A Novel Approach for Representing Temporal Knowledge Graphs

Chenze Huang, Ying Zhong

Abstract—**Temporal completion of knowledge graphs is of significant importance in real-world applications. However, previous research has mostly focused on stationary knowledge graphs, largely ignoring the dynamic evolutionary properties of facts. Furthermore, the volatility and scarcity of temporal knowledge graphs, along with the intricate temporal dependency characteristics, render existing models incapable of accurately representing facts that undergo temporal changes. To more accurately represent changes in entities over time, we provide a learning algorithm for representing temporal knowledge graphs using quaternion rotation. This approach characterises the evolution of entities as a temporal rotation transformation in quaternion space. The Hamiltonian product in quaternion space is more effective at capturing potential interdependencies than the Ermitian inner product in complex number space. This results in a learning process that is both more efficient and more expressive. The model shows exceptional performance, as evidenced by experimental results on benchmark datasets.**

Index Terms—**Knowledge representation learning, complex network, deep learning, quaternion embedding, temporal knowledge graph**

I. INTRODUCTION

Knowledge graph technology is essential in the field of knowledge representation and reasoning. A knowledge graph is an extensive semantic network consisting of nodes and edges [1]–[3]. It can be thought of as a graph-based data structure. A knowledge graph is typically represented as a triplet, consisting of a head entity, a tail entity, and a connection between the two entities. Popular knowledge graphs such as FreeBase, YAGO and Schema.org are examples of typical knowledge graphs [4]. A knowledge graph can effectively convey the connections between things using a well-organised framework, offering the benefits of extensibility and comprehensibility. The knowledge graph has been helpful in areas such as system recommendation and knowledge retrieval.

Typically, knowledge graphs are very large [5], such as the Google Knowledge Graph, which contains tens of billions of relationships and entities [6]. In general, the knowledge graph is still incomplete and there is a serious data sparsity problem [7], i.e. many facts in the knowledge graph lack corresponding relationships or entities. This missing information can limit the performance of the knowledge graph and its wider applications, so completing the knowledge graph is very important. Knowledge graph completion [8] is a type of knowledge reasoning that aims to predict missing facts based on existing facts in the knowledge graph, it can also be called link prediction. In the early stage, the scale of knowledge graph was mainly expanded by artificial construction, but this way is inefficient and difficult to cope with super-large scale knowledge graph. Knowledge representation learning represents entities and relationships in low-dimensional vectorisation [1], [2], [4], which can effectively solve the problem of sparse data. At present, knowledge representation learning has become the basis of large-scale knowledge graph construction and application, and is also the most mainstream knowledge graph completion technology.

However, current studies focus mainly on the stationary knowledge graph, making it difficult to understand the temporal dynamics of information. In fact, many facts change over time. For example, the triad of facts consisting of South Korean President Moon Jae-in was exclusively in place from May 2017 to May 2021. However, there have been other cases where different triads have been established, such as South Korea, President Park Geun-hye and South Korea, President Lee Myung-bak. Ignoring the time element when answering the question ”Who is the President of South Korea?” is likely to lead to inaccurate results. Therefore, there is great value in conducting research on knowledge graphs that incorporate temporal information.

In this paper, we present a learning algorithm called TKGR for representing temporal knowledge graphs. TKGR uses quaternion rotation to accurately capture the evolutionary process of entities over time. It defines the transformation of entities over time as a temporal rotation in quaternion space. The Hamiltonian product in quaternion space provides a more efficient and expressive learning process compared to the Ermitian inner product in complex number space, allowing for a better capture of potential interdependencies. The main contributions of this study are outlined below.

- To make efficient use of temporal information and to accurately represent changes in entities over time, we propose a learning technique that represents temporal knowledge graphs using quaternion rotation. The model represents the temporal progression of entities using a rotation transformation in quaternion space, which increases the efficiency and expressiveness of the learning process.
- Our method models the temporal and relational evolution of knowledge graphs through Hamiltonian product-based quaternion rotations, leveraging their properties for precise representation of relational data, outperforming the Ermitian product in complex numbers.
- Finally, we validate the effectiveness and appropriateness of the TKGR model through extensive experimental comparisons with state-of-the-art algorithms from multiple perspectives.

Manuscript received November 27, 2023; revised April 18, 2024.

Chenze Huang is a PhD candidate of Research and Development Institute of Northwestern Polytechnical University, Shenzhen, Guangdong 518057, China (Corresponding author e-mail: yxy_0713@qq.com).

Ying Zhong is a lecturer of Research and Development Institute of Northwestern Polytechnical University, Shenzhen, Guangdong 518057 China (Corresponding author e-mail: luo_xj@mail.nwpu.edu.cn).

The rest of this paper is structured as follows. Section II introduces the related work, Section III gives a detailed description of the related theory of quaternions. Section IV describes the TKGR model. Section V presents specific experimental results. The research work is summarized in Section VI.

II. RELATED WORK

Embedding techniques can transform entities and relationships in a knowledge network into low-dimensional continuous vectors. This section deals specifically with static knowledge graph embedding and temporal knowledge graph embedding.

A. Static Knowledge Graph Embedding

Various knowledge graph embedding approaches have been presented by researchers for static knowledge graphs. The methods can be divided into three main groups: (1) knowledge graph embedding based on matrix decomposition, (2) knowledge graph embedding based on convolutional neural networks, and (3) knowledge graph embedding based on graphs.

TransE [9] was first proposed to solve models of knowledge graph embedding that use a score function to map entities h, t and relations r: $h + r \approx t$. The aim is to minimise the distance between h and t. However, TransE is limited in its ability to handle complex relations. It performs well only on 1 to 1 relations and cannot handle 1 to N and N to 1 relations. To overcome these limitations, several extended models have been proposed. One such model is TransH [10], which introduces a hyperplane for each relation onto which entities are projected. This allows TransH to handle both 1 to N and N to 1 relations. Another model is TransR [10], which uses a separate relation-specific matrix to transform the entities into a different vector space. This allows TransR to effectively handle different types of relations. TransD [11] is another extended model that addresses the limitations of TransE. It introduces additional matrices to model the semantic meaning of entities and relations separately. This allows TransD to capture more fine-grained information and handle complex relationships.

Matrix factorization techniques such as DistMult [12], RESCAL [13], HolE [14], ComplEx [15], and SimplE [16] convert relationships into matrix-based linear operations on entity representations. In contrast, CNN-based approaches like ConvE [17] and ConvKB [5] utilize convolution to model entity and relation interactions, transforming them into multi-dimensional matrices and enhancing the expressiveness of the embeddings through feature extraction. Additionally, graph-based methods are considered, with DeepWalk [18] pioneering the generation of vector representations for network nodes through a random walk model, and R-GCN [19], a graph neural network variant, being among the first to aggregate neighborhood information for learning node embeddings with equal weights. However, these strategies fall short in capturing temporal dynamics in temporal knowledge graphs, rendering them less applicable for such scenarios.

B. Temporal Knowledge Graph Embedding

Previous research has mainly focused on improving static knowledge graph techniques by integrating temporal elements. One prominent model, TTransE [16], pioneered the learning of entity and relation embeddings by representing transitions between temporal relations using temporal constraints. For example, it considers the sequential relation between the events ”diedIn” and ”wasBornIn”. The fusion of temporal information from static models does not simply extend the above models. The ConT model extends the static knowledge graph model to include temporal knowledge graphs [20]. This allows the model to retain plot data and extrapolate to incorporate new information. HyTE [21] incorporates time into the entity relationship framework by linking each timestamp to the relevant hyperplane in a clear and direct manner. HyTE employs time-sensitive guidance to enhance knowledge graph reasoning and prognosticates the temporal validity of relationships missing temporal data. TA-DisMult [22] addresses the task of predicting the temporal knowledge graph by capturing the encoding of potential entities and connection types. The model uses recurrent neural networks to acquire time-sensitive representations of relationship types, allowing their integration with established potential factor decomposition techniques for temporal information fusion.

DE-SimplE [23] augments a traditional knowledge graph framework with a dynamic entity embedding mechanism, facilitating the depiction of temporal entity characteristics across various time frames. It differs from current temporal knowledge graph embedding methods, which only provide static attributes of entities. SubEE [23] uses a fixed size vocabulary to assign labels to entities, relationships, timestamps and locations. Meanwhile, the framework leverages a spatial-temporal communication layer to extract the underlying feature vectors within the knowledge network. BoxTE [24] is an extended version of the static knowledge graph embedding model, it exhibits rich expressiveness and robust inductive capacity in temporal settings.

While existing methods for temporal knowledge graph completion are promising, they often fail to capture the graph structure and temporal relationships. Unlike these approaches, which tend to focus on either entity features or connections without integration, TKGR uniquely captures both temporal dynamics and graph structure, resulting in more accurate embeddings for these tasks.

III. THEORY OF QUATERNIONS

This part offers an initial summary of the conceptual underpinnings pertaining to quaternions. The quaternionic structure $\mathbb{H}$ represents a broadening of the realm of complex numbers, commonly comprising one real element and three imaginary units, as outlined further in the text.

$$q = q_r + q_i\boldsymbol{i} + q_j\boldsymbol{j} + q_k\boldsymbol{k} \tag{1}$$

where q_r, q_i, q_j, and q_k are the corresponding real coefficients. $\boldsymbol{i}$, $\boldsymbol{j}$ and $\boldsymbol{k}$ are the imaginary units. If q_j=q_k=0, it is the conventional complex form. The mentioned imaginary components conform to a specific set of guidelines.

$$\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = \boldsymbol{ijk} = -1 \tag{2}$$

Some other rules that do not satisfy commutativity can be deduced from this rule, such as ***ij=k***, ***ji=-k***, ***jk=i***, ***ki=j***, ***kj=-i***, and ***ik=-j*** . A quaternion vector $\boldsymbol{q} \in \mathbb{H}^n$ is defined as follows.

$$\boldsymbol{q} = \boldsymbol{q}_r + \boldsymbol{q}_i \boldsymbol{i} + \boldsymbol{q}_j \boldsymbol{j} + \boldsymbol{q}_k \boldsymbol{k} \tag{3}$$

where $\boldsymbol{q}_r$, $\boldsymbol{q}_i$, $\boldsymbol{q}_j$ and $\boldsymbol{q}_k$ are *n*-dimensional real vectors. These definitions are used in this paper to model relational data. Some common arithmetic rules for quaternions are defined as follows.

Conjugate: The definition of the conjugate for quaternion q from $\mathbb{H}$ is defined as follows.

$$\bar{q} = q_r - q_i \boldsymbol{i} - q_j \boldsymbol{j} - q_k \boldsymbol{k} \tag{4}$$

Norm: The norm of quaternion $\boldsymbol{q} \in \mathbb{H}$ is defined as follows.

$$\|q\| = \sqrt{q_r^2 + q_i^2 + q_j^2 + q_k^2} \tag{5}$$

By means of this operation, the definition of the quaternion of the unit $q^{\triangleleft}$ is obtained.

$$q^{\triangleleft} = \frac{q}{\|q\|} \tag{6}$$

The unit representation of the quaternion vector $\boldsymbol{q} \in \mathbb{H}^n$ is defined below.

$$\boldsymbol{q}^{\triangleleft} = \frac{\boldsymbol{q}_r + \boldsymbol{q}_i \boldsymbol{i} + \boldsymbol{q}_j \boldsymbol{j} + \boldsymbol{q}_k \boldsymbol{k}}{\sqrt{\boldsymbol{q}_r^2 + \boldsymbol{q}_i^2 + \boldsymbol{q}_j^2 + \boldsymbol{q}_k^2}} \tag{7}$$

Addition: The addition operation of two quaternions $q_1 = q_{r1} + q_{r1}\boldsymbol{i} + q_{j1}\boldsymbol{j} + q_{k1}\boldsymbol{k}$ and $q_2 = q_{r1} + q_{r2}\boldsymbol{i} + q_{j2}\boldsymbol{j} + q_{k2}\boldsymbol{k}$ is defined as follows.

$$\begin{aligned} q_1 + q_2 = & (q_{r1} + q_{r2}) + (q_{i1} + q_{i2})\boldsymbol{i} \\ & + (q_{j1} + q_{j2})\boldsymbol{j} + (q_{k1} + q_{k2})\boldsymbol{k} \end{aligned} \tag{8}$$

Scalar multiplication: The multiplication of a scalar λ with a quaternion q from the set $\mathbb{H}$ is specified as follows.

$$\lambda q = \lambda q_r + \lambda q_i \boldsymbol{i} + \lambda q_j \boldsymbol{j} + \lambda q_k \boldsymbol{k} \tag{9}$$

Inner product: Similar to the inner product operation on vectors, the result of the inner product operation on quaternions can be obtained by computing the product of the corresponding components of two quaternions $q_1 \in \mathbb{H}$ and $q_2 \in \mathbb{H}$ and then summing them.

$$q_1 \cdot q_2 = q_{r1} q_{r2} + q_{i1} q_{i2} + q_{j1} q_{j2} + q_{k1} q_{k2} \tag{10}$$

The inner product operation of two quaternion vectors $q_1 \in \mathbb{H}^n$ and $q_2 \in \mathbb{H}^n$ is defined below.

$$\boldsymbol{q}_1 \cdot \boldsymbol{q}_2 = \boldsymbol{q}_{r1}^T \boldsymbol{q}_{r2} + \boldsymbol{q}_{i1}^T \boldsymbol{q}_{i2} + \boldsymbol{q}_{j1}^T \boldsymbol{q}_{j2} + \boldsymbol{q}_{k1}^T \boldsymbol{q}_{k2} \tag{11}$$

Hamiltonian product: Hamiltonian product is also called quaternion multiplication. The Hamiltonian product $\otimes$ of two quaternions $q_1 \in \mathbb{H}$ and $q_2 \in \mathbb{H}$ is defined as

$$\begin{aligned} q_1 \otimes q_2 = & (q_{r1} q_{r2} - q_{i1} q_{i2} - q_{j1} q_{j2} - q_{k1} q_{k2}) \\ & + (q_{i1} q_{r2} + q_{r1} q_{i2} - q_{k1} q_{j2} + q_{j1} q_{k2})\boldsymbol{i} \\ & + (q_{j1} q_{r2} + q_{k1} q_{i2} + q_{r1} q_{j2} - q_{i1} q_{k2})\boldsymbol{j} \\ & + (q_{k1} q_{r2} - q_{j1} q_{i2} + q_{i1} q_{j2} + q_{r1} q_{k2})\boldsymbol{k} \end{aligned} \tag{12}$$

The Hamiltonian product of two quaternion vectors $q_1 \in \mathbb{H}^n$ and $q_2 \in \mathbb{H}^n$ is defined as

$$\begin{aligned} \boldsymbol{q}_1 \otimes \boldsymbol{q}_2 = & (\boldsymbol{q}_{r1} \circ \boldsymbol{q}_{r2} - \boldsymbol{q}_{i1} \circ \boldsymbol{q}_{i2} - \boldsymbol{q}_{j1} \circ \boldsymbol{q}_{j2} - \boldsymbol{q}_{k1} \circ \boldsymbol{q}_{k2}) \\ & + (\boldsymbol{q}_{i1} \circ \boldsymbol{q}_{r2} + \boldsymbol{q}_{r1} \circ \boldsymbol{q}_{i2} - \boldsymbol{q}_{k1} \circ \boldsymbol{q}_{j2} + \boldsymbol{q}_{j1} \circ \boldsymbol{q}_{k2})\boldsymbol{i} \\ & + (\boldsymbol{q}_{j1} \circ \boldsymbol{q}_{r2} + \boldsymbol{q}_{k1} \circ \boldsymbol{q}_{i2} + \boldsymbol{q}_{r1} \circ \boldsymbol{q}_{j2} - \boldsymbol{q}_{i1} \circ \boldsymbol{q}_{k2})\boldsymbol{j} \\ & + (\boldsymbol{q}_{k1} \circ \boldsymbol{q}_{r2} - \boldsymbol{q}_{j1} \circ \boldsymbol{q}_{i2} + \boldsymbol{q}_{i1} \circ \boldsymbol{q}_{j2} + \boldsymbol{q}_{r1} \circ \boldsymbol{q}_{k2})\boldsymbol{k} \end{aligned} \tag{13}$$

where $\circ$ represents the multiplication between corresponding elements. It can be seen from Eq.(12) that the Hamiltonian product has non commutativity, i.e. $q_1 \otimes q_2 \neq q_2 \otimes q_1$.

IV. PROPOSED MODEL

We use (h, r, t, τ) to represent quaternions in a temporal knowledge graph, A time-aware knowledge graph $\mathcal{G}$ may be regarded as an assembly of quaternion elements. For each training quaternion in the graph, it is necessary to generate corresponding negative samples using negative sampling technology to support effective representation learning of entities and relations. This section employs the notation $\mathcal{G}\prime$ to denote the collection of negative instances generated by swapping the head or tail entities. Considering a knowledge graph that evolves over time, the model aims to master a compact representation using quaternions and to establish a scoring formula $g(h, r, t, \tau)$ that quantifies the compatibility of the head entity, tail entity, relationship, and associated time point with the real numbers. The objective of this evaluation metric is to assess the veracity of quads, where genuine and functional quads receive higher scores compared to those deemed ineffectual.

A. *Model Definition*

The model aims to employ quaternion embeddings for representing entities, relationships, and temporal data. Given a quaternion (h, r, t, τ), the quaternion representations $\boldsymbol{q}_h$, $\boldsymbol{q}_r$, $\boldsymbol{q}_t$ and $\boldsymbol{q}_\tau \in \mathbb{H}^n$ corresponding to each constituent element are defined as follows.

$$\boldsymbol{q}_h = \boldsymbol{a}_h + \boldsymbol{b}_h \boldsymbol{i} + \boldsymbol{c}_h \boldsymbol{j} + \boldsymbol{d}_h \boldsymbol{k} \tag{14}$$

$$\boldsymbol{q}_r = \boldsymbol{a}_t + \boldsymbol{b}_t \boldsymbol{i} + \boldsymbol{c}_t \boldsymbol{j} + \boldsymbol{d}_t \boldsymbol{k} \tag{15}$$

$$\boldsymbol{q}_t = \boldsymbol{a}_h + \boldsymbol{b}_h \boldsymbol{i} + \boldsymbol{c}_h \boldsymbol{j} + \boldsymbol{d}_h \boldsymbol{k} \tag{16}$$

$$\boldsymbol{q}_\tau = \boldsymbol{a}_\tau + \boldsymbol{b}_\tau \boldsymbol{i} + \boldsymbol{c}_\tau \boldsymbol{j} + \boldsymbol{d}_\tau \boldsymbol{k} \tag{17}$$

where the coefficients of the real part as well as each imaginary part unit are *n*-dimensional vectors in real space.

While static knowledge graph completion models are able to learn multiple relational interactions between entities, they ignore the temporal factor and are therefore unable to reason effectively on temporal knowledge graphs. To address this problem, our model defines temporal evolution as a Hamiltonian product-based rotation transformation in quaternion space. Specifically, the model uses the unit vector form $\boldsymbol{q}_\tau^{\triangleleft}$ of the temporal quaternion to represent the rotation operator, and rotates the head entity vector $\boldsymbol{q}_h$ and the tail entity vector $\boldsymbol{q}_t$ to obtain a temporal entity representation, respectively.

$$\boldsymbol{q}_{h,\tau} = \boldsymbol{q}_h \otimes \boldsymbol{q}_\tau^{\triangleleft} \tag{18}$$

$$\boldsymbol{q}_{t,\tau} = \boldsymbol{q}_t \otimes \boldsymbol{q}_\tau^{\triangleleft} \tag{19}$$

The quaternion vectors of the head and tail entities can be considered as a point in the quaternion space, while $\boldsymbol{q}_\tau^{\triangleleft}$ denotes a rotation transformation. After obtaining the time-dependent entities, the model uses the relation as a rotation transformation based on the Hamiltonian product, with the aim of rotating the time-dependent head entity to the vicinity of the time-dependent tail entity through the relation transformation. The general structure of the model is shown in Figure 1. To effectively represent the rotation transformation, we normalise the relation quaternion vector. For real quaternions, the model expects to satisfy $\boldsymbol{q}_{h,\tau} \otimes \boldsymbol{q}_\tau^{\triangleleft} \approx \boldsymbol{q}_{t,\tau}$. With two Hamiltonian product-based rotations, the entities, relations and timestamps can be allowed to interact sufficiently to capture the potential interdependencies between these three.

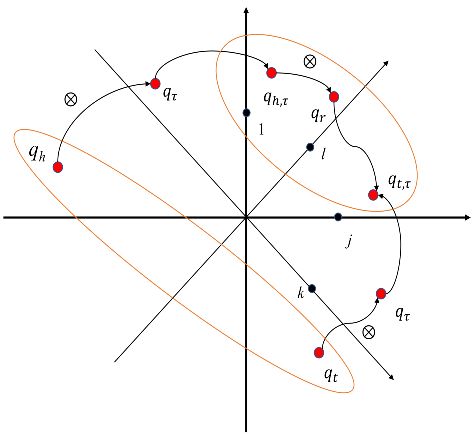

Fig. 1: Schematic diagram of quaternion rotation.

Differing from the Euler distance-based scoring methods of earlier studies, our model employs the angle between vectors as a metric for their similarity. For assessing the quaternion compatibility, the scoring function $g\ (h,\ r,\ t,\ \tau)$ of our model is established thusly.

$$g\left(h, r, t, \tau\right) = \boldsymbol{q}_{h,\tau} \otimes \boldsymbol{q}_r^{\triangleleft} \cdot \boldsymbol{q}_{t,\tau} \tag{20}$$

B. Model Training

For parameter learning of the model, it's essential to establish a matching loss function derived from the scoring function. Within the training data, for every quadruple, we create a negative example by substituting either the head or tail entity with an alternative from the entity set. The loss function is then formulated such that the score assigned to the model's negative example is less than that of the positive example.

$$L\left(\theta\right) = -\sum_{(h,r,t,\tau)} \left(\frac{\exp\left(g\left(h,r,t,\tau\right)\right)}{\sum_{T1=(h,r,t\prime,\tau)\in\mathcal{G}} \exp\left(T1\right)} + \frac{\exp\left(g\left(h,r,t,\tau\right)\right)}{\sum_{T2=(h',r,t,\tau)\in\mathcal{G}'} \exp\left(T2\right)} \right) \tag{21}$$

where θ denotes the learnable parameters of the model.

We introduce parametric regularization terms to the entity, relationship, and timestamp embeddings to avoid overfitting and enhance the model's capacity to generalize unseen data, as indicated by Equation (22).

$$L_{regular}\left(\theta\right) = \sum_{(h,r,t,\tau)} \left(\|\boldsymbol{q}_h\|_2^2 + \|\boldsymbol{q}_r\|_2^2 + \|\boldsymbol{q}_t\|_2^2 + \|\boldsymbol{q}_\tau\|_2^2 \right) \tag{22}$$

Furthermore, we assume that the existing knowledge of past time intervals can be used to accurately capture the progression pattern of the whole graph during model training. The literature [25] presents experimental results indicating that subgraphs of successive time steps exhibit minimal changes, and the embedding space of successive time steps should also exhibit smoothness. Consequently, the TKGR model must have smoothing requirements on successive time intervals.

$$L_{smooth}\left(\theta\right) = \frac{1}{\mid \mathcal{T} \mid -1} \sum_{i=1}^{|\mathcal{T}|-1} \mid \boldsymbol{q}_{\tau_{i+1}} - \boldsymbol{q}_{\tau_i} \mid_2^2 \tag{23}$$

where $\mid \mathcal{T} \mid$ denotes the number of time steps. By adding parameter regularization constraints as well as temporal smoothing constraints to the model, the final loss function of the TKGR model is shown below.

$$\mathcal{L}\left(\theta\right) = L\left(\theta\right) + \lambda_1 L_{regular}\left(\theta\right) + \lambda_2 L_{smooth}\left(\theta\right) \tag{24}$$

where λ_1 and λ_2 represents the coefficient of the regularization term.

Meanwhile, a suitable initialisation method can improve the training speed and reduce the risk of gradient explosion or gradient disappearance. It has been shown that the parameters of the hyper-complex representation cannot simply be initialised randomly. Thus, we use a specific approach to initialize parameters and facilitate the TKGR model's convergence.

$$w_r = \varphi \cos\left(\theta\right) \tag{25}$$

$$w_i = \varphi \mathcal{Q}_{img_i}^{\triangleleft} \sin\left(\theta\right) \tag{26}$$

$$w_j = \varphi \mathcal{Q}_{img_j}^{\triangleleft} \sin\left(\theta\right) \tag{27}$$

$$w_k = \varphi \mathcal{Q}_{img_k}^{\triangleleft} \sin\left(\theta\right) \tag{28}$$

where w_r, w_i, w_j, and w_k represent the real and imaginary parts of the initialized quaternion, respectively. θ is randomly generated over the interval $[-\pi, \pi]$. $\mathcal{Q}_{img}^{\triangleleft}$ denotes the normalised unit quaternion, generated according to a uniform distribution in the interval $[0,1]$. φ is randomly generated based on the value of quaternion and the chosen initialization principle.

Finally, we use the AdaGrad algorithm to optimise the objective function. Algorithm 1 describes the learning process of the TKGR model.

Algorithm 1 Learning process of the TKGR model

Input: Temporal knowledge graph entity set $\mathcal{E}$. Relationship set $\mathcal{R}$. Timestamp set $\mathcal{T}$. Training set $\mathcal{G}_{train} = \{(h,r,t,\tau)\}$. The dimension of quaternion vector *n*. Learning rate α. Batch size *b*. Regular term coefficient λ_1 and λ_2. Number of negative samples γ. Number of model iterations *M*.

Output: Vector representation of entities, relationships, and timestamps

Initialize model parameters according to Equation (21)
for $epoch$ =1 to M **do**
    $S_{batch} \leftarrow Sample\,(\mathcal{G}_{train}, b)$
    for each$(h,r,t,\tau) \in S_{batch}$ **do**
        $S\,(h\prime, r, t\prime, \tau) \leftarrow Sample\left(S_{(h,r,t,\tau)}\right)$
    end for
    Apply Equation (16) to compute the scores for positive and negative instances
    Determine the loss function value by applying Equation (20)
    Update model parameters
end for

C. Learning Different Relationship Patterns

Some existing temporal knowledge graph embedding models may have limitations in learning complex relationship patterns, while this model can simultaneously model complex relationship types such as symmetric relationships, antisymmetric relationships, and inverse relationships. We first provide definitions of these complex relationships and then prove that this model has this modelling capability.

Symmetric relationship: A connection $r\,(h,t,\tau) \rightarrow r\,(t,h,\tau)$ is considered symmetric if *h*, *t*, and τ exist.

Anti symmetric relationship: If variables *h*, *t* and τ satisfy the condition $r\,(h,t,\tau) \rightarrow r\,(t,h,\tau)$, then the relation *r* is called antisymmetric.

Inverse relationship: If variables *h*, *t* and τ result in the transformation $r_2\,(h,t,\tau) \rightarrow r_1\,(t,h,\tau)$, then r_1 is considered to be the inverse of r_2.

Given a fact quaternion (*h*, *t*, τ), If *r* is a symmetric relationship, then both $\boldsymbol{q}_{h,\tau} \otimes \boldsymbol{q}_r^{\triangleleft} = \boldsymbol{q}_{t,\tau}$ and $\boldsymbol{q}_{t,\tau} \otimes \boldsymbol{q}_r^{\triangleleft} = \boldsymbol{q}_{h,\tau}$ are satisfied, which is equivalent to $\boldsymbol{q}_r^{\triangleleft}$ degenerating into a scalar. Since this model uses the angle between vectors to measure similarity, the length of the vector does not affect the final scoring function, so the imaginary part of the relationship $\boldsymbol{q}_r$ will be 0. On the contrary, if r is an antisymmetric relationship, then it is sufficient to make the imaginary part of $\boldsymbol{q}_r$ not zero.

Given two fact quaternions (h, r_1, t, τ) And (h, r_2, t, τ), So r_1 and r_2 are a pair of inverse relationships, which is easy to verify, that is, this model can use a pair of conjugate quaternions to express a pair of inverse relationships. The specific verification process is given below.

$$\begin{aligned}
&\boldsymbol{q}_{h,\tau} \otimes \boldsymbol{q}_r^{\triangleleft} \cdot \boldsymbol{q}_{t,\tau} \\
&= a_h a_t a_r - b_h b_t b_r - c_h a_t a_r - d_h a_t d_r + b_h b_t a_r + a_h b_t b_r \\
&- d_h b_t c_r + c_h b_t d_r + c_h c_t a_r + d_h c_t b_r + a_h c_t c_r - b_h c_t d_r \\
&+ c_h c_t a_r + d_h c_t b_r + a_h c_t c_r - b_h c_t d_r
\end{aligned} \tag{29}$$

$$\begin{aligned}
&\boldsymbol{q}_{t,\tau} \otimes \bar{\boldsymbol{q}}_r^{\triangleleft} \cdot \boldsymbol{q}_{h,\tau} \\
&= a_h a_t a_r + a_h b_t b_r + a_h c_t c_r + a_h d_t d_r + b_h b_t a_r - b_h a_t b_r \\
&+ b_h d_t c_r - b_h c_t d_r + c_h c_t a_r - c_h d_t b_r - c_h a_t c_r + c_h b_t d_r \\
&+ d_h c_t a_r + d_h c_t b_r - d_h b_t c_r - d_h a_t d_r
\end{aligned} \tag{30}$$

According to the expansion results, it can be found that these two formulas are completely consistent overall, so $\boldsymbol{q}_{h,\tau} \otimes \boldsymbol{q}_r^{\triangleleft} \cdot \boldsymbol{q}_{t,\tau} = \boldsymbol{q}_{t,\tau} \otimes \bar{\boldsymbol{q}}_r^{\triangleleft} \cdot \boldsymbol{q}_{h,\tau}$.

V. EXPERIMENTS AND RESULT ANALYSIS

The goal of this section is to validate the excellence of our proposed model. Firstly, we list the state-of-the-art algorithms and experimental setups, and then we conduct experimental comparisons from different perspectives, and the following are the specific experimental sessions.

A. Datasets

We opt for the trio of prevalent temporal knowledge graph benchmarksłICEWS14, ICEWS05-15, and GDELTłtypically used to appraise completion models in this domain. TABLE I lists the detailed statistical information of these three datasets.

ICEWS14: This dataset consists of four tuples taken from social news related to political events. ICEWS14 is a subset of ICEWS that focuses on events from the year 2014. It has 7,128 entities, 230 relationships and spans 365 time steps.

ICEWS05-15: ICEWS05-15 is a subset of the ICEWS dataset containing events that occurred between 2005 and 2015. The dataset contains 10,488 entities, 251 relationships and spans 4,017 time steps.

GDELT: This dataset is a collection of human social relationships. We isolate a subset from the source [26] that represents events that occurred between 2015 and 2016. This subset contains 500 entities, 20 relationships and spans 366 time steps.

B. Evaluation Metrics

The evaluation metrics employed for completing dynamic knowledge graphs are consistent with those utilized in static knowledge graph completion. First, For every quaternion within the test data, we produce two variations by interchanging either the leading or trailing entity. Any potential quaternion found in $\mathcal{G}$ must be removed. The exact formula is given below.

$$h_{candidate} = \{(h', r, t, \tau) \mid h' \in \varepsilon, (h', r, t, \tau \notin \mathcal{G})\} \tag{31}$$

$$h_{candidate} = \{(h, r, t', \tau) \mid t' \in \varepsilon, (h, r, t', \tau \notin \mathcal{G})\} \tag{32}$$

TABLE I: DETAILS OF THE DATASETS

Datasets	Entities	Relations	Time Steps	Training	Validation	Test
ICEWS14	7,128	230	365	72,826	8,941	8,963
ICEWS05-15	10,488	251	4,017	386,962	46,275	46,092
GDELT	500	20	366	2,735,685	341,961	341,961

It is then necessary to calculate the ratings of the test quaternion and all candidate quaternions, and rank the ratings in descending order to obtain the ranking of the test quaternion in each of the two candidate sets, which is denoted by f_h and f_t, respectively. Based on this ranking, we select MRR and Hit@k (k=1,3,10) as the model evaluation metrics. MRR represents the reciprocal of the mean rank, while Hit@k denotes the proportion of correctly predicted items within the top k positions. The calculation for MRR is provided below.

$$MRR = \frac{1}{2\cdot \mid \mathcal{G}_{test} \mid} \sum_{(h,r,t,\tau)\in\mathcal{G}_{test}} \left(\frac{1}{f_h} + \frac{1}{f_t}\right) \tag{33}$$

where $\mid \mathcal{G}_{test} \mid$ is the size of the test set. A higher MRR value indicates an improvement in the model's performance. The formula for hit@k is shown below.

$$Hit@k = \frac{1}{2\cdot \mid \mathcal{G}_{test} \mid} \sum_{(h,r,t,\tau)\in\mathcal{G}_{test}} \left(I\left(f_h \leqslant k\right) + I\left(f_t \leqslant k\right)\right) \tag{34}$$

where $I\left(\cdot\right)$ denotes the indicator function, e.g. if $f_h \leqslant k$, then $I(f_h \leqslant k) = 1$, otherwise $I(f_h \leqslant k) = 0$. In the experiments, the Hit@k results were multiplied by 100.

C. Experimental Setup

Our model is trained utilizing the PyTorch framework, which harnesses the potent capabilities of the GeForce RTX 2080 GPU to enhance its performance. For the coefficients λ_1 and λ_2 of the parameter regularisation term and the temporal smoothing constraint term, we assign the same values to both coefficients as they have the same number of paradigms. We then select the most appropriate values from the range $\{0.1, 0.01, 0.001\}$.

The dimensionality of the embeddings is set to 500, with the learning rate adjusted to 0.1. The negative to positive sample ratio is 500 for both ICEWS datasets and 5 for the GDELT dataset. The batch size is set to 512. Different hyperparameters are used for each dataset to optimise the training given their different sizes. For example, GDELT is much larger than the other two datasets, resulting in smaller hyperparameters. The model trains 5,000 times on ICEWS14 and ICEWS05-15, and 1,000 times on GDELT. The training process uses the AdaGrad optimisation algorithm within the small batch random gradient descent approach.

D. Baselines

We compare different knowledge graph embedding techniques, including TransE [9], DistMult [27], SimplE [28], ConT [20], TTransE [29], HyTE [21], TA-DisMult [22], DE-SimplE [30], RoAN [31], SubEE [23], T-GAE [32], GLANet [27], BoxTE [24], DualMatch [33], DKGE [23] and TLmod [34]. These methods use different mechanisms to handle entity relationships, temporal information, global and local structure, and reasoning processes, with the aim of improving the expressiveness and inference accuracy of knowledge graphs.

E. Link Prediction

Tables II and III present the TKGR and baseline model outcomes for forecasting temporal connections across three distinct datasets. The empirical findings indicate that TKGR surpasses the prevalent baseline model in the majority of metrics, affirming the proficiency of TKGR. Quaternion rotation operations based on Hamiltonian products allow for a more comprehensive interaction between entities, time and relationships. On the ICEWS05-15 dataset, TKGR showed greater improvement than the TeRo model on MRR and Hit@10, with increases of 2.55% and 1.38% respectively. The ICEWS14 dataset contains a larger number of entities, relationships, timestamps and training samples. TKGR has the ability to describe complicated relationship types, including symmetric/anti-symmetric and inverse relationships, making it well suited to complicated datasets.

While TKGR's performance on the GDELT dataset may not stand out significantly among other advanced models, it remains impressive. Compared to the BoxTE, T-GAE and RoAN models, there is a slight improvement in all indicators, but the level of performance improvement is not very significant. In the GDELT dataset, the number of entities and relationships is quite small. Conversely, the volume of both the training and evaluation datasets is considerably substantial. This poses a greater challenge to the models, and as a result the test scores for all the models are not particularly high.

In addition, traditional representation learning algorithms such as TransE and DistMult models are mainly used for static knowledge graphs. As a result, their performance in temporal link prediction is generally unsatisfactory. This is because both algorithms ignore the influence of temporal information and are therefore unable to effectively capture the temporal dynamic properties of knowledge graphs. In conclusion, the results of the experiments show that TKGR is both appropriate and necessary. This capability arises from its adeptness at leveraging temporal data, which enables it to accurately capture the mutable characteristics of entities over time.

F. Impact of Regularisation

The final objective function is modified by TKGR to incorporate parameter regularisation and temporal smoothing constraints. These modifications are based on previous results. We conduct an empirical validation with the ICEWS14 dataset to illustrate the effects of these two regularization elements on the model's efficacy. This is done by observing the variation of the results and analysing them when these

TABLE II: LINK PREDICTION RESULTS ON DATASETS ICEWS14 AND ICEWS05-15

Model	ICEWS14				ICEWS05-15			
	MRR	Hit@1	Hit@3	Hit@10	MRR	Hit@1	Hit@3	Hit@10
TransE	0..28	9.4	-	63.7	0.294	9.0	-	66.3
DistMult	0.439	32.3	-	67.2	0.456	33.7	-	69.1
SimplE	0.458	34.1	51.6	68.7	0.478	35.9	53.9	70.8
ConT	0.185	11.7	20.5	31.5	0.163	10.5	18.9	27.2
TTransE	0.255	7.4	-	60.1	0.271	8.4	-	61.6
HyTE	0.297	10.8	41.6	65.5	0.316	11.6	44.5	68.1
TA-DistMult	0.477	36.3	-	68.6	0.474	34.6	-	72.8
DE-SimplE	0.526	41.8	59.2	72.5	0.513	39.2	57.8	74.8
RoAN	0.560	45.3	63.1	74.2	0.587	47.1	66.9	79.7
SubEE	0.519	43.5	62.4	72.1	0.572	46.3	65.4	78.1
GLANet	0.509	44.6	63.1	73.5	0.564	46.7	65.3	78.7
T-GAE	0.554	44.8	62.1	73.8	0.552	47.5	65.1	77.2
DKGE	0.549	45.2	62.9	73.9	0.591	47.3	65.8	79.2
BoxTE	0.549	45.2	62.9	74.1	0.594	47.9	66.3	78.2
TKGR	**0.568**	**45.9**	**63.6**	**74.8**	**0.602**	**48.3**	**67.9**	**80.8**

TABLE III: LINK PREDICTION RESULTS ON THE DATASET GDELT

Model	MRR	Hit@1	Hit@3	Hit@10
TransE	0.113	0.0	15.8	31.2
DistMult	0.196	11.7	20.8	34.8
SimplE	0.206	12.4	22.0	36.6
ConT	0.144	8.0	15.6	26.5
TTransE	0.115	0.0	16.0	31.8
HyTE	0.118	0.0	16.5	32.6
TA-DistMult	0.206	12.4	21.9	36.5
DE-SimplE	0.230	14.1	24.8	40.3
RoAN	0.231	13.9	23.7	38.4
SubEE	0.224	12.4	22.6	37.8
GLANet	0.235	13.6	23.9	37.4
T-GAE	0.217	12.4	22.6	36.7
DKGE	0.225	13.1	22.9	37.1
BoxTE	0.229	13.7	23.5	37.5
TKGR	**0.244**	**14.2**	**25.0**	**38.5**

two regularisation terms are not used in the objective function, while the remaining parameters are kept at their original values.

TABLE IV displays the findings from the pair of comparative tests conducted. The data in the table indicates that the regularization component exerts a more pronounced influence on the outcomes. For instance, the Mean Reciprocal Rank (MRR) and Hit@1 metrics enhance the model's performance by 1% and 1.8% respectively when time smoothing constraints and parameter regularization are applied, as opposed to the model's performance in their absence. Therefore, these two regularisation terms are more beneficial to the model.

G. Evolution of Relations

Investigations conducted thus far reveal that various models for temporal knowledge graph completion employ temporal data in distinct manners. The TA-TransE model combines relational and temporal information to acquire a relational representation that contains temporal information. The HyTE model is able to learn temporal representations of entities as well as interpersonal relations. TKGR in this study only represents the evolution of entities and continues to use static representations where relationships are concerned.

This section aims to confirm the rationality behind the TKGR model's development. we set up the experiments in two different ways: the first is to perform temporal rotation transformations only on relations, while entities remain static representations; the second is to perform rotation transformations on both entities and relations simultaneously. The ICEWS14 dataset is chosen as the experimental dataset, and the parameters of the model are assumed to be compatible with the parameter values described in Section 5.3. TABLE V presents the outcomes of the conducted experiment.

If we focus only on the temporal evolution of relations, then the performance of the model variant suffers in all metrics. On the other hand, if we focus on the temporal evolution of entities and relations simultaneously, the performance of the variant model is comparable to that of TKGR. The findings from the experimental outcomes lead to this inference. This could be associated with the observation that datasets typically contain a much larger quantity of entities compared to relationships.

As a result, the temporal evolution of entities is a more accurate indicator of the efficient use of temporal information. In addition, it appears that the nature of entities is more likely to change over time, whereas the nature of relationships tends to remain more consistent. This can be understood from real life experience. In partnerships, for example, the state of the same individual is likely to be different at different times before and after, whereas the opposite is true because of the nature of the relationship. TKGR focuses on the development of entities over time. This is done to preserve the simple nature of the model and the geometric significance of the changes that occur within the relations.

H. Parameter Adjustment

In this section, we focus on the regularisation coefficients λ_1 and λ_2. Identify and select the ideal parameters. We start by iterating the algorithm a thousand times. Subsequently, we

TABLE IV: EFFECT OF REGULARISATION TERMS ON THE DATASET ICEWS14

Loss Function	MRR	Hit@1	Hit@3	Hit@10
$L(\theta) + \lambda_1 L_r(\theta) + \lambda_2 L_s(\theta)$	0.568	45.9	63.6	74.8
$L(\theta)$	0.563	45.1	63.2	74.4

TABLE V: EXPERIMENTAL RESULTS FOR DIFFERENT VARIANT MODELS ON THE DATASET ICEWS14

Variations	MRR	Hit@1	Hit@3	Hit@10
$\boldsymbol{q}_{h,\tau} \otimes \boldsymbol{q}_r^{\triangleleft} \cdot \boldsymbol{q}_{t,\tau}$	0.568	45.9	63.6	74.8
$\boldsymbol{q}_h \otimes \boldsymbol{q}_{r,\tau}^{\triangleleft} \cdot \boldsymbol{q}_t$	0.563	45.1	63.2	74.4
$\boldsymbol{q}_{h,\tau} \otimes \boldsymbol{q}_{r,\tau}^{\triangleleft} \cdot \boldsymbol{q}_{t,\tau}$	0.563	45.1	63.2	74.4

assess the outcomes of the Hit@10 metric on the validation subset to ascertain the optimal parameters.

Figure 2 illustrates the impact of adjusting the regularization factor's magnitude on the model's performance using the ICEWS14 dataset. The findings indicate that optimal performance of the model is achieved when the parameter λ is set to the values of λ_1 and λ_2, with each being 0.01. As a result, λ_1 and λ_2 are both set to 0.01 in the trials described above.

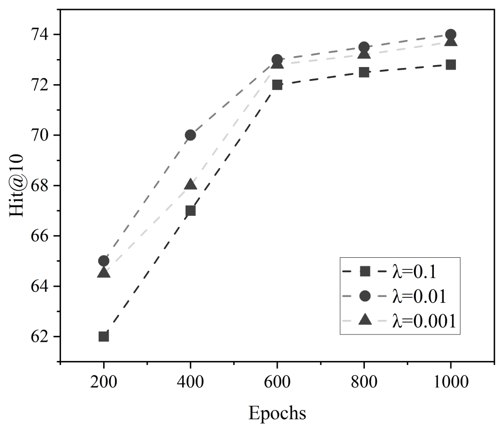

Fig. 2: The influence of regularization coefficient on the TKGR model.

VI. CONCLUSION

This research presents a learning methodology anchored in quaternion rotations for the portrayal of temporal knowledge graphs, enhancing the depiction of entity development and mutual connections compared to traditional complex number-based systems. The superior performance of the model is supported by benchmark dataset experiments. We plan to extend negative sampling strategies, which are crucial for training, to better adapt to the temporal dimension of knowledge graphs.

