Cross-relational attention mechanism-driven graph neural network and Apriori algorithm are used for knowledge reasoning in knowledge graphs

Kuang Wei, Yifei Wei, Huimei Wen, Yuhong Su(✉)

School of Geophysics and Information Technology, China University of Geosciences (Beijing), Beijing, China
`weik@email.cugb.edu.cn, 2741495630@qq.com`

Abstract. Knowledge reasoning for knowledge graphs refers to predicting unknown relationships within a knowledge graph to achieve automatic completion and expansion of knowledge. In the context of knowledge graph link prediction tasks, this paper proposes an improved algorithm for knowledge graph link prediction based on the graph neural network with relation-specific attention mechanism and the Apriori association rule mining algorithm. This approach addresses the challenges of information interaction between nodes with different relationships in traditional knowledge reasoning methods based on the heterogeneous graph paradigm. It aims to reduce the complexity of knowledge reasoning, improve reasoning efficiency and quality. The proposed method utilizes the relation-specific attention mechanism to facilitate information propagation across multiple relationships and enhance the performance of the graph neural network. The Apriori association rule mining algorithm is employed for data preprocessing to filter out irrelevant information in the reasoning inputs, thereby improving the quality of reasoning results. The effectiveness of the proposed approach is demonstrated through comparative experiments with graph neural network models such as GCN, GAT, and R-GCN on the FB15K-237 and WN18 datasets. Ultimately, the model achieves an MRR of 0.2753 and 0.9054 on the FB15K-237 and WN18 datasets when trained on subgraphs of size 80,000.

Keywords: Knowledge graph, Link prediction, Cross-relation attention mechanism, Apriori algorithm.

1 Introduction

Knowledge graph reasoning [1] refers to the process of using existing knowledge and rules to generate new knowledge through inference, thereby filling in the gaps and incompleteness in the knowledge graph.

Graph neural networks [2] are a series of deep learning algorithms used for learning and representing graph data. With the introduction and development of graph neural networks, more and more researchers have begun applying them to knowledge graph reasoning. In 2017, Schlichtkrull et al. [3] proposed a relational graph convolutional network (RGCN) that applies graph neural networks to knowledge graph reasoning. They use convolutional calculations to embed entities and relationships in the knowledge graph and provide an implementation approach for node classification and link prediction tasks in knowledge graph reasoning. The above methods for knowledge graph embedding often only consider the information of the triplets themselves and fail to capture the inherent complexity and implicit information in the neighborhood surrounding the triplets. To address this issue, Nathani et al. [4] proposed a novel graph attention network (GAT) method for feature embedding based on attention mechanisms to capture entity and relationship features in a given entity's neighborhood. Furthermore, their model includes relation clustering and multi-hop relationships, and it has been validated on multiple datasets, showing significant performance improvements compared to existing state-of-the-art methods.

However, there are still several limitations to knowledge reasoning based on graph neural networks. First, it lacks long-range reasoning capability. Most graph neural network-based knowledge graph reasoning methods rely on local information for reasoning, which may result in a lack of global information, especially in scenarios that require long-range reasoning. Second, it requires significant computational resources for large-scale knowledge graphs. Particularly, when there are a large number of entities and relationships, training and inference times can increase significantly, and issues such as gradient vanishing and exploding may occur during the training process. Additionally, many methods perform poorly when dealing with large-scale knowledge graphs due to data sparsity issues. Third, it is sensitive to noise and erroneous information. Knowledge graphs often contain erroneous facts and noise, which can lead to inaccurate reasoning. Graph neural network-based methods typically learn from these erroneous and noisy information, thereby affecting the accuracy of reasoning. Fourth, it lacks interpretability. Graph neural networks are often considered black-box models, making it difficult to interpret their reasoning results. This lack of interpretability is problematic in certain application domains such as law and medical decision-making that require interpretable models. Fifth, entities and relationships in knowledge graphs usually have diverse types and attributes, forming a heterogeneous graph structure. This heterogeneity needs to be considered during reasoning because different types of entities and relationships may have different rules, constraints, and semantics. To address these issues, our method provides the following three contributions.

