Relational Context Modeling for Improved Knowledge Graph Completion

Guoqi Lin, Qi Li*

***Abstract*—Knowledge graphs (KGs) structure knowledge, but are typically incomplete. Link prediction or knowledge graph completion (KGC) builds on KG facts to infer missing facts. Previous embedding models cannot capture the expressive aspects of deeper, multi-layered models. These systems also assign a static embedding to each object and relationship, ignoring the fact that they behave differently in different graph settings, which limits their performance. In this paper, we propose a method that merges the reception weighted key value model with the TuckER model to overcome these limitations, called RCME. RWKV models sequential information and allows dynamic embeddings, while TuckER provides robust relational decoding. Embedding provides more expressive representations. Our strategy outperforms several state-of-the-art models on link prediction and triple classification on benchmark datasets.**

***Index Terms*—knowledge graphs, knowledge completion, triple classification, link prediction**

I. Introduction

KNOWLEDGE graphs (KGs) have emerged as a substantial knowledge resource used in various applications, including recommendation systems [1] and question-answering systems [2]. A knowledge graph can be conceptualized as a directed graph structured in the form of triples, each consisting of a head entity, a relation and a tail entity [3]. Despite their usefulness, knowledge graphs are often characterized by incompleteness, with many missing links. Link prediction, also known as knowledge graph completion (KGC), is an automated reasoning technique aimed at inferring missing components within knowledge graphs [4], [5]. One prominent method for KGC is knowledge graph embedding (KGE), which attempts to map entities and relations into a low-dimensional continuous vector space [6]. Fig. 1 provides a visual representation of a knowledge graph and demonstrates the application of knowledge graph completion.

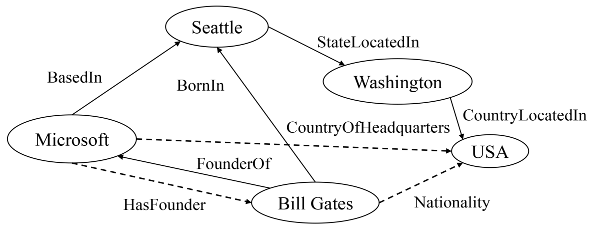

Fig. 1: An example of a knowledge graph with a task for predicting possible links (dotted lines represent potential links)

Manuscript received January 5, 2025; revised March 31, 2025.

Guoqi Lin is an Associate Researcher of Shaoxing University, Shaoxing, Zhejiang 312000, China (e-mail: 2010001537@usx.edu.cn).

Qi Li is a Lecturer of Shaoxing University, Shaoxing, Zhejiang 312000, China (e-mail: lsongru1106@163.com).

Current approaches to embedding knowledge graphs include translation-based models [7], semantic matching models [8], and neural network-based models [9]. Translation-based models establish linear translation rules that relate the head entity to the tail entity. Semantic matching models use various scoring algorithms to evaluate the similarity of embeddings between entities and relationships. However, these methods face challenges in handling the high-dimensional embeddings required to encapsulate rich information while managing large knowledge graphs, potentially leading to overfitting and increased computational complexity [10], [11]. They typically generate a single static representation that fails to capture the nuanced meanings of entities and links in different contexts. In addition, their reliance on predominantly additive or multiplicative operations limits their expressiveness [12]. In contrast, neural network-based models exploit different neural architectures to generate expressive representations from raw embeddings, achieving commendable results in KGC tasks [13]–[15].

Receptance weighted key value (RWKV) [16], a sequence model that combined the advantages of RNNs and transformers, has seen widespread application. Designed for computational efficiency and expressiveness, RWKV enhances attention mechanisms via multiple linear methods, replacing point product interactions with channel-directed attention to increase efficiency. TuckER [17], which uses Tucker decomposition on binary tensors, excels in predicting links in knowledge graphs. In this paper, we propose to integrate RWKV as an encoder and TuckER as a decoder. This framework exploits RWKV's sequential information modelling and dynamic embeddings, together with TuckER's relational decoding robustness, to achieve more expressive representations of complex data structures. Our work contributes to the field through the following three key innovations.

- We propose a novel hybrid architecture that uses the RWKV model to encode sequential data into dynamic embeddings and the TuckER model to decode relations. This design allows us to exploit the temporal dynamics captured by RWKV and the relational richness modelled by TuckER.
- Our method introduces advanced mechanisms for learning from both sequential patterns and relational structures within the data. By integrating RWKV's ability to capture time-evolving features with TuckER's strength in handling multi-relational data, it significantly improves the system's ability to understand and represent complex relationships, which is critical for tasks such as link prediction and triple classification.
- To validate the effectiveness of our proposed method, we conduct experiments on several benchmark datasets. The empirical results show that our approach consistent-

ly outperforms several state-of-the-art models in both link prediction and triple classification tasks.

II. Related Work

A common method for knowledge graph completion is based on knowledge graph embedding (KGE) techniques. The goal of KGE is to embed the representations of entities and relations in a low-dimensional, continuous vector space. KGE-based approaches can be divided into three main types: translation-based models [7], semantic matching models [8], and neural network-based models [9].

Translation-based models use relations to formulate linear translation rules between head and tail entities. The paradigmatic translation-based model, TransE [11], represents entities and relations as vectors. The core concept of TransE is that the sum of the head entity vector and the relation vector approximates the tail entity vector, thereby representing all entities and relations. Although TransE is simple and efficient, it faces challenges in modelling complex relationships. TransH [18], TransR [19], TransD [20] and TranSparse [21] transform entities and relations into subspaces to deal with 1-*N*, *N*-1 and *N*-*N* relations. RotateE [10] defines each relation as a complex vector space rotation from head to tail. HAKE [5], which embeds entities in polar coordinates, captures the semantic hierarchy in knowledge graphs and differentiates entities at different levels. HousE [4] identifies important relational patterns through a householder parameterisation.

Semantic matching models [22] evaluate the similarity of entity relation embeddings via scoring functions. A typical tensor decomposition model is RESCAL [2]. It represents pairwise entity relation interactions by a three-way tensor factorisation. DistMult [1] streamlines RESCAL by transforming the relation matrix into a diagonal matrix, thus reducing the number of training parameters, but it only handles symmetric relations. ComplEx [10] handles asymmetric relations better by extending matrix decomposition to the complex domain. ANALOGY [23] captures the analogous properties of entities and relations through linear mapping. TuckER [17] calculates a validation score for each triple using the Tucker decomposition.

Neural network approaches [23] use different models to derive comprehensive representations from embeddings for tasks such as connection prediction. An early method was the feedforward neural tensor network (NTN) [15]. Convolutional neural network (CNN)-based models [14], including ConvE [12], ConvKB [9] and InteractE [15], detect complex feature interactions. Graph-based models [13], such as R-GCN [24] and CompGCN [25], use structural and neighbourhood data to improve the quality of the representation. Transformer-based models [26] have improved knowledge graph completion, with KG-BERT and CoKE using BERT and Transformer encoders to process triples. HittER [24] uses transformer blocks to generate relational embeddings for entity neighbours.

Current knowledge graph embedding models face limitations in contextual adaptability and computational efficiency, as translation-based and semantic matching approaches rely on static embeddings that fail to capture dynamic contextual nuances [24]–[26], while neural network-based methods struggle with temporal dynamics despite their expressiveness [9], [12], [17]. To address these challenges, we propose a hybrid approach combining RWKV's sequential processing capability for generating dynamic embeddings with TUCKER's robust multi-relational decoding, aiming to enhance context-aware representations while maintaining computational efficiency.

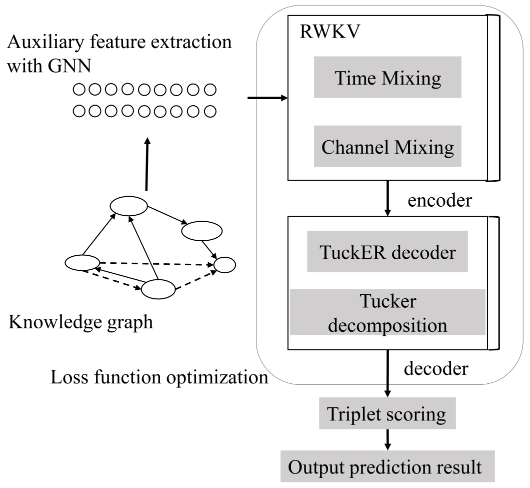

Fig. 2: Architecture of our method

III. Proposed Model

This section presents a strategy for predicting connections in knowledge graph. It combines two different but complementary architectures: the RWKV model for input encoding and the TuckER model for decoding and classification. The hybrid method exploits RWKV's unique way of handling sequential data during encoding and TuckER's advanced tensor decomposition for efficient decoding. The model is trained to predict the missing entity in a given input context. The overall framework of the approach is shown in Fig. 2.

A. Encoder

1) Auxiliary Feature Extraction with Graph Neural Networks: The GNN framework consists of several layers where each layer aggregates information from neighbouring nodes to update node representations. In particular, we use a graph attention network (GAT) to highlight important connections between entities. The attention mechanism allows the model to weight different neighbours differently, thereby capturing the most relevant information for each entity.

Let $\mathbf{h}_i^{(l)}$ denote the embedding of node i in layer l. The updated embedding $\mathbf{h}_i^{(l+1)}$ is computed as follows:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{j\in\mathcal{N}(i)} \alpha_{ij}\mathbf{W}\mathbf{h}_j^{(l)}\right) \tag{1}$$

where $\mathcal{N}(i)$ represents is the set of neighbours of node i, α_{ij} is the attention coefficient between nodes i and j, $\mathbf{W}$ is a learnable weight matrix, and σ is an activation function such as ReLU.

Once the GNN phase has produced improved embeddings, these are fed into the RWKV encoder. This pre-processing step ensures that the dynamic embeddings generated by RWKV benefit from a richer context derived from the graph structure. Thus, while the core functionality of the

RWKV encoder and TuckER decoder remains unchanged, their performance is potentially improved due to the richer input features.

2) RWKV Framework: The RWKV framework consists of a series of stacked residual blocks. Each block consist of a time-mixing sub-block and a channel-mixing sub-block. This design effectively combines the advantages of recurrent neural networks (RNNs) and attention mechanisms.

The receptor vector, denoted *R*, captures information from previous states. The weight vector *W* is a model parameter that accounts for positional weight decay. The key vector *K* plays a similar role to the key in traditional attention models, and the value vector *V* is analogous to the value in standard attention mechanisms.

3) Time Mixing Mechanism: The time mixing mechanism processes an input sequence $x = (e_h, e_r)$, where e_h is the head entity embedding, and e_r is the relation embedding. The resulting output embedding $o = (o_h, o_r)$ encapsulates the contextual information and dependencies within the input sequence. The output is determined by the following equation.

$$o_t = W_o \cdot \sigma\left(R_t\right) \cdot \left(\sum_{i=1}^{t} w_{k_i v_i t}\right) \tag{2}$$

where $t \in \{h, r\}$ and W_o is the weighting matrix of the output vector. The term $\sigma\left(R_t\right)$ is the sigmoid of the receptance vector, calculated as

$$R_t = \mu_r \cdot x_t + (1 - \mu_r) \cdot R_{t-1} \tag{3}$$

where x_t is the input embedding at step *t* and μ_r is an interpolation coefficient. The term $w_{k_i v_i t}$ is consistent with the techniques used in the Attention-Free Transformer, expressed as

$$w_{k_i v_i t} = u_t \cdot W_k \cdot k_i + (1 - u_t) \cdot W_v \cdot v_i \tag{4}$$

where u_t is a specialized weighting factor for the current input, and *w*, *u*, *k*, *v* are derived from the *K*, *V* vectors respectively. The role of u_t is to provide a distinct attention channel for the current step, thus addressing potential information degradation associated with *w*. The key and value vectors are calculated similarly, with μ_k and μ_v as the interpolation coefficients, and W_k and W_v as the weighting matrices for the key and value vectors respectively.

4) Channel Mixing Operation: In the channel mixing operation, the modified input sequence is transformed.

$$\begin{aligned} &Modified\ sequence = \\ &W_c \cdot ReLU\left(W_c \cdot Modified\ sequence\right) \end{aligned} \tag{5}$$

where W_c is the separate weighting matrix for the transformed vectors, and ReLU is the activation function used for the output.

5) Integration of Modules in RWKV Blocks: Each RWKV block processes the input by sequentially applying a dropout-enhanced time mix followed by a channel mix.

$$Output = Dropout(TimeMixing) + Input \tag{6}$$

This design introduces dropout layers in front of the remaining links to prevent overfitting. These components enable the RWKV architecture to effectively handle sequences, preserve causality, and improve model robustness through its RNN-inspired mechanism.

B. Decoder

1) Tucker Decomposition Application: The TuckER model, based on Tucker decomposition, is used for decoding and classification. Tucker decomposition is a technique that decomposes a high-dimensional tensor into a product of a low-dimensional core tensor and several matrix factors. In the context of knowledge graph completion, the graph is represented as a set of triplets (e_h, *r*, e_t), where e_h is the head entity, *r* is the relation, and e_t is the tail entity. After obtaining the output representations of the head entity and the relation, a third-order tensor is constructed, where each element is a triplet tuple valued 1 if the triplet is valid, and 0 otherwise. Tucker decomposition is then used to decode the embedded information. Let $\mathcal{C}$ denote the output representation of the encoder, where $\mathcal{C}_h$ and $\mathcal{C}_r$ are the output representations of the head entity and relation, respectively. The evaluation function in the decoder is defined as follows.

$$Score = \mathcal{G}\left(\times n\right) \mathcal{C}_h \left(\times n\right) \mathcal{C}_r \tag{7}$$

where $\mathcal{G}$ is the learnable core tensor of the Tucker decomposition, and $(\times n)$ denotes the *n*-mode tensor product.

C. Training Enhancements

To better capture the bidirectional nature of relations within a knowledge graph, we augment our training dataset with inverse triples (*t*, *r*-1, *h*). This reciprocal learning approach ensures that the model learns both the direct relationship between entities and their inverse counterparts. By including inverse triples, we mitigate potential biases towards specific relation orientations and enrich the representation of relational patterns, thereby enhancing the model's understanding of bidirectional relationships.

We use an advanced frequency-based weighted sampling technique. The probability *P*(*e*) of a negative sample is proportional to its frequency of occurrence in the knowledge graph.

$$P\left(e\right) = \frac{f(e)^{\alpha}}{\sum_{e' \in E} f(e')^{\alpha}} \tag{8}$$

where *f*(*e*) represents the frequency of the entity *e* in the dataset, and α is a tunable parameter that controls the emphasis on frequent entities. This method ensures that negative samples are more representative of the true distribution, leading to improved generalization and robustness to unseen data.

We implement an adaptive 1-*N* scoring mechanism, where each pair (*h*, *r*) is scored against all possible entities as the target *t*. Unlike static methods, this approach dynamically adjusts the scoring threshold based on the difficulty of distinguishing correct from incorrect triples. For a given triple (*h*, *r*, *t*), the score *s*(*h*, *r*, *t*) is computed as follows.

$$s(h, r, t) = g(f(h, r, t)) + \beta \bullet d(h, r) \tag{9}$$

where $g(\bullet)$ is a non-linear activation function applied to the feature vector, $f(h, r, t)$ represents the head entity, relation and tail entity, β is a learnable parameter, and $d(h, r)$ is a measure of difficulty derived from the context of the head entity and relation. This adaptive scoring improves accuracy by accounting for different levels of ambiguity within different parts of the knowledge graph.

Moreover, we adopt a curriculum learning strategy during training. Starting with simpler tasks or easier to predict triples, the model gradually progresses to more complex tasks. The complexity of the training instances is determined by a predefined metric, such as the average distance of the embeddings or the historical prediction accuracy. The transition between levels of difficulty is controlled by a schedule that can be linear, exponential, or adaptive based on the model's performance.

$$\lambda(t) = min(1, \lambda, t) \tag{10}$$

where $\lambda(t)$ is the proportion of complex triples included at time step *t*, and λ is a rate parameter that controls the speed of the transition. This method speeds up convergence and ensures a stable learning process. To optimise the training process, we use a hybrid loss function that combines the cross-entropy loss with a margin-based ranking loss. The total loss L for a given batch is defined as

$$L = L_{ce} + \lambda L_{rank} \tag{11}$$

where L_{ce} is the cross-entropy loss that ensures accurate classification of positive and negative triples, and L_{rank} is the margin-based ranking loss that promotes a certain margin Δ between the scores of positive and negative triples.

$$L_{rank} = \sum_i max(0, \Delta - s(h_i, r_i, t_i) + s(h_i, r_i, t_i')) \tag{12}$$

where $s(h_i, r_i, t_i)$ and $s(h_i, r_i, t_i')$ are the scores for positive and negative triples, respectively, and λ is a balancing factor that controls the contribution of the ranking loss relative to the cross-entropy loss. Algorithm 1 describes the RCME process in detail.

Algorithm 1 RCME

```
Input: Head entity $e_h$, Relation r, Adjacency info N, Training data $\mathcal{D}$
Output: Predicted tail entity $e_t$ or triple validity
1: $h_i^{(l+1)} \leftarrow \text{GAT}(h_j^{(l)}, \mathcal{N}(i))$ (GNN Feature Extraction)
2: $o_t \leftarrow \text{RWKV}(e_h, e_r)$ (RWKV Time/Channel Mixing)
3: Score $\leftarrow \mathcal{G} \times_n c_h \times_n c_r$ (Tucker Decomposition)
4: $\mathcal{D}^- \leftarrow \text{InverseTriples}(\mathcal{D})$ (Add inverse triples)
5: $P(e) \leftarrow \frac{f(e)^\alpha}{\sum f(e')^\alpha}$ (Weighted negative sampling)
6: $s(h, r, t) \leftarrow g(f(h, r, t)) + \beta \cdot d(h, r)$ (Adaptive scoring)
7: $L \leftarrow L_{ce} + \lambda L_{rank}$ (Hybrid loss)
8: Update model parameters via Adam optimizer
```

IV. Experiments

The evaluation framework employs four benchmark datasets spanning distinct knowledge domains to ensure comprehensive validation. The biomedical domain is represented by UMLS [25], containing clinical entities and therapeutic relationships. Cross-domain analysis utilizes FB15k [24] with its 1,345 heterogeneous relations and its refined subset FB13 [13] focusing on core semantic patterns. Demographic attribute modeling is enabled through YAGO3-10 [26], which captures interpersonal characteristics across 46 social relations.

Our computational architecture was implemented in PyTorch with NVIDIA RTX 4090 acceleration. Hyperparameters were empirically optimized through systematic grid searches: embedding dimensions $k \in \{64, 96, 128, 192, 256\}$, RWNV block depth $\ell \in \{2, 4, 6, 8\}$, and adaptive dropout rates $\eta \in [0.2, 0.5]$ across network layers. The Adam optimizer was configured with learning rates $\alpha \in \{5 \times 10^{-4}, 10^{-2}\}$ alongside standard momentum parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$). Training procedures employed fixed batch sizes ($B = 512$) across maximum 500 epochs, with early termination triggered by development set MRR plateaus. Final model selections were determined through five-fold cross-validation on each dataset's validation subset to prevent overfitting.

A. Baseline

To ensure fairness, we exclude models that use auxiliary data such as text and focus on two categories: triple-based methods that rely solely on KG structural information (e.g. TransE [23], DistMult [1], ComplEx [10], RotatE [10], TuckER [17], ConvE [20], CoKE [22], HAKE [5], and HousE [4]) and contextual methods that incorporate graph structure or logic rules (e.g. Neural-LP [21], R-GCN [20], Rlogic [19] and ChatRule [18]).

B. Evaluation Metrics

For link prediction assessment, we implement entity ranking through inverse relation scoring:

1. Head Entity Ranking: For each test triplet $(h, r, t) \in \mathcal{S}$, compute $\phi(t, r^{-1}, h')$ for all $h' \in \mathcal{E}$ to determine h's rank ρ_h.

2. Tail Entity Ranking: Calculate $\phi(h, r, t')$ for all $t' \in \mathcal{E}$ to derive t's rank ρ_t. Relations remain fixed during scoring to avoid semantic ambiguity.

The evaluation employs two rigorously defined metrics:

Mean Reciprocal Rank (MRR):

$$\text{MRR} = \frac{1}{|\mathcal{S}|} \sum_{i=1}^{|\mathcal{S}|} \frac{1}{\rho_i} \tag{13}$$

where ρ_i denotes the rank of the i-th true triplet.

Hits@k:

$$\text{Hits@k} = \frac{1}{|\mathcal{S}|} \sum_{i=1}^{|\mathcal{S}|} \mathbb{I}(\rho_i \leq k) \tag{14}$$

with $\mathbb{I}(\cdot)$ as the binary indicator function.

Consistent with [12], we apply filtered ranking by excluding training-set triplets during candidate generation to prevent data leakage.

TABLE II: Triple classification accuracy.

Method	FB13	FB15K
NTN	86.9	84.2
TransE	81.2	78.6
TransH	82.4	79.8
TransR	81.5	82.4
DistMult	85.7	84.3
ComplEx	83.1	88.5
CoKE	87.4	88.9
RCME	88.9	89.7

C. Link Prediction

Link prediction involves predicting the missing head or tail entity in a triple, which as serves as a common evaluation task for knowledge graph completion models. TABLE I shows the performance of our proposed method compared to existing representative models on the FB15k, YAGO3-10, and UMLS datasets. Our method excels in performance, consistently outperforming both triple-based and context-based methods.

On FB15k, while our method slightly lags behind the transformer-based CoKE in Hit@1 and Hit@10, the difference is minimal, and our model outperforms other competitors across the board. On the large scale YAGO3-10 dataset, our approach achieves the best results, demonstrating its effectiveness on datasets with many entities. This success highlights the robustness of our method in dealing with complexity and scale of large knowledge graphs.

On the smaller UMLS dataset, our approach also delivers excellent performance, achieving the highest scores across on all metrics. This shows that our method is not only effective on large datasets, but also performs well on smaller ones, demonstrating its ability to handle different data sizes efficiently.

D. Triple Classification

Triple classification focuses on determining the validity of a given triple (h, r, t). It is a binary classification task explored for evaluation in [16]. For this task, the FB13 and FB15k datasets are used. For triple classification a relation specific threshold δ_r is set. For a triple (h, r, t), if the dissimilarity score of f_r is below δ_r, it's a false fact, otherwise it's a true fact. Different relations have different δ_r values. Settings from the link prediction task are used. All parameters are optimized on validation datasets for maximum accuracy. Finally, our method is compared with TransE, TransH, TransR, Distmult, ComplEx, and CoKE.

As shown in TABLE II, the results indicate that our proposed model, RCME, achieves the highest accuracy of 88.9% on the FB13 dataset, outperforming all other baseline methods, including the transformer-based method CoKE. On the FB15K dataset, RCME also shows strong performance with an accuracy of 89.7%, slightly lower than CoKE but better than other models. These results highlight the effectiveness of RCME in capturing the underlying relationships between entities and relations in knowledge graphs. Therefore, in the triple classification task, the proposed method can accurately determine whether the given triple is correct or not.

TABLE IV: Comparison of Decoders with Score Functions on FB15k

Decoders	FB15k		
	MRR	H@1	H@10
MLP	0.81	80.2	88.7
TransE	0.85	72.4	87.6
DistMult	0.78	73.5	88.3
ComplEx	0.81	77.9	90.3
TuckER	0.86	82.1	91.5

E. Ablation Study

A comprehensive ablation study is performed on the F-B15k and YAGO3-10 datasets to evaluate the significance of the Tucker decomposition decoder and RWKV components. In the experiments, the RWKV encoder and the Tucker decomposition decoder are removed one at a time. When the Tucker decomposition decoder is removed, the embedding generated by RWKV is used directly as the final predicted entity embedding. When RWKV is removed, the model essentially reverts to the original TuckER model. The results of our ablation study on FB15k and YAGO3-10 are showed in TABLE III. The integration of both modules gives the best results, confirming that both components are essential to achieve optimal performance.

In addition, experiments are carried out on FB15k using the following decoders: MLP, TransE, DistMult and ComplEx. To ensure a fair comparison, all other experimental settings are maintained as kept as consistent as possible. TABLE IV shows that the results indicate that our model equipped with the TuckER decoder achieves the best performance. This shows the effectiveness of the TuckER model when acting as a decoder, as it is able to capture more complex interactions between entities and relations.

F. Parameter sensitivity analysis

In this section, we systematically investigate the impact of key hyperparameters on model performance, with a particular focus on the number of RWKV layer and the embedding dimensionality. Our experimental framework first evaluates the effects of layer configuration by testing architectures containing 2 to 10 RWKV layers in 2-layer increments, as shown in the left panel of Fig. 3. Each configuration underwent independent training with consistent evaluation metrics (MRR and Hits@10). In particular, the performance metrics demonstrated remarkable stability across different layer depths-MRR values fluctuated within a narrow range of 0.02, while Hits@10 maintained over 98% consistency across layer variations.

These empirical results suggest that the number of RWKV layers has a negligible impact on model effectiveness within the current experimental paradigm. Interestingly, even the minimal 2-layer configuration achieved comparable performance to deeper architectures. This stability implies that extending the layer depth beyond 4 layers provides diminishing returns while significantly increasing the computational overhead. Therefore, an optimal implementation should prioritise shallower architectures (2-4 layers) to maintain model efficiency without compromising performance metrics.

We investigate the influence of the size of the embedding dimensions size on the performance of the model. Embedding dimensions, which transform the features of entities

TABLE I: Link Prediction of various methods on FB15k, YAGO3-10, and UMLS datasets

model	FB15k			YAGO3-10			UMLS		
	mrr	h@1	h@10	mrr	h@1	h@10	mrr	h@1	h@10
TransE	0.37	22.9	46.5	0.32	23.6	47.1	0.65	51.7	88.6
DistMult	0.62	52.2	81.4	0.35	26.1	55.4	0.89	90.4	97.3
ComplEx	0.63	58.4	84.3	0.39	32.4	59.7	0.88	85.4	96.7
RotatE	0.75	72.7	87.6	0.52	41.2	67.3	0.87	81.2	96.3
TuckER	0.78	73.8	89.1	0.45	38.3	64.5	0.88	82.4	97.1
ConvE	0.73	67.2	86.9	0.49	45.1	59.8	0.87	89.3	98.2
CoKE	0.81	82.5	90.1	0.53	47.2	66.7	0.92	90.1	97.7
HAKE	0.83	81.5	90.5	0.53	45.7	69.1	0.93	90.5	98.2
House	0.79	75.8	89.4	0.55	48.4	70.3	0.92	91.2	93.5
Neural-LP	0.75	75.2	83.1	0.46	42.8	69.4	0.71	57.6	93.2
R-GCN	0.68	59.3	83.2	0.22	20.8	41.6	0.69	52.3	90.8
Rlogic	0.34	19.4	48.5	0.32	25.6	50.7	0.65	54.8	92.1
ChatRule	0.29	20.6	56.7	0.46	35.6	32.8	0.76	68.4	93.7
RCME	0.86	82.6	91.4	0.68	51.3	71.4	0.96	92.7	99.3

TABLE III: Ablation Study Results on FB15k and YAGO3-10

Ablation	FB15k			YAGO3-10		
	MRR	H@1	H@10	MRR	H@1	H@10
RCME	0.87	82.6	91.4	0.62	52.3	71.8
w/o Tucker Decomposition Decoder	0.84	76.4	90.1	0.57	49.5	69.7
w/o RWKV (Viewed as TuckER)	0.82	75.3	88.7	0.48	42.7	65.9

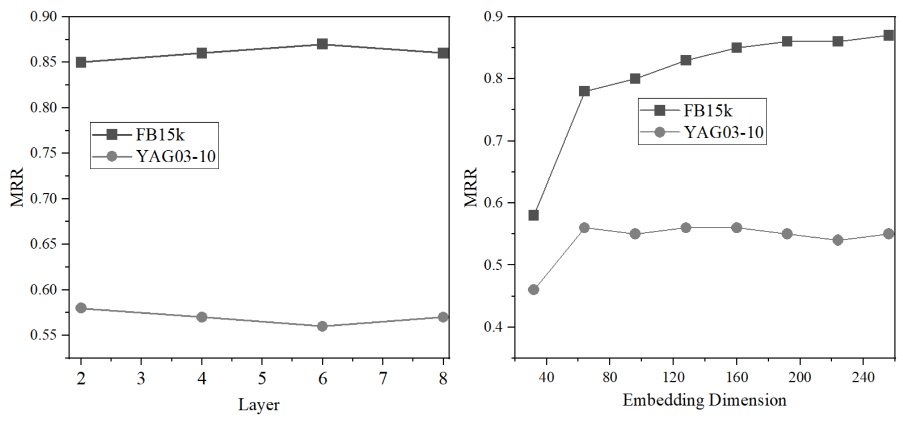

Fig. 3: Experimental results of parameter sensitivity analysis.

and relations into a lower dimensional space, are essential hyperparameters. Changes in the dimension size can have a significant impact on the performance of the model. As shown in the right-hand panel of Fig. 3, we conducted experiments with a variety of embedding dimensions. On the FB15k dataset, the performance of the model approached the optimal level when the dimension was set to 128. In contrast, on the YAGO3-10 dataset, the model reached its peak performance starting from 64 dimensions. This indicates that the model can work effectively with relatively small embedding sizes.

G. RCME in recommendation systems

RCME's ability to model complex relationships and contextual dependencies makes it highly effective for recommendation systems. By integrating knowledge graphs with collaborative filtering data, RCME captures intricate user-item interactions and diverse preferences that traditional methods often miss. This dual coding mechanism ensures that recommendations are not only personalised but also contextually relevant, overcoming cold-start issues and improving accuracy in sparse data scenarios. For example, in an e-commerce platform, the knowledge graph can represent users, products and interactions (such as ”User A-purchase-item X”). The RCME model can improve recommendation accuracy by capturing complex user-product interaction patterns. For example, if the model predicts that a user is likely to purchase a ”wireless headset”, it takes into account not only the user's purchase history, but also the purchase behaviour of similar users and the attributes of the headset (such as brand, price, noise reduction) to generate more accurate recommendations.

V. Conclusion

In this study, we introduce a novel knowledge graph completion approach, called RCME, which integrates integrates the RWKV encoder (for sequential modelling and dynamic embeddings) and the TuckER decoder (for relational reasoning). Experiments on link prediction and triple classification show superior performance over state-of-the-art models, demonstrating RCME's ability to generate expressive representations using this architecture. Future directions include extending RCME to complex tasks such as multi-relational reasoning and temporal knowledge graphs, incorporating multimodal data (text/image), and optimising computational efficiency for scalability.

References

[1] D. Zhang, W. Feng, Z. Wu, G. Li, and B. Ning, "Cdrgn-sde: Cross-dimensional recurrent graph network with neural stochastic differential equation for temporal knowledge graph embedding," *Expert Systems with Applications*, vol. 247, p. 123295, 2024.

[2] P. Rosso, D. Yang, N. Ostapuk, and P. Cudré-Mauroux, "Reta: A schema-aware, end-to-end solution for instance completion in knowledge graphs," in *Proceedings of the Web Conference 2021*, 2021, pp. 845–856.

[3] X. Ge, Y. C. Wang, B. Wang, C.-C. J. Kuo *et al.*, "Knowledge graph embedding: An overview," *APSIPA Transactions on Signal and Information Processing*, vol. 13, no. 1, 2024.

[4] C. Liu, Z. Wei, and L. Zhou, "Contrastive predictive embedding for learning and inference in knowledge graph," *Knowledge-Based Systems*, vol. 307, p. 112730, 2025.

[5] Y.-L. Li, X. Liu, X. Wu, Y. Li, Z. Qiu, L. Xu, Y. Xu, H.-S. Fang, and C. Lu, "Hake: a knowledge engine foundation for human activity understanding," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 7, pp. 8494–8506, 2022.

[6] Y. Han and H. Dai, "Research on network traffic classification based on graph neural network." *IAENG International Journal of Computer Science*, vol. 51, no. 12, pp. 2043–2050, 2024.

[7] T. Ebisu and R. Ichise, "Generalized translation-based embedding of knowledge graph," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 5, pp. 941–951, 2019.

[8] P. Luo, X. Zhu, T. Xu, Y. Zheng, and E. Chen, "Semantic interaction matching network for few-shot knowledge graph completion," *ACM Transactions on the Web*, vol. 18, no. 2, pp. 1–19, 2024.

[9] K. Wang, Y. Xu, and S. Luo, "Tiger: Training inductive graph neural network for large-scale knowledge graph reasoning," *Proceedings of the VLDB Endowment*, vol. 17, no. 10, pp. 2459–2472, 2024.

[10] Y. Wang, Y. Peng, and J. Guo, "Enhancing knowledge graph embedding with structure and semantic features," *Applied Intelligence*, vol. 54, no. 3, pp. 2900–2914, 2024.

[11] S. M. Asmara, N. A. Sahabudin, N. S. N. Ismail, and I. A. A. Sabri, "A review of knowledge graph embedding methods of transe, transh and transr for missing links," in *2023 IEEE 8th International Conference On Software Engineering and Computer Systems (ICSECS)*, 2023, pp. 470–475.

[12] S. Schramm, C. Wehner, and U. Schmid, "Comprehensible artificial intelligence on knowledge graphs: A survey," *Journal of Web Semantics*, vol. 79, p. 100806, 2023.

[13] L. Li, X. Zhang, Y. Ma, C. Gao, J. Wang, Y. Yu, Z. Yuan, and Q. Ma, "A knowledge graph completion model based on contrastive learning and relation enhancement method," *Knowledge-Based Systems*, vol. 256, p. 109889, 2022.

[14] H. Liu, Y. Li, M. Tsang, and Y. Liu, "Costco: A neural tensor completion model for sparse tensors," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 324–334.

[15] Q. Li, J. Yao, X. Tang, H. Yu, S. Jiang, H. Yang, and H. Song, "Capsule neural tensor networks with multi-aspect information for few-shot knowledge graph completion," *Neural Networks*, vol. 164, pp. 323–334, 2023.

[16] Z. Zhu, W. Shao, and D. Jiao, "Tls-rwkv: Real-time online action detection with temporal label smoothing," *Neural Processing Letters*, vol. 56, no. 2, p. 57, 2024.

[17] C. Peng, F. Xia, M. Naseriparsa, and F. Osborne, "Knowledge graphs: Opportunities and challenges," *Artificial Intelligence Review*, vol. 56, no. 11, pp. 13 071–13 102, 2023.

[18] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, "Unifying large language models and knowledge graphs: A roadmap," *IEEE Transactions on Knowledge & Data Engineering*, vol. 36, no. 07, pp. 3580–3599, 2024.

[19] K. Cheng, J. Liu, W. Wang, and Y. Sun, "Rlogic: Recursive logical rule learning from knowledge graphs," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 179–189.

[20] S. Xiong, Y. Yang, A. Payani, J. C. Kerce, and F. Fekri, "Teilp: Time prediction over knowledge graphs via logical reasoning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 14, 2024, pp. 16 112–16 119.

[21] B. Shang, Y. Zhao, Y. Liu, and C. Wang, "Attention-based exploitation and exploration strategy for multi-hop knowledge graph reasoning," *Information Sciences*, vol. 653, p. 119787, 2024.

[22] Z. Li, H. Liu, Z. Zhang, T. Liu, and N. N. Xiong, "Learning knowledge graph embedding with heterogeneous relation attention networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 8, pp. 3961–3973, 2021.

[23] R. Pan, Y. Wang, and Z. Wang, "A dga domain name detection model based on a hybrid deep neural network with multi-dimensional features," *IAENG International Journal of Computer Science*, vol. 52, no. 1, pp. 11–22, 2025.

[24] Z. Hu, V. Gutiérrez-Basulto, Z. Xiang, R. Li, and J. Z. Pan, "Hyper-former: Enhancing entity and relation interaction for hyper-relational knowledge graph completion," in *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2023, pp. 803–812.

[25] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE transactions on neural networks and learning systems*, vol. 33, no. 2, pp. 494–514, 2021.

[26] Z. Bi, S. Cheng, J. Chen, X. Liang, F. Xiong, and N. Zhang, "Relphormer: Relational graph transformer for knowledge graph representations," *Neurocomputing*, vol. 566, p. 127044, 2024.

Guoqi Lin was born in Zhejiang, China, in 1983. He is currently affiliated with Shaoxing University. His research interests mainly focus on computer-related fields, especially in the areas of artificial intelligence algorithms and their applications. He has made remarkable contributions to academic research. He has published several high-quality papers in prestigious international computer science journals, exploring innovative ways to improve the efficiency and accuracy of machine learning algorithms. For example, his research on optimising deep neural network architectures has attracted considerable attention from the academic community. He has also actively participated in various international academic conferences, sharing his latest research findings and exchanging ideas with scholars from around the world.

Qi Li was born in Jiangsu, China, in 1987. He began his academic journey at the prestigious Northwestern Polytechnical University, where he pursued his passion for engineering and technology. In 2015, he completed his Master of Science degree in Communication Engineering, demonstrating his expertise in the field. He continued his commitment to higher education and research. He remained at Northwestern Polytechnical University to continue his studies. In 2019, he received his Doctor of Philosophy degree in Computer Science, marking a significant milestone in his academic career. He is currently employed at Shaoxing University, where he contributes his knowledge and skills to the academic community. His research interests include graph neural networks and network security.