d \times 1} \) to the capsule in the second layer. Then, the capsule generates a vector output \( e \in \mathbb{H}^{d \times 1} \) by performing a non-linear squashing function. Finally, vector output \( e \) is multiplied by the weight matrix \( W \in \mathbb{H}^{d \times 1} \) to produce a score that is used to determine the correctness of a given triple. The above operation process is shown by the (26) and (27).

\[
\mathbf{v}_i = W_i \times \mathbf{v}_i; \quad s = \sum c_i \mathbf{v}_i \quad (26)
\]

\[
e = \frac{s}{1 + \|s\|}; \quad f = e \cdot W \quad (27)
\]

where \( c_i \) are the coupling coefficient determined by the routing process, as shown Algorithm 1.

The detailed optimization procedure is described in Algorithm 2. In CapS-QuaR, we employ QuaR-trained triples matrices to initialize entity and relationship embeddings (line 15). At each main iteration of the algorithm, we employ convolution operations and dynamic routing to train the matrix. First, we sample a mini-batch \( b \) from the training
Algorithm 2 CapS-QuaR Model Algorithm

1. QuaR Input: Training set \( S = (h, r, t) \), entities and relation sets \( E \) and \( R \), margin \( \gamma \), embeddings dim. \( K \).
2. Initialize: \( r \leftarrow \text{uniform}(\frac{-\sqrt{2}}{K}, \frac{\sqrt{2}}{K}) \) for each \( r \in R \).
3. \( e \leftarrow \text{uniform}(\frac{-\sqrt{2}}{K}, \frac{\sqrt{2}}{K}) \) for each \( e \in E \).
4. /* There are different iterations for different datasets. */
5. for \( \text{iteration} = 1, 2, ..., N \) do
6. \( e \leftarrow \frac{e}{\|e\|} \) for each \( e \in E \).
7. /* Sample a minibatch \( b \) */
8. \( S_b \leftarrow \text{sample}(S, b) \).
9. /* Initialize the set of pairs of triples */
10. \( T_b \leftarrow \emptyset \).
11. for \( (h, r, t) \in S_b \) do
12. /* Sample a corrupted triple */
13. \( (h', r', t') \leftarrow \text{sample}(S_{(h, r, t)}) \).
14. \( T_b \leftarrow T_b \cup \{(h, r, t), (h', r', t')\} \).
15. end
16. Update embeddings w.r.t:
17. \[-\frac{1}{K} \sum_{i=1}^{n} \log(f_i(h_i, t'_i) - \gamma) - \log(\gamma - f_i(h, t)) \]
18. /* Input triplet matrix trained by QuaR */
19. CapS-QuaR Input: \( \left[ [h, r, t] \in H^{K \times 3} \right] \)
20. for \( \text{iteration} = 1, 2, ..., N \) do
21. /* Iterative operation 7-12 lines */
22. end
23. Calculate \( f_r(h, t) = \|\text{caps}(g([h, r, t] * \Omega))\| \cdot W \)
24. Update loss function w.r.t:
25. \[ \sum_{(h, r, t) \in T_b} \ln(1 + \exp(\theta \cdot f_r(h, t))) + \lambda \|W\|_2^2 \]
26. end

set \( S \) (line 7). For each such triple, we then sample a single corrupted triple (line 10). Finally, score prediction and loss correction are conducted on these mini-batches (line 19 and line 21 respectively). The parameters are then updated by using the Adam optimizer. The algorithm is stopped based on its performance on a validation set.

Formally, we define the score function \( f \) for CapS-QuaR as follows:

\[ f_r(h, t) = \|\text{caps}(g([h, r, t] * \Omega))\| \cdot W \]  
(28)

where the set of filters \( \Omega \) and \( W \) are shared hyper-parameters in the convolution layer; \( * \) indicates a convolution operator; and the capsule network operator is denoted by \( \text{caps} \). The activation function is ReLU. We minimize the loss function \( L \) as the final training goals, shown as follows:

\[ L = \sum_{(h, r, t) \in \{T \cup T'\}} \ln(1 + \exp(\theta \cdot f_r(h, t))) + \lambda \|W\|_2^2 \]  
(29)

where \( \lambda \) is the weight of the regularization term \( \|W\|_2^2 \).

TABLE 3. Statistics of the datasets used in this paper.

| Datasets     | #entity | #relation | #training | #validation | #test  |
|--------------|---------|-----------|-----------|-------------|-------|
| WN18RR       | 40,943  | 11        | 86,835    | 3,034       | 3,134 |
| FB15k-237    | 14,541  | 237       | 272,115   | 17,535      | 20,466|

In order to prevent the model from overfitting, we employ the \( L_2 \) regularization on the weight vector \( W \). In addition, \( T \) and \( T' \) are collections of correct and corrupted triples, respectively, and the corrupted triples are generated by corrupting the correct triples. The value of \( \theta \) depends on whether the triple is correct or not, shown as follows:

\[ \text{in which, } (h, r, t) \in \begin{cases} T, & \theta = -1 \\ T', & \theta = 1 \end{cases} \]  
(30)

A set of corrupted triples constructed according to (31), is composed of training triples with the head entity or tail entity replaced by a random entity.

\[ T' = \{(h, r, t) | h' \in E \} \cup \{(h, r, t) | t' \in E \} \]  
(31)

We employ the Adam optimizer [37] to train CapS-QuaR and use ReLU as the activation function of the model.

IV. EXPERIMENTS

A. LINK PREDICTION EVALUATION

1) EXPERIMENTAL SETUP

a: DATASETS

We evaluate our proposed models on two benchmark datasets: WN18RR [35] and FB15k-237 [38]. The statistics of WN18RR and FB15k-237 are summarized into Table 3. In addition, FB15k is a subset of Freebase, and its main relation patterns are symmetry/anti-symmetry and reversal. WN18 is a subset of WordNet, and its main relation patterns are symmetry/anti-symmetry and reversal.

Since WN18 and FB15k contain many reversible relations, the existence of these relations will significantly improve the experimental results. Therefore, WN18RR and FB15k-237 are created to solve this reversible relation problem in WN18 and FB15k, for which the knowledge graph completion task will be more realistic.

WN18RR [35] is a subset of WN18 [18] which contains 93,003 triples with 40,943 entities and 11 relations. On WN18RR, one of the main relation patterns is the symmetry pattern since almost each word has a symmetric relation in WN18RR, e.g., also see and similar to [13].

FB15k-237 [38] is a subset of FB15k [18] which contains 310,116 triples with 14,541 entities and 237 relations. On FB15k-237, the main relationship patterns are symmetric/anti-symmetric and combination, such as nationality and place_of_birth [13].
**b: EVALUATION SETTINGS**

Link prediction aims to predict the missing entity given a relationship and another entity, i.e., inferring a head entity \( h \) given \((r, t)\) or inferring a tail entity \( t \) given \((h, r)\). The results could be obtained from ranking the scores calculated by the score function on test triples.

In testing phase, we use the same ranking procedure as in TransE. For each triple in the test set, we replace the head or tail entity with all entities to create a set of corrupted triples. These corrupted triples use the score function \( f_r(h, t) \) to calculate the similarity scores, and then rank them. We rank the correct test triples and corrupted triples in descending order of their scores. In fact, a corrupted triple may also exist in knowledge graphs, which should be also considered correct. Therefore, we evaluate the performance of link prediction in the filtered setting: we rank test triples against all other candidate triples not appearing in the training, validation, or test set, where candidates are generated by corrupting head entities or tail entities: \((h’, r, t)\) or \((h, r, t’)\). For evaluation, we use three evaluation metrics used across the link prediction literature: Mean Rank (MR), Mean Reciprocal Rank (MRR) and Hits at N (H@N). MR measures the average rank of all correct triples with lower value representing better performance. MRR is the average inverse rank for correct triples. H@N measures the proportion of correct triples in the top N triples. Lower MR, higher MRR or higher H@N mean better performance. Final scores on the test set are reported for the model obtaining the highest H@N on the validation set.

**c: TRAINING SETTINGS**

*QuaR:* We use Adam [37] as the optimizer and fine-tune the hyper-parameters on the validation dataset. We implemented our model using PyTorch [38] and tested it on a single GPU.

We train QuaR for 3,000 epochs, using a grid search of hyper-parameters: the dimensionality of embeddings \( k \in \{125, 250, 500, 1000\} \), batch size \( b \in \{256, 512, 1024\} \), the learning rate \( \lambda \in \{0.00001, 0.00005, 0.0001, 0.0005\} \) and fixed margin \( \gamma \in \{3, 6, 9, 12, 18\} \). The real and imaginary parts of the entity embeddings are uniformly initialized, and the phases of the relationship embedding are uniformly initialized between 0 and 2\( \pi \). No regularization is used since we find that the fixed margin \( \gamma \) could prevent our model from over-fitting. The highest Hits@10 scores for our QuaR on the validation set are obtained when using \( k = 500, b = 512, \gamma = 6 \) and the initial learning rate at \( 5e^{-5} \) on WN18RR; and \( k = 1000, b = 1024, \gamma = 9 \) and the initial learning rate at \( 1e^{-5} \) on FB15k-237.

*CapsS-QuaR:* We employ the QuaR model and the ConvKB [26] model to implement CapsS-QuaR. We use entity and relation embeddings produced by QuaR to initialize entity and relation embeddings in CapsS-QuaR. We use the pre-trained 100-dimensional embeddings of entities and relations to train the QuaR model on WN18RR and FB15K-237. All algorithms are implemented in Python, and all experiments are conducted on a server with 1755 MHz 23 GD6 GeForce RTX 2080 Ti GPU and 64 GB memory.

For ConvKB, the number of filters is set to \( |\omega| = N \in \{50, 100, 200, 400\} \), the Adam initial learning rate in \( \gamma \in \{0.00001, 0.00005, 0.0001, 0.0005\} \), and the model is searched 3000 times by using the hyper-parametric grid. For each 100 training epochs, monitor the Hits@10 score to select the optimal hyper-parameters. We finally obtain the highest Hits@10 scores on the validation set when using \( N = 400, K = 100 \) and \( \gamma = 0.00005 \) on WN18RR; and \( N = 50, K = 100 \) and \( \gamma = 0.00001 \) on FB15k-237.

Like ConvKB, we use the embedding of entities and relationships obtained from QuaR to initialize CapsS-QuaR. The batch size and number of neurons within the capsule in the second capsule layer are set as 128 and 8 \((d=8)\) respectively. And the number of iterations in the routing algorithm \( m \) is set to \( \{1, 3, 5, 7, 9\} \). The weight \( W \) is initialized by the truncated normal distribution function and finally determined during the training phase of the CapsS-QuaR. We run CapsS-QuaR up to 500 epochs and monitor the Hits@10 score after each 10 training epochs to select optimal hyper-parameters. The highest Hits@10 scores for our CapsS-QuaR on the validation set are obtained when using \( m = 1, N = 400, K = 100 \) and the initial learning rate at \( 1e^{-5} \) on WN18RR; and \( m = 1, N = 50, K = 100 \) and the initial learning rate at \( 1e^{-4} \) on FB15k-237.

2) **LINK PREDICTION RESULTS**

Link prediction aims to predict the missing head entity or tail entity for a triple. In this task, for the absence of a head entity or tail entity, the system is required to rank a set of candidate entities from the knowledge graph, rather than just giving one optimal result. We compare CapsS-QuaR to several previous state-of-the-models, including TransE [18], DistMult [23], ComplEx [24], ConvKB [26], RotatE [13] and ConvE [35], as well as our quaternion model QuaR, to empirically show the importance of modeling and inferring the missing triples for link prediction task.