Firstly, implementation and improvement of knowledge graph link prediction models. Knowledge graph embedding and heterogeneous graph decomposition are applied to the knowledge graph. A cross-relation attention mechanism is introduced to allow each node to better capture both local and global information in the entire graph structure. To address the complexity of knowledge reasoning caused by the large number of entity relationships and the complex internal structure of large-scale knowledge graphs, training subgraph sampling, positive and negative sampling, and block diagonal matrix decomposition are introduced. These techniques reduce the computational complexity of model training and improve the model's generalization ability.

Secondly, introducing the Apriori key rule mining algorithm for data preprocessing. In this study, the Apriori key rule mining algorithm is introduced. Entities in the knowledge graph are split into individual itemsets based on different relationship types.

Prior to knowledge reasoning, association rule mining is conducted on the entities to mine effective relationship pairs that meet the specified confidence and support thresholds. These mined relationships are then used to construct the input for knowledge reasoning by creating the triplets to be evaluated.

Thirdly, two sets of comparative experiments were conducted on the FB15K-237 and WN18 datasets. The first set of experiments involved comparing the improved model proposed in this study with traditional graph neural network models based on GCN, GAT, and R-GCN under different training subgraph sizes to verify the superiority of the proposed model. The second set of experiments involved a comparison after integrating the Apriori algorithm. Inference experiments were conducted on both datasets to examine the impact of different support thresholds on the reasoning process. Through data analysis, the effectiveness of introducing the Apriori algorithm in improving the quality and efficiency of knowledge reasoning was demonstrated.

The remaining sections of this paper are organized as follows. In Section 2, we presented relevant prior works. In Section 3, we present the design and analysis of the knowledge graph reasoning algorithm that integrates a graph neural network with a cross-relation attention mechanism and association rule mining. In Section 4, we conduct extensive experimental evaluations. In Section 5, we summarize the paper.

2 Related Work

Knowledge graph reasoning can be divided into three main directions: traditional rule-based reasoning, distributed representation-based reasoning, and neural network-based reasoning.

Traditional rule-based reasoning methods for knowledge graphs can be classified into two categories: rule-based reasoning and ontology-based reasoning. Rule-based reasoning relies on a predefined set of rules and utilizes logical inference to deduce new knowledge. Many publicly available large-scale knowledge graphs, such as NELL [5] and YAGO [6], rely on rule-based reasoning methods for knowledge base expansion. The advantage of rule-based reasoning is that it can directly leverage domain expertise, but it requires significant human effort and time to develop rules, and the scalability of rule-based reasoning is limited. On the other hand, ontology-based reasoning is a reasoning approach based on ontology. It involves parsing, reasoning, and combining ontologies to derive new knowledge. Researchers have proposed semi-automatic ontology construction approaches to address the problem of complex schema construction in RDF knowledge bases, aiming to achieve more powerful queries, consistency checks, debugging, and improved reasoning [7]. Jiang et al. [8] proposed a system based on Markov Logic Networks for cleaning raw knowledge bases. It allows scalable systems like NELL to perform probabilistic inference and addresses the problem of network-scale reasoning using ontology constraints and confidence values from the original system, as well as manually labeled data that can be used to calibrate the confidence scores of the original system or learn the effectiveness of individual extraction patterns.

Overall, traditional reasoning methods often require the instantiation of specific rules or ontology constraints. When the knowledge graph contains a large amount of information, the instantiation process becomes a massive undertaking, making it challenging to meet the efficiency requirements of many tasks. Additionally, rules and ontology constraints often require manual intervention and filtering to ensure reliability, and they may not perform optimally in terms of recall. Due to these limitations, traditional reasoning methods for knowledge graphs are often only suitable for specific scenarios.

The central idea of methods based on distributed representation learning is to find a mapping function that maps symbolic representations to a vector space for numerical representation. This helps alleviate the curse of dimensionality and captures implicit associations between entities and relationships. Importantly, these methods allow for direct and fast computation. Common distributed representation methods include the TransE [9] series of algorithms, RESCAL [10], DistMult [11], among others, which can be used for downstream tasks such as node classification. In the DistMult model, entities and relationships are embedded in a low-dimensional space, and the similarity between entities and relationships is computed using dot products. Specifically, for a given triplet (h, r, t), the scoring function is computed as shown in Formula 1:

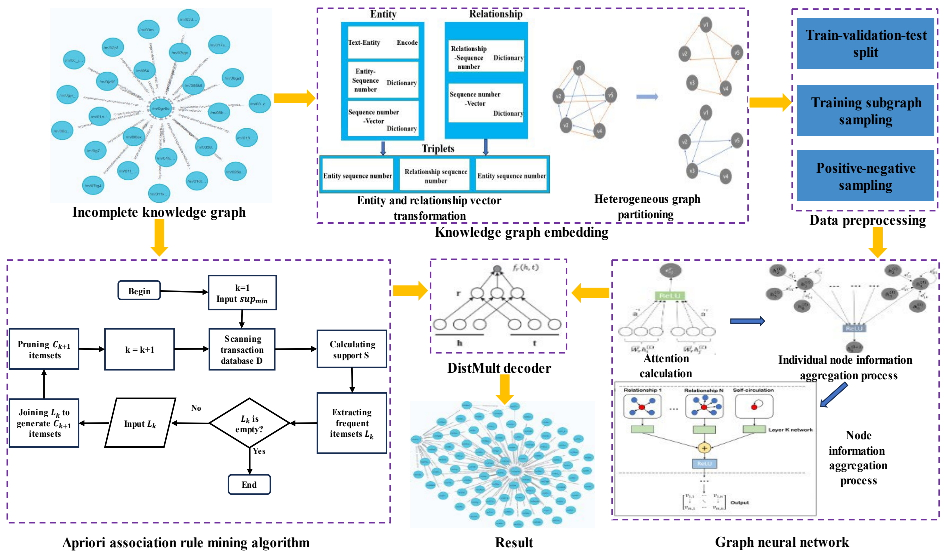
$$\text{score}\ (h, r, t) = \langle h, r, t \rangle = \sum_{i=1}^{d} h_i \cdot r_i \cdot t_i \qquad (1)$$

Where $\langle h, r, t \rangle$ represent the dot product of the vectors h, r, and t, which is used to measure the validity of the triplet. However, distributed representation-based reasoning methods capture limited deep-level information and can only obtain more features by increasing the dimensionality.

Neural network-based knowledge graph reasoning is a hot research direction in the field of knowledge graphs. It aims to utilize deep learning techniques to discover new, unknown knowledge from the knowledge graph or to infer new conclusions using known knowledge. Neural network-based knowledge graph reasoning has a wide range of applications, such as intelligent question answering, recommendation systems, and natural language processing. The core idea of neural network-based knowledge reasoning is to represent entities and relationships in the knowledge graph as vectors and use these vectors for reasoning. The advantage of this approach is that it can handle high-dimensional, nonlinear, and complex relationships. It can also capture semantic information between entities and relationships by learning representations extracted from the data. For example, Socher et al. [12] proposed a neural tensor network (NTN) for inferring relationships between two entities. They evaluated the performance of the model by predicting additional true relationships between entities in subsets of WordNet and FreeBase knowledge graphs, and the experimental results showed higher accuracy compared to previous models. Shi et al. [13] introduced a shared-variable neural network model called ProjE, which learns joint embeddings of entities and edges in the knowledge graph and fills in missing information in the knowledge graph by modifying the standard loss function. ProjE's parameter size is smaller than most existing methods, and its performance on standard datasets was better than the previous state-of-the-art methods. The experiments also demonstrated ProjE's ability to accurately determine the correctness of many declarative statements.

The Apriori algorithm is a classic algorithm for association rule mining [14]. It narrows down the search space of candidate itemsets by utilizing prior knowledge of frequent itemsets. The advantages of the Apriori algorithm are that it can handle large-scale datasets and is relatively simple and easy to understand. However, it also has some drawbacks. It cannot handle sparse data and involves a significant amount of redundant computation. Additionally, as it needs to scan the transaction database multiple times, its efficiency is relatively low. Therefore, in practical applications, the Apriori algorithm is often used in combination with other optimization methods.

