Neural Backward Chaining Logic Algorithm Based on Dynamic Knowledge Region Segmentation in Knowledge Graph Completion

Chunmao Liu[1,2], Somkiat Tuntiwongwanich[1*], Thiyaporn Kantathanawat[1]
[1]School of Industrial Education and Technology, King Mongkut's Institute of Technology Ladkrabang, Bangkok, 10520, Thailand
[2]School of Electronic and Information Engineering, Henan Polytechnic Institute, Nanyang, 473000, China
E-mail: somkit_l@126.com
*Corresponding author

Keywords: internet, knowledge graph, recurrent neural networks, neural backward chain reasoning, unified medical language system

Received: October 23, 2024

With the rapid development of the Internet, the information age has arrived in a comprehensive way. The explosive growth of information data has greatly increased people's demand for knowledge management. Knowledge graph, as an effective structured knowledge representation tool, greatly enhances the organization and retrieval capabilities of information. However, in practical applications, knowledge graphs often face problems of knowledge loss and incompleteness, which severely limit their widespread application. To address this issue, this study proposes a neural backward chain inference method based on dynamic knowledge region generation. This method overcomes the bottleneck of performance degradation of traditional static methods on large datasets by introducing a dynamic knowledge region generation mechanism, which significantly improves the completion effect of knowledge graphs. The experiment was conducted on the Unified Medical Language System dataset and the Nations dataset. The results showed that when the size of the Unified Medical Language System dataset reached 2500, the accuracy of the proposed method reached 0.85. It was 6.25%, 20%, and 51.8% higher than the 0.80 of the neural backward chain inference method generated by static knowledge regions, 0.70 of the conditional theorem prover algorithms, and 0.56 of the traditional neural theorems proving algorithm, respectively. In the Nations dataset, the accuracy of the proposed method was 0.80 at the same data scale, which was significantly better than other methods. In addition, the method based on dynamic knowledge region generation reduced the iteration time to 1.4 seconds, which was about 52% and 63% higher than the static method's 2.9 s and the traditional method's 3.8 s, respectively. The research results indicate that the proposed neural backward chain logic algorithm based on dynamic knowledge region generation exhibits good performance in completing knowledge graphs.

Povzetek: Opisana metoda za dopolnjevanje znanstvenih grafov uporablja nevralno obratno verigo sklepanja s dinamičnim generiranjem znanstvenih področij, kar povečuje kvaliteto v primerjavi s tradicionalnimi statičnimi metodami.

1 Introduction

With the development of the Internet, the information age has arrived, and the way people live and work has changed radically. The development of mobile Internet has made the web no longer serve only specific professionals, but become a tool used by all people [1]. The sheer volume of information makes it particularly difficult for people to query information sets, and the emergence of knowledge graph (KG) solves this problem. KG is a graphical database for organizing and representing structured knowledge, but there are some problems with KG, KG may have incomplete information and some information about relationships or attributes between entities may be missing [2]. This may be due to the fact that the information has not yet been added to the graph, or due to the dynamic nature of knowledge, new information is generated but not yet updated to the graph. It is the missing and incomplete nature of the knowledge in KGs that poses a significant limitation to the use of KGs. The research aims to address the problems of low efficiency and insufficient accuracy in KG completion in large-scale and dynamic environments. A neural backward chain inference (BCI) method based on dynamic knowledge region generation is proposed. This method assumes that by dynamically adjusting the knowledge region, the accuracy and computational efficiency of KG completion can be significantly improved. It is hypothesized that this method will demonstrate higher accuracy and lower computation time than traditional static methods on datasets of different sizes and types. This proves the effectiveness of the dynamic knowledge region generation mechanism in KG completion. In this research, a neural theorem provers (NTPs) model based on BCI is proposed, and a recurrent neural network (RNN)-based NTPs model is proposed for the

improvement of the model in response to the shortcomings of the model in terms of computational efficiency. Aiming at the problem that RNNNNTPs are not ideal for dynamic knowledge area (KA) complementation, a conditional theorem provers (CTPs) algorithm is proposed to improve the NTPs. There are four primary sections to this research. The first includes a synopsis of previous studies on the subject of KG research. The second part is a review of the main methods used in this research, and the third part is the model results obtained by applying the methods to the research and analyzing the results. The fourth part is a summary of all the above studies and the outlook for future research.

2 Related works

The emergence of KG has brought multiple benefits to Internet information. Shen et al. discovered that substantial semantic information may be gleaned from the relationships and attributes of KGs, aiding in the creation of possible semantic representations of KGs. Knowledge representation approaches could reduce variation between distinct knowledge domains by aligning things through their conversion into spatial vectors. Nevertheless, previous methods neglected the distinctions between relations by using the same optimization objective for triples under various relations. The study team employed an adaptive margin strategy in training and provided an entity alignment method based on the TransE model to address this issue. The results of the study demonstrated that the proposed method had significant improvement compared to the baseline model [3]. Wu et al. found that due to the large amount of item information in KG that could help recommender systems to develop users, it becomes the most important source of auxiliary information. However, there were some problems in this application scenario. To address this problem, the research team proposed a multi-context-aware recommendation algorithm based on KG. According to experimental data, the suggested approach could handle KG faults more effectively [4]. Hussey et al. proposed to advance nursing theory through nursing KG using health informatics standards and translating nursing knowledge for nursing research. Experimental results showed that the proposed KG was able to gain valuable insights into nursing interventions and inform future big data science [5].

Lin et al. discovered that knowledge representation is a crucial stage in building domain KG, and that domain KG has emerged as a study issue in the artificial intelligence era. However, there were some gaps in the construction of domain KGs. Therefore, the research team studied the classification of KGs, and the study analyzed the unresolved issues and future research trends of knowledge representation in domain KG research [6]. Haoran et al. found that KG complementation could solve the problem of sparse data in KGs. However, under-utilization of structural information around nodes was the main problem of the traditional KG complementation model, which led to relatively homogeneous encoded information. To address this problem, the research team proposed a new KG complementation model for encoding and decoding feature information. The study's findings showed that the suggested model functioned well on the used dataset [7]. According to Yutian et al., KG was a helpful tool and resource for describing items and connections in tasks involving natural language processing. The function of logic rules and the impact of false negative samples on knowledge embedding, however, were not fully taken into account by traditional methods of KG embedding, and the current KGs were insufficient. The model could well enhance the complementing effect of KG, according to experimental results [8].

In summary, many scholars have already had research on KG and achieved certain results. In this study, a BCI-based NTPs model is proposed, and for the shortcomings of the computational efficiency of this model, the model is improved by proposing an RNN-based NTPs model, and for the problem that RNNNNTPs are not ideal for dynamic KG complementation, the NTPs are improved and a CTPs algorithm is proposed. The comparison table of literature review is shown in Table 1.

Table 1: Related works

Research	Method	Research content	Dataset used	Key performance metrics	Reference
Shen et al. (2022)	Entity alignment method based on transe model with adaptive margin strategy	Proposed an entity alignment method addressing the issue of using the same optimization objective for triples under different relations in knowledge graphs, enhancing alignment through an adaptive margin strategy	Knowledge graph embedding datasets	Significantly improved entity alignment compared to baseline models	[3]
Wu et al. (2022)	Multi-context-aware recommendation algorithm based on	Developed a multi-context-aware recommendation	Recommendation system related datasets	Effectively addressed defects in knowledge	[4]

	knowledge graph	algorithm leveraging project information in knowledge graphs to enhance recommendation systems, addressing deficiencies in knowledge graphs		graphs, improving recommendation accuracy	
Hussey et al. (2021)	Method using health informatics standards and translated nursing knowledge	Constructed a nursing knowledge graph by combining health informatics standards and nursing knowledge to advance nursing theory	Nursing big data	Provided valuable insights for nursing interventions, supporting nursing theory and big data science research	[5]
Lin et al. (2021)	Classification study of domain knowledge graphs, analyzing knowledge representation issues and trends	Researched the classification of domain knowledge graphs, analyzed unresolved issues in knowledge representation within domain knowledge graph studies, and proposed future research directions	AI domain related datasets	Analyzed challenges in knowledge representation in domain knowledge graph studies and proposed future research directions	[6]
Haoran et al. (2022)	Knowledge graph completion model based on graph convolutional networks (GCN) with multi-information fusion and high-dimensional structure analysis weighting	Proposed a new feature encoding and decoding knowledge graph completion model using GCN for multi-information fusion and high-dimensional structure analysis	Knowledge graph completion related datasets	The proposed model performs well on the used datasets, improving the accuracy of knowledge graph completion	[7]
Yutian et al. (2021)	Knowledge graph embedding model based on soft rules and adversarial learning	Combined soft rules and adversarial learning to propose a new knowledge graph embedding method, enhancing knowledge graph completion	Knowledge graph embedding datasets (specific datasets not detailed)	Effectively improved knowledge graph completion, enhancing the robustness of embeddings	[8]

3 Knowledge graph complementation model based on neural backward chain logic algorithm

The first section of this chapter introduces KGs and builds a BCI-based model for NTPs. The second chapter addresses the problems of the complementary model and proposes a neural BCI method based on dynamic KG generation to complement KGs.

3.1. Modeling of NTPs based on backward chain reasoning

KG usually adopts the ternary form to represent structured semantic knowledge. The ternary consists of subject, predicate and object, which are used to describe the relationship between the entities, where the entity is the most basic element, which is generally a real-world physical object or an abstract concept, and the relationship is the link connecting the entities, and in the construction of KG, it will generally contain three phases: information extraction, knowledge fusion, and knowledge processing [9]. Inductive logic programming (ILP) is a machine learning method that combines inductive reasoning and logic programming. It aims to automatically learn logic programs from given facts and rules for reasoning and decision making. The basic idea of ILP is to reason and make decisions by generalizing general logical rules from examples. It does this by representing examples as logical facts and goals, and then finding logical rules that satisfy the goals by using

logical reasoning and search algorithms, the flow of which is shown in Fig. 1.

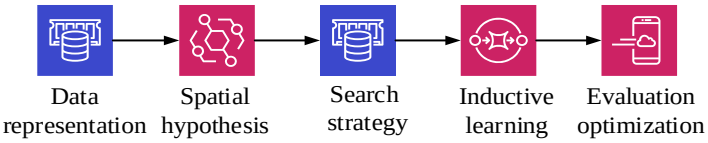


Figure 1: ILF flow

In Fig. 1, the ILF algorithmic process is mainly divided into five types, which are data representation, hypothesis space, search strategy, inductive learning, and evaluation and optimization. Among them, data representation refers to the form of transforming example data into logical facts, i.e., mapping attributes and relations in data into logical predicates. Hypothesis space refers to the determination of the hypothesis space of logical rules, which is usually realized by defining the formal syntax and constraints of logical rules. Search strategy refers to the selection of an appropriate search strategy to search for logical rules in the hypothesis space. Inductive learning refers to the continuous adaptation of logical rules during the search process, based on a given objective function and learning algorithm, and the evaluation of their accuracy and generalization ability on example data. Evaluation optimization refers to evaluating the performance of the logic rules obtained by learning and optimizing and correcting the rules according to the evaluation metrics of the learning algorithm [10-11].

RNN is a neural network model for processing sequential data or data with time dependency [12]. Unlike traditional feed-forward neural networks, RNN has recurrent connections that can transfer information within the network, giving it memory capability, the RNN recurrent hidden state update formula is shown in equation (1).

$$h_t = \begin{cases} 0, & t = 0 \\ \varphi(h_{t-1}, x_t), & otherwise \end{cases} \tag{1}$$

In equation (1), h_t denotes the cyclic hidden state of the RNN, φ denotes the nonlinear function, and x denotes the sequence. Equation (2) displays the update of the cyclic hidden state.

$$h_t = g(Uh_{t-1} + Wx_t) \tag{2}$$

In equation (2), g represents a smooth bounded function and W and U denote constant parameters. Conventional RNNs are prone to the problem of gradient vanishing or gradient explosion during the training process, which leads to difficulties in capturing long-term temporal dependencies. To solve this problem, some improved RNN structures, such as gated recurrent unit (GRU) and Long Short-Term Memory (LSTM), have emerged to deal with long-term dependencies efficiently by introducing a gating mechanism, and the structure of LSTM is shown in Fig. 2.

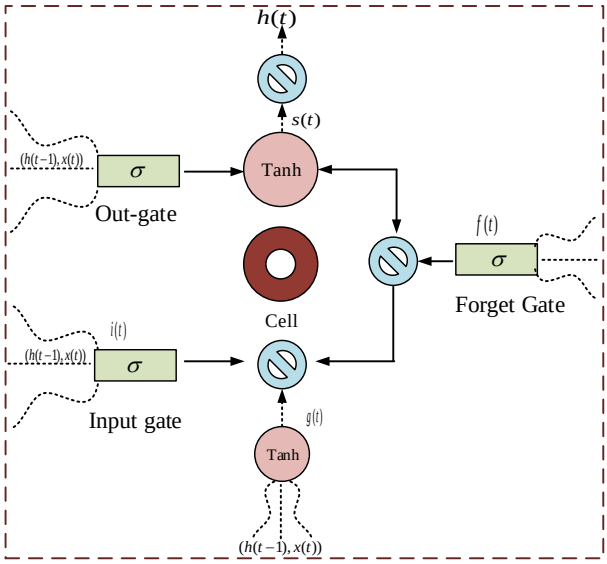


Figure 2: Structure of LSTM

In Fig. 2, the input layer input at moment t is $x(t)$, $h(t-1)$ is the hidden layer at moment $t-1$, and the internal state s is the secret key for neuron activation and also determines the network structure of the LSTM. Equation (3) shows the output of the LSTM unit, which is a value $f(t)$ between 0 and 1. This value is obtained through $h(t-1)$ and $x(t)$, representing a trade-off between retaining or discarding the internal states. The activation function used is a sigmoid function.

$$h_t^j = o_t^j \tanh(c_t^j) \tag{3}$$

In equation (3), o_t^j denotes the output gate that controls exposure, c_t^j denotes the memory cell. Equation (4) displays the expression for the output gate.

$$o_t^j = \sigma(W_o x_t + U_o h_{t-1} + V_o c_t)^t \tag{4}$$

In equation (4), σ represents the sigmoid function and V_o represents the diagonal matrix. The degree of forgetting of existing memories is regulated by the forgetting gate, and the new memory content is controlled by the updating gate, and the two gates are calculated as shown in equation (5).

$$\begin{cases} f_t^{\,j} = \sigma(U_f h_{t-1} + W_f x_t + V_f c_{t-1})^j \\ i_t^{\,j} = \sigma(U_i h_{t-1} + W_i x_t + V_i c_{t-1})^j \end{cases} \quad (5)$$

In equation (5), V_i and V_j denote diagonal matrices. Based on the current input and the concealed state of the previous instant, the input gate uses a sigmoid function to determine which information needs to be added to memory. Using a sigmoid function, the forgetting gate determines whether data should be erased from memory based on the input at hand and the hidden state of the preceding instant. Using a sigmoid function, the output gate determines which data should be output depending on the input at hand and the hidden state from the previous instant.

In the complementation of KG, when there are missing relations or entities in the triad, all relations or entities in the KG are generally complemented to the triad, and the triad for which the filling is performed is used as a query option, sorted and scored, and the relations or entities with the highest scores are used as the final result output [13]. BCI based NTPs apparatus are characterized by micro-computability and strong interpretability of rules and results. The core of RNN is the GRU, which is an improved RNN structure with good performance and computational efficiency. The GRU model incorporates update and reset gates to regulate the flow of information at every moment, thereby enabling the model to selectively retain or forget information. This approach effectively solves the gradient vanishing problem that traditional RNNs frequently encounter when processing long sequences. In practical implementation, RNNs utilize the features of each entity as input, combining them with the relevant relationships in the graph to achieve inference. During each inference, the RNN computes the current hidden state based on the current input and the previous hidden state. This recursive structure allows the model to maintain a "contextual" memory of the entire KG at each time step, taking into account the dependencies between entities and relationships in the graph during knowledge completion. In NTPs, there are three main modules, which are unfication (unify), or and. the computational process of NTPs can be approximated as a search. That is, or indicates that any rule in the knowledge base serves the goal of the proof, and indicates that the premises of the rules must be recursively proved. RNN is used as the regulation and facts for the knowledge generation operation because the proof process of NTPs is top-down, the unify operation needed by each layer will be increased step by step, and there is a sequential relationship between the upper layers and lower layer's rule facts. Its structure is shown in Fig. 3.

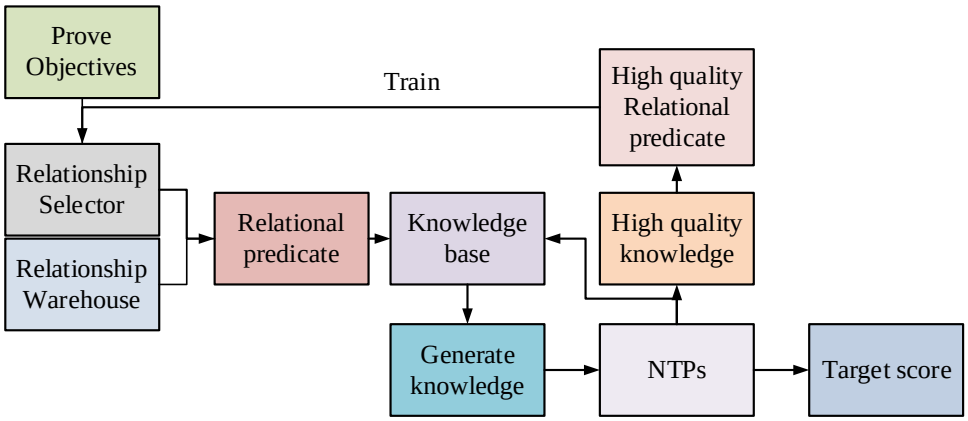


Figure 3: RNNNTPs model structure

In Fig. 3, the proof target is firstly input into the relation selector, and the relations in the relation warehouse are selected through the relation selector. The knowledge in the knowledge base is selected according to the selected relation predicates, the generated knowledge is fed into NTPs to generate high quality knowledge, and the generated new rules are added to the knowledge base, the high-quality relation predicates are refined by the high-quality knowledge, and in turn the relation selector is trained, and finally the NTPs outputs the target values by the generated knowledge. In this study, RNN selector is used to predict the sequence and the generated sequence is shown in equation (6) [14].

$$h_0 = f(r) \quad (6)$$

In equation (6), where f denotes the linear transformation, the relationship between different positions is predicted by the gating unit and the expression is shown in equation (7).

$$h_t = GRU(h_{t-1}, g([r_{t-1}, r_t])) \quad (7)$$

In equation (7), g denotes the linear transformation and $[r, r_t]$ denotes the connection between the previous relation and the current relation. The generated relation set is divided by softmax function to get the next relation. In RNNNTPs, the relation selector generates only the relation set generates the knowledge set by

matching algorithm and the expression is shown in equation (8).

$$\begin{cases} KB_{gen} = Select(KBs, \mathrm{Re}l_{gen}) \\ Select(KB, \mathrm{Re}l) = KB' \end{cases} \tag{8}$$

In equation (8), KB denotes the knowledge base and $\mathrm{Re}l$ denotes the set of relations. Since NTPs have the problem of slow computational efficiency, the modules in NTPs are improved to address this problem. The Or module is improved by reconstructing it so that it only considers the knowledge provided by the knowledge generator. By establishing a threshold to govern the Unify module, it may be made better. If a similarity surpasses the threshold, it will be added to the high-quality knowledge set [15]. Fig. 4 depicts the revised algorithm's flow.

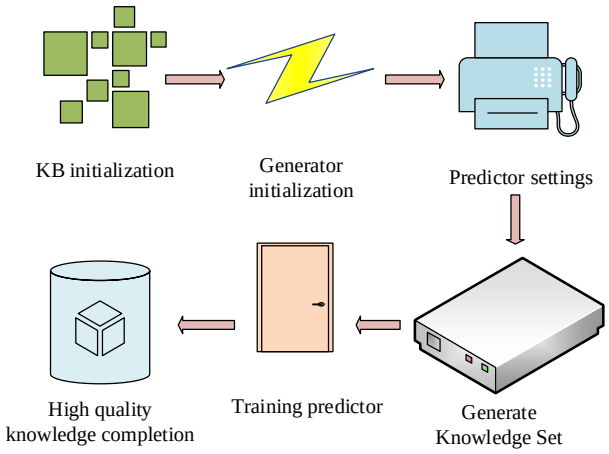


Figure 4: Flowchart of the improved algorithm

In Fig. 4, firstly the ternary dataset is input and then it is preprocessed, secondly the GRU-based RNN is built, then the predictor setting is performed and ComplEx is used as the optimization function for the NTPs. The knowledge set is generated by the predictor setting, the individual relations in the target are proved by the RNN selector to generate the set of relations. To create the knowledge set, the relations in the set are matched in the KBs. The predictor is then trained to add the closest knowledge into the KBs for knowledge complementation.

3.2. Knowledge graph complementation model based on dynamic knowledge search region segmentation

RNNNTPs substantially improve the computational efficiency of NTPs, but there are some problems, in the relation selection, some relations will appear more frequently, and some simple relations will be matched with more knowledge. This presents a significant challenge for the next connection selection process. To address this, the CTP algorithm is used to optimize the model by incorporating the neural layer of knowledge selection into the predictor. This further increases the accuracy of the link selection process. CTPs are computational models that prove given conditional propositions through the use of conditionals and inference rules, and can be used in areas such as logical reasoning, theorem proving, and formal verification [16]. The main use of CTPs is to automate the theorem proving process. CTPs algorithm can receive conditions and propositions to be proved and try to prove these propositions using logical inference rules and reasoning strategies, the model of RNN-CTPs algorithm is shown in Fig. 5.

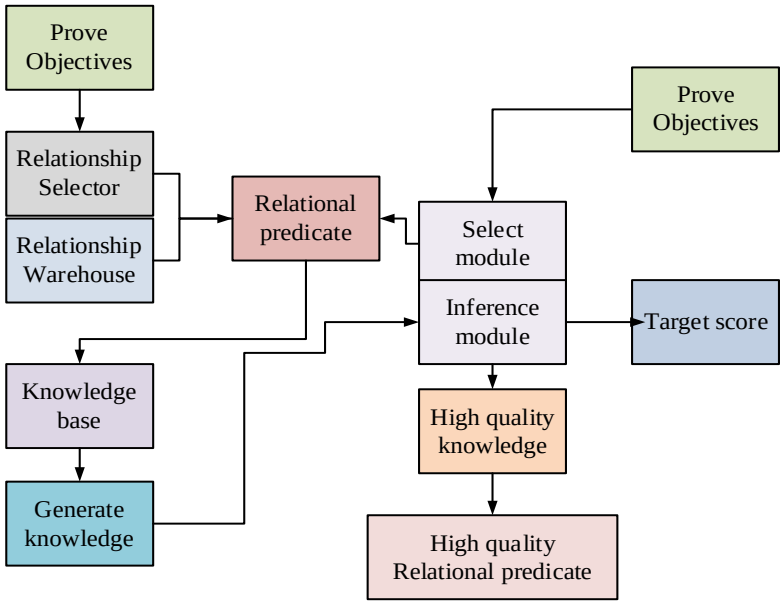


Figure 5: Model flow of RNNCTPs algorithm

In the relation selector, the sequence relations are predicted by RNN relation selector, after initializing the RNN selector, the relations at different positions are predicted by gating unit, the generated rule set is divided by softmax function to get the next rule body, and the number of generated relations is shown in equation (9).

$$relNum = batchSize \times size(\mathrm{Re}\, l_s) \times rel_\lambda \times KB_\lambda \quad (9)$$

In equation (9), $relNum$ denotes the generated relations and $batchSize$ is the proof objectives. $relNum$ denotes the hyperparameter whose purpose is to adjust the relations generated, and $size(\mathrm{Re}\, l_s)$ denotes the relations present in the relation repository. KB_λ denotes the degree to which the generated knowledge matches the relations. In the relation warehouse, dynamic expansion coefficients are used and the dynamic change of expansion coefficients is shown in equation (10).

$$ep_i^{new} = ep_i^{last} - \kappa(KB_\lambda^{new} - KB_\lambda^{last}) \quad (10)$$

In equation (10), κ represents the hyperparameter that controls the diffusion coefficient, and KB_λ^{new} and KB_λ^{last} are the current KB_λ and the KB_λ of the previous cycle, respectively. CTPs, as an improved version of NTPs, reduces the amount of knowledge that needs to be learned by increasing the neural layer used for knowledge selection, which improves the efficiency of computation. To achieve better computation, CTPs will choose alternative functions dependent on the object. The selection module is based on the attention function, as illustrated in equation (11).

$$f_i(x) = \alpha E_R \quad (11)$$

In equation (11), E_R denotes the embedding matrix of predicates, and α denotes the attention distribution of predicate verbs in the matrix, the expression of which is shown in equation (12).

$$\alpha = soft\max(W_i x) \in \Delta^{|R|-1} \quad (12)$$

In equation (12), α denotes the attention distribution of the predicate verbs in the matrix and W_i denotes the elements embedded in the matrix. CTPs go through the selection module to generate a set of knowledge by proving the goal. After each generation of knowledge set by the model, variables need to be recorded and the expression is shown in equation (13).

$$KB_{\lambda_i} = \frac{\left|KB_{Used_i}\right|}{\left|KB_{gen_i}\right|} \quad (13)$$

In equation (13), KB_{λ_i} denotes the set of variables, KB_{Used_i} denotes the knowledge that is involved in the process at this time. Moreover, KB_{gen_i} represents the knowledge produced by the CTPs following the feeding of inputs. The flow of the algorithmic model of the RNNCTPs is shown in Fig. 6.

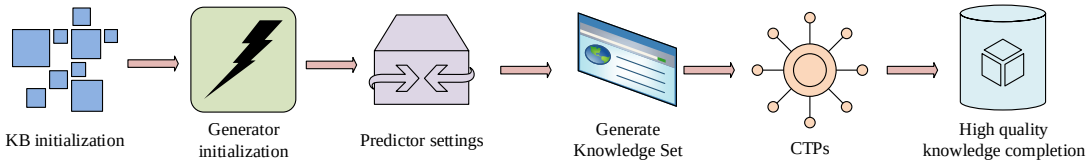


Figure 6: Model flow of RNNCTPs algorithm

Firstly, the ternary dataset is input and then it is preprocessed, secondly the GRU based RNN is built and then the predictor setting is performed. The predictor setting creates the knowledge set. The RNN selector proves each individual relation in the target to create the set of relations. Moreover, the KBs match the relations in each set to create the knowledge set. The predictor is then trained to add the closest knowledge to the KBs for knowledge complementation. The training of the CTPs module already includes the selection of the module, so the amount of knowledge used for the unify module to run is under control. Each iteration produces limb utilization as shown in equation (14).

$$KB_\lambda = \begin{cases} KB_\lambda + \tau(KB_{\lambda mean} - KB_\lambda), if \left|KB_{\lambda mean} - KB_\lambda\right| > \varsigma \\ \dfrac{KB_\lambda + KB_{\lambda mean}}{2}, \qquad else \end{cases} \quad (14)$$

In equation (14), τ A denotes the learning rate of KB_λ update. ς denotes the learning threshold, which can be fine-tuned if the difference between $KB_{\lambda mean}$ and KB_λ is too large.

```
RNN-CTPs

# Input: Dataset X (features) and Y (labels), RNN parameters, CTPs parameters
# Output: Predicted labels Y_pred

# Step 1: Initialization
1. Initialize RNN structure (e.g., number of layers, number of hidden units).
2. Define CTPs parameters (e.g., tree depth, splitting criterion such as Gini index or information gain).

# Step 2: Training Phase
2.1 Feature Extraction with RNN:
   for each sample (x, y) in (X, Y):
      # Pass input sequence x through RNN to extract features
      h_t = RNN(x_t; W_rnn)  # W_rnn is the weight matrix of the RNN
      h_T = FinalState(h_t)  # h_T is the final hidden state representation of the sequence

   # Collect the final hidden states h_T as the feature set for all samples

2.2 Build Conditional Probability Trees (CTPs):
   1. Initialize root node with all training samples and their extracted features h_T.
   2. While the stopping criterion is not met (e.g., max depth or minimum samples in a node):
      a. For each node, calculate the optimal splitting condition based on features h_T
         and maximize conditional probabilities of labels.
      b. Split the data into child nodes based on the chosen condition.
   3. At each leaf node, compute the probability distribution of labels and store it.

# Step 3: Prediction Phase
3.1 Predict for each test sample x_test:
   a. Pass x_test through the RNN to obtain feature representation h_test.
   b. Traverse the CTPs using h_test, following the splitting conditions until reaching a leaf node.
   c. At the leaf node, select the class with the highest probability as the predicted label y_pred.

# Step 4: Output
4. Return all predicted labels Y_pred.
```

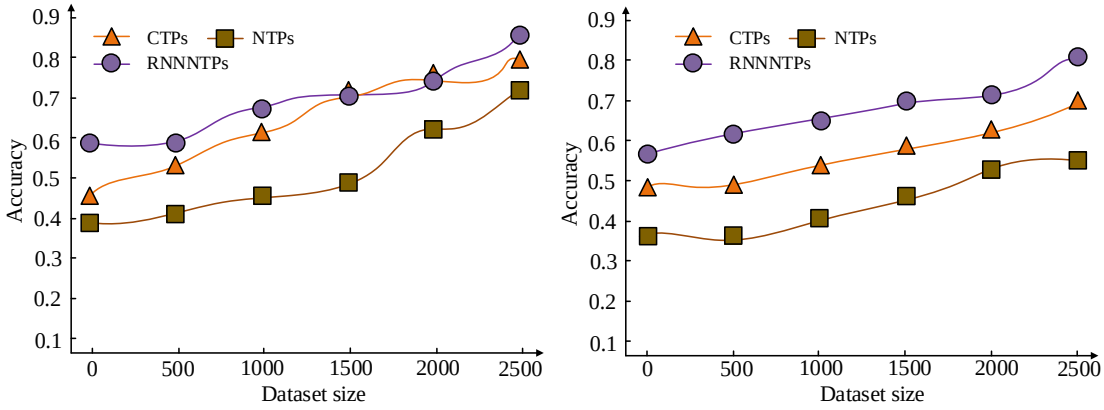
Figure7: RNN CTPs pseudocode

4 Performance analysis of knowledge graph complementation model based on neural backward chain logic algorithm

In the first section of this section, the performance comparison is carried out for static KG segmentation of KG using UMLS dataset and Nation's dataset, and the results of the study show that it performs well in the UMLS dataset. Therefore, the performance of KG complementary model based on dynamic KG segmentation is analyzed using UMLS dataset.

4.1. Performance analysis of knowledge Graph complementary model based on static knowledge area segmentation

The server CPU used in this study is Inter(R) Core (TM) i5-10210U with 16GB of RAM, GPU is GeForce RTX 2080 operating system is Windows 10 with 8G of RAM. Using a systematic hyperparameter selection method, different combinations of hyperparameters such as learning rate, batch size, GRU layers, number of hidden units, and similarity threshold are first traversed by grid search, and the performance of each parameter set is evaluated on the validation set to finally select the optimal parameter combination. To ensure the robustness of the selection, k-fold cross-validation is used to divide the training data into k subsets, and multiple training and validation are performed to calculate the average performance of each parameter group. At the same time, an early-stop mechanism is introduced to stop training if the performance of the validation set did not significantly improve in 10 consecutive training rounds to prevent overfitting. The specific hyperparameter settings are as follows: learning rate set to 0.001, batch size set to 64, GRU layers set to 2, number of hidden units set to 128, similarity threshold set to 0.7, Dropout rate set to 0.5, optimization algorithm using Adam, iteration count set to 300, embedding dimension set to 100, activation function set to ReLU, and batch normalization applied in the model. To ensure the robustness of the results, each experiment is run independently at least five times with the same hyperparameter settings, using different random seeds to avoid random effects, and taking the average of key performance indicators such as accuracy, iteration time, and knowledge utilization, while calculating the standard deviation to evaluate the stability of the results. When comparing performance, paired t-test and analysis of variance methods are used with a significance level of α=0.05 to ensure that performance differences between different models are statistically significant. At the same time, a 95% confidence interval is calculated to provide an uncertainty assessment for performance estimation and increase the reliability of result interpretation. The study is conducted using the Unified Medical Language System (UMLS) dataset and the Nations dataset. UMLS is a medical knowledge representation and retrieval system developed by the National Institutes of Health. Its purpose is to facilitate the interoperability and exchange of medical information so that different medical information resources can be connected and shared. UMLS contains many medical vocabularies, coding systems, and standardized terminology, including 49 predicates, 135 constants, and 6529 true facts. It also unifies terminology, coding, and standards from various medical disciplines. Nations has 2565 true facts, 111 unary predicates, 14 constants, and 56 binary predicates. The performance of the proposed model is analyzed on both datasets and the results are shown in Fig. 8.



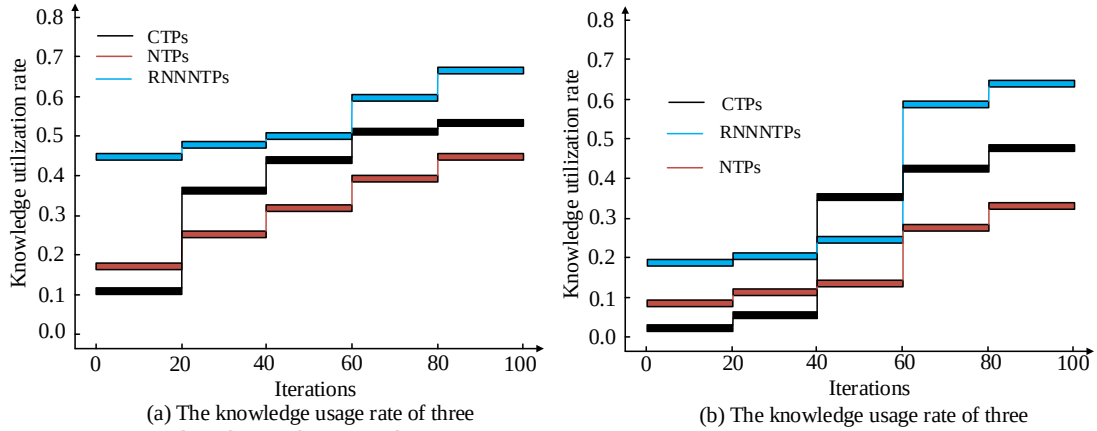
(a) Performance of Three Algorithms on UMLS Datasets

(b) Performance of Three Algorithms on the Nations Dataset

Figure 8: Comparison of accuracy of three algorithm models in different datasets

Fig. 8(a) shows the performance of three algorithms on the UMLS dataset, while Fig. 8(b) shows the performance of the three algorithms on the Nations dataset. According to Fig. 8(a), in the UMLS dataset, as the dataset increases, the accuracy of RNNNTPs, CTPs, and NTPs algorithm models also fluctuates and increases. Among them, the RNNNTPs algorithm model shows higher performance. When the dataset size is 2500, the accuracy of RNNNTPs, CTPs, and NTPs algorithm models are 0.88, 0.79, and 0.71, respectively. As shown in Fig. 8(b), in the Nations dataset, the accuracy of the three algorithms also increases with the size of the dataset, but the training performance is relatively poor compared to the UMLS dataset. When the dataset size is 2500, the accuracy of RNNNTPs, CTPs, and NTPs algorithm models are 0.80, 0.70, and 0.56, respectively. The experimental results show that the proposed hybrid algorithm model exhibits high accuracy among the three algorithms and performs well in training on the UMLS dataset. Compare the performance of three algorithms at different iteration times, and the results are shown in Fig. 9.

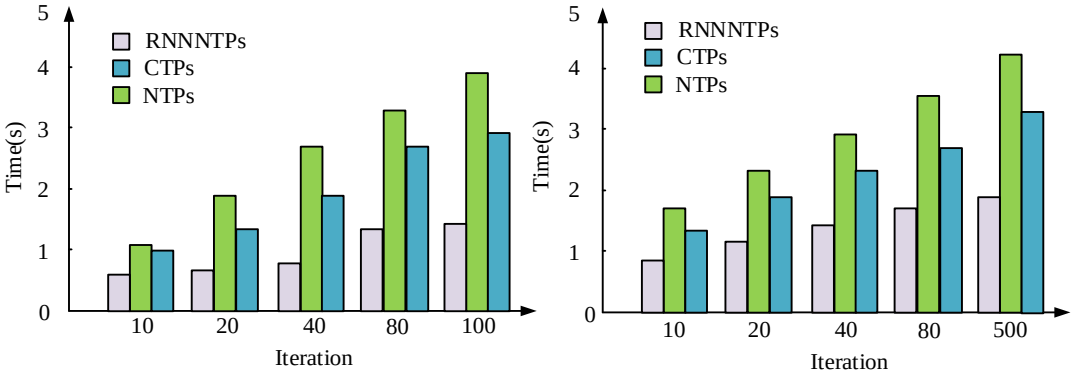
Fig. 9(a) shows the knowledge utilization rates of three algorithms in the UMLS dataset, while Fig. 9(b) shows the knowledge utilization rates of the three algorithms in the Nations dataset. As shown in Fig. 9(a), with the increase of iteration times, the knowledge utilization rate of the three algorithms also continues to rise. When the number of iterations reaches 100, the knowledge utilization rates of RNNNTPs, CTPs, and NTPs algorithm models in the UMLS dataset are 0.68, 0.53, and 0.46, respectively. In Fig. 9(b), with the increase of iteration times, the knowledge utilization rate of the three algorithm models also continues to rise, but it is significantly lower than that of the UMLS dataset. When the number of iterations reaches 100, the knowledge utilization rates of the three algorithm models are 0.65, 0.48, and 0.32, respectively. The experimental results show that the proposed RNNNTPs algorithm model has better performance compared to other algorithm models, and each algorithm model performs well in the UMLS dataset. The iteration times of the three algorithms are compared for different numbers of iterations, and the results are presented in Fig. 10.



(a) The knowledge usage rate of three algorithms in the UMLS dataset

(b) The knowledge usage rate of three algorithms in the Nations Dataset

Figure 9: Knowledge utilization rate of three algorithms at different iteration times



(a) Iteration time of three algorithms in the UMLS dataset

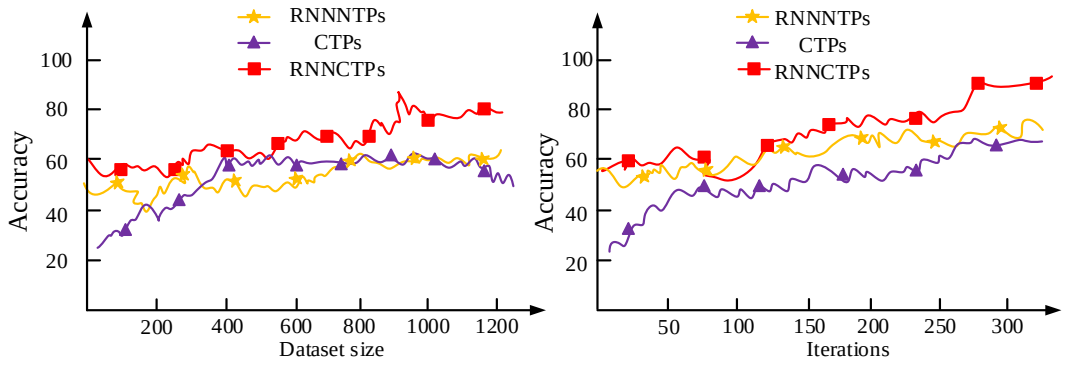
(b) Iteration time of three algorithms in the Nations dataset

Figure 10: Iteration time in different datasets

Fig. 10(a) represents the iteration time of the three algorithms in the UMLS dataset and Fig. 10(b) represents the iteration time of the three algorithms in the Nations dataset. In Fig. 10, the RNNNTPs algorithm model exhibits less time used for iterations than the other two methods in all three methods. The iteration times of the RNNNTPs, CTPs and NTPs algorithm models in the UMLS dataset are 1.4s, 2.9s and 3.8s, respectively, when the number of iterations is 100. In the Nations dataset, the iteration times of the three algorithms are 1.9s, 3.2s and 4.3s, respectively. The findings revealed that the proposed RNNNNTPs algorithm has a lower iteration time under different iteration number of iterations have low iteration time.

4.2. Performance analysis of knowledge graph complementation model based on dynamic knowledge area segmentation

The findings are displayed in Fig. 11 after the superior UMLS dataset is chosen and the model performance is examined to compare each model's performance.

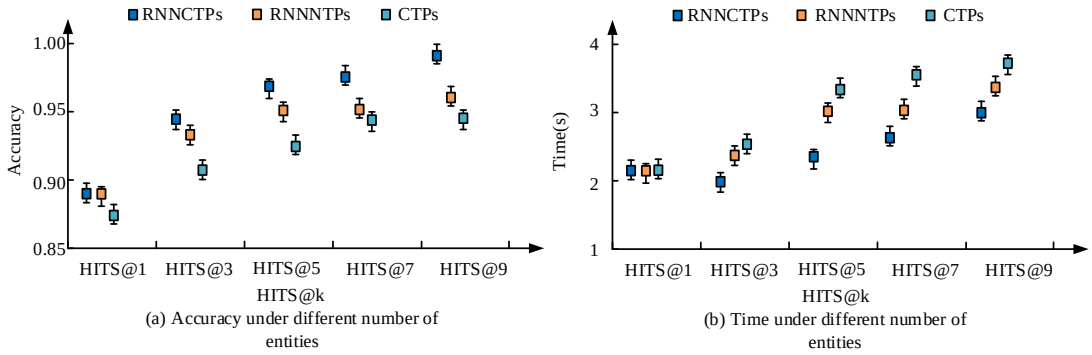


(a) Accuracy under different dataset sizes

(b) Accuracy under different iterations

Figure 11: Accuracy of three algorithmic models with different dataset sizes and number of iterations

Fig. 11(a) represents the accuracy comparison of the three algorithms under different dataset sizes, and Fig. 11(b) represents the accuracy comparison of the three algorithms under different iteration numbers. In Fig. 11(a), the accuracy of the three models fluctuates and rises as the dataset increases. The accuracy of the RNNCTPs, RNNNNTPs, and CTPs algorithmic models are 0.82, 0.64, and 0.57, respectively, when the dataset size is 1200. In Fig. 11(b), the accuracy of the three models fluctuates and rises as the iterations increases. The accuracy of the three algorithmic models fluctuates and rises when the iterations is 300, the accuracy rates of the three algorithmic models are 0.94, 0.76, and 0.78, respectively. The study's findings show that, out of the three models, the suggested RNNCTPs model—which is based on dynamic KG segmentation—performs better. The HITS@k evaluation metric is used to compare the three methods. The findings are displayed in Fig. 12 and this metric indicates the percentage of real entities that are present in the top k entities of the sorted ranked list. Fig. 12(a) shows the accuracy of the three algorithms for different entities, while Fig. 12(b) shows the iteration time for the same. Among the three algorithmic models, the proposed RNNCTPs model based on dynamic KG segmentation has a better performance. The comprehensive performance of the model is analyzed, and the results are shown in Table 2.



(a) Accuracy under different number of entities

(b) Time under different number of entities

Figure 12: Table comparing the performance of the three algorithms

Table 2: Model comprehensive performance analysis

Dataset size	Model	ACC	Calculation overhead/s	Mean response time/s	Training time/s	Iteration time/s	Model stability
1000	RNNCTPs	0.75	12.3	1.2	85	1.5	0.03
	RNNNTPs	0.7	15.4	1.8	92	2.1	0.05
	CTPs	0.6	18.2	2.0	99	2.5	0.06
	NTPs	0.45	21.6	2.3	110	3.2	0.08
1500	RNNCTPs	0.78	15.6	1.4	130	1.7	0.02
	RNNNTPs	0.74	18.0	1.9	138	2.3	0.04
	CTPs	0.68	20.1	2.2	145	2.8	0.05
	NTPs	0.5	23.0	2.6	155	3.6	0.07
2000	RNNCTPs	0.82	18.5	1.5	170	1.9	0.01
	RNNNTPs	0.78	21.2	2.0	180	2.5	0.03
	CTPs	0.72	23.4	2.4	190	3.0	0.04
	NTPs	0.55	26.2	2.8	200	3.9	0.06
2500	RNNCTPs	0.85	20.7	1.6	210	1.4	0.02
	RNNNTPs	0.80	23.0	2.2	225	2.7	0.04
	CTPs	0.75	25.3	2.6	240	3.2	0.05
	NTPs	0.56	28.6	3.0	260	4.3	0.07

As demonstrated in Table 2, the RNNCTPs model displays superior accuracy and reduced computational overhead across various dataset sizes, particularly at a dataset size of 2500, where the accuracy reaches 0.85. Concurrently, the response time and computational overhead of the RNNCTP model are notably lower compared to other models. Furthermore, the iteration and training times of RNNCTPs are relatively brief, suggesting their efficacy in handling large-scale datasets. For other models, such as RNNNTPs, CTPs, and NTPs, the gap in accuracy and computational cost gradually becomes apparent as the dataset size increases, especially for the 2500 scale dataset, where the performance advantage of the RNNCTPs model is particularly prominent. The experimental outcomes reveal that the proposed RNNCTPs model has stability and robustness under different input scales, providing strong support for its wide application. Fifty people with certain medical knowledge are selected and randomly divided into five groups, and the models are scored separately. Moreover, Table 3 displays the outcomes.

Table 3: User evaluation form

Model	Group 1	Group 2	Group 3	Group 4	Group 5	Average
RNNCTPs	96.4	87.6	94.1	86.2	85.2	89.9
RNNNTPs	91.2	87.3	86.1	80.4	77.6	84.5
CTPs	87.5	72.6	78.2	77.3	70.4	77.2

In Table 3, the ratings of the five groups of users for the RNNCTPs model are 96.4, 87.6, 94.1, 86.2, 85.2, 89.9, with an average of 89.9. The ratings of the RNNNNTPs model are 91.2, 87.3, 86.1, 80.4, 77.6, and 84.5, with an average of 84.5. The ratings of the CTPs model are 87.5, 72.6, 78.2, 77.3, 70.4, 77.2, and 77.2, respectively. are

87.5, 72.6, 78.2, 77.3, 70.4, 77.2 with an average score of 77.2. According to the experimental findings, there is more satisfaction with the suggested RNNCTPs algorithmic model.

5 Discussion

To complete the KG, a neural backward chain logic algorithm based on dynamic knowledge region generation was proposed. The experimental results showed that this method was significantly better than traditional static methods and traditional neural theorem proving algorithms in terms of accuracy and computational efficiency. First, the proposed RNNCTPs model achieved accuracy of 0.85 and 0.80 on the Unified Medical Language System and Nations datasets, respectively, showing significant improvements over the neural BCI method generated from static knowledge regions and traditional algorithms. This advantage was mainly due to the introduction of a dynamic knowledge region generation mechanism. It allowed the model to adaptively focus on relevant knowledge regions based on the current inference context, thus avoiding the performance degradation problem of static methods when processing large datasets. In addition, the integration of the CTP algorithm further optimized the knowledge selection process. By introducing a CTP in the prediction layer, the model could more accurately select relevant knowledge, reduce the search space, and accelerate the inference speed, achieving significant reductions in iteration time of about 52% and 63%. The CTP improved the accuracy of complex relationship matching by optimizing the knowledge selection process, and outperformed the soft rule and graph adversarial learning models proposed by Yutian et al [8] in complex KGs. In addition, compared to Shen L et al.'s improvement of the TransE model through an adaptive margin strategy [3], this study had higher efficiency in completing tasks. Testing on the Nations dataset further validated the robustness of the model with an accuracy of 0.80. The application of RNNs, especially the use of GRUs, enabled the model to effectively remember and utilized the information obtained during previous inference processes, thus improving the adaptability and accuracy of the model in processing dynamic and large KGs. Meanwhile, the improved Unify module filtered out low-quality knowledge by setting similarity thresholds, ensuring high accuracy and efficiency in the inference process. The user evaluation results further validated the practicality and superiority of the model. The RNNCTPs model achieved an average user satisfaction score of 89.9, significantly higher than other models, indicating its potential value and good performance in practical applications. This technology had two primary contributions. First, it enhanced the efficacy of KG completion. Second, it offered a practical solution for the management and application of large-scale dynamic KGs. This development was significant from both a technical and practical standpoint, and it held considerable potential for future applications. Nevertheless, further validation was necessary to ascertain the model's generalizability across diverse dataset types. Future research can test it on more diverse datasets and explore its integration with other artificial intelligence technologies to further improve the model's performance and adaptability. In summary, the neural backward chain logic algorithm based on dynamic knowledge region generation provides an efficient, accurate, and scalable solution for KG completion tasks, thus promoting the development of intelligent knowledge management systems. Although the research has mainly been validated on the unified medical language system and Nation’s datasets, in order to more comprehensively evaluate the performance of the proposed model, future experiments should be extended to different datasets and real-world application scenarios to further validate the model's applicability and robustness. Outside the medical field, applications can be explored in the financial sector for risk management, credit scoring and other tasks, or in recommendation systems in e-commerce to help automate product recommendations and user behavior analysis. These scenarios typically have different data characteristics and more complex practical requirements, which can provide a more challenging validation of the model's generalization ability.

6 Conclusion

The internet has made an abundance of information available due to the accelerated development of information technology in recent years. This research proposed a neural BCI method based on dynamic KG generation for complementary KG. The experimental results indicated that the knowledge utilization rates of the RNNNTPs, CTPs, and NTPs algorithmic models in the UMLS dataset were 0.68, 0.53, and 0.46, respectively, when the iterations reached 100. In the Nations dataset, the knowledge utilization rates of the three algorithmic models were 0.65, 0.48, and 0.32, respectively, when the iterations reached 100. The iteration times of the RNNNTPs, CTPs, and NTPs algorithm models were 1.4s, 2.9s, and 3.8s, respectively, in the UMLS dataset when the iterations were 100. The iteration times of the three algorithms were 1.9s, 3.2s, and 4.3s, respectively, in the Nations dataset. In the case where the dataset size was 1200, the RNNCTPs, RNNNTPs, and CTPs algorithm models were 0.82, 0.64, and 0.57, respectively. At an iteration number of 300, the three algorithm models were 0.94, 0.76, and 0.78, respectively. The study’s findings demonstrate the potential of the RNNCTPs algorithmic model as a supplement for big KGs. The superior performance of RNN CTPs over RNNNTPs and CTPs is due to their model architecture and adaptability to dataset features. RNN CTPs combine the sequence modeling capability of RNN with the conditional probability splitting mechanism of CTPs. It can capture complex temporal dependencies and efficiently segment feature spaces. This has significant advantages in diverse or high-dimensional data. In contrast, RNNNTPs rely solely on deep network capture patterns and lack the interpretability of CTPs, which can fail when feature relationships are more complex. Although CTPs have

strong interpretability, they lack the ability to extract deep features, which limits their performance on data with strong nonlinear relationships. The feature distribution and complexity of the dataset exacerbate these differences, further enhancing the advantages of RNN CTPs, especially when dealing with time series or nested patterns. The RNN part could capture rich features through long-range dependencies, and the expansion of data scale could further enhance the feature extraction capabilities, while CTPs achieved efficient splitting through conditional probability and can adapt to a wider sample distribution. However, the increase in computational complexity when training larger datasets become a challenge, especially as the sequence computation of RNNs and the tree depth of CTPs coulf lead to a decrease in training time and inference efficiency. The experimental design could select large datasets such as Freebase or DBpedia to evaluate accuracy, AUC, and time cost. The expected results showed that increasing the data size improved the model performance.

In the future, with the further development of technology and the verification of more practical applications, the proposed methods are expected to play a greater role in the construction and maintenance of KGs. These methods are poised to become an indispensable key technology in intelligent reasoning, decision support systems, and big data analysis. Therefore, the research not only provides new ideas for the field of KG completion, but also opens up new possibilities for a wider range of future application scenarios.

Funding

The research is supported by science and technology research project of Henan Province: 232102210193; Research subject of Computer basic Education in China: 2023-AFCEC-227.

References

[1] Gao L, Qiu J, Chen G. Software Test Data Management Based on Knowledge Graph. Informatica, 2024, 48(16), 241-251. https://doi.org/10.31449/inf.v48i16.6416

[2] Liu J, Wang F, Song B, Wang X. Design of an Intelligent Classification Model for Interior Design Knowledge Graph Based on Simulated Annealing Algorithm. Informatica, 2024, 48(12):44-57. https://doi.org/10.31449/inf.v48i12.6029

[3] Shen L, He R, Huang S. Entity alignment with adaptive margin learning knowledge graph embedding. Data & Knowledge Engineering, 2022, 139(5):101-132. https://doi.org/10.1016/j.datak.2022.101987

[4] Wu C, Liu S, Zeng Z, Chen M, Alhudhaif A. Knowledge graph-based multi-context-aware recommendation algorithm. Information Sciences, 2022, 595(12):179-194. https://doi.org/10.1016/j.ins.2022.02.054

[5] Hussey P, Das S, Farrell S. A Knowledge Graph to Understand Nursing Big Data: Case Example for Guidance. Journal of Nursing Scholarship, 2021, 53(3):323-332. https://doi.org/10.1111/jnu.12650

[6] Lin J, Zhao Y, Huang W, Liu C, Pu H. Domain knowledge graph-based research progress of knowledge representation. Neural computing & applications, 2021, 33(2):126-129. https://doi.org/10.1007/s00521-020-05057-5

[7] Haoran N, Haitao H E, Jianzhou F, Junlan N, Yangsen Z, Jiadong R. Knowledge Graph Completion Based on GCN of Multi-Information Fusion and High-Dimensional Structure Analysis Weight. Chinese Journal of Electronics, 2022, 32(2):387-396. https://doi.org/10.1049/cje.2021.00.080

[8] Yutian W, Jiamin C, Ling S, Yutian W. Improving Knowledge Graph Completion Using Soft Rules and Adversarial Learning. Chinese Journal of Electronics, 2021,30(4):623-655. https://doi.org/10.1049/cje.2021.05.004

[9] He L, Ye W, Wang Y X, Feng H, Chen B, Liang D. Using knowledge graph and RippleNet algorithms to fulfill smart recommendation of water use policies during shale resources development. Journal of Hydrology, 2023, 617(23):128-137. https://doi.org/10.1016/j.jhydrol.2022.128970

[10] Ma J, Zhou C, Wang Y, Guo Y, Hu G. PTrustE: A high-accuracy knowledge graph noise detection method based on path trustworthiness and triple embedding. Knowledge-based systems, 2022, 256(28):1-14. https://doi.org/10.1016/j.knosys.2022.109688

[11] Wang L, Zhang Y. Digital Dissemination of Information Based on Knowledge Graph Recommendation Algorithm. Informatica, 2024, 48(19). https://doi.org/10.31449/inf.v48i19.6561

[12] Gao L, Qiu J, Chen G. Software Test Data Management Based on Knowledge Graph. Informatica, 2024,48(16). https://doi.org/10.31449/inf.v48i16.6416

[13] Wang X, Liu A, Kara S. Constructing Product Usage Context Knowledge Graph Using User-Generated Content for User-Driven Customization. Journal of mechanical design, 2023, 145(4):323-342. https://doi.org/10.1115/1.4056321

[14] Bhosle K, Musande V. Evaluation of Deep Learning CNN Model for Recognition of Devanagari Digit. Artif. Intell. Appl.2023, 1(2):114-118. https://doi.org/10.47852/bonviewaia3202441

[15] Zhao N, Long Z, Wang J, Zhao Z. AGRE: A knowledge graph recommendation algorithm based on multiple paths embeddings RNN encoder. Knowledge-based systems, 2023, 259(10):1-8. https://doi.org/10.1016/j.knosys.2022.110078

[16] Wang H, Wang Y, Li J, Luo T. Degree aware based adversarial graph convolutional networks for entity alignment in heterogeneous knowledge graph. Neurocomputing, 2022, 487(28):99-109. https://doi.org/10.1016/j.neucom.2022.02