Link prediction results on all two datasets are shown in Table 4 and Table 5. We can see that CapsS-QuaR is better than many of the state-of-the-art models. CapsS-QuaR performs better than its closely related embedding model QuaR on both experimental datasets (except MR on WN18RR and MR on FB15k-237), especially on FB15k-237 where our model obtains significant improvements of 0.525 – 0.358 = 0.162 in MRR (which is about 46.6% relative improvement), and 61.8% – 56.0% = 5.8% absolute improvement in Hits@10.

In addition, CapsS-QuaR gains better scores than CapsE on both datasets, thus demonstrating the usefulness of using quaternions to represent entities and relationships. Especially on FB15k-237, CapsS-QuaR achieves improvements of 303 – 238 = 65 in MR (which is about 27% relative improvement) and 61.8% – 59.3% = 2.5% in Hits@10, while both CapsE and ConvKB produce similar MR scores. CapsE also obtains 54.7% relatively higher MRR score than the relation embedding model RotatE on FB15k-237. This
In the dataset FB15k-237, we can see that the quaternion-based embedding model QuaR is superior to all the state-of-the-art models in MR. QuaR outperforms RotatE on all metrics on FB15K-237. The results also demonstrate that QuaR can effectively capture the symmetry, anti-symmetry and combination patterns since they account for a large portion of the relations in these two datasets. In addition, KBAT is better than all models in Hits@10 and second only to QuaR in MR, which can show that KBAT can be effectively used for knowledge graph completion tasks.

In order to confirm the ability of quaternions to model diverse types of relationships, we use different relationships for representation learning on WN18RR. Table 6 summarizes the MRR for each relationship on WN18RR, confirming the superior representation capability of CapS-QuaR in exploring the capsule network to knowledge graph completion.

In this paper, we classify the relationships in the FB15k-237 test set into three categories: symmetry, anti-symmetry and combination. Fig. 4 shows the results of Hits@10 and MRR for predicting head and tail entities w.r.t each relationship category on FB15k-237. CapS-QuaR is superior to CapsE in both head entity prediction and tail entity prediction. This also shows that the combination of quaternion and capsule network can significantly improve the experimental results on the link prediction task. Figs. 5 and 6 show the Hits@10 and MRR scores for each relationship on WN18RR, respectively. As can be seen from Figs. 5 and 6, CapS-QuaR is better than CapsE in many specific relationships, for example: also see, has_part, Instance_hypernym, member_of_domain_usage, member_of_domain_region. We see that the length and orientation of each capsule in the first layer can help model the important entries in the corresponding dimension, So CapS-QuaR can work well on 1-M and M-1 relationship categories on test triples.

3) MODEL ANALYSIS

a: NUMBER OF PARAMETERS COMPARISON

Table 7 shows the amount of parameters comparison between QuaR and RotatE. On WN18RR and FB15K-237, compared to the benchmark model RotatE, the amount of QuaR parameters is reduced by 70.2% and 53%. QuaR reduces the parameter size of the complex-valued counterpart RotatE largely, saving up to 70% parameters while maintaining superior performance.

b: HADAMARD PRODUCT BETWEEN HEAD AND TAIL ENTITIES

We re-ran the experiment using different score functions. It can be seen from Table 8 that on two benchmark datasets WN18RR and FB15K-237, compared to the benchmark model RotatE, the amount of QuaR parameters is reduced by 70.2% and 53%. QuaR reduces the parameter size of the complex-valued counterpart RotatE largely, saving up to 70% parameters while maintaining superior performance.

c: CAPSULE NETWORK MODELING RELATIONSHIP TRIPLES

From Table 9, we can clearly see that CapS-QuaR has better performance than its closely related CapsE model based on the capsule network and has better experimental results on the two datasets WN18RR and FB15K-237. CapS-QuaR uses quaternion-embedded triples as input, so that capsules have certain specific semantic information, which provides important reference information for completing missing
relationships between entities. We convert the head and
tail quaternions into corresponding capsules for processing,
obtain higher-level capsules through routing operations, and
treat the size of the capsules as the probability of determining
whether the triples are correct. Thus, we have
\[ f_r(h, t) = \|caps(g([h, r, t] \ast \Omega))\| \cdot W. \]

B. SEARCH PERSONALIZATION EVALUATION

For a given user, when the user submits a query, the search system returns some query documents. Our method is to
re-rank the returned documents, that is, the higher the relevance, the higher the ranking. Following [43], we represent the submitted query, user and the returned documents as a triple \((query, user, document)\), that is \((h, r, t)\). The triple captures how much a user is interested in a document given a query. Therefore, we can evaluate the effectiveness of our CapS-QuaR for the search personalization task.

1) EXPERIMENTAL SETUP

a: DATASET

We evaluate our proposed CapS-QuaR on the SEARCH17 [43] dataset. In addition, we also added some entities, relationships and triples in SEARCH17. SEARCH17 contains the query logs of 306 anonymous users, provided by a
A log entity contains not only the user identifier, the query, the top 10 query documents, but also the user’s dwell time when the document was clicked. SEARCH17 dataset is divided into three classes: training, validation and test sets. The training, validation, and test sets include 5,658, 1,184, and 1,860 valid triples; and 40,239, 7,882, and 8,540 invalid triples, respectively. The three relation patterns are typical in the SEARCH17 dataset. Based on the analysis of SEARCH17, the symmetry pattern could be found to be the main relation pattern among the three patterns mentioned above, such as (sq-337441, r-6, d-141697) and (d-141697, r-6, sq-337441). And 77.3%, 21% and 1.7% of the test triples in SEARCH17 contain symmetry, anti-symmetry and combination relations, respectively [11].

**b: EVALUATION SETTINGS**

We use CapS-QuaR to re-rank the original list of documents returned by the search engine. The specific operations are as follows: (I) First, we train the proposed method CapS-QuaR, using the trained model to calculate the score for each \((h, r, t)\) triple. (II) Then, we rank the score of each triple in descending order to obtain a new ranking list. We use two standard evaluation metrics in document ranking: Mean Reciprocal Rank (MRR) and Hits@1. For each metric, the higher value shows the better ranking performance.

**c: TRAINING SETTINGS**

We initialize user profile, query and document embeddings for the baselines QuaR and ConvKB, and our CapS-QuaR. We trained the LDA topic model of 200 topics only on the relevant documents extracted from the query log. Then we used the LDA model to infer the topic probability distribution for each returned document. We use the topic proportion vector of each document as its document embedding (i.e. \(k = 200\)). For experiments with CapS-QuaR, we set batch size to 128. The number of neurons within the capsule in the second capsule layer is set as 8 (\(d = 8\)). And the number of iterations in the routing algorithm is set to 2 (\(m = 2\)). For the training model, we selected the number of filters among \(\{100, 200, 400, 500\}\), the initial learning rate \(\gamma\) of the Adam optimizer among \(\{0.00001, 0.00005, 0.0001, 0.0005\}\). We also use ReLU as the activation function. We run the

![FIGURE 5. Hits@10 (in %) on the WN18RR test set w.r.t each relation.](image-url)
model up to 200 epochs and perform a grid search to choose optimal hyper-parameters on the validation set. We obtain the highest MRR score when using $N = 400$ and the initial learning rate at $5 \times 10^{-5}$.

2) MAIN RESULTS

As Table 10 shown, compared to other traditional models, CapS-QuaR produces better ranking results in search personalization task. This shows that the combination of quaternions and capsule networks can effectively improve the ranking quality of search personalization systems. In addition, CapS-QuaR performs better than its closely related neural network model CapsE on SEARCH17, our model

![FIGURE 6. MRR on the WN18RR test set w.r.t each relation.](image)

![FIGURE 7. The learning curves of the proposed CapS-QuaR model.](image)

| Models  | MRR  | H@1  |
|---------|------|------|
| SE[*]   | 0.559| 38.5 |
| CI[*]   | 0.597| 41.6 |
| SF[*]   | 0.631| 45.2 |
| TransE[*] | 0.645| 48.1 |
| ConvKB  | 0.750| 59.9 |
| CapsE   | 0.760| 62.1 |
| CapS-QuaR | 0.784| 63.7 |

[*] denotes the results reported in [43]. Hits@1 (H@1) is reported in %.
obtains significant improvements of 0.784 - 0.760 = 0.024 in MRR (which is about 3.2% relative improvement), and 63.7% - 62.1% = 1.6% absolute improvement in Hits@1. Specifically, our CapS-QuaR model achieves the highest performances in both MRR and Hits@1.