3 Proposed knowledge reasoning algorithms in the knowledge graph



Incomplete knowledge graph
Entity
Relationship
Text-Entity
Encode
Entity-Sequence number
Dictionary
Sequence number -Vector
Dictionary
Relationship -Sequence number
Dictionary
Sequence number -Vector
Dictionary
Triplets
Entity sequence number
Relationship sequence number
Entity sequence number
Entity and relationship vector transformation
Heterogeneous graph partitioning
Knowledge graph embedding
Train-validation-test split
Training subgraph sampling
Positive-negative sampling
Data preprocessing
Begin
k=1
Input sup_{min}
Pruning C_{k+1} itemsets
k = k+1
Scanning transaction database D
Calculating support S
Joining L_k to generate C_{k+1} itemsets
Input L_k
No
L_k is empty?
Yes
Extracting frequent itemsets L_k
End
Apriori association rule mining algorithm
$f_r(h, t)$
r
h
t
DistMult decoder
Result
Attention calculation
Individual node information aggregation process
Relationship 1
Relationship N
Self-circulation
Layer K network
Output
Node information aggregation process
Graph neural network

Fig. 1. The architecture diagram of our algorithm.

We utilize the framework of R-GCN [15] to integrate node attention mechanisms across relationships in the network to aggregate information about nodes. We also use the Apriori algorithm to con-struct triples for evaluation. Finally, we input the triples into the model and use the DistMult decoder to calculate the scores and rankings of the triples, obtaining the final link prediction results.

3.1 Node information aggregation based on cross-relation attention mechanism

As shown in **Fig. 1**, the first step is to encode the entity set V in the knowledge graph as text. Then, the encoded entities are mapped to corresponding indices, typically starting from 0 and incrementing sequentially. A dictionary is created to map each index to its corresponding entity in the knowledge graph. During the model inference process, the entity information is uniformly processed using these indices. Next, a dictionary is built to map indices to vectors. This allows each index to correspond to a low-dimensional vector representation of the final transformed entity. For the edge set E in the knowledge graph, each edge e_i is transformed into a relationship type r_i, and dictionaries are created to establish mappings between relationships and indices, as well as between indices and relationship vectors. This completes the transformation of the knowledge graph into $G = (V, \varepsilon, R)$, where the knowledge in the graph is represented as vectors, serving as the input for subsequent algorithms.

After converting entities and relationships in the knowledge graph into vectors, a graph neural network structure is used to aggregate information from nodes, enabling link prediction in the existing knowledge graph. The entire knowledge graph is represented as a collection of vectors, denoted as $G = (V, \varepsilon, R)$. $\{V\} = \{1,2,3, \dots, n\}$ is the set of indices for all entities in the knowledge graph, $\{\varepsilon\}$ represents the set of all edges, and $\{R\}$ represents the set of relationships. Assuming that the head node of a triplet is $v_i \in V$, the tail node is $v_j \in V$, and the relationship between the nodes is $r \in R$, this triplet can be represented as $(v_i, r, v_j) \in \varepsilon$. In GCN [16], the embedding vector of a node can be computed using the formula (2).

$$h^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}h^{(l)}W^{(l)}\right) \tag{2}$$

Where $h^{(l)}$ is the embedding vector of nodes in the l-th layer, $W^{(l)}$ is the trainable weight matrix of the l-th layer, $\tilde{A} = A + l$ is the sum of the adjacency matrix and the self-connection matrix of the graph., $\widetilde{D}$ is the degree matrix of $\tilde{A}$, and σ is the activation function.

Based on Equation (2), the information aggregation process in GCN can be represented using Equation (3).

$$h_i^{(l+1)} = \sigma\left(\sum_{j\in N_i}\frac{1}{\sqrt{\tilde{d}_i\tilde{d}_j}}h_j^{(l)}W^{(l)}\right) \tag{3}$$

Where $h_i^{(l+1)}$ represents the embedding vector of node i in the $l+1$-th layer, $W^{(l)}$ is the trainable weight matrix, N_i is the set of neighbors of node i, and $\tilde{d}_i$ and $\tilde{d}_j$ are the degrees of node i and node j in the matrix $\tilde{A}$ respectively.

In this study, we consider the heterogeneity [17] of the knowledge graph. Therefore, the first step is to convert entities and relationships in the knowledge graph into vectors. Additionally, mappings between indices and vectors are constructed for entities and relationships. By utilizing the correspondence between indices and vectors, we can conveniently split the knowledge graph into different heterogeneous graphs for processing, based on different relationships. This is reflected in the construction of different weight matrices for different relationships, instead of using a single weