applied sciences

MDPI

Article

Memory-Enhanced Knowledge Reasoning with Reinforcement Learning

Jinhui Guo, Xiaoli Zhang *, Kun Liang and Guoqiang Zhang

The College of Artificial Intelligence, Tianjin University of Science and Technology, Tianjin 300457, China; guowudi@mail.tust.edu.cn (J.G.); liangkun@tust.edu.cn (K.L.); zhangguoqiang@mail.tust.edu.cn (G.Z.)
* Correspondence: zhangxiaoli@tust.edu.cn

Abstract: In recent years, the emergence of large-scale language models, such as ChatGPT, has presented significant challenges to research on knowledge graphs and knowledge-based reasoning. As a result, the direction of research on knowledge reasoning has shifted. Two critical issues in knowledge reasoning research are the algorithm of the model itself and the selection of paths. Most studies utilize LSTM as the path encoder and memory module. However, when processing long sequence data, LSTM models may encounter the problem of long-term dependencies, where memory units of the model may decay gradually with an increase in time steps, leading to forgetting earlier input information. This can result in a decline in the performance of the LSTM model in long sequence data. Additionally, as the data volume and network depth increase, there is a risk of gradient disappearance. This study improved and optimized the LSTM model to effectively address the problems of gradient explosion and gradient disappearance. An attention layer was employed to alleviate the issue of long-term dependencies, and ConvR embedding was used to guide path selection and action pruning in the reinforcement learning inference model. The overall model achieved excellent reasoning results.

Keywords: knowledge reasoning; knowledge graph completion; reinforcement learning; LSTM

Citation: Guo, J.; Zhang, X.; Liang, K.; Zhang, G. Memory-Enhanced Knowledge Reasoning with Reinforcement Learning. *Appl. Sci.* **2024**, *14*, 3133. https://doi.org/10.3390/app14073133

Academic Editor: Tobias Meisen

Received: 11 March 2024
Revised: 2 April 2024
Accepted: 5 April 2024
Published: 8 April 2024

1. Introduction

A knowledge graph (KG) [1] is a network of concepts where the fundamental element is a triple in the form of (entity, relationship, entity). Many knowledge graphs, including WordNet [2], NELL [3], and Freebase [4], have been developed and successfully applied in intelligent service areas like information retrieval, recommendation systems, and question answering systems. Since these large-scale knowledge graphs are often incomplete and require constant supplementation, knowledge reasoning [5,6] involves deducing new entities or relationships from existing data, thus continually enhancing the knowledge graph. As a result, knowledge reasoning has emerged as a hot topic in the field of knowledge graph research in recent years.

Currently, studies on KG reasoning can be broadly categorized into three types: embedding-based reasoning, logic-based multi-hop reasoning, and path-based multi-hop reasoning, as illustrated in Figure 1. Multi-hop refers to the process of going from one node to another node in a graph network through multiple intermediate nodes. Multi-hop knowledge graph inference starts from one entity, goes through multiple entities and relationships, and finally reaches the target entity, focusing on learning the logical rules between semantics from the relational path of the knowledge graph. One popular approach for knowledge reasoning is to use neural networks to learn entity and relation embeddings [7–9]. These methods represent entities and relations as low-dimensional dense vectors and use the similarity between vectors to reason about relationships between entities or determine the truth of a given triplet, thus completing the KG. However, their capacity to capture multi-hop path relationships is limited since they rely on the triplet

representation of the KG, making them less applicable for more complex reasoning tasks. As a result, another solution is to combine multi-hop path information between entity pairs for knowledge reasoning. The Path Ranking Algorithm (PRA) [10] uses a random walk based on a restart inference mechanism to perform multiple bounded depth-first searches and utilizes supervised learning to choose more reasonable paths, thereby mitigating the aforementioned issue. Recently, multi-hop reasoning has been formulated as a Markov decision process (MDP) [11] due to its interpretability and excellent performance. Reinforcement learning (RL) [12,13] has been used to conduct efficient path search. DeepPath [14] is the first multi-hop reasoning method to transfer RL to KGs, wherein entities are mapped to states and relations to actions. The goal is to use RL to sample relationships and expand the path. These methods present new research directions for KG reasoning.

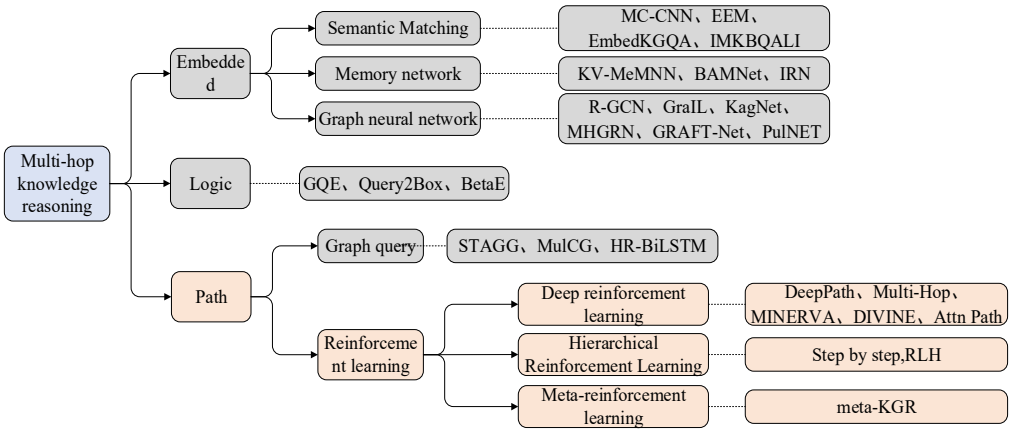

Figure 1. Status of knowledge reasoning research.

However, using RL for path search in actual knowledge reasoning tasks is inefficient. Firstly, most RL methods fail to embed entities and relations effectively when constructing the KG environment, resulting in a low success rate for the agent's path search. Secondly, the KG has many invalid paths. For example, given the triplets Person A, Field, knowledge graph; Person B, Teacher, Person A; Person B, Field, knowledge graph; Person B, Collaborate, Person C; Person C, Teacher, Person D; and Person D, Field, knowledge graph, it is possible to infer Person C, Field, knowledge graph, as depicted in Figure 2.

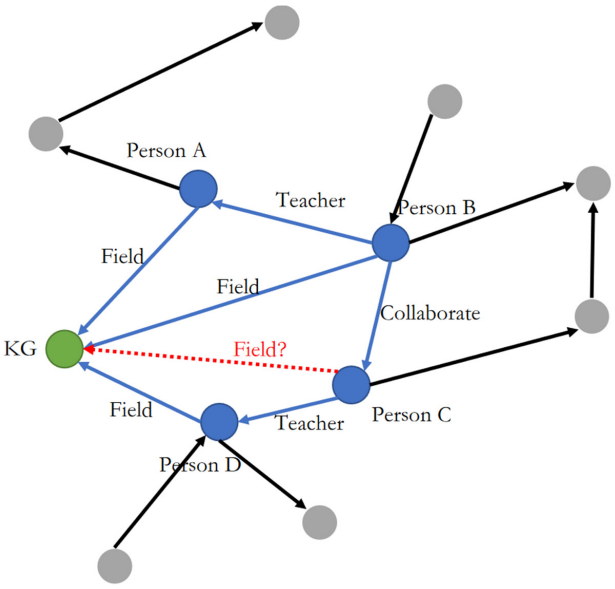

Figure 2. The most basic triplet example.

During the reasoning process, the relationship "Field" plays a critical role, and the gray dots represent invalid actions due to the inability of gray entities to serve as the subject or object of "Field". When the agent selects an invalid action during the traversal process, it retraces its steps and makes a new selection. However, there is a possibility that the agent might repeatedly choose the same invalid action, leading to inefficiency. Similarly, there may be instances where the agent selects the same valid path repeatedly, resulting in a loop.

Presently, the latest research progress on reinforcement learning is as follows. Hildebrandt et al. [15] proposed a knowledge graph reasoning method based on the dynamics of discussion, which defines a tuple classification task as a game between two opposing reinforcement learning agents. In this method, a searcher iteratively completes the model's training by searching for parameters that can persuade the discriminator to accept its position. The downside of this method might be that it could require extensive iterations and interactions, which can be computationally expensive, and it may be difficult to find a global optimum due to getting stuck in local optima. In response to the issue of multiple semantics for entities or relations in the multi-hop inference process, Wan et al. [16] proposed the RLH model that includes a hierarchical decision-making process. In this model, the high-level strategy learns from historical information, while the low-level strategy is responsible for learning specific sub-actions and the division of the action space. The limitation of this method could be the coordination and efficiency between the high-level and low-level strategies, as well as the acquisition and application of historical information in complex environments. To address the problem of sparse reward signals in knowledge reasoning, the RuleGuider [17] model uses high-quality rules generated by symbolic reasoning methods to provide reward supervision for a walk-based agent. This model combines a symbolic logic rule generation model and a walk-based rule guide inference model, enhancing the interpretability of the inference paths. However, this model's flaw might be the difficulty of generating high-quality rules and the reliance on the quality of the rules, which may not cover all inference paths sufficiently. The DacKGR [18] model, targeting the problem of multi-hop reasoning on sparse graphs, utilizes embedding models to predict the current state's action space and dynamically adds additional reliable actions to the path search space to alleviate sparsity issues in multi-hop reasoning. However, this approach may lead to an overly large state space due to the dynamic addition of actions, increasing the complexity of the search. Fu et al. [19] proposed the CPL model for open-domain knowledge graph reasoning tasks, which jointly trains a multi-hop reasoning agent and a fact extraction agent, using facts extracted from external data to assist the path reasoning process. The downside of this approach might lie in its dependency on external data and the challenge of effectively integrating external data to ensure the consistency of the information. At the same time, the LSTM model is used in all of the above methods. Typically, reinforcement learning with LSTM networks is used for knowledge graph reasoning, storing the agent's historical actions. However, the LSTM model is prone to overfitting and struggles to capture long-term dependencies in long sequence data, which can result in a performance decline on new data. All of these issues lead to a reduction in the accuracy of path selection and may make it challenging for the agent to obtain rewards from the policy network during the initial traversal stage. Typically, reinforcement learning with LSTM networks is used for knowledge graph reasoning, storing the agent's historical actions. However, the LSTM model is prone to overfitting and struggles to capture long-term dependencies in long sequence data, which can result in a performance decline on new data. All of these issues lead to a reduction in the accuracy of path selection and may make it challenging for the agent to obtain rewards from the policy network during the initial traversal stage.

The problems of the low efficiency and accuracy issues in path selection can actually be solved through "Strong memory ability". "Strong memory" refers to the ability to effectively retain and retrieve information across multiple iterations or steps of the learning process. This ability is critical for tasks that involve understanding long-term relationships or making inferences across multiple steps. Existing methods may lack this capability

due to limitations in information representation or model architecture. Therefore, in our work, we propose a fusion of representation learning and reinforcement learning, combining ConvR embeddings and reinforcement learning to form a knowledge graph reasoning method with strong memory ability, named ConvR-based reinforcement learning knowledge graph reasoning with "Strong memory ability" (ConvRL-KGM). The proposed model transforms the path selection problem into a sequence decision problem and makes three key contributions.

1. Using ConvR embeddings to generate vector representations of entities and relationships in the knowledge graph, the intelligent agent can accurately search for paths during its interaction with the knowledge graph environment, thus enhancing the efficiency of the reasoning method.
2. To address the problem of invalid actions, random dropout operations are performed on actions to reduce their interference with the intelligent agent. This enables the agent to receive comprehensive training and enhances the model's generalization ability by encouraging the policy network to select different relationships.
3. To overcome the limitations of the LSTM model, this paper proposes an improved SLSTM (SELU-activated LSTM) model that uses SLSTM to store the agent's historical actions and introduces self-attention to force the agent to choose different actions and avoid constant pauses on the same entity node. This approach also prevents overfitting to the training data and helps the model find higher success rate paths for reasoning tasks during the training process, thereby improving the model's "Strong memory ability".

2. Related Work

Our work mainly uses the combination of knowledge graph embedding and reinforcement learning and improves the memory module for knowledge reasoning, so this chapter will summarize the current work in this order.

2.1. Embedded Model-Based Reasoning

Embedding-based reasoning is a technique that maps entities and relations to a vector space, enabling reasoning by computing similarity between vectors. Several methods exist for embedding-based reasoning, including (1) translation-based models, which aim to learn embeddings by translating relations from the head entity to the tail entity, (2) bilinear models, which apply bilinear transformations to entities and relations, (3) factorization models, which represent the model as a tensor decomposition of a three-way tensor, and (4) neural network-based models, which employ fully connected neural networks, convolutional neural networks, and graph convolutional neural networks to encode semantic information. The most classical model among these is the Trans series model, illustrated in Figure 3.

TransE, proposed by Bordes et al. [8] in 2013, which regards the relationship as a certain translation between entity pairs, performs well when dealing with simple relationships, but when faced with 1-N, N-1, N-N, etc., there will be errors in complex relationships. TransH, proposed by Wang et al. [20], solves the limitations of TransE in dealing with complex relationships by setting a relationship hyperplane so that entities under different relationships have different representations. TransD, proposed by Ji et al. [21], considers the diversity of entities and relationships at the same time. By setting two projection matrices to project the head and tail entities into the relationship space, the model is more flexible. The TransE-KC model, proposed by Jincheng Xu et al. [22], learns triplets at the same time as information; the network structure characteristics and semantic information of the knowledge graph are also considered to enhance the representation effect of triples. ComplEx [23] was proposed by Trouillon et al., and the model is a machine learning model used for processing text similarity tasks. It is mainly used for learning text representation, especially for complex texts with multiple entities and relationships. The ComplEx model improves its performance in handling complex entities and relationships by combining word embeddings and entity relationship embeddings to learn the representation of text.

TuckER [24], proposed by Balažević, represents the knowledge map as a third-order binary tensor, and each element corresponds to a triplet, which has a strong ability to learn features. ConvE [9], proposed by T Dettmers et al., takes the form of convolution, defined by a single convolution layer, a projection layer with embedding dimensions, and an inner product layer, which can capture complex information.

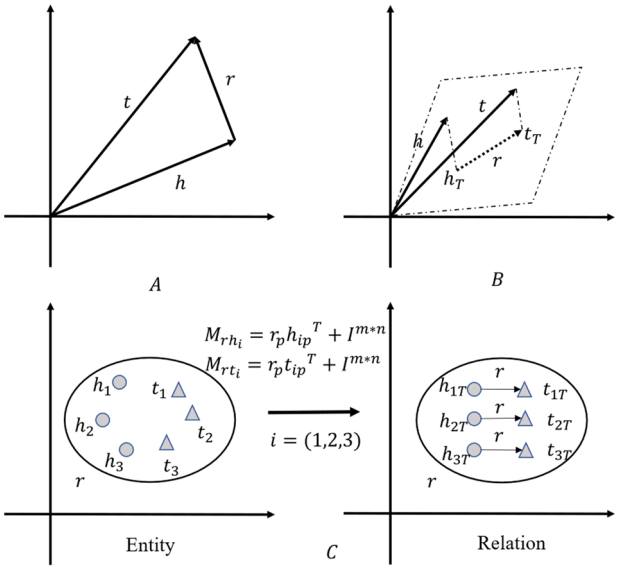

Figure 3. Trans series model (partial, PS: A is TransE, B is TransH, C is TransD).

2.2. Inference Based on Reinforcement Learning

The reasoning method based on reinforcement learning considers the path walk between entities as a Markov decision process and uses a policy-based agent to search for the reasoning path. Xiong et al. proposed the first reinforcement learning method, DeepPath [14], which considers the learning path in the knowledge map, as shown in Figure 4.

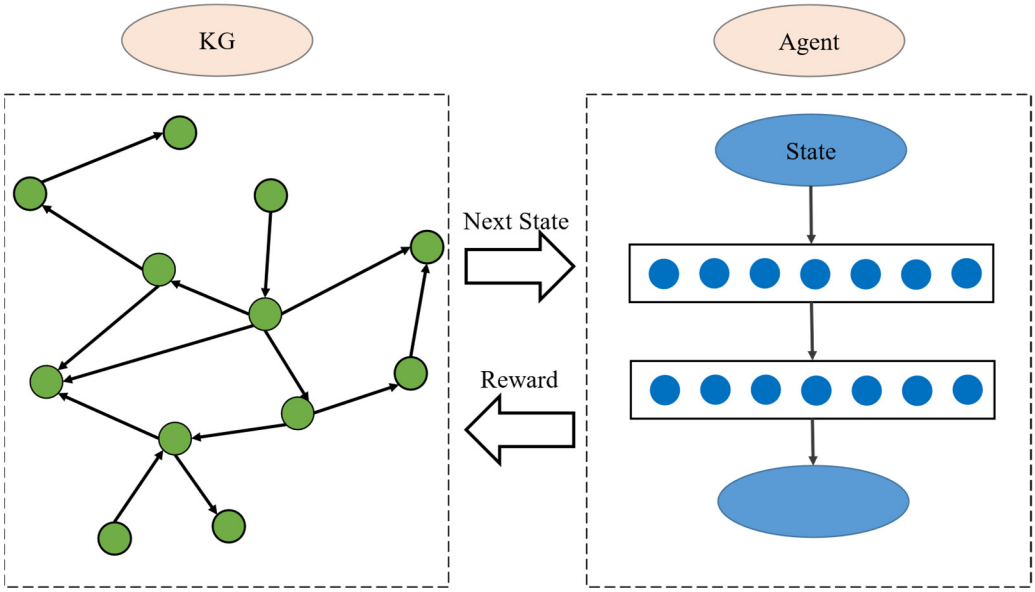

Figure 4. DeepPath model diagram.

MINERVA, proposed by Das et al. [25], is an end-to-end model for multi-hop KG query answering using reinforcement learning training. M-Walk, proposed by Shen et al. [26], is

an agent comprising an RNN and a Monte Carlo tree. It is used to encode the state of the path and generate valid paths. DIVINE, proposed by Li et al. [27], is a framework based on generative adversarial learning, which enhances RL-based reasoning in knowledge graphs by learning reasoning strategies and reward functions. Meilicke et al. [28] proposed a rule-based multi-hop inference model and introduced reinforcement learning to guide the rule sampling process, which helps to obtain more valuable rules. The RuleGuider proposed by Lei et al. [17] is an approach that leverages top-quality rules created by symbol-based methods to offer reward supervision for agents. TransPath, proposed by Cui Yuanning et al. [29], improved the success rate of path selection by adding a single-step walk to select the source task of effective actions in addition to the target task. Wan et al. [30]. proposed that human-like reasoning decomposes the entire reasoning process into a two-level policy hierarchy for the encoding of historical information and the learning of structured action spaces, improving multiple semantic issues.

2.3. Memory Module

Most of the memory modules that encode historical information in reinforcement learning knowledge reasoning use the LSTM model, as shown in Figure 5.

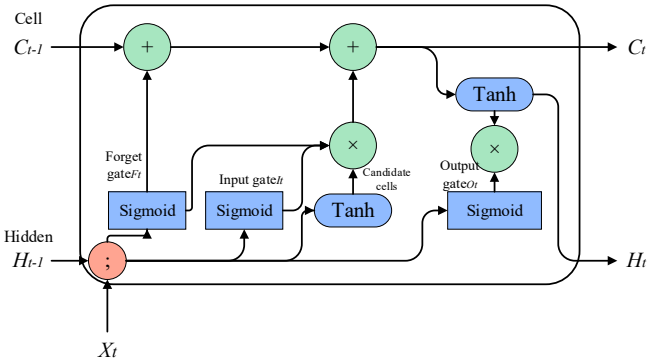

Figure 5. LSTM model.

The LSTM model typically requires a substantial amount of training data and computing resources to train, particularly when the model scales up. As the model size increases, training time and computing costs rise sharply, and overfitting becomes more likely, leading to poor performance on new data. Gers et al. [31] proposed a coupled forgetting mechanism in 2000 that merged the forget gate and input gate. They added the concept of Peephole Connection in 2001 and a cell state signal to each gate's input. Chung et al. [32] proposed the GRU model, which only has two gates, the reset gate and the update gate, removing the output gate in LSTM. GRU has fewer parameters, trains faster, and requires relatively fewer samples but is not suitable for large-sample training. Shi et al. [33]. proposed ConvLSTM to build a spatio-temporal sequence prediction model that captures both temporal and spatial information. The fully connected weight in LSTM is replaced by convolution, and each gate in the LSTM unit performs convolution operations, capturing underlying spatial features. Shen et al. [34]. designed ON-LSTM, a unique LSTM structure that encodes the hierarchical structure of sentences, thus improving LSTM's expressive ability.

Few methods combine embedding models and reinforcement learning, and reinforcement learning does not fully utilize an agent's historical path information during reasoning. To address these issues, in our work, we use the embedding ConvR method, which embeds entities and relationships in a low dimension to create the knowledge map environment. The information line is encoded to prune the agent's output path, and the SLSTM network is introduced as a memory component to store historical paths. Additionally, the self-attention mechanism is used to enhance path selection efficiency.

3. Reinforcement Learning Framework Algorithm Based on ConvR Embedding

The goal of knowledge reasoning is to predict the reliable path between entity pairs. Therefore, in order to improve the overall quality and efficiency of prediction, in our work, we propose a knowledge reasoning method ConvRL-KGM (ConvR-based Reinforcement Learning Knowledge Graph Reasoning with Strong Memory) based on reinforcement learning of ConvR embedding, which will find the path between entity pairs. Possible relationships and path information are transformed into sequential decision-making problems for reinforcement learning. The model is shown in Figure 6.

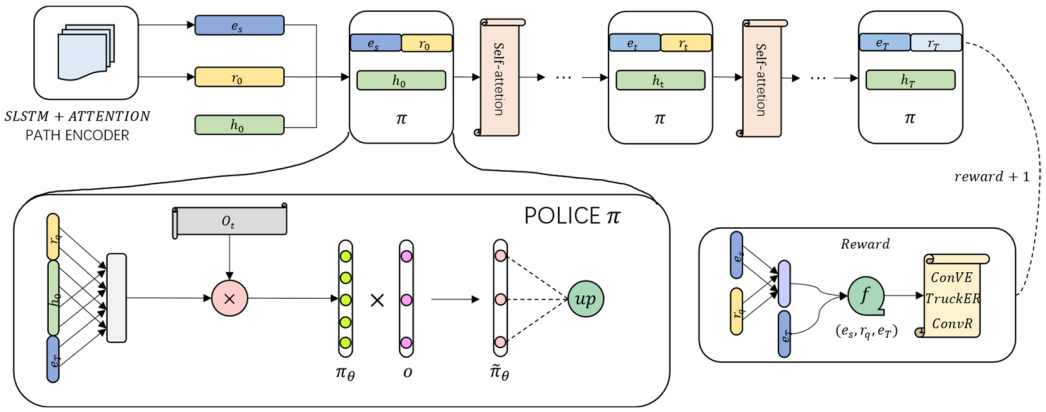

Figure 6. ConvR-based Reinforcement Learning Knowledge Graph Reasoning with Strong Memory framework. The blue of e_s represents the entity input at each step, the yellow of r_0 represents the query relationship input at each step, the green of h_0 represents the historical memory recorded in the path of each step, the pink of self-attention represents the self-attention module, and in the strategy π, green and purple represent the θ updates of the parameters.

First, we use ConvR embedding to map entities and relationships into vectors containing triplet semantic information and score the path corresponding to each entity. Then, we use the continuous vectorized representation of entities and relationships obtained by ConvR embedding as a neural network based on reinforcement learning. The input of the network enables the model to make full use of the existing triplet information of the knowledge graph, and the RL agent trains the strategy network during the walking process. Finally, action pruning and SLSTM+self-attention are used to control the path selection search and path and take effective actions in the reinforcement learning strategy to improve the overall performance of the model.

3.1. SLSTM

The core idea of the LSTM model is to use the "cell state" C_t. The "cell state" acts like a conveyor belt that runs directly throughout the entire chain, with only a small amount of linear interaction. Information flowing through it can easily be kept unchanged. LSTM removes or adds information to the cell state using a carefully designed structure called "gates". A gate is a way of selectively allowing information to pass through, and it consists of a sigmoid neural network layer and a pointwise multiplication operation. Three gate functions are introduced in LSTM: the input gate, forget gate, and output gate, which are used to control input values, memory values, and output values, respectively.

However, the LSTM model may face the problem of long-term dependencies when processing long sequence data, where earlier inputs have less impact on subsequent predictions. This is because the memory unit of the LSTM model may gradually decay as the time step increases, causing it to gradually forget earlier input information. This may lead to a decrease in the performance of the LSTM model in long sequence data. Moreover, as the amount of data increases and the network deepens, there is a risk of gradient vanishing. Therefore, in our work, we propose an improved LSTM algorithm—the SELU-activated

LSTM (SLSTM) algorithm—to address the problems of long-term dependencies, gradient vanishing, and exploding. The model is shown in Figure 5.

Based on the latest research results on activation functions, the improved LSTM model makes the state h_t and c_t enter an activation function during the gate computation process, and the output end also needs to go through an activation function. The activation function selected is the SELU function [35], which ensures that the gradient of each layer is transmitted in a stable manner during the optimization process, and the SELU function has the function of automatically converging the activation values of neurons to zero mean and unit variance, achieving the purpose of self-normalization. In addition, the normalization of SELU restricts the range of variance and also ensures robustness in multi-level training, effectively avoiding the problem of gradient vanishing and exploding during the training process. The function is presented in Eq. (1) as follows:

$$selu(x) = \lambda \begin{cases} x \; if \; x > 0 \\ \alpha(e^x - 1) \; if \; x < 0 \end{cases} \tag{1}$$

where x refers to the value entered into the activation function; both λ and α are hyper-parameters, and their values are obtained by proof rather than training. λ in formula 1 is the scaling coefficient, α is the scaling parameter, and the values of λ and α are derived mathematically, where λ is usually chosen as 1.0507 and α is usually chosen as 1.6733.

Then, the input gate is removed and combined with the forget gate, and the addition of new information and the retention of the old state are set as complementary values (with a sum of 1). That is, we only forget when new information needs to be added, and we only add new information when forgetting is necessary. For knowledge reasoning, in order to ensure the accuracy of reasoning, long-term effective memory is required. Therefore, combined with the "peephole connections" proposed by Gers and Schmidhuber, 2000 [36], the hidden gate is connected to the state of the previous unit, and the output gate is connected to the current unit. Status connection is established. The overall improved SLSTM model is shown in Figure 7.

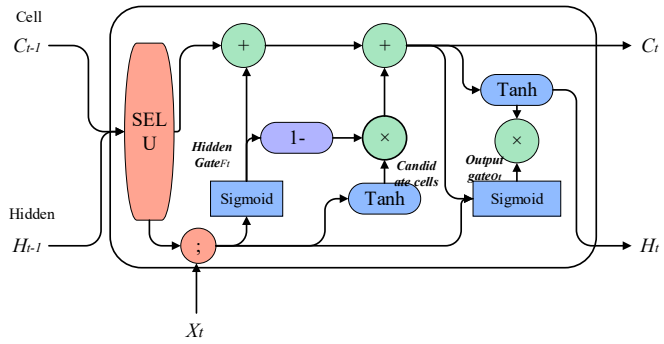

Figure 7. SELU−activated LSTM framework.

3.1.1. Input Preprocessing

The SELU function has the function that the excitation value of neurons automatically converges to zero mean and unit variance so as to achieve the purpose of self-normalization, and the normalization of SELU limits the range of variance, which can also ensure it in multi-level training. It maintains robustness, thereby effectively avoiding the problem of gradient disappearance and explosion during training. First, we use SELU to preprocess the input, and the process is shown in Figure 8.

$$h_{t-1} = SELU(H_{t-1}) \tag{2}$$

$$c_{t-1} = SELU(C_{t-1}) \tag{3}$$

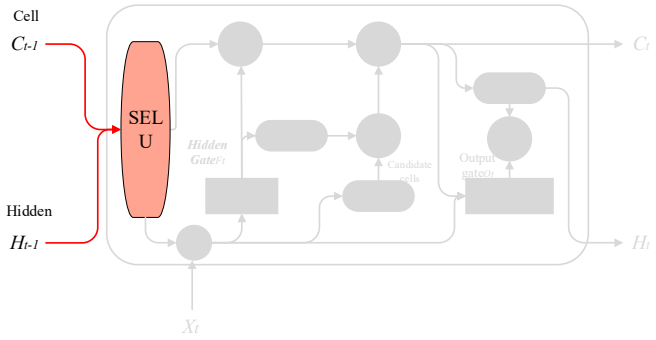

Figure 8. Input pre-training process.

3.1.2. Hidden Gate

This step is performed to decide how much information in the cell state to keep and to selectively forget the information in the cell state in the previous step. The achieved effect is the same as the forget gate in LSTM. For example, if you want the model to obtain a new subject, it will automatically hide the old subject. For example, "*She works so hard, so I. . .*", and when the model starts to deal with "I", it will hide the previous subject "she".

Implementation method: The sigmoid layer h_{t-1} in the previous step and X_t in this step are used as inputs, and then a value between 0 and 1 is output for each number in C_{t-1}, which is recorded as F_t, indicating how much information to keep (1 means completely retained, 0 means completely discarded). The process is shown in Figure 9.

$$F_t = \sigma(W_F \cdot [c_{t-1}, h_{t-1}, X_t] + b_F) \tag{4}$$

where c_{t-1} is the cell state at the previous time, h_{t-1} is the hidden state at the previous time, X_t is the value of the input sequence at the current time, W_F and b_F are the weights and biases, respectively, and σ represents the sigmoid function.

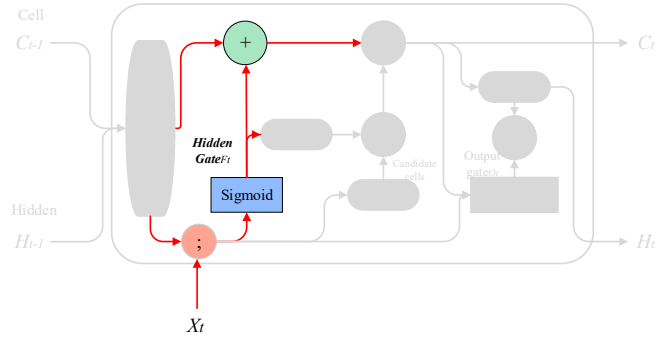

Figure 9. Hidden gate process.

3.1.3. Memory Merge

This step decides what to store in the cell state and selectively records new information in the cell state. For example, we want to add new subject categories to the cell state to replace old subjects that need to be forgotten. For example, "*She works so hard, so I. . .*", and when the model starts processing "I", it will update the subject "I" into the cell.

Implementation method: The model uses the result obtained by the hidden gate to negate so as to determine the update value. The probability is expressed as $1 - F_t$, and the *tanh* layer creates a vector of candidate values $\widetilde{C}_t$ that will be added to the cell state, which is then combined to update the cell state. The process is shown in Figure 10.

$$\widetilde{C}_t = \tanh(W_C \cdot [h_{t-1}, X_t] + b_C) \tag{5}$$

$$C_t = F_t \times c_{t-1} + (1 - F_t) \times \widetilde{C}_t \tag{6}$$

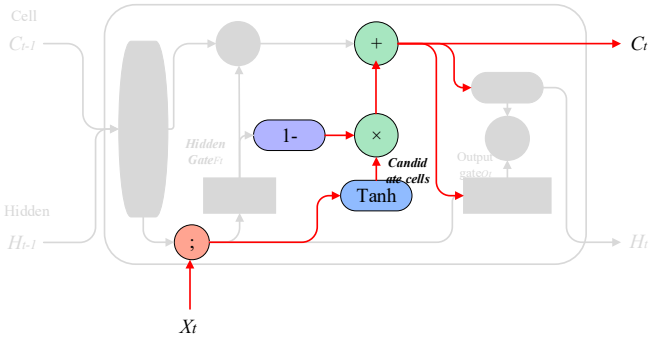

Figure 10. Memory merge process.

3.1.4. Output Gate

This step determines what kind of predictions to make.

Implementation method: The model utilizes a sigmoid layer to determine which parts of the output cell state C_t are significant. Subsequently, the cell state is passed through a tanh layer to ensure the values range between -1 and 1. The final output H_t is obtained by multiplying the output of the sigmoid layer with the output of the tanh layer. The process can be represented mathematically as follows:

Output gate calculation

$$out_t = W_o \cdot [c_{t-1}, h_{t-1}, X_t] + b_o \tag{7}$$

Final output calculation

$$H_t = out_t \times \tanh(C_t) \tag{8}$$

out_t is the output gate activation vector, W_o and b_o are the weights and bias for the output gate, c_{t-1} and h_{t-1} are the previous cell state and hidden state, respectively, X_t is the current input, H_t is the final output at the time step, and C_t is the current cell state. The process is shown in Figure 11.

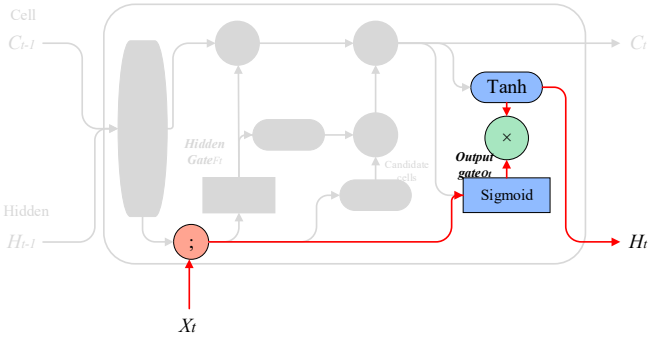

Figure 11. Output gate process.

3.1.5. The Final Memory of the Current Time Step

In order to ensure the consistency of input and output, therefore, the final output memory also needs to be processed as follows:

$$H_t^{out} = SELU(H_t) \tag{9}$$

3.2. ConvR Embedding Model

The ConvE model is insufficient to comprehensively capture the interactions between input entities and relations, as it models interactions in the region of matrix adjacency of input entities and relations. In order to maximize the interaction between entities and relations, Jiang [37] et al. proposed the ConvR model, which continues the framework of ConvE. The difference is that the relation is used as the convolution kernel. The essence lies

in adaptively building the relation embedding into a convolution kernel and interacting with the entity embedding so that each convolution is 100% interactive. The overall model is shown in Figure 12.

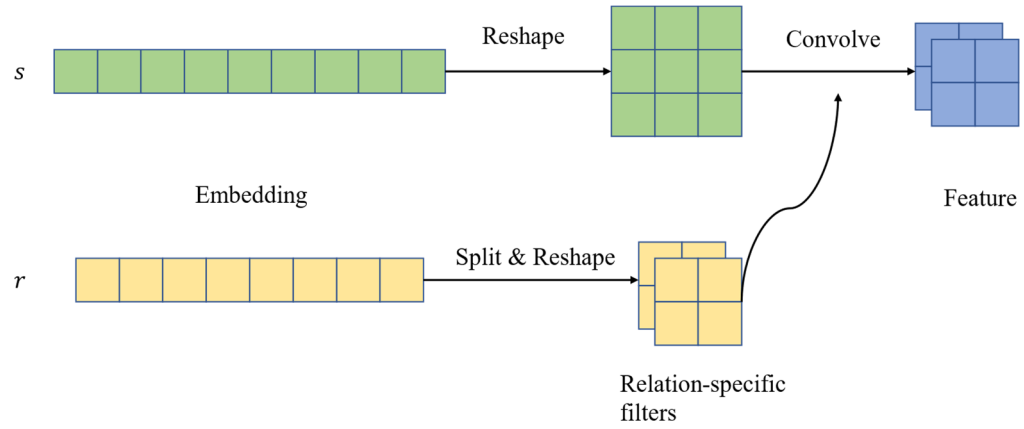

Figure 12. ConvR model diagram.

Firstly, after being given a triplet, the relational representation should be split and reshaped into a set of filters. Next, the relational representation of the head entity should be reshaped and used as the input for the convolutional layer. Then, the filter should be applied to the input to capture interactions between filters (part of the relational representation) and different regions of the input (entity representation). Finally, the convolutional features are projected and matched to the representation of the tail entity. In comparison to ConvE, which utilizes a global filter, ConvR employs an adaptive filter that is constructed from a relational representation and uses the relation as a convolution kernel.

In order to use the relation as the convolution kernel, the relation embedding is first divided into several small blocks $r^1 \dots r^n$ of the same size. This process is called "Split". At the same time, because 2D convolution is used, each small block $r^\ell \in \mathbb{R}^{d_r/c}$, is reshaped into a height h, width w, and convolution kernel $R^\ell \in \mathbb{R}^{h*w}$, $d_r = chw$.

During convolution, the relational specialized convolution kernel R^ℓ is used for operations, and the mathematical expression of the convolution process is as follows:

$$c^\ell_{m,n} = f\left(\sum_{i,j} s_{m+i-1,n+j-1} \times r^\ell_{i,j}\right) \tag{10}$$

where $c^\ell_{m,n}$ represents the response of the l-th filter at position (m, n) on the convolved feature map. The function f is an activation function, such as ReLU, etc. $s_{m+i-1,n+j-1}$ is a part of the input entity embedding, and $r^\ell_{i,j}$ is the value at position (i, j) in the l-th filter. The entire summation process is the convolution operation, which calculates the dot product of the filter with the entity embedding at each position.

For instance, when training on the FB15K-237 dataset, the data that participate in the convolution primarily include entity embeddings and relation embeddings. Entity embeddings are representations of entities obtained through pre-training of the model, while relation embeddings are a series of convolutional filters obtained through splitting and reshaping the relational representations. These embeddings are updated along with the model parameters during the training process to ultimately capture the complex interactions between entities.

3.3. Reasoning Framework Algorithm

ConvRL-KGM is a method of transforming simple reasoning tasks into Markov decision problems, where the state transitions and action selection of the agent are all performed

within a knowledge graph environment. This section mainly focuses on how to convert knowledge graphs into decision problems for intelligent agents.

3.3.1. State

The relationship selection strategy is not only dependent on the current entity information but also on the queried relationship. At time step t, the state s_t is composed of the source entity e_s, the queried relationship r_q, and the current entity e_t. Its representation is as follows:

$$s_t = (e_s, r_q, e_t) \tag{11}$$

Given a pair of entity relationships (e_s, r_q, e_{target}), the initial state is represented as (e_s, r_q, e_s), and the agent starts its traversal from the source entity e_s.

3.3.2. Action Space

Let O_t denote all the relationships in the knowledge graph $\mathcal{G}$ that are associated with the current entity e_t at state s_t.

$$O_t = \{r | e_t, (e_t, r, e')\} \tag{12}$$

where e' represents the tail entity. To select the action o_t from O_t based on the policy π, a fixed time step limit is imposed for the search process to facilitate a comprehensive search.

3.3.3. State Transition

State transition refers to the process by which an agent moves from the current state to the next state based on the action taken. Specifically, the agent, in the current state, selects an action, and the transition to the next state is determined by the interaction with the environment, represented by the knowledge graph $\mathcal{G}$. The state transition probability P is denoted as follows:

$$P : S \times O \to S \tag{13}$$

$$P = (S_{t-1} = s' | S_t = s, O_t = o) \tag{14}$$

3.3.4. Construction of Reward Function Based on Knowledge Embeddings

In the previous reinforcement learning strategy, the agent only received binary rewards based on the observed answers in $\mathcal{G}$.

$$R(s_t) \begin{cases} 1, \ if \ e_t = e_{target} \\ 0, \ else \end{cases} \tag{15}$$

However, $\mathcal{G}$ is inherently incomplete, and this approach assigns the same reward to negative outcomes as positive outcomes. To address this, we adopt the ConvR embedding approach and set soft rewards for result entities whose correctness are unknown. Formally, the embedding model maps entities E and relations R into a vector space and uses the ConvR composite function to estimate the likelihood of each fact. Therefore, we shape the policy with the following reward function:

$$R(s_T) = R(s_t) + (1 - R(s_t)) ConvR(e_t, r, e') \tag{16}$$

The environment returns intermediate rewards $R(s_t)$ for subsequent reward estimation. If the agent reaches the target entity, a reward of 1 is returned at the current time step; otherwise, a penalty less than 1 is returned. To evaluate the reliability of the predicted triplet (e_t, r, e'), we utilize the pre-trained embedding model ConvR to score it.

3.3.5. Policy

1. Dynamic policy space pruning

Due to the fact that invalid paths are often more numerous than correct paths and are easier to detect, the burden of path search is increased, especially when the KG grows with the increase in path length, resulting in an exponential increase in action space and thus increasing the search burden. This phenomenon is even more severe for entities with large outdegrees (i.e., entities with many connecting relations). Enumerating all possible relation paths between entity pairs on a large-scale knowledge graph is infeasible, so conducting effective path exploration and identifying reasoning paths is particularly important.

Therefore, we leverage the advantage of ConvR embedding to obtain, in the action space, all tail entities that are connected to the corresponding head entity and relation. Specifically, for the current entity and selected relation, we compute the probability of all entities becoming the tail entity and then select the top K entities as our action space, where K is predetermined for each dataset. A large K value may lead to information confusion, while a small K value may lead to insufficient information. Therefore, we can refine the action space to enable the goal to reach the target entity.

2. Policy network

Each entity and relation in the knowledge graph (KG) are represented by a low-dimensional dense vector with semantic information through ConvR embeddings. In order to construct a search sequence more effectively, the agent needs to not only understand the current information but also the past events that have occurred. Therefore, we utilize the SLSTM algorithm proposed in this paper to encode the historical information of the search history $(e_s, r_1, e_1, r_2, \ldots e_t)$ into a hidden state denoted as h_t, representing the historical information at time step t. The output of the historical information is then passed through a self-attention layer. Assuming the initial state is $h_0 = 0$, the search history $h_t = (e_s, r_1, e_1, r_2, \ldots e_t)$ is composed of the action sequence taken from step 1 to step t.

$$h_0 = SLSTM(0, r_0) \tag{17}$$

$$h_t = SLSTM((h_{t-1}, c_{t-1}), o_{t-1}) \tag{18}$$

In the self-attention layer, we first set up a linear function f_s, which should be twice the size of h_t due to the use of bidirectional SLSTM in reinforcement learning knowledge reasoning.

$$f_s = f(h_t * h_t, 1) \tag{19}$$

Then, we calculate the attention weights using the SoftMax function.

$$a_s = softmax(f_s) \tag{20}$$

Finally, we obtain the final output h_t by multiplying the attention weights with the linear function f_s.

$$h_t = a_s \cdot f_s \tag{21}$$

The reinforcement learning policy π, which maps state s_t to action space O_t, can be represented as $\pi : s_t \rightarrow O_t$, where $O_t \in R^{|O_t| \times d_1}$ and d_1 is the embedding dimension.

$$\pi_\theta(o_t|s_t) = \sigma\big(O_t \times W_2 ReLU\big(W_1\big[e_t; r_q; h_t\big]\big)\big) \tag{22}$$

where σ represents the SoftMax function, ";" denotes the concatenation operation, $W_1 \in R^{(d_1+d_2+d_3)\times d_4}$, $W_2 \in R^{d_1 \times d_4}$, and d_2 is the entity dimension, d_3 is the dimension of historical hidden states, and d_4 is the number of hidden units in the fully connected layer. Then, a dropout method is added to make the process more random. The upgraded strategy is expressed as follows:

$$\widetilde{\pi}_\theta(o_t|s_t) = \pi_\theta(o_t|s_t) \cdot m + \varepsilon \tag{23}$$

where m is sampled from a Bernoulli distribution, which randomly masks some options in its action space. ε is a very small value when $m = 0$.

3. Optimization of the Policy Function

The model employs the REINFORCE gradient policy method to optimize the parameters θ of the policy network π_θ following the formula

$$J(\theta) = \mathbb{E}_{(e_h,r,e_t)\epsilon\mathcal{G}}[\mathbb{E}_{o_1,\ldots,o_T\sim\pi_\theta} Reward(s_T|(e_s,r))] \tag{24}$$

where $J(\theta)$ represents the batch reward and $\mathbb{E}$ denotes the expectation over different triplets in the training set. The REINFORCE gradient policy method utilizes a series of historical trajectories (iterating over all triplets in $\mathcal{G}$) generated by the current policy to estimate the stochastic gradient, which is then used to update the parameters as follows:

$$\nabla_\theta J(\theta) \approx \nabla_\theta \sum_t Reward(s_T|(e_s,r)) log\pi_\theta(o_t|s_t) \tag{25}$$

The training process of the model is shown in Algorithm 1

Algorithm 1: Policy Network

Initialize $s_t \leftarrow 0$, time step $T \leftarrow 0$
Initialize the policy network π_θ
Encode historical information h_t using SLSTM+self-attention
for n $\leftarrow 1$ to Epoch num **do**
  for t $\leftarrow 1$ to Time step **do**
    Initialize s_t
    Generate action space $o_t = \pi_\theta(o_t|s_t)$
    Update hidden state h_t
    Obtain reward R
  end for
  Optimize parameters θ
end for

4. Experiment Settings

4.1. Dataset

In this work, we conducted experiments using three datasets: (1) the NELL-995 dataset, which was generated from the 995th dump of the "Never Ending Language Learning" project and (2) the FB15K-237 dataset, which is a subset of FB15K with reverse relations removed. It is a knowledge base where all entities are linked to the Wikipedia database. It contains facts related to real-world entities, such as movies, actors, awards, sports, etc. It has rich relationship types and is suitable for tasks such as reasoning and link prediction in knowledge graphs. (3) We also used the Kinship dataset, which is a knowledge graph dataset representing kinship relations among family members. The dataset is shown in Table 1.

Table 1. The specific format of the datasets.

Dataset	#Ent	#Rel	#Fac	#Meandegree
FB15K-237%	14,505	237	310,116	21.38
NELL-995	75,492	200	154,213	4.07
Kinship	104	25	8544	85.15

These three datasets are publicly available in the field of knowledge graph research. They have different scales, domains, and task characteristics, making them suitable for different research needs and having high application value in relevant fields.

FB15K-237 (https://www.microsoft.com/en-us/download/details.aspx?id=52312) (accessed on 10 March 2024): This dataset contains facts about real-world entities and is suitable for studying tasks such as reasoning and link prediction in knowledge graphs.

NELL-995 (https://download.csdn.net/download/guotong1988/10313268?utm_source=bbsseo) (accessed on 10 March 2024): The NELL-995 dataset contains basic semantic relationships between data, such as city, company, emotion, and sports team. It also covers multiple domains, including characters, locations, organizations, time, events, etc. The NELL-995 dataset has a rich set of entities and relationship types, making it suitable for diverse research in the field of knowledge graph.

Kingship (https://gitcode.com/Colinasda/KGdatasets/overview?utm_source=csdn_github_accelerator) (accessed on 10 March 2024): The Kinship dataset primarily includes familial relationships between family members, such as parents, children, siblings, grandparents, great grandparents, etc. It is used for research and analysis in fields such as familial relationships, family structure, sociology, anthropology, etc., including social network analysis, family history research, population studies, and more.

4.2. Compare Models

We compared ConvRL-KGM with existing knowledge graph reasoning methods, including DeepPath, MINERVA, and multi-hop KG, as well as two recent reinforcement learning-based models, TuckRL and RLH, for multi-hop reasoning.

DeepPath (https://github.com/xwhan/DeepPath) (accessed on 10 March 2024): A study that initially applies reinforcement learning for knowledge graph reasoning.

MINERVA (https://github.com/shehzaadzd/MINERVA) (accessed on 10 March 2024): An end-to-end model trained with reinforcement learning for answering multi-hop KG queries.

Multi-hop KG (https://github.com/JiapengWu/MultiHopKG?tab=readme-ov-file) (accessed on 10 March 2024): Proposed pre-training supervision to estimate unseen rewards and introduced a random masked edge mechanism to explore more diverse paths.

TuckRL (http://www.c-s-a.org.cn/html/2022/9/8681.html) (accessed on 10 March 2024): Utilizes TuckER embeddings to encode states and trains the entire policy network using MDP.

RLH (https://opus.lib.uts.edu.au/bitstream/10453/157768/2/0267.pdf) (accessed on 10 March 2024): Adopts a hierarchical reinforcement learning approach, where knowledge reasoning is separated into two parts, actions, and relations, with no mutual influence.

4.3. Evaluation Index

1. Hits@K metric

The Hits@K metric, where K represents a specific value, such as 1, 3, or 10, is used to measure the performance of knowledge graph reasoning algorithms. Hits@K calculates the proportion of correctly predicted triples that are ranked within the top K positions after scoring. It is computed as the cumulative sum of correctly predicted triples ranked within the top K positions divided by the total number of triples in the test set, resulting in a value between 0 and 1. "$rank_i$" refers to the ordering of a pair of relations and entities by replacing the head or tail entity with any other entity (n-1 in total, leaving the other entity and the relationship unchanged, changing only one of the entities), thus obtaining (n-1), a new triplet of relations. Then, the entity relation distance is calculated for these triples, and the n-1 triples are arranged in order of distance from smallest to largest. A higher Hits@K value indicates better performance of the reasoning algorithm. The formula for calculating Hits@K is as follows:

$$Hits@K = \sum_{i} \frac{(rank_i) < K}{N} \tag{26}$$

2. Mean Reciprocal Rank (MRR)

The Mean Reciprocal Rank (MRR) is a metric that reflects the overall ranking of correct triples in the candidate list after scoring by the model's scoring function. The MRR

calculates the average reciprocal rank of correctly predicted triples based on their ranking in the list of candidates according to their scores. The reciprocal rank is obtained by taking the inverse of the rank so that higher ranks have smaller reciprocal ranks. The MRR is computed as the average of reciprocal ranks of correctly predicted triples, resulting in a value between 0 and 1. A higher MRR value closer to 1 indicates better performance of the model. The formula for calculating MRR is as follows:

$$MRR = \frac{1}{N}\sum_{i}\frac{1}{rank_i} \tag{27}$$

5. Experimental Results and Analysis

In this section, a series of experiments were conducted to validate the effectiveness of the proposed framework for multi-hop knowledge graph reasoning. Firstly, the proposed framework was evaluated on four benchmark datasets and compared with state-of-the-art models. Secondly, ablation studies were conducted on the proposed model to demonstrate the necessity of each module. Thirdly, our framework was evaluated based on different relationship types and unseen queries. Finally, we investigated how different embedding dimensions and maximum path lengths affect the performance of our framework.

5.1. Model Comparison

We compare the proposed ConvRL-KGM with the baseline models set in Section 4.2, and the results are presented in Table 2.

Table 2. Experimental comparison results.

Model	NELL-995				FB15K-237				KINSHIP			
	@1	**@3**	**@10**	**MRR**	**@1**	**@3**	**@10**	**MRR**	**@1**	**@3**	**@10**	**MRR**
DeepPath	52.6	61.5	77.3	63.8	16.9	24.8	35.7	22.7	-	-	-	-
MINERVA	66.3	77.3	83.1	72.5	21.7	32.9	45.6	29.3	60.5	-	92.4	72
Mulithop KG	65.6	79.2	84.4	72.7	34.6	44.3	58.1	42.1	78.9	92.7	98.0	86.5
TuckRL	65.8	78.5	83.5	73.1	36.9	46.3	59.1	42.6	-	-	-	-
RLH	69.2	-	87.3	72.3	-	-	-	-	-	-	-	-
Ours	69.9	81.6	86.9	75.1	36.5	45.8	59.5	42.7	79.2	92.6	98.2	86.9

According to Table 2, it can be seen that our proposed model, ConvRL-KGM, outperforms most of the models. Although the embedding-based model is relatively simple, its overall performance on multiple datasets is very significant. The reason may be that the embedding-based method can map each triple of the KG to the embedding space, thereby encoding the connectivity of the entire graph. ConvRL-KGM also takes advantage of this feature of the embedding model and achieves good experimental results. It shows significant improvements on the NELL-995 and KINSHIP datasets compared to other methods, especially the FB15K-237 dataset, where the correct rate of action selection is low in other models due to the long path length between entities. In ConvRL-KGM, using ConvR embedding actions for action pruning and SLSTM encoding paths effectively avoids selecting invalid actions. Furthermore, on the large-scale NELL-995 dataset, our proposed method shows significant improvement compared to multi-hop KG using ConvE, as ConvR directly removes the convolution kernel and uses the relationship directly as the convolution kernel, making each convolution 100% interactive and improving the accuracy by enhancing the role of the relationship.

5.2. Ablation Experiment

To verify the correctness and effectiveness of the added model, we conducted three sets of experiments using a controlled variable method: (1) removing ConvR embedding and using ConvE as the embedding model, (2) using LSTM as the memory module, and (3) removing the self-attention layer. We compared their MRRs with the entire model on the

development set, as shown in Table 3. In most datasets, deleting each component resulted in a significant performance decrease. Generally, removing action dropout had a greater impact, indicating that thorough exploration of the path space is crucial in the dataset.