To illustrate our training progress, we plot the performance of CapS-QuaR on the validation set for each epoch in Fig. 7. When the initial learning rate is $5 \times 10^{-3}$, as the number of iterations increases, we can observe that the epoch is stable at 60 and the best MRR is about 0.78. In addition, the performance of the CapS-QuaR model is improved with the number of filters increasing since capsules can encode more useful properties for a larger embedding size.

V. CONCLUSION

In this paper, we first propose QuaR, a knowledge graph embedding model that uses quaternion to represent knowledge graph. QuaR is advantageous with its capability in modeling several pivotal relation patterns, expressing with higher degrees of freedom as well as its good generalization. Then, we present CapS-QuaR, the core model of this paper, a novel knowledge graph embedding model that combines quaternions and capsule networks. The quaternion embeddings of entity and relationship viewed as input in CapS-QuaR, are more expressive than the traditional real number embedding.

In our experiments, we evaluate our models by link prediction task and search personalization task. The experimental results show that our models achieve consistent and significant improvements compared to the state-of-the-art baselines. In the future, we will consider adding a large amount of entity description text to improve the accuracy of experimental results.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers and the editors for helping to improve this work.

REFERENCES

[1] F. M. Suchanek, G. Kasneci, and G. Weikum, “Yago: A core of semantic knowledge,” in Proc. 16th Int. Conf. World Wide Web, 2007, pp. 697–706.
[2] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, “Freebase: A collaboratively created graph database for structuring human knowledge,” in Proc. ACM SIGMOD Int. Conf. Manage. Data, 2008, pp. 1247–1250.
[3] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer, “DBpedia: A large-scale, multilingual knowledge base extracted from Wikipedia,” Semantic Web, vol. 6, no. 2, pp. 167–195, Jan. 2015.
[4] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, “Collaborative knowledge base embedding for recommender systems,” in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Aug. 2016, pp. 353–362.
[5] Y. Hao, Y. Zhang, K. Liu, S. He, Z. Liu, H. Wu, and J. Zhao, “An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge,” in Proc. ACL, 2017, pp. 221–231.
[6] C. Xiong, R. Power, and J. Callan, “Explicit semantic ranking for academic search via knowledge graph embedding,” in Proc. ACL, Apr. 2017, pp. 1271–1279.
[7] B. Yang and T. Mitchell, “Leveraging knowledge bases in LSTMs for improving machine reading,” in Proc. 55th Annu. Meeting Assoc. Comput. Linguistics, vol. 1, 2017, pp. 1436–1446.
[8] R. Socher, D. Chen, C. D. Manning, and A. Ng, “Reasoning with neural tensor networks for knowledge base completion,” in Proc. Adv. Neural Inf. Process. Syst., 2013, pp. 926–934.
[9] R. West, E. Gabrilovich, K. Murphy, S. Sun, R. Gupta, and D. Lin, “Knowledge base completion via search-based question answering,” in Proc. ACL, 2014, pp. 515–526.
[10] A. Bordes, J. Weston, R. Collobert, and Y. Bengio, “Learning structured embeddings of knowledge bases,” in Proc. AAAI, San Francisco, CA, USA, 2011, pp. 301–306.
[11] T. Vu, T. D. Nguyen, D. Q. Nguyen, and D. Phung, “A capsule network-based embedding model for knowledge graph completion and search personalization,” in Proc. NAACL, New Orleans, LA, USA, 2019, pp. 697–706.
[12] Q. Wang, Z. Mao, B. Wang, and L. Guo, “Knowledge graph embedding: A survey of approaches and applications,” IEEE Trans. Knowl. Data Eng., vol. 29, no. 12, pp. 2724–2743, Dec. 2017.
[13] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang, “Rotate: Knowledge graph embedding by relational rotation in complex space,” in Proc. ICLR, 2019, pp. 1–18.
[14] P. Hao, Y. Wang, C. Liu, B. Wang, and H. Wu, “A novel non-probabilistic reliability-based design optimization algorithm using enhanced chaos control method,” Comput. Methods Appl. Mech. Eng., vol. 318, pp. 572–593, May 2017.
[15] Z. Meng, Z. Zhang, D. Zhang, and D. Yang, “An active learning method combining Kriging and accelerated chaotic single loop approach (AK-ACSLA) for reliability-based design optimization,” Comput. Methods Appl. Mech. Eng., vol. 357, Dec. 2019, Art. no. 112570.
[16] Z. Meng, Z. Zhang, and G. Li, “An active weight learning method for efficient reliability assessment with small failure probability,” Struct. Multidisciplinary Optim., vol. 61, pp. 1157–1170, Nov. 2019.
[17] Z. Meng, D. Zhang, G. Li, and B. Yu, “An importance learning method for non-probabilistic reliability analysis and optimization,” Struct. Multidisciplinary Optim., vol. 59, no. 4, pp. 1255–1271, Apr. 2019.
[18] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, “Translating embeddings for modeling multi-relational data,” in Proc. Adv. Neural Inf. Process. Syst., vol. 26, 2013, pp. 2787–2795.
[19] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in Proc. NIPS, 2013, pp. 3111–3119.
[20] Z. Wang, J. Zhang, J. Feng, and Z. Chen, “Knowledge graph embedding by translating on hyperplanes,” in Proc. 26th AAAI Conf. Artif. Intell., 2014, pp. 1112–1119.
[21] Z. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, “Learning entity and relation embeddings for knowledge graph completion,” in Proc. 29th AAAI Conf. Artif. Intell., 2015, pp. 2181–2187.
[22] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, “Knowledge graph embedding via dynamic mapping matrix,” in Proc. 53rd Annua. Meeting Assoc. Comput. Linguistics, 7th Int. Joint Conf. Natural Lang. Process., 2015, pp. 687–696.
[23] B. Yang, W.-T. Yih, X. He, J. Guo, and L. Deng, “Embedding entities and relations for learning and inference in knowledge bases,” in Proc. Int. Conf. Learn. Represent., 2015, pp. 1–12.
[24] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, “Complex embeddings for simple link prediction,” in Proc. 33rd Int. Conf. Mach. Learn., 2016, pp. 2071–2080.
[25] S. Zhang, Y. Tay, L. Yao, and Q. Liu, “Quaternion knowledge graph embeddings,” 2019, arXiv:1904.10281. [Online]. Available: http://arxiv.org/abs/1904.10281.
[26] D. Q. Nguyen, T. D. Nguyen, D. Q. Nguyen, and D. Phung, “A novel embedding model for knowledge base completion based on convolutional neural network,” in Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol., 2018, pp. 327–333.
[27] S. Sabour, N. Frost, and G. E. Hinton, “Dynamic routing between capsules,” in Proc. NIPS, 2017, pp. 3856–3866.
[28] J. Hui. (2017). Understanding Matrix Capsules With EM Routing. Accessed: Nov. 10, 2019. [Online]. Available: http://arxiv.org/abs/1711/11492.
[29] T. Parcollet, M. Morchid, P.-M. Bousquet, R. Dufour, G. Linarès, and R. De Mori, “Quaternion neural networks for spoken language understanding,” in Proc. 12th Int. Conf. Structure Multidisciplinary Optim., vol. 357, Dec. 2019, Art. no. 112570.
[30] T. Parcollet, M. Morchid, and G. Linarès, “Quaternion denoising understanding Matrix Capsules With EM-routing Capsule-Network.”
[31] C. J. Gaudet and A. S. Maida, “Deep quaternion networks,” in Proc. Int. Joint Conf. Neural Netw., 2018, pp. 1–8.

[32] T. Parcollet, Y. Zhang, M. Morchid, C. Trabelsi, G. Linares, R. De Mori, and Y. Bengio, “Quaternion convolutional neural networks for end-to-end automatic speech recognition,” 2018, arXiv:1806.07789. [Online]. Available: http://arxiv.org/abs/1806.07789.

[33] T. Parcollet, M. Ravanelli, M. Morchid, G. Linares, C. Trabelsi, R. De Mori, and Y. Bengio, “Quaternion recurrent neural networks,” 2018, arXiv:1806.04418. [Online]. Available: http://arxiv.org/abs/1806.04418.

[34] Y. Lin, Z. Liu, and M. Sun, “Knowledge representation learning with entities, attributes and relations,” in Proc. IJCAI, Jan. 2016, pp. 41–52.

[35] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, “Convolutional 2D knowledge graph embeddings,” in Proc. 32nd AAAI Conf. Artif. Intell., 2018, pp. 1811–1818.

[36] D. Q. Nguyen, K. Sirts, L. Qu, and M. Johnson, “STransE: A novel embedding model of entities and relationships in knowledge bases,” in Proc. Conf. North Am. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol., 2016, pp. 460–466.

[37] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in Proc. Int. Conf. Learn. Represent. (ICLR), 2015, pp. 1–15.

[38] K. Toutanova and D. Chen, “Observed versus latent features for knowledge base and text inference,” in Proc. CVSC, 2015, pp. 57–66.

[39] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in PyTorch,” in Proc. Adv. Neural Inf. Process. Syst. (NIPS) Workshop, 2017, pp. 1–4.

[40] L. Cai and W. Y. Wang, “KBGAN: Adversarial learning for knowledge graph embeddings,” in Proc. Conf. North Am. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol., 2018, pp. 1470–1480.

[41] Y. Pinter and J. Eisenstein, “Predicting semantic relations using global graph properties,” in Proc. Adv. Empirical Methods Natural Lang. Process., 2018, pp. 1741–1751.

[42] D. Nathani, J. Chauhan, C. Sharma, and M. Kaul, “Learning attention-based embeddings for relation prediction in knowledge graphs,” 2019, arXiv:1906.01195. [Online]. Available: http://arxiv.org/abs/1906.01195.

[43] T. Vu, D. Q. Nguyen, M. Johnson, D. Song, and A. Willis, “Search personalization with embeddings,” in Proc. Ence. Conf. Inf. Retr., 2017, pp. 598–604.

HENG CHEN received the B.S. degree in computer science and technology from Jilin University, Changchun, China, in 2004, and the M.S. degree in computer application technology from Dalian Jiaotong University, Dalian, China, in 2007. He is currently pursuing the Ph.D. degree in computer application technology with Dalian Maritime University. He is also an Associate Professor with the Research Center for Language Intelligence, Dalian University of Foreign Languages. His major research interests include machine learning and knowledge graph completion.

WEIMEI WANG received the B.S. degree in computer science and technology from Jining Medical University, Jining, China, in 2018. She is currently pursuing the M.S. degree in computer science and technology with Dalian Maritime University. Her major research interests include machine learning and knowledge graph completion.

GUANYU LI received the B.S. degree in electronic computer and the M.S. degree in computer application technology from Dalian Maritime University, Dalian, China, in 1985 and 1993, respectively, and the Ph.D. degree in management science and engineering from the Dalian University of Technology, Dalian, in 2010. He is currently a Professor and a Ph.D. Supervisor with the School of Information Science and Technology, Dalian Maritime University. His major research interests include machine learning, intelligent information processing, and knowledge graph completion.

YIMIN SHI received the B.S. and M.S. degrees in computer science and technology from Dalian Maritime University, Dalian, China, in 1989 and 1992, respectively. She is currently an Associate Professor with the School of Information Science and Technology, Dalian Maritime University. Her major research interests include machine learning, intelligent information processing, and knowledge graph completion.

H. Chen et al.: Quaternion-Embedded Capsule Network Model for Knowledge Graph Completion