REFERENCES

[1] X. Zeng, X. Tu, Y. Liu, X. Fu, and Y. Su, "Toward better drug discovery with knowledge graph," *Current Opinion in Structural Biology*, vol. 72, pp. 114–126, 2022.

[2] J. Wang, B. Wang, J. Gao, X. Li, Y. Hu, and B. Yin, "Tdn: Triplet distributor network for knowledge graph completion," *IEEE Transactions on Knowledge & Data Engineering*, no. 01, pp. 1–13, 2023.

[3] C. Huang and Y. Zhong, "A network representation learning method fusing multi-dimensional classification information of nodes," *IAENG International Journal of Computer Science*, vol. 50, no. 1, pp. 94–105, 2023.

[4] C. Peng, F. Xia, M. Naseriparsa, and F. Osborne, "Knowledge graphs: Opportunities and challenges," *Artificial Intelligence Review*, pp. 1–32, 2023.

[5] Z. Li, Y. Zhao, Y. Zhang, and Z. Zhang, "Multi-relational graph attention networks for knowledge graph completion," *Knowledge-Based Systems*, vol. 251, p. 109262, 2022.

[6] F. Wang, Z. Zheng, Y. Zhang, Y. Li, K. Yang, and C. Zhu, "To see further: Knowledge graph-aware deep graph convolutional network for recommender systems," *Information Sciences*, vol. 647, p. 119465, 2023.

[7] W. Jin, B. Zhao, H. Yu, X. Tao, R. Yin, and G. Liu, "Improving embedded knowledge graph multi-hop question answering by introducing relational chain reasoning," *Data Mining and Knowledge Discovery*, vol. 37, no. 1, pp. 255–288, 2023.

[8] B. Zhou, X. Shen, Y. Lu, X. Li, B. Hua, T. Liu, and J. Bao, "Semantic-aware event link reasoning over industrial knowledge graph embedding time series data," *International Journal of Production Research*, vol. 61, no. 12, pp. 4117–4134, 2023.

[9] A. Bordes, N. Usunier, A. Garcia-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Proceedings of the 26th International Conference on Neural Information Processing Systems-Volume 2*, 2013, pp. 2787–2795.

[10] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28, no. 1, 2014.

[11] X. Wang, S. Lyu, X. Wang, X. Wu, and H. Chen, "Temporal knowledge graph embedding via sparse transfer matrix," *Information Sciences*, vol. 623, pp. 56–69, 2023.

[12] M. M. Li, K. Huang, and M. Zitnik, "Graph representation learning in biomedicine and healthcare," *Nature Biomedical Engineering*, vol. 6, no. 12, pp. 1353–1369, 2022.

[13] L. Bai, D. Chai, and L. Zhu, "Rlat: Multi-hop temporal knowledge graph reasoning based on reinforcement learning and attention mechanism," *Knowledge-Based Systems*, vol. 269, p. 110514, 2023.

[14] M. Nickel, L. Rosasco, and T. Poggio, "Holographic embeddings of knowledge graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[15] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *International Conference on Machine Learning*. PMLR, 2016, pp. 2071–2080.

[16] P. Baghershahi, R. Hosseini, and H. Moradi, "Self-attention presents low-dimensional knowledge graph embeddings for link prediction," *Knowledge-Based Systems*, vol. 260, p. 110124, 2023.

[17] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[18] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 701–710.

[19] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*. Springer, 2018, pp. 593–607.

[20] Y. Ma, V. Tresp, and E. A. Daxberger, "Embedding models for episodic knowledge graphs," *Journal of Web Semantics*, vol. 59, p. 100490, 2019.

[21] S. S. Dasgupta, S. N. Ray, and P. Talukdar, "Hyte: Hyperplane-based temporally aware knowledge graph embedding," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 2001–2011.

[22] A. Garcia-Duran, S. Dumančić, and M. Niepert, "Learning sequence encoders for temporal knowledge graph completion," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 4816–4821.

[23] G. Wan, Z. Zhou, Z. Zheng, and B. Du, "Sub-entity embedding for inductive spatio-temporal knowledge graph completion," *Future Generation Computer Systems*, vol. 148, pp. 240–249, 2023.

[24] J. Messner, R. Abboud, and I. I. Ceylan, “Temporal knowledge graph completion using box embeddings,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 7, 2022, pp. 7779–7787.

[25] Y. Xu, S. Sun, H. Zhang, C. Yi, Y. Miao, D. Yang, X. Meng, Y. Hu, K. Wang, H. Min *et al.*, “Time-aware graph embedding: A temporal smoothness and task-oriented approach,” *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 16, no. 3, pp. 1–23, 2021.

[26] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder–decoder for statistical machine translation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014, p. 1724.

[27] J. Wang, X. Lin, H. Huang, X. Ke, R. Wu, C. You, and K. Guo, “Glanet: temporal knowledge graph completion based on global and local information-aware network,” *Applied Intelligence*, vol. 53, no. 16, pp. 19 285–19 301, 2023.

[28] S. M. Kazemi and D. Poole, “Simple embedding for link prediction in knowledge graphs,” in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018, pp. 4289–4300.

[29] T. Jiang, T. Liu, T. Ge, L. Sha, B. Chang, S. Li, and Z. Sui, “Towards time-aware knowledge graph completion,” in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016, pp. 1715–1724.

[30] R. Goel, S. M. Kazemi, M. Brubaker, and P. Poupart, “Diachronic embedding for temporal knowledge graph completion,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 04, 2020, pp. 3988–3995.

[31] L. Bai, X. Ma, X. Meng, X. Ren, and Y. Ke, “Roan: A relation-oriented attention network for temporal knowledge graph completion,” *Engineering Applications of Artificial Intelligence*, vol. 123, p. 106308, 2023.

[32] X. Hou, R. Ma, L. Yan, and Z. Ma, “T-gae: A timespan-aware graph attention-based embedding model for temporal knowledge graph completion,” *Information Sciences*, vol. 642, p. 119225, 2023.

[33] X. Liu, J. Wu, T. Li, L. Chen, and Y. Gao, “Unsupervised entity alignment for temporal knowledge graphs,” in *Proceedings of the ACM Web Conference 2023*, 2023, pp. 2528–2538.

[34] S. R. Bhat, R. Bhat, and S. G. Bhat, “A comprehensive analysis of total and semi-total graphs.” *Engineering Letters*, vol. 32, no. 1, pp. 21–29, 2024.