Table 3. Results of ablation experiments.

Model	NELL-995				FB15K-237				KINSHIP			
	@1	@3	@10	MRR	@1	@3	@10	MRR	@1	@3	@10	MRR
ConvRL-KGM	69.9	81.6	86.9	75.1	36.5	45.8	59.5	42.7	79.2	92.6	98.2	86.9
ConvRL-KGM (-ConvR)	65.6	79.2	84.4	72.7	34.6	44.3	58.1	42.1	78.9	92.7	98.0	84.5
ConvRL-KGM (-SLSTM)	63.7	75.6	83.2	72.1	33.1	43.8	57.2	41.2	77.3	90.2	96.1	84.9
ConvRL-KGM (self-attention)	68.9	80.2	84.5	74.1	34.8	44.3	58.3	41.6	78.9	91.9	97.2	86.2

5.3. Path Diversity Exploration

To compute the total number of unique paths explored by the agent during training and visualize their changes, we include edge labels and intermediate entities when calculating unique paths. This is shown in Figure 13.

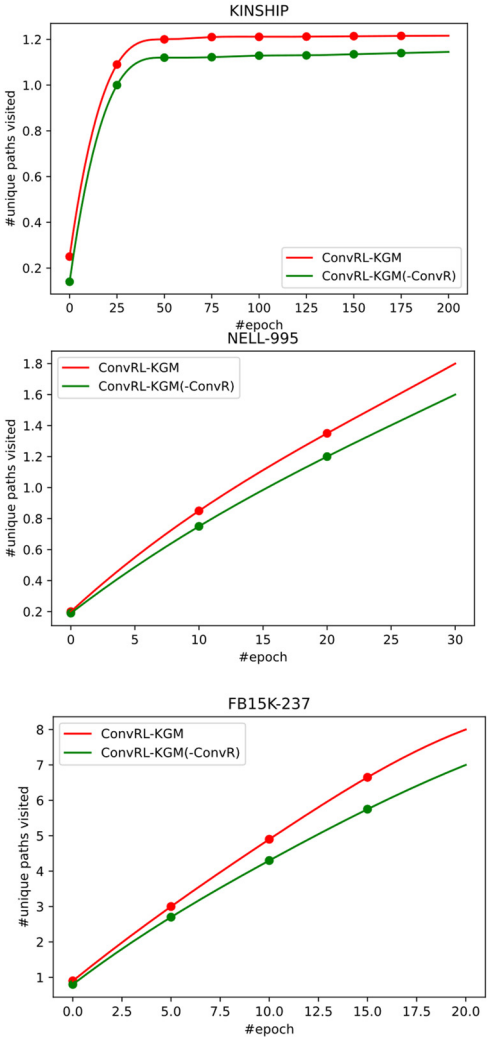

Figure 13. Number of unique paths in KINSHIP, NELL-995, and FB15K-237.

In Figure 13, it is observed that a large number of paths need to be explored by the agent before reaching a good performance level across all datasets. As training progresses, the speed of path discovery slows down. On the smaller $\mathcal{G}$ (KINSHIP), the rate of encountering new paths decreases significantly after a certain number of epochs, and the development set accuracy tends to stabilize accordingly. On larger $\mathcal{G}$s (FB15k-237 and NELL-995), there is no clear slowdown before severe overfitting occurs and development set performance begins to decrease. One possible reason is that larger $\mathcal{G}$s are more sparsely connected, so the efficiency of obtaining generalizable knowledge from $\mathcal{G}$ by sampling and exploring a limited proportion of the path space is lower.

Although removing some modules significantly reduces the effectiveness of path exploration, removing the ConvR reward shaping slightly improves the paths accessed during training across all datasets. This suggests that the correlation between explored paths and development set performance is not strictly positive. Typically, the best-performing model is not the one that explores the most paths.

Influence of maximum path length on prediction datasets

In the ConvRL-KGM framework, the maximum length of the pre-specified path is provided for the agent to search. The influence of different maximum path lengths was studied through experiments. The agent can traverse the graph in a shorter number of steps for datasets with shorter path lengths, while larger datasets require longer steps. Experiments were conducted on the maximum lengths of {2, 3, 4, 5} on NELL-995. The results are shown in Figure 14.

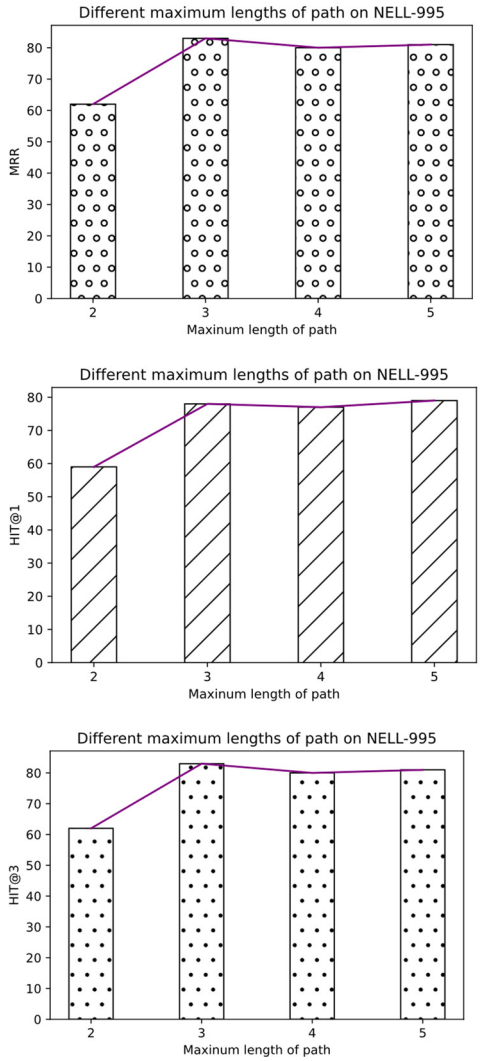

Figure 14. *Cont.*

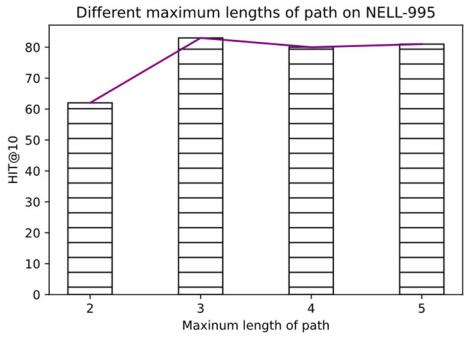

Figure 14. Different maximum path lengths on the NELL-995 dataset.

For larger datasets, the growth rate of performance slows down when the maximum length of the path exceeds three. Although the agent does not need excessively long paths, it can still use different strategies to reach the correct answer, showing strong flexibility.

6. Conclusions and Future Work

In this paper, we propose a knowledge graph inference model based on embedding-based reinforcement learning, ConvRL-KGM. This method adopts a new path traversal and reward strategy based on embedded reinforcement learning, which integrates ConvR's action pruning mechanism, SLSTM neural network, and self-attention mechanism into the relationship selection and path traversal process of reinforcement learning agents.

The improved SLSTM model based on LSTM effectively solves the problem of constant pauses at the same node in the process of knowledge reasoning and can more accurately capture the complex dependence between entities and relations, which is very suitable for complex relations and deep understanding tasks. Thanks to the self-attention mechanism, the LSTM can be endowed with better sequence processing capabilities, so that the model can better capture long-distance dependencies, thus achieving more accurate reasoning. According to the existing technology, traditional LSTM tends to have the problem of gradient disappearance and explosion when dealing with long sequences, resulting in limited feature learning. The SLSTM proposed in this study effectively solves this problem, which can better learn effective feature representations, and is more suitable for dynamically changing knowledge graphs, such as real-time monitoring systems and social media analysis. In addition, the SLSTM model can better deal with structured knowledge data and unstructured text data and is suitable for the scenario of integrating multiple data types. In addition, ConvRL-KGM, compared with the previous reinforcement learning methods, is more conducive to the agent to effectively mine high-quality paths in the inference process so as to complete the knowledge graph inference task. ConvR embedment is introduced as a reward function so that the complex relationship between entities can be fully learned in the inference process, which is of great importance in improving the accuracy and efficiency of inference. Traditional multi-hop reasoning is not effective in the face of complex reasoning because the interaction between entities cannot be fully utilized. The ConvR embedding introduced in this study effectively solves the limitations of traditional multi-hop reasoning.

Meanwhile, this method also has certain limitations. For example, it relies heavily on data quality, as errors or incomplete data will affect the accuracy of reasoning, and the generalization ability is limited, so it needs to be adjusted for specific tasks. In future studies, it is necessary to explore the impact of different reinforcement learning methods on reasoning, and the generalization ability of the model needs to be further improved to make it adapt to more fields and tasks. It is considered to combine the improvements in this study with other cutting-edge technologies, such as Variational Autoencoders (VAEs), to explore new application scenarios and functions. At the same time, meta-learning can be introduced to solve the inference problem of small sample data, which may bring better inference performance.

Author Contributions: J.G.: literature research, data processing, methods, and writing; X.Z.: methods, innovation analysis, and supervision; K.L.: funding acquisition, supervision, and writing—review. G.Z.: literature research and experimental analysis. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Natural Science Foundation of China (No. 62377036).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Lin, J.; Zhao, Y.; Huang, W.; Liu, C.; Pu, H. Domain knowledge graph-based research progress of knowledge representation. *Neural Comput. Appl.* **2021**, *33*, 681–690. [CrossRef]
2. Miller, G.A. WordNet: A lexical database for English. *Commun. ACM* **1995**, *38*, 39–41. [CrossRef]
3. Carlson, A.; Betteridge, J.; Kisiel, B.; Settles, B.; Hruschka, E.; Mitchell, T. Toward an architecture for never-ending language learning. In Proceedings of the 24th AAAI Conference on Artificial Intelligence, Atlanta, GA, USA, 11–15 July 2010; pp. 1306–1313.
4. Bollacker, K.; Evans, C.; Paritosh, P.; Sturge, T.; Taylor, J. *Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge*; ACM: Vancouver, BC, Canada, 2008; pp. 1247–1250.
5. Chen, X.; Jia, S.; Xiang, Y. A review: Knowledge reasoning over knowledge graph. *Expert Syst. Appl.* **2020**, *141*, 112948. [CrossRef]
6. Zeng, X.; Tu, X.; Liu, Y.; Fu, X.; Su, Y. Toward better drug discovery with knowledge graph. *Curr. Opin. Struct. Biol.* **2022**, *72*, 114–126. [CrossRef] [PubMed]
7. Yang, B.; Yih, W.T.; He, X.; Gao, J.; Deng, L. Embedding entities and relations for learning and inference in knowledge bases. *arXiv* **2014**, arXiv:1412.6575.
8. Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; Yakhnenko, O. Translating embeddings for modeling multi-relational data. In Proceedings of the 26th International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–8 December 2013; Curran Associates Inc.: Lake Tahoe, NV, USA, 2013; pp. 2787–2795.
9. Dettmers, T.; Minervini, P.; Stenetorp, P.; Riedel, S. Convolutional 2D knowledge graph embeddings. In Proceedings of the 32nd AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; pp. 1811–1818.
10. Lao, N.; Mitchell, T.M.; Cohen, W.W. Random walk inference and learning in a large-scale knowledge base. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Edinburgh, UK, 27–29 July 2011; ACM: Edinburgh, UK, 2011; pp. 529–539.
11. Bellman, R. A Markovian decision process. *J. Math. Mech.* **1957**, *6*, 679–684. [CrossRef]
12. Williams, R.J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* **1992**, *8*, 229–256. [CrossRef]
13. Akanksha, E.; Sharma, N.; Gulati, K. Review on reinforcement learning, research evolution and scope of application. In Proceedings of the 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 8–10 April 2021; pp. 1416–1423.
14. Xiong, W.H.; Hoang, T.; Wang, W.Y. DeepPath: A reinforcement learning method for knowledge graph reasoning. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Copenhagen, Denmark, 9–11 September 2017; Association for Computational Linguistics: Copenhagen, Denmark, 2017; pp. 564–573.
15. Hildebrandt, M.; Serna, J.A.Q.; Ma, Y.; Ringsquandl, M.; Joblin, M.; Tresp, V. Reasoning on knowledge graphs withdebate dynamics. *Proc. AAAI Conf. Artif. Intell.* **2020**, *34*, 4123–4131.
16. Wan, G.; Pan, S.; Gong, C.; Zhou, C.; Haffari, G. Reasoning like human: Hierarchical reinforcement learning for knowledge graph reasoning. In Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on ArtificialIntelligence, Montreal, QC, Canada, 19–27 August 2021; pp. 1926–1932.
17. Lei, D.; Jiang, G.; Gu, X.; Sun, K.; Mao, Y.; Ren, X. Learning collaborative agents with rule guidance forknowledge graph reasoning. In Proceedings of the 2020 Conference on EmpiricalMethods in Natural Language Processing, Virtual, 16–20 November 2020; pp. 8541–8547.
18. Lv, X.; Han, X.; Hou, L.; Li, J.; Liu, Z.; Zhang, W.; Zhang, Y.; Kong, H.; Wu, S. Dynamic anticipation and completion for multi-hopreasoning over sparse knowledge graph. In Proceedings of the 2020 Conferenceon Empirical Methods in Natural Language Processing (EMNLP), Virtual, 6–20 November 2020; pp. 5694–5703.
19. Fu, C.; Chen, T.; Qu, M.; Jin, W.; Ren, X. Collaborative policy learning for open knowledge graphreasoning. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, Hong Kong, China, 3–7 November 2019; pp. 2672–2681.
20. Wang, Z.; Zhang, J.; Feng, J.; Chen, Z. Knowledge graph embedding by translating on hyperplanes. In Proceedings of the 28th AAAI Conference on Artificial Intelligence, Québec City, QC, Canada, 27–31 July 2014; pp. 1112–1119.

21. Ji, G.; He, S.; Xu, L.; Liu, K.; Zhao, J. Knowledge graph embedding via dynamic mapping matrix. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, Beijing, China, 26–31 July 2015; Association for Computational Linguistics: Beijing, China, 2015; pp. 687–696.
22. Xu, J.; Ge, Y.; Wu, Z. An Improved Translation-Based Method for Knowledge Graph Representation. In Proceedings of the 2020 3rd International Conference on E-Business, Information Management and Computer Science, Wuhan, China, 5–6 December 2020; pp. 555–560.
23. Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, É.; Bouchard, G. Complex embeddings for simple link prediction. In Proceedings of the 33nd International Conference on Machine Learning, New York, NY, USA, 20–22 June 2016; PMLR: New York, NY, USA, 2016; pp. 2071–2080.
24. Balažević, I.; Allen, C.; Hospedales, T.M. TuckER: Tensor factorization for knowledge graph completion. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, Hong Kong, China, 3–7 November 2019; Association for Computational Linguistics: Hong Kong, China, 2019; pp. 5185–5194.
25. Das, R.; Dhuliawala, S.; Zaheer, M.; Vilnis, L.; Durugkar, I.; Krishnamurthy, A.; Smola, A.; McCallum, A. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In Proceedings of the 6th International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
26. Shen, Y.; Chen, J.; Huang, P.S.; Guo, Y.; Gao, J. M-Walk: Learning to walk over graphs using Monte Carlo tree search. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 3–8 December 2018; Curran Associates Inc.: Montreal, QC, USA, 2018; pp. 6787–6798.
27. Li, R.P.; Cheng, X. DIVINE: A generative adversarial imitation learning framework for knowledge graph reasoning. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, Hong Kong, China, 3–7 November 2019; Association for Computational Linguistics: Hong Kong, China, 2019; pp. 2642–2651.
28. Meilicke, C.; Chekol, M.W.; Fink, M.; Stuckenschmidt, H. Reinforced anytime bottom up rule learning for knowledge graph completion. *arXiv* **2000**, arXiv:2004.04412.
29. Cui, Y.; Li, J.; Chen, Y.; Lu, Z. TransPath: A Knowledge Reasoning Method Based on Deep Transfer Reinforcement Learning. *J. Chin. Mini-Micro Comput. Syst.* **2022**, *43*, 536–543.
30. Gers, F.A.; Schmidhuber, J.; Cummins, F. Learning to forget: Continual prediction with LSTM. *Neural Comput.* **2000**, *12*, 2451–2471. [CrossRef]
31. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv* **2014**, arXiv:1412.3555.
32. Shi, X.; Chen, Z.; Wang, H.; Yeung, D.Y.; Wong, W.K.; Woo, W.C. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 802–810.
33. Shen, Y.; Tan, S.; Sordoni, A.; Courville, A. Ordered neurons: Integrating tree structures into recurrent neural networks. *arXiv* **2018**, arXiv:1810.09536.
34. Klambauer, G.; Unterthiner, T.; Mayr, A.; Hochreiter, S. Self-normalizing neural networks. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.
35. Gers, F.A.; Schmidhuber, J. Recurrent nets that time and count. In Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium, Como, Italy, 27–27 July 2000; Volume 3, pp. 189–194.
36. Jiang, X.; Wang, Q.; Wang, B. Adaptive convolution for multi-relational learning. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA, 2–7 June 2019; Volume 1 (Long and Short Papers), pp. 978–987.
37. Krzyston, J.; Bhattacharjea, R.; Stark, A. Complex-Valued Convolutions for Modulation Recognition using Deep Learning. In Proceedings of the 2020 IEEE International Conference on Communications Workshops (ICC Workshops), Dublin, Ireland, 7–11 June 2020. [CrossRef]