Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.070493

ARTICLE

# Dynamic Knowledge Graph Reasoning Based on Distributed Representation Learning

**Qiuru Fu[1], Shumao Zhang[1], Shuang Zhou[1], Jie Xu[1,*], Changming Zhao[2], Shanchao Li[3] and Du Xu[1,*]**

[1]School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu, 611731, China

[2]School of Computer Science, Chengdu University of Information Technology, Chengdu, 610103, China

[3]Department of Computer Science and Engineering, University of North Texas, Denton, TX 76207, USA

*Corresponding Authors: Jie Xu. Email: xuj@uestc.edu.cn; Du Xu. Email: xudu@uestc.edu.cn

Received: 17 July 2025; Accepted: 01 October 2025; Published: 09 December 2025

**ABSTRACT:** Knowledge graphs often suffer from sparsity and incompleteness. Knowledge graph reasoning is an effective way to address these issues. Unlike static knowledge graph reasoning, which is invariant over time, dynamic knowledge graph reasoning is more challenging due to its temporal nature. In essence, within each time step in a dynamic knowledge graph, there exists structural dependencies among entities and relations, whereas between adjacent time steps, there exists temporal continuity. Based on these structural and temporal characteristics, we propose a model named "DKGR-DR" to learn distributed representations of entities and relations by combining recurrent neural networks and graph neural networks to capture structural dependencies and temporal continuity in DKGs. In addition, we construct a static attribute graph to represent entities' inherent properties. DKGR-DR is capable of modeling both dynamic and static aspects of entities, enabling effective entity prediction and relation prediction. We conduct experiments on ICEWS05-15, ICEWS18, and ICEWS14 to demonstrate that DKGR-DR achieves competitive performance.

**KEYWORDS:** Dynamic knowledge graph reasoning; recurrent neural network; graph convolutional network; graph attention mechanism

## 1 Introduction

Since Google first introduced the concept of Knowledge Graphs (KG) in 2012 [1], they have been widely applied in various real-world scenarios. Despite their large scale, existing KGs still suffer from sparsity and incompleteness, with many implicit facts not directly represented [2]. Knowledge Graph Reasoning (KGR) aims to address these problems by inferring missing knowledge from known facts, thereby completing the KG.

However, most existing KGR approaches focus only on static knowledge graphs, assuming that facts do not change over time. This overlooks the temporal nature of knowledge. Time is critical in representing knowledge, as facts are often only valid during specific time periods. For instance, the fact *(Country, president, Person)* holds only for a certain term, and different individuals may hold the position before and after this period. Without temporal information, answering the question "Who is the president of a given country?" may lead to ambiguous or incorrect results. Therefore, it is essential to incorporate temporal dimensions into KG modeling.

A Dynamic Knowledge Graph (DKG) can be regarded as a sequence of knowledge graphs evolving over time. As shown in Fig. 1, each timestamp corresponds to a distinct knowledge graph. A fact at time step $t$ is represented as a triple $(s_t, r_t, o_t)$, where $s_t$, $r_t$, and $o_t$ denote the subject, relation, and object, respectively. Dynamic Knowledge Graph Reasoning (DKGR) aims to infer missing or future facts. Depending on the inference target, DKGR can be categorized into:

- **Entity Prediction:** Given $(?, r_t, o_t)$, predict the missing head entity; or given $(s_t, r_t, ?)$, predict the missing tail entity at time $t$.
- **Relation Prediction:** Given $(s_t, ?, o_t)$, predict the missing relation at time $t$.

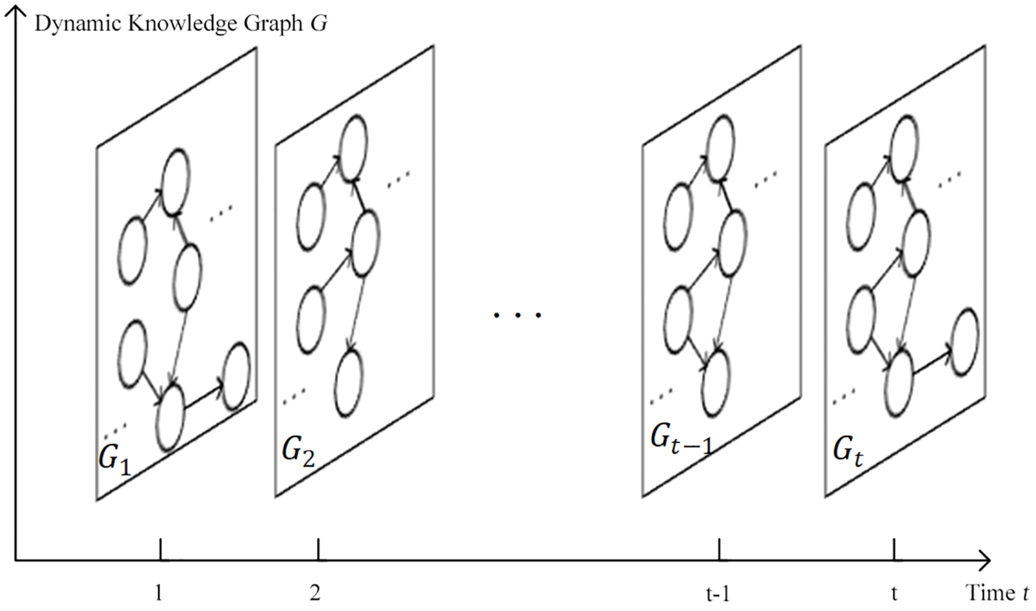

**Figure 1:** Illustration of a dynamic knowledge graph

Compared to static KGR, DKGR is more challenging due to the temporal changes in graph structure. MLPs [3] rely on logical rules and ignore temporal dynamics. TTransE [4] incorporates time into relation embeddings but follows the static reasoning paradigm of TransE [5], thus overlooking rich historical information. CyGNet [6] considers historical facts but only repeated ones. EvoKG [7] models both temporal and structural information but neglects static entity attributes. Both tensor decomposition-based and neural network-based DKGR models face limitations in representation capacity.

In this work, we model DKGs as a sequence of evolving knowledge graphs. At each time step, entities are connected by relations, forming a structure with strong dependencies. The graph structure encodes how entities are semantically linked through relations, and these dependencies provide essential contextual information for representation learning. Capturing such structural dependencies ensures that the learned embeddings reflect not only individual entities but also their interconnected roles in the knowledge graph. We refer to the sequential correlations of entity and relation representations across adjacent time steps as temporal continuity. Because the evolution of knowledge graphs typically follows smooth and progressive trends, modeling temporal continuity enables the representation learning process to capture dynamic patterns. In addition to temporal dynamics, we also consider static attributes of entities that do not change over time. These attributes, such as type, category, or descriptive features, serve as stable semantic signals that complement the dynamic structural and temporal information, thereby enhancing the expressiveness of the learned representations. Based on these ideas, we propose a distributed representation learning model called DKGR-DR, designed to effectively capture structural dependencies, temporal continuity, and static attributes for DKGR tasks.

The main contributions of this paper are summarized as follows:

- We propose DKGR, a unified framework for both entity prediction and relation prediction tasks in dynamic knowledge graphs. The framework is composed of three major components: a representation learning module, a static attribute fusion module, and a prediction module, which work together to capture both temporal dynamics and static semantic information.
- To effectively model the evolving nature of knowledge graphs, we design a representation learning module that integrates recurrent neural networks with graph neural networks. This hybrid design enables the model to capture dynamic temporal dependencies through recurrent neural networks, while simultaneously learning static structural patterns of entities and relations from the graph topology using relational graph convolutional network.
- In order to enrich entity and relation representations with auxiliary information, we construct three types of static attribute graphs. These attribute graphs encode complementary semantic information such as entity categories, textual descriptions, and relation hierarchies, thereby providing additional contextual signals beyond the temporal interactions in the dynamic knowledge graph.
- We introduce a static attribute fusion module that leverages a relation-aware graph attention mechanism to integrate structural and attribute-based signals. By dynamically adjusting attention weights according to relation types, this module effectively balances heterogeneous sources of information, leading to more robust and expressive entity and relation representations.

This paper is organized as follows. Section 2 reviews related works on knowledge graph reasoning, highlighting the strengths and limitations of existing static and dynamic methods. Section 3 formulates the problem by introducing the task settings, notations, and objectives of entity and relation prediction in dynamic knowledge graphs. Section 4 presents the proposed DKGR-DR model in detail, including the representation learning module, the construction of static attribute graphs, and the relation-aware graph attention mechanism for attribute fusion. Section 5 reports the experimental setup, datasets, baselines, evaluation metrics, and results, along with ablation studies to validate the effectiveness of the model. Finally, Section 6 concludes the work by summarizing the key contributions.

## 2 Related Works

Static knowledge graph reasoning methods can be broadly categorized into three classes. Translation-based models [5,8–12] model relations as translation operations in the embedding space, assuming that the tail entity should be close to the head entity after applying the relation-specific translation. Tensor decomposition-based models [13–17] reconstruct KGs by factorizing high-dimensional adjacency tensors into low-dimensional latent representations. Neural network-based models [18–21] directly learn a plausibility scoring function for factual triples using deep neural architectures. Beyond these general categories, several graph neural network approaches have been proposed for multi-relational graphs: Schlichtkrull et al. [22] introduced the Relational Graph Convolutional Network (R-GCN) tailored for KG structures, while Veličković et al. [23] proposed the Graph Attention Network (GAT), which leverages attention mechanisms to enhance representation learning on graph-structured data.

While static knowledge graph reasoning methods have achieved remarkable success in capturing structural dependencies within fixed relational data, they inherently overlook the temporal dimension of knowledge. However, in many real-world applications, facts are not static but evolve over time, with entities and relations continuously emerging, disappearing, or changing their states. To address this limitation, research has shifted toward dynamic knowledge graph (DKG) reasoning, which explicitly incorporates temporal information into the inference process.

Representative models have approached temporal knowledge graph reasoning from different perspectives. TTransE [4], as one of the earliest temporal extensions of TransE [5], augments the score function with temporal parameters to obtain time-aware embeddings. Building on this idea, HyTE [24] adapts the TransH [8] framework by projecting entities onto time-specific hyperplanes, thereby allowing the representation space to evolve with temporal contexts. DE-SimplE [25] further advances temporal modeling by encoding entities across multiple time steps using historical embeddings to capture long-term dependencies and entity evolution. These embedding-based extensions demonstrate the importance of explicitly incorporating temporal signals, but they remain limited in capturing complex structural patterns and often struggle with scalability.

Another line of research integrates sequential and recurrent architectures. TA-DistMult [26] employs recurrent neural networks to model temporal dependencies, enabling relation embeddings to adapt dynamically over time. RE-NET [27] extends this paradigm by sequentially encoding historical contexts of entities and relations, effectively leveraging temporal sequences to improve predictive performance. Complementary to these approaches, CyGNet [6] focuses on recurring temporal patterns in entity prediction, addressing the cyclic or periodic nature of temporal interactions. These models highlight the effectiveness of sequence modeling for temporal reasoning, yet they often overlook the integration of static semantic attributes and fine-grained relation-specific signals.

More recently, advanced frameworks have been proposed that combine temporal reasoning with attention or multimodal information. For example, TANGO [28] integrates graph convolutional and temporal convolutional networks with gating and attention mechanisms to jointly capture relational and temporal dynamics. In the financial domain, KGTransformer [29] applies attention-based GNNs to contextual signals for enhanced link prediction and investment strategies. TD-RKG [30] further explores dynamic fusion by integrating recurrent, implicit, and attention-based encoding to improve temporal reasoning. While these approaches achieve strong performance, they tend to treat structural information and static attributes in isolation, limiting the robustness and expressiveness of learned representations.

In summary, existing studies provide valuable strategies for incorporating temporal information into knowledge graph reasoning, ranging from temporal embedding extensions to recurrent architectures and attention-based models. However, most of them do not fully integrate temporal dynamics, graph structural patterns, and static semantic attributes in a unified framework. This gap motivates our proposed DKGR-DR, which leverages RNNs and GNNs for temporal and structural modeling, constructs multiple static attribute graphs, and introduces a relation-aware graph attention module to fuse heterogeneous information effectively.

## 3 Problem Formulation

A dynamic knowledge graph (DKG) can be represented as a sequence of static knowledge graphs: $\mathcal{G} = \{G_1, G_2, \ldots, G_t, \ldots\}$. At each timestamp $t$, a knowledge graph $G_t$ consists of a set of factual triples $(s_t, r_t, o_t)$, where $s_t$ and $o_t$ are the head and tail entities, respectively, and $r_t$ is the relation. Let $\mathcal{E}$ denote the set of entities, $\mathcal{R}$ the set of relations, and $\mathcal{F}_t$ the set of facts at time $t$.

We assume that reasoning over the facts at time $t$ depends only on the most recent $m$ snapshots of the knowledge graph sequence: $\mathcal{G}_{t-m+1:t} = \{G_{t-m+1}, \ldots, G_t\}$, which we refer to as the historical knowledge graphs.

A representation learning algorithm aims to encode the historical knowledge graphs into distributed embeddings for entities and relations. Specifically, let $E_t \in \mathbb{R}^{|\mathcal{E}|\times d}$ denote the entity embedding matrix and $R_t \in \mathbb{R}^{|\mathcal{R}|\times d}$ the relation embedding matrix at time $t$, where $d$ is the embedding dimension.

Based on these representations, we define two reasoning tasks:

- **Task 1: Entity Prediction.** Given a query triple $(s_t, r_t, ?)$, the goal is to compute the conditional probability vector over all possible tail entities, using the head entity, the relation, and the historical knowledge graphs:

$$p\left(o|s, r, \mathcal{G}_{t-m+1:t}\right) = p\left(o|s, r, E_t, R_t\right). \tag{1}$$

- **Task 2: Relation Prediction.** Given a query triple $(s_t, ?, o_t)$, the goal is to compute the conditional probability vector over all possible relations, using the head and tail entities and the historical knowledge graphs:

$$p\left(r|s, o, \mathcal{G}_{t-m+1:t}\right) = p\left(r|s, o, E_t, R_t\right). \tag{2}$$

## 4 The DKGR-DR Model

DKGR-DR models structural dependency, temporal continuity, and static attributes of entities for dynamic knowledge graph reasoning. It supports both entity prediction and relation prediction. As shown in Fig. 2, the model consists of a distributed representation learning module, a static attribute fusion module, and a prediction module.

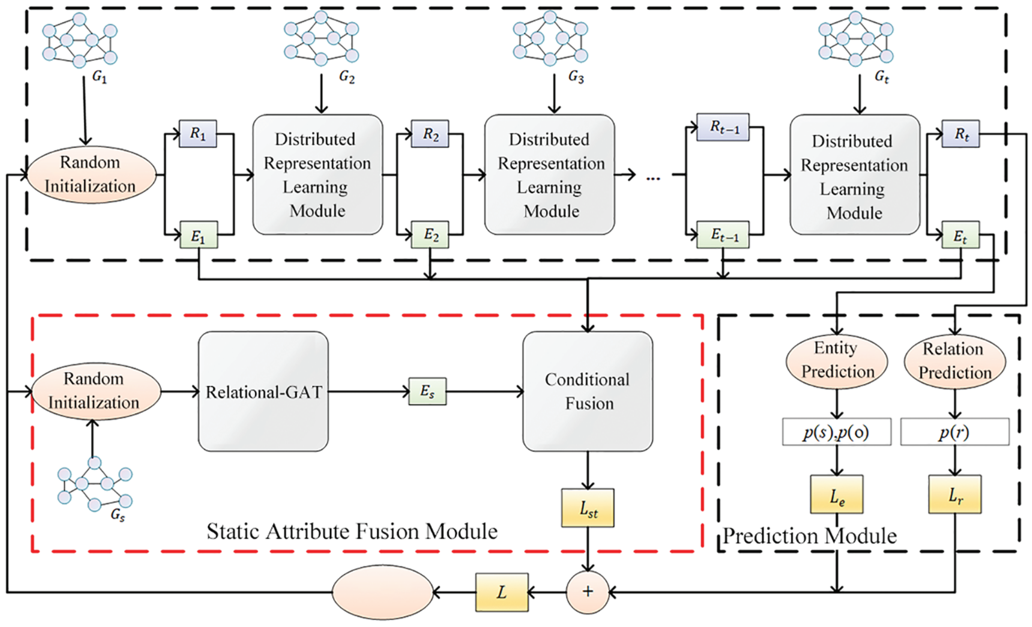

**Figure 2:** Architecture of the DKGR-DR model

### *4.1 Distributed Representation Learning Module*

This module is designed to jointly capture the structural and temporal dynamics of dynamic knowledge graphs. Instead of treating each snapshot independently, DKGR-DR learns how entities and relations evolve over time while preserving graph connectivity information at each step. To achieve this, we combine recurrent neural networks with relational graph convolutional networks. This hybrid design allows the model to benefit from the complementary strengths of both sequence modeling and graph representation learning. The module structure is illustrated in Fig. 3.

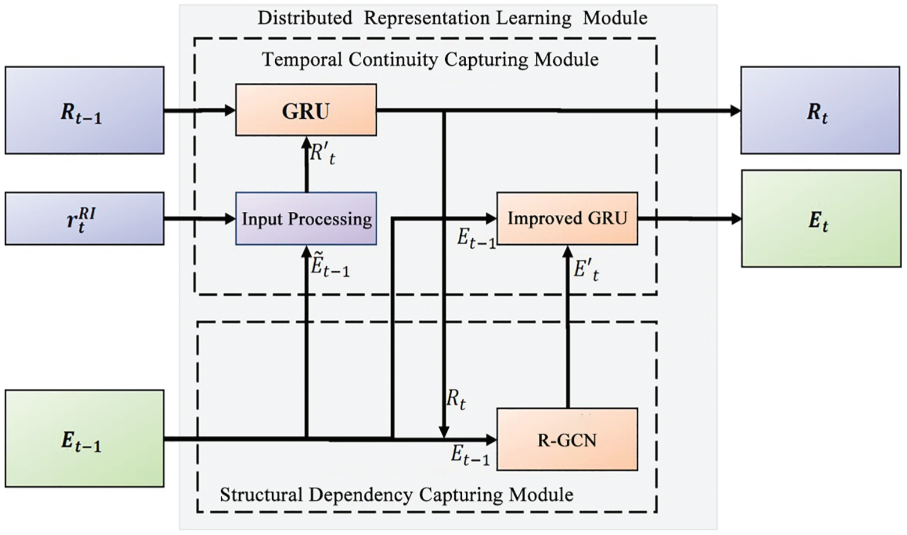

**Figure 3:** Structure of the distributed representation module

In Fig. 3, the distributed embedding matrices of entities and relations at time step $t-1$, denoted as $E_{t-1}$ and $R_{t-1}$, and the randomly initialized relation vectors $r_t^{RI}$ in the knowledge graph $G_t$ at time $t$ are considered. Given $R_I$ as input, the distributed representation learning unit can learn the distributed embedding matrices of entities and relations at time $t$, respectively, denoted as $E_t$ and $R_t$. The distributed representation learning unit mainly consists of the following three components:

1. Structure dependency capture module based on R-GCN. By feeding $E_{t-1}$ and $R_{t-1}$ into an $\omega$-layer R-GCN network, the structural dependencies between entities and relations can be learned. The output serves as the input matrix $E'_t$ for the enhanced GRU module. Implementation details are provided in Section 4.1.1.
2. Temporal continuity and relation learning module based on GRU. In Fig. 3, $\tilde{E}_{t-1}$ (i.e., $\{E^r_{t-1}\}$) represents the set of entity embeddings at time $t-1$ associated with relation $r$ at time $t$. After processing $\tilde{E}_{t-1}$ and $r_t$ through the input processing module, the input matrix $R'_t$ for the GRU module is obtained. By feeding $R'_t$ and $R_{t-1}$ into the GRU network, historical information carried by temporal continuity can be learned. The relation embedding matrix at time $t$, which contains historical information, is denoted as $R_t$. Implementation details are provided in Section 4.1.2.
3. Temporal continuity and entity learning module based on enhanced GRU. Similar to the GRU-based temporal continuity and relation learning module, the entity learning module takes $E'_t$ and $E_{t-1}$ as inputs. $E'_t$ is learned from the structure dependency capture module based on R-GCN. After passing through the enhanced GRU, the distributed entity embedding matrix $E_t$ at time $t$ is obtained, which contains rich historical information. Implementation details are provided in Section 4.1.2.

### *4.1.1 Structural Dependency Module*

At each timestamp, entities interact through diverse relations, forming a graph structure that encodes local dependencies. We employ R-GCN to model these structural patterns. R-GCN explicitly distinguishes different relation types when aggregating neighborhood information, ensuring that semantic differences between relations (e.g., *ally* vs. *attack*) are preserved. Moreover, the self-loop mechanism ensures that even isolated entities can maintain meaningful representations. After several layers of R-GCN, we obtain a structurally enriched entity embedding matrix $E'_t$, where each entity representation incorporates both its own features and aggregated information from its relational context.

As illustrated in Fig. 4, the input of the R-GCN-based structural dependency capturing module consists of the relation embedding matrix $R_t$ at time step $t$ and the entity embedding matrix $E_{t-1}$ from the previous time step. Let the entity to be updated be denoted as $e_t$. R-GCN partitions the entity matrix $E_{t-1}$ according to the categories of relations in $R_t$ into $N$ types of entities, represented as $\{r_t^1, r_t^2, \ldots, r_t^N\}$, along with the target entity $e_t$. Furthermore, for each relation type, entities are divided into incoming and outgoing ones according to the direction of relation links, resulting in $2N$ types of entities: $\{r_t^1(\text{in}), r_t^1(\text{out}), r_t^2(\text{in}), r_t^2(\text{out}), \ldots, r_t^N(\text{in}), r_t^N(\text{out})\} \cup \{e_t\}$. After classification, the $2N$ types of entities are transformed through a relation-specific transformation function, while $e_t$ is updated through a self-loop operation. Finally, the transformed features are aggregated and passed through an activation function for nonlinear transformation, yielding the updated feature vector of entity $e_t$. It classifies neighboring entities based on relation types and then aggregates them to update entity representations:

$$e_{i,t}^{l+1} = \sigma\left(\sum_{r_t \in R_t} \sum_{j \in N_i^r} \frac{1}{c_{i,r_t}} W_r^l e_{j,t}^l + W_0^l e_{i,t}^l\right) \tag{3}$$

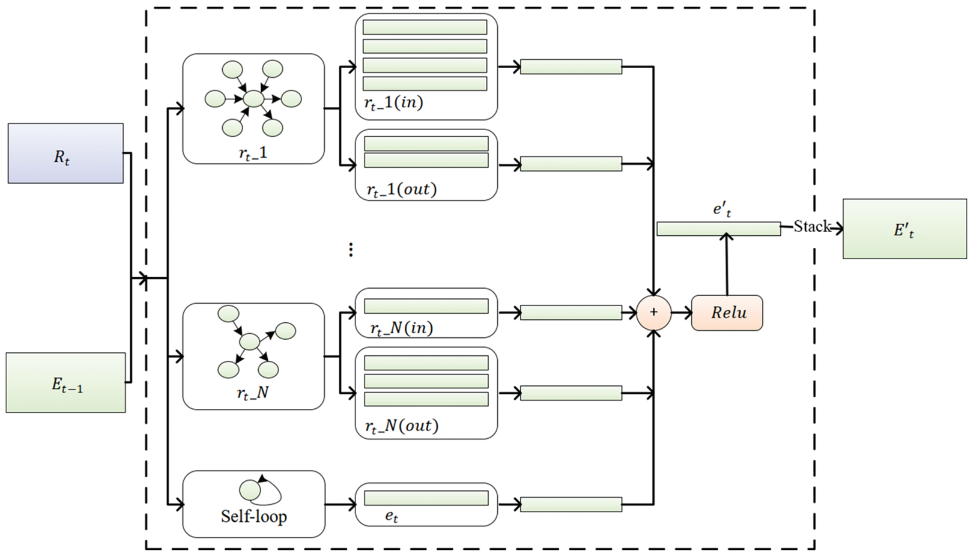

**Figure 4:** Structure of the structural dependency module

Here, $e_{i,t}^l \in \mathbb{R}^d$ is the embedding of entity $i$ at time $t$ and layer $l$ (e.g., if $d = 100$, then $e_{i,t}^l = [0.2 - 0.1 \ldots 0.05]$). $R_t$ is the set of relations at time $t$ (e.g., $\{interact, cooperate, attack\}$). $N_i^r$ denotes the neighbors of $i$ connected via relation $r$ (e.g., if $i$ = USA and $r$ = ally, then $N_i^r = \{\text{UK}, \text{France}\}$). $W_r^l \in \mathbb{R}^{d \times d}$

and $W_0^l \in \mathbb{R}^{d\times d}$ are weight matrices. $c_{i,r_t}$ is a normalization factor (scalar). This formulation allows the model to update all entities, including those without neighbors, using self-loop mechanisms.

From Eq. (3), it can be observed that R-GCN is able to aggregate relational information and update the feature vector of any entity, regardless of whether it serves as the head or the tail entity, because it explicitly distinguishes the directionality of relations. For entities that are not involved in any facts, namely those without neighboring entities, self-loop operations can still be applied to update their feature vectors. In practice, R-GCN derives new representations of entities at each time step based on the observed facts among them, while the self-loop operation can be regarded as a form of self-learning for entities.

By passing each entity in $E_{t-1}$ through an $\omega$-layer R-GCN, we obtain entity feature vectors $e_t'$ that encode structural dependencies. Stacking all $e_t'$ at time step $t$ yields the structural dependency matrix $E_t'$. This matrix is subsequently employed to learn temporally continuous distributed embeddings of entities, which ultimately facilitate dynamic knowledge graph reasoning.

### *4.1.2 Temporal Continuity Module*

While R-GCN captures structural dependencies within a single snapshot, temporal reasoning requires tracking how entities and relations evolve over time. To this end, we design a temporal continuity module based on gated recurrent units (GRUs). For relations, the GRU takes as input both the aggregated embeddings of related entities from the previous timestamp and a learnable base embedding, enabling it to update relation states in a history-aware manner. For entities, we use a simplified GRU that adaptively blends past embeddings with current structural features, effectively controlling how much historical information is retained or forgotten. This mechanism allows the model to capture both short-term dynamics (e.g., sudden events) and long-term trends (e.g., stable alliances).

As shown in Fig. 5, the input processing module takes as input the distributed embedding matrix of entities $E_{t-1}$ from the previous time step and the randomly initialized relation vectors $r_t$ from the knowledge graph $G_t$ at time step $t$. The relation input $R_t'$ for the GRU module is then generated by the relation initializer $RI$, where $R_t'$ is constructed by stacking the relation vectors $r_t'$ at time step $t$. And $r_t'$ is computed as follows:

$$r_t' = \left[\text{pooling}\left(E_{t-1,\mathcal{E}_{r,t}}\right); r_t^{RI}\right] \tag{4}$$

Here, $\text{pooling}() \in \mathbb{R}^d$ denotes mean pooling, which aggregates the embeddings of entities connected to relation $r$ at time step $t-1$. $r_t^{RI} \in \mathbb{R}^d$ is the randomly initialized embedding of relation $r$ at time $t$. $r_t' \in \mathbb{R}^{2d}$, and stacking across all relations gives $R_t' \in \mathbb{R}^{|R|\times 2d}$.

The relation embeddings are updated as:

$$R_t = \text{GRU}(R_t', R_{t-1}) \tag{5}$$

The output $R_t$ is obtained from the GRU module after selectively forgetting and retaining historical information. In the GRU, the reset gate is responsible for short-term memory, determining how much past information to preserve or discard, while the update gate controls long-term memory, deciding which information should be replaced. The inputs $R_t'$ and $R_{t-1}$ are combined through the $\tanh(\cdot)$ function to produce a candidate representation $\tilde{R}_t$, where the influence of $R_{t-1}$ on $\tilde{R}_t$ is regulated by the reset gate. Finally, the update gate integrates $R_{t-1}$ and $\tilde{R}_t$ to form the output $R_t$. The combination strategy is determined by the update gate: if the gate value is 0, then the previous state $R_{t-1}$ is completely forgotten and $R_t = \tilde{R}_t$; if the gate value is 1, then the previous state $R_{t-1}$ is fully retained.

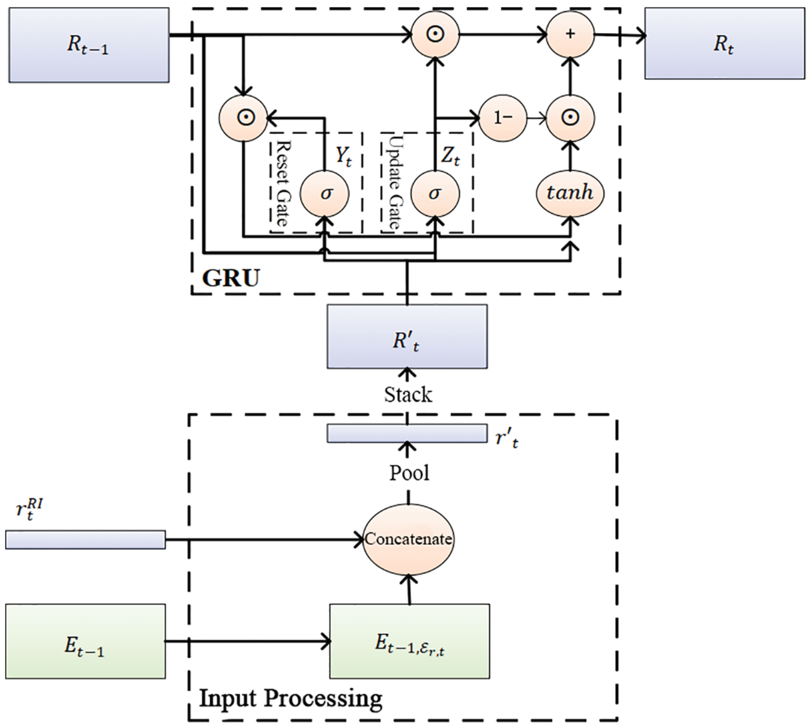

**Figure 5:** Structure of the temporal continuity module

Given that the entity set is usually orders of magnitude larger than the relation set and that entity representations exhibit relatively smooth temporal variations, employing a full GRU for all entities would lead to prohibitive computational cost and vanishing gradients. To mitigate those issues, we adopt a simplified GRU that retains only the update gate. This design effectively balances the preservation of historical information with the incorporation of current structural features, while maintaining training stability in large-scale dynamic knowledge graphs. Entity embeddings are updated using a simplified GRU with fewer parameters:

$$U_t = \sigma(W_U E'_t + b_U) \tag{6}$$

$$E_t = U_t \odot E_{t-1} + (1 - U_t) \odot E'_t \tag{7}$$

Here, $\odot$ is the element-wise product; $W_U$ and $b_U$ are trainable parameters. $E_t \in \mathbb{R}^{|\mathcal{E}|\times d}$, with $|\mathcal{E}|$ entities. This mechanism selectively retains or forgets temporal information.

## 4.2 Static Attribute Fusion Module

Dynamic interactions alone may not fully capture the semantic characteristics of entities. For example, two countries with similar political systems may exhibit correlated behaviors even if they rarely co-occur in temporal graphs. To address this, we introduce a static attribute fusion module.

### 4.2.1 Static Attribute Graph Construction

We incorporate simple static attributes such as entity type and country. In ICEWS14, many entity names include country information (e.g., *Government_(Nigeria)*). Such entities are represented with type and country triples, e.g., (Government, isA, Government), (Government, country, Nigeria). For entities like

*France*, we use (France, is, France). We construct a static attribute graph $G_s$ accordingly for each ICEWS subset. These additional graphs provide stable, complementary signals that are invariant across time.

### *4.2.2 Static Attribute Learning*

To effectively learn from attribute graphs, we introduce a relation-aware graph attention network (Relational-GAT). Unlike conventional GATs that treat all edges equally, Relational-GAT dynamically adjusts attention scores based on relation types, thereby prioritizing more informative attribute connections. This produces enhanced static embeddings $E_s$, where entities are contextualized not only by their temporal interactions but also by their semantic attributes.

$G_s$ is a small-scale knowledge graph with only three relation types. We design a relation-aware GAT, called Relational-GAT, to assign importance scores:

$$\alpha_{ij} = \frac{\exp(\mathrm{LeakyReLU}(a^T[We_{s,i} \| We_{s,j} \| Wr_{i,j}]))}{\sum_{k\in N_i} \exp(\mathrm{LeakyReLU}(a^T[We_{s,i} \| We_{s,k} \| Wr_{i,k}]))} \tag{8}$$

Here, $e^l_{s,i} \in \mathbb{R}^d$ is the embedding of entity $i$ in $G_s$. $Wr_{i,j} \in \mathbb{R}^d$ is the relation embedding (e.g., "isA").

When aggregating entity features, the Relational-GAT does not include an additional self-loop module for learning the entity's own features. This is because, during the construction of $G_s$, the neighbors of each entity are derived from the entity itself and already contain its feature information. After learning the relation-specific importance coefficients $\alpha_{ij}$ between two entities and ignoring self-loops. The entity update is:

$$e^{l+1}_{s,i} = \sigma\left(\sum_{j\in N_i} \alpha_{ij} W_e e^l_{s,j}\right) \tag{9}$$

Here, $\alpha_{ij} \in \mathbb{R}$ denotes the importance coefficient of neighbor entity $e_j \in \mathbb{R}^d$ with respect to relation $r_{ij} \in \mathbb{R}^d$ for entity $e_i \in \mathbb{R}^d$; $\exp(\cdot)$ represents the exponential operation; $\mathrm{LeakyReLU}(\cdot)$ is the nonlinear activation function; $s^\top \in \mathbb{R}^{2d}$ denotes the transpose of the weight vector; $W \in \mathbb{R}^{d\times d}$ is a learnable linear transformation matrix; $||$ indicates vector concatenation; $N_i$ is the set of all neighbor entities of $e_i$.

Unlike other graph neural networks that update only node representations while ignoring edge updates, the Relational-GAT-based static attribute capturing module not only formulates an update rule for entities but also designs a feed-forward neural network to update relation representations. The relation update uses a feedforward network:

$$r^{l+1}_i = \sigma(W_r r^l_i) \tag{10}$$

In the equation, $r^{l+1}_i$ denotes the representation of relation $r_i$ at layer $l+1$; $\sigma(\cdot)$ is the sigmoid activation function; $W_b \in \mathbb{R}^{d\times d}$ is the linear transformation matrix for relation updates; $r^l_i$ denotes the representation of relation $r_i$ at layer $l$. All entity embeddings in $G_s$ are stacked into matrix $E_s$.

### *4.2.3 Static Attribute Fusion*

Finally, DKGR-DR fuses static and dynamic representations in a principled manner. The key motivation is that static attributes provide stable semantic anchors, whereas dynamic embeddings capture time-sensitive interactions and may vary significantly across timestamps. Simply concatenating the two types of embeddings is insufficient, as this may lead to inconsistent representations where temporal fluctuations override static semantics. To address this issue, DKGR-DR enforces a geometric alignment constraint that regulates the angle between static and dynamic embeddings over time.

Concretely, the model encourages the dynamic embedding of an entity to remain close to its corresponding static embedding in the vector space, but without forcing them to be identical. This design ensures that while dynamic representations are free to evolve according to temporal signals (e.g., changing alliances or sudden events), they are still guided by invariant semantic properties (e.g., a country always being a *government* or belonging to a certain region). This balance improves robustness to noisy or sparse temporal interactions and enhances interpretability, because the dynamic embedding trajectories remain consistent with domain knowledge encoded in static attributes.

Formally, we constrain the angle between static and dynamic embeddings. Let $\theta_\tau$ denote the maximum allowable angle deviation at lag $\tau$:

$$\theta_\tau = \min(\gamma\tau, 90^\circ), \tag{11}$$

where $\gamma$ is a hyperparameter controlling how fast the constraint is relaxed. At earlier timestamps, the alignment is strict, ensuring that static information strongly influences the embeddings. As time progresses, the constraint gradually loosens, allowing dynamic information to dominate when necessary.

The cosine similarity between the static embedding $e_{s,i}$ and the dynamic embedding $e_{t-m+\tau,i}$ should therefore satisfy:

$$\cos(e_{s,i}, e_{t-m+\tau,i}) \geq \cos(\theta_\tau). \tag{12}$$

This guarantees that the two embeddings are never too far apart in angular space, even if the dynamic embedding drifts due to temporal changes. The corresponding fusion loss at time $t$ is defined as:

$$L_{st}(\tau) = \sum_{i=0}^{|\mathcal{E}|-1} \max\Big(\cos(e_{s,i}, e_{t-m+\tau,i}) - \cos(\theta_\tau), 0\Big), \tag{13}$$

where $e_{s,i} \in \mathbb{R}^d$ and $e_{t-m+\tau,i} \in \mathbb{R}^d$ denote the static and dynamic embeddings of entity $i$, respectively, at the corresponding time steps; $\theta_\tau$ is the angular threshold at relative time $\tau$. This loss penalizes violations of the angular constraint across entities, ensuring geometric consistency between static and dynamic embeddings. Aggregating over the most recent $m$ timestamps gives the overall fusion loss:

$$L_{st} = \sum_{\tau=0}^{m} L_{st}(\tau). \tag{14}$$

Intuitively, static embeddings act as semantic "anchors," preventing entity trajectories from drifting too far in representation space, whereas the gradual relaxation of constraints allows flexibility for temporal adaptation. As a result, the fusion mechanism enables DKGR-DR to capture both the stability of static knowledge and the variability of temporal dynamics within a unified framework.

### *4.3 Loss Function and Optimization*

After obtaining the predicted probabilities for entity and relation predictions, we define the loss functions to optimize the model.

Entity prediction is treated as a multi-label classification problem. Let $y_t^e \in \{0,1\}^{|\mathcal{E}|}$ denote the label vector for entity prediction at time $t$, where $y_{t,i}^e = 1$ if entity $i$ appears as the object in a fact at time $t$, and 0 otherwise. Let $p_i(o \mid s, r, E_t, R_t)$ denote the predicted probability of entity $i$ being the object given the subject $s$, relation $r$, and the entity and relation embeddings $E_t \in \mathbb{R}^{|\mathcal{E}|\times d}$ and $R_t \in \mathbb{R}^{|\mathcal{R}|\times d}$ at time $t$. The entity prediction loss is then:

$$L_e = \sum_{t=0}^{T-1} \sum_{(s,r,o,t)\in\mathcal{F}_t} \sum_{i=0}^{|\mathcal{E}|-1} y_{t,i}^{e} \log p_i(o \mid s, r, E_t, R_t), \tag{15}$$

where $\mathcal{F}_t$ denotes the set of observed facts at time $t$, $T$ is the total number of timestamps, and $d$ is the embedding dimension.

Similarly, relation prediction is formulated as a multi-label classification problem. Let $y_t^r \in \{0,1\}^{|\mathcal{R}|}$ denote the label vector for relation prediction at time $t$, and $p_i(r \mid s, o, E_t, R_t)$ denote the predicted probability of relation $i$ given the subject $s$ and object $o$ at time $t$. The relation prediction loss is defined as:

$$L_r = \sum_{t=0}^{T-1} \sum_{(s,r,o,t)\in\mathcal{F}_t} \sum_{i=0}^{|\mathcal{R}|-1} y_{t,i}^{r} \log p_i(r \mid s, o, E_t, R_t), \tag{16}$$

where $|\mathcal{R}|$ is the total number of relations.

Finally, the total loss function combines the entity prediction loss, relation prediction loss, and the static attribute fusion loss:

$$L = \lambda_1 L_e + \lambda_2 L_r + \lambda_3 L_{st}, \tag{17}$$

where $\lambda_1$, $\lambda_2$, and $\lambda_3$ are hyperparameters controlling the relative importance of each component in the overall optimization.

## 5 Experiments

To evaluate the effectiveness of the proposed DKGR-DR model, we conduct comparative experiments against both static and dynamic knowledge graph reasoning models across three dynamic knowledge graph datasets. For code requests or further information, please contact the corresponding author.

### *5.1 Experimental Setup*

#### *5.1.1 Datasets*

We utilize three publicly available dynamic knowledge graph datasets: ICEWS05-15, ICEWS18, and ICEWS14 [31]. Table 1 summarizes the statistics of these datasets. The time interval indicates that, at each time step, all events or states occurring on that day are recorded.

**Table 1:** Statistics of datasets used in experiments

| Dataset | #Entities | #Relations | Train | Valid | Test | Time interval |
|---|---|---|---|---|---|---|
| ICEWS05-15 | 10,094 | 251 | 368,868 | 46,302 | 46,159 | 24 h |
| ICEWS18 | 23,033 | 256 | 373,018 | 45,995 | 49,545 | 24 h |
| ICEWS14 | 6869 | 230 | 74,845 | 8514 | 7371 | 24 h |

#### *5.1.2 Evaluation Metrics*

We employ standard evaluation metrics for knowledge graph reasoning: Mean Reciprocal Rank (MRR) and Hits@N (N = 1, 3, 10). Mean Rank (MR) is generally considered less informative and is thus omitted.

### *5.1.3 Experimental Environment*

All experiments are implemented in Python using the PyTorch framework. The hardware environment includes an AMD Ryzen 7 5800U processor with 16 cores at 1.90 GHz and an NVIDIA GeForce RTX 3080 GPU.

### *5.1.4 Hyperparameters*

The hyperparameters are set as follows: embedding dimension $d$ = 200; R-GCN layers $\omega$ = 2; learningate = 0.001; batch size = 64; epochs = 500; margin $\gamma$ = 10; Relational-GAT layers $n$ = 2. The history length $m$ varies per dataset: ICEWS14 ($m$ = 3), ICEWS18 ($m$ = 6), and ICEWS05-15 ($m$ = 10). $\lambda_1$ = 0.6, $\lambda_2$ = 0.2, and $\lambda_3$ = 0.2. The Adam optimizer is employed for training. These settings were determined through extensive experiments on the validation split of each dataset (as listed in Table 1), and the final reported results are obtained on the disjoint test set.

## ***5.2 Experimental Results and Analysis***

To validate the effectiveness of DKGR-DR, we compare it with both static and dynamic knowledge graph reasoning models. Static models include ComplEx [16], R-GCN [22], ConvE [18], and RotatE [32]. Dynamic models encompass HyTE [24], TA-DistMult [26], RE-NET [27], CyGNet [6], T-GAP [33], EvoKG [7], TANGO [28], KGTransformer [29] and TD-RKG [30]. For models without publicly available code, we adopt the results reported in their respective papers.

### *5.2.1 Entity Prediction Results*

The results of the entity prediction task are presented in Tables 2 and 3. All evaluation metrics are reported as percentages, with the "%" symbol omitted for clarity. The best-performing results for each metric are shown in **bold**, while the second-best results are <u>underlined</u>. DKGR-DR(-S) denotes the variant of our model without the static attribute fusion module.

**Table 2:** Entity prediction results (I)

| **Model** | **ICEWS05-15** | | | | **ICEWS18** | | | |
|---|---|---|---|---|---|---|---|---|
| | **MRR** | **Hits@1** | **Hits@3** | **Hits@10** | **MRR** | **Hits@1** | **Hits@3** | **Hits@10** |
| ComplEx | 20.26 | 6.66 | 26.43 | 47.31 | 15.45 | 8.04 | 17.19 | 30.73 |
| R-GCN | 27.13 | 18.83 | 30.41 | 43.16 | 15.05 | 8.13 | 16.49 | 29.00 |
| ConvE | 31.40 | 21.56 | 35.70 | 50.96 | 22.56 | 13.63 | 25.41 | 41.43 |
| RotatE | 19.01 | 10.42 | 21.35 | 36.92 | 11.63 | 6.47 | 12.31 | 28.03 |
| HyTE | 16.05 | 6.53 | 20.20 | 34.72 | 7.41 | 3.10 | 7.33 | 16.01 |
| TA-DistMult | 27.51 | 17.57 | 31.46 | 47.32 | 16.42 | 8.60 | 18.13 | 32.51 |
| RE-NET | 23.91 | 15.99 | 26.63 | 42.70 | 26.81 | 16.43 | 30.58 | 45.92 |
| CyGNet | 35.46 | 25.44 | 40.20 | 54.47 | 24.98 | 15.54 | 28.58 | 43.54 |
| T-GAP | 37.13 | 26.06 | 41.66 | 57.27 | 26.33 | 18.57 | 32.02 | 44.18 |
| EvoKG | 39.54 | 27.05 | 43.43 | 58.09 | 29.18 | 19.32 | <u>33.94</u> | 45.92 |
| TANGO | 21.83 | 10.32 | 26.35 | 41.62 | – | – | – | – |
| TD-RKG | 38.63 | 28.14 | 43.47 | 59.11 | – | – | – | – |

(Continued)

**Table 2 (continued)**

| Model | ICEWS05-15 | | | | ICEWS18 | | | |
|---|---|---|---|---|---|---|---|---|
| | **MRR** | **Hits@1** | **Hits@3** | **Hits@10** | **MRR** | **Hits@1** | **Hits@3** | **Hits@10** |
| DKGR-DR(-S) | 42.35 | 29.51 | 46.41 | 62.18 | 29.46 | 20.20 | 32.49 | 48.30 |
| DKGR-DR | **42.78** | **29.65** | **48.69** | **66.07** | **30.05** | **20.43** | **37.07** | **53.30** |

**Table 3:** Entity prediction results (II)

| Model | ICEWS14 | | | |
|---|---|---|---|---|
| | **MRR** | **Hits@1** | **Hits@3** | **Hits@10** |
| ComplEx | 19.06 | 10.09 | 22.00 | 36.41 |
| R-GCN | 15.03 | 9.42 | 16.12 | 31.47 |
| ConvE | 21.64 | 11.30 | 23.16 | 38.37 |
| RotatE | 9.79 | 6.41 | 9.37 | 22.24 |
| HyTE | 16.78 | 2.13 | 24.84 | 43.94 |
| TA-DistMult | 26.22 | 16.83 | 29.72 | 45.23 |
| CyGNet | 24.68 | 15.35 | 28.88 | 43.16 |
| T-GAP | 26.14 | 16.36 | 30.21 | 46.82 |
| EvoKG | 27.18 | 14.23 | 30.84 | 47.67 |
| KGTransformer | 23.98 | – | 26.89 | 41.22 |
| TANGO | 26.89 | 12.462 | 29.57 | 44.68 |
| TD-RKG | **36.70** | **26.46** | **41.20** | 54.79 |
| DKGR-DR(-S) | 29.65 | 17.51 | 34.75 | 54.20 |
| DKGR-DR | 30.07 | 17.37 | 36.31 | **57.11** |

As shown in Table 2, the proposed DKGR-DR outperforms the baseline models on all evaluation metrics in ICEWS05-15 and ICEWS18. In particular, DKGR-DR demonstrates clear superiority over the four static reasoning models. This improvement can be attributed to the use of GRU and its modified variant within DKGR-DR, which effectively capture the temporal continuity between adjacent timestamps. It consistently outperforms other dynamic baselines. The model performs better on ICEWS05-15 compared to ICEWS18. This is because ICEWS05-15 contains knowledge graphs from 4017 timestamps—approximately ten times more than the other two datasets. The richer structural dependencies and temporal continuity allow DKGR-DR, through R-GCN, to better model entity representations.

As shown in Table 3, DKGR-DR achieves excellent results on ICEWS14 across most baselines. The model also surpasses the majority of dynamic reasoning models. As noted in [7], GRU addresses the inherent issues of gradient vanishing and explosion in traditional RNNs, leading to more stable training and improved performance. It is worth noting, however, that DKGR-DR performs worse than TD-RKG on the ICEWS14 dataset. This is likely due to the fact that ICEWS14 contains only one year of data, making it relatively sparse and lacking in clear temporal evolution patterns—conditions under which DKGR-DR's dynamic modeling capabilities are less effective. In contrast, TD-RKG incorporates a dynamic global information attention layer that captures deeper semantic relationships across timestamps and entity types, allowing it to leverage more informative temporal cues in such sparse scenarios.

### *5.2.2 Relation Prediction Results*

For the relation prediction task, only a subset of models from the entity prediction experiments were selected, as not all models are designed for relation prediction. We report the results using two evaluation metrics: MRR and Hits@10. Hits@1 and Hits@3 are excluded because the differences among models on these metrics are minimal and do not clearly distinguish model performance. Specifically, we select ConvE [18] and ConvTransE [34] from the static models, and RGCRN [35] from the dynamic models, and these models are more representative of their respective categories and share similarities with our proposed model. The experimental results are shown in Table 4.

**Table 4:** Comparison of relation prediction results (All values are percentages; **bold** indicates the best result; underline indicates the second best result.)

| Model | MRR | | | Hits@10 | | |
|---|---|---|---|---|---|---|
| | ICEWS14 | ICEWS18 | ICEWS05-15 | ICEWS14 | ICEWS18 | ICEWS05-15 |
| ConvE | 38.80 | 37.73 | 37.89 | 54.32 | 57.79 | 55.98 |
| ConvTransE | 38.00 | 38.40 | 38.26 | 58.01 | 58.40 | 59.52 |
| RGCRN | 37.14 | 38.04 | 38.37 | 65.71 | 63.54 | 68.73 |
| DKGR-DR(-S) | <u>40.90</u> | <u>39.65</u> | <u>19.92</u> | <u>70.81</u> | <u>69.81</u> | <u>72.56</u> |
| DKGR-DR | **41.42** | **40.76** | **42.28** | **72.30** | **71.70** | **73.98** |

As shown in Table 4, DKGR-DR achieves the best performance in terms of both MRR and Hits@10 across all four datasets. DKGR-DR(-S) ranks second in all cases. These results indicate that structural dependencies, temporal continuity, and static entity attributes all contribute positively to dynamic relation reasoning.

## *5.3 Ablation Study*

To evaluate the effectiveness of each neural network module in the DKGR-DR model, we designed ablation experiments by replacing individual modules.

In the encoder distributed representation learning unit, to capture structural dependencies among entities, we proposed an R-GCN-based structural dependency capturing module. To examine its contribution, we replaced this module with randomly initialized embedding vectors while keeping other settings unchanged, denoted as DKGR-DR(-RGCN).

To learn relation embeddings with temporal continuity, we proposed a GRU-based temporal continuity and relation learning module within the encoder. Under the same conditions, we replaced this module with a convolutional operation, denoted as DKGR-DR(-GRU).

After obtaining distributed representations of entities and relations, we designed a ConvE-based decoder. To verify its effectiveness, we replaced ConvE with a simple fully-connected layer while keeping other components unchanged, denoted as DKGR-DR(-ConvE).

Similarly, DKGR-DR(-S) denotes the variant of the model without incorporating the static view.

Among the three dynamic knowledge graph datasets—ICEWS14, ICEWS18 and ICEWS05-15—we chose ICEWS18 for the ablation experiments. Although ICEWS14 and ICEWS18 both record events within a single year with the same temporal granularity, ICEWS18 contains nearly four times more entities than ICEWS14. While ICEWS05-15 has almost as many facts as ICEWS18, it spans nearly 10 years, resulting in relatively few facts per timestamp. Therefore, ICEWS18 was used for the ablation experiments.

Using ICEWS18 as the ablation dataset, the results for the entity prediction task are presented in Table 5.

**Table 5:** Ablation study results on the ICEWS18 Dataset (All values are percentages; **bold** indicates the best result.)

| Model | MRR | Hits@1 | Hits@3 | Hits@10 |
|---|---|---|---|---|
| DKGR-DR(-RGCN) | 18.30 | 7.68 | 20.97 | 35.42 |
| DKGR-DR(-GRU) | 25.90 | 15.55 | 32.17 | 45.65 |
| DKGR-DR(-ConvE) | 28.16 | 18.76 | 34.45 | 48.38 |
| DKGR-DR(-S) | 29.46 | 20.20 | 32.49 | 48.30 |
| DKGR-DR | **30.05** | **20.43** | **37.07** | **53.30** |

From the results in Table 5, several observations can be made. First, removing the R-GCN-based structural dependency module (DKGR-DR(-RGCN)) leads to the most significant performance drop across all metrics, highlighting the importance of capturing structural dependencies among entities. Second, omitting the GRU-based temporal continuity and relation learning module (DKGR-DR(-GRU)) also noticeably degrades performance, indicating that modeling temporal dynamics of relations is crucial for accurate prediction. Third, replacing the ConvE decoder with a simple fully-connected layer (DKGR-DR(-ConvE)) results in moderate performance decline, suggesting that the ConvE decoder effectively captures complex entity-relation interactions. Fourth, the variant without static view incorporation (DKGR-DR(-S)) exhibits lower Hits@3 and MRR compared to the full model, demonstrating that integrating static attributes helps stabilize entity representations and improve prediction accuracy.

Overall, the ablation study confirms that each module in DKGR-DR contributes positively to the model's performance. The combination of structural dependency modeling, temporal continuity learning, relation-aware decoding, and static attribute integration enables the model to achieve the best results on ICEWS18, validating the of DKGR-DR design for dynamic knowledge graph reasoning.

## 6 Conclusions

This paper proposes a model, named DKGR-DR, which learns distributed representations of entities and relations by capturing three key properties of dynamic knowledge graphs (DKGs): structural dependencies, temporal continuity, and static entity attributes. To capture temporal continuity, we design an enhanced GRU network. Additionally, we construct three static attribute datasets and introduce a Relational-GAT network to model entity static attributes. Overall, the DKGR-DR model supports both entity prediction and relation prediction. We conduct comparative experiments on three widely used dynamic reasoning datasets. The experimental results demonstrate the effectiveness of the proposed model.

**Acknowledgement:** Not applicable.

**Funding Statement:** This work was supported by National Natural Science Foundation of China (Grant Nos. 62071098, U24B20128); Sichuan Science and Technology Program (Grant No. 2022YFG0319).

**Author Contributions:** The authors confirm contribution to the paper as follows: Conceptualization, Qiuru Fu; methodology, Qiuru Fu, Shumao Zhang, Jie Xu and Du Xu; software, Qiuru Fu, Shumao Zhang and Shuang Zhou; validation, Shanchao Li and Du Xu; formal analysis, Qiuru Fu, Shanchao Li and Du Xu; investigation, Shumao Zhang,

Shanchao Li and Du Xu; resources, Qiuru Fu; data curation, Qiuru Fu, Jie Xu and Changming Zhao; writing—original draft preparation, Qiuru Fu; writing—review and editing, Qiuru Fu, Shumao Zhang, Jie Xu and Changming Zhao; supervision, Jie Xu and Changming Zhao; project administration, Jie Xu, Changming Zhao and Du Xu. All authors reviewed the results and approved the final version of the manuscript.

**Availability of Data and Materials:** The datasets generated or analyzed during the current study are available from the corresponding authors on reasonable request.

**Ethics Approval:** Not applicable.

**Conflicts of Interest:** The authors declare no conflicts of interest to report regarding the present study.

## References

1. Singhal A. Introducing the knowledge graph: things, not strings; 2012 [Internet]. [cited 2025 Sep 25]. Official Google Blog. Available from: https://www.blog.google/products/search/introducing-knowledge-graph-things-not.
2. Dong X, Gabrilovich E, Heitz G, Horn W, Lao N, Murphy K, et al. Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14. New York, NY, USA: Association for Computing Machinery; 2014. p. 601–10. doi:10.1145/2623330.2623623.
3. Chekol M, Pirrò G, Schoenfisch J, Stuckenschmidt H. Marrying uncertainty and time in knowledge graphs. Proc AAAI Conf Artif Intell. 2017;31(1):10495. doi:10.1609/aaai.v31i1.10495.
4. Jiang T, Liu T, Ge T, Sha L, Chang B, Li S, et al. Towards time-aware knowledge graph completion. In: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers. Osaka, Japan: The COLING 2016 Organizing Committee; 2016. p. 1715–24.
5. Bordes A, Usunier N, Garcia-Duran A, Weston J, Yakhnenko O. Translating embeddings for modeling multi-relational data. In: Burges CJ, Bottou L, Welling M, Ghahramani Z, Weinberger KQ, editors. Advances in neural information processing systems. San Francisco, CA, USA: Curran Associates, Inc.; 2013. Vol. 26, p. 2787–95. doi:10.5555/2999792.2999923.
6. Zhu C, Chen M, Fan C, Cheng G, Zhang Y. Learning from history: modeling temporal knowledge graphs with sequential copy-generation networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. Washington, DC, USA: AAAI Press; 2021. Vol. 35, p. 4732–40. doi:10.1609/aaai.v35i5.16604.
7. Park N, Liu F, Mehta P, Cristofor D, Faloutsos C, Dong Y. EvoKG: jointly modeling event time and network structure for reasoning over temporal knowledge graphs. In: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM '22). New York, NY, USA: Association for Computing Machinery; 2022. p. 794–803. doi:10.1145/3488560.3498451.
8. Wang Z, Zhang J, Feng J, Chen Z. Knowledge graph embedding by translating on hyperplanes. In: AAAI'14: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence; 2014 Jul 27–31; Québec City, QC, Canada. p. 1112–9. doi:10.1609/aaai.v28i1.8870.
9. Lin Y, Liu Z, Sun M, Liu Y, Zhu X. Learning entity and relation embeddings for knowledge graph completion. In: AAAI'15: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence; 2015 Jan 25–30; Austin, TX, USA. p. 2181–7. doi:10.1609/aaai.v29i1.9491.
10. Ji G, He S, Xu L, Liu K, Zhao J. Knowledge graph embedding via dynamic mapping matrix. In: Zong C, Strube M, editors. Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing. Beijing, China: Association for Computational Linguistics; 2015. p. 687–96. doi:10.3115/v1/P15-1067.
11. Xiao H, Huang M, Hao Y, Zhu X. TransA: an adaptive approach for knowledge graph embedding. arXiv:1509.05490. 2015.

12. Xiao H, Huang M, Zhu X. TransG: a generative model for knowledge graph embedding. In: Erk K, Smith NA, editors. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics. Berlin, Germany: Association for Computational Linguistics; 2016. p. 2316–25. doi:10.18653/v1/P16-1219.
13. Nickel M, Tresp V, Kriegel HP. A three-way model for collective learning on multi-relational data. In: Proceedings of the 28th International Conference on International Conference on Machine Learning, ICML'11. Madison, WI, USA: Omnipress; 2011. p. 809–16. doi:10.5555/3104482.3104584.
14. Yang B, Yih SWt, He X, Gao J, Deng L. Embedding entities and relations for learning and inference in knowledge Bases. arXiv.1412.6575. 2015.
15. Nickel M, Rosasco L, Poggio T. Holographic embeddings of knowledge graphs. In: AAAI'16: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence; 2016 Feb 12–17; Phoenix, AZ, USA. p. 1955–61. doi:10.5555/3016100.3016172.
16. Trouillon T, Welbl J, Riedel S, Gaussier E, Bouchard G. Complex embeddings for simple link prediction. In: Balcan MF, Weinberger KQ, editors. Proceedings of the 33rd International Conference on Machine Learning. New York, NY, USA: PMLR; 2016. Vol. 48, p. 2071–80. doi:10.5555/3045390.3045609.
17. Liu H, Wu Y, Yang Y. Analogical inference for multi-relational embeddings. In: International Conference on Machine Learning. New York, NY, USA: PMLR; 2017. p. 2168–78. doi:10.5555/3305890.3305905.
18. Dettmers T, Minervini P, Stenetorp P, Riedel S. Convolutional 2D knowledge graph embeddings. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence, AAAI'18/IAAI'18/EAAI'18. Washington, DC, USA: AAAI Press; 2018. p. 1811–8. doi:10.5555/3504035.3504256.
19. Kazemi SM, Poole D. SimplE embedding for link prediction in knowledge graphs. In: Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R, editors. Advances in neural information processing systems. San Francisco, CA, USA: Curran Associates, Inc.; 2018. Vol. 31, p. 4289–300. doi:10.5555/3327144.3327341.
20. Nguyen DQ, Vu T, Nguyen TD, Nguyen DQ, Phung D. A capsule network-based embedding model for knowledge graph completion and search personalization. In: Burstein J, Doran C, Solorio T, editors. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Minneapolis, MN, USA: Association for Computational Linguistics; 2019. p. 2180–9. doi:10.18653/v1/N19-1226.
21. Vashishth S, Sanyal S, Nitin V, Agrawal N, Talukdar P. InteractE: improving convolution-based knowledge graph embeddings by increasing feature interactions. Proc AAAI Conf Artif Intell. 2020;34(3):3009–16. doi:10.1609/aaai.v34i03.5694.
22. Schlichtkrull M, Kipf TN, Bloem P, van den Berg R, Titov I, Welling M. Modeling relational data with graph convolutional networks. In: Gangemi A, Navigli R, Vidal ME, Hitzler P, Troncy R, Hollink L, et al. editors. The semantic web. Cham, Switzerland: Springer International Publishing; 2018. p. 593–607. doi:10.1007/978-3-319-93417-4_38.
23. Veličković P, Casanova A, Liò P, Cucurull G, Romero A, Bengio Y. Graph attention networks. In: 6th International Conference on Learning Representations, ICLR 2018; 2018 Apr 30–May 3; Vancouver, BC, Canada. p. 1–12. doi:10.17863/CAM.48429.
24. Dasgupta SS, Ray SN, Talukdar P. HyTE: hyperplane-based temporally aware knowledge graph embedding. In: Riloff E, Chiang D, Hockenmaier J, Tsujii J, editors. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. Brussels, Belgium: Association for Computational Linguistics; 2018. p. 2001–11. doi:10.18653/v1/D18-1225.
25. Goel R, Kazemi SM, Brubaker M, Poupart P. Diachronic embedding for temporal knowledge graph completion. Proc AAAI Conf Artif Intell. 2020;34(4):3988–95. doi:10.1609/aaai.v34i04.5815.
26. García-Durán A, Dumančić S, Niepert M. Learning sequence encoders for temporal knowledge graph completion. In: Riloff E, Chiang D, Hockenmaier J, Tsujii J, editors. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. Brussels, Belgium: Association for Computational Linguistics; 2018. p. 4816–21. doi:10.18653/v1/D18-1516.

27. Jin W, Qu M, Jin X, Ren X. Recurrent event network: autoregressive structure inferenceover temporal knowledge graphs. In: Webber B, Cohn T, He Y, Liu Y, editors. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). Stroudsburg, PA, USA: Association for Computational Linguistics; 2020. p. 6669–83. doi:10.18653/v1/2020.emnlp-main.541.
28. Wang Z, Ding D, Ren M, Conti M. TANGO: a temporal spatial dynamic graph model for event prediction. Neurocomputing. 2023;542(1):126249. doi:10.1016/j.neucom.2023.126249.
29. Li XV, Sanna Passino F. FinDKG: dynamic knowledge graphs with large language models for detecting global trends in financial markets. In: Proceedings of the 5th ACM International Conference on AI in Finance, ICAIF '24. New York, NY, USA: Association for Computing Machinery; 2024. p. 573–81. doi:10.1145/3677052.3698603.
30. Chen H, Zhang M, Chen Z. Temporal knowledge graph reasoning based on dynamic fusion representation learning. Expert Syst. 2025;42(2):e13758. doi:10.1111/exsy.13758.
31. Boschee E, Lautenschlager J, O'Brien S, Shellman S, Starz J, Ward M. ICEWS coded event data. Harvard Dataverse. 2015. doi:10.7910/DVN/28075.
32. Sun Z, Deng ZH, Nie JY, Tang J. RotatE: knowledge graph embedding by relational rotation in complex space. arXiv.1902.10197. 2019.
33. Jung J, Jung J, Kang U. Learning to walk across time for interpretable temporal knowledge graph completion. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, KDD '21. New York, NY, USA: Association for Computing Machinery; 2021. p. 786–95. doi:10.1145/3447548.3467292.
34. Shang C, Tang Y, Huang J, Bi J, He X, Zhou B. End-to-end structure-aware convolutional networks for knowledge base completion. In: Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, AAAI'19/IAAI'19/EAAI'19. Washington, DC, USA: AAAI Press; 2019. p. 3060–7. doi:10.1609/aaai.v33i01.33013060.
35. Seo Y, Defferrard M, Vandergheynst P, Bresson X. Structured sequence modeling with graph convolutional recurrent networks. In: Neural Information Processing: 25th International Conference, ICONIP 2018. Berlin/Heidelberg, Germany: Springer-Verlag; 2018. p. 362–73. doi:10.1007/978-3-030-04167-0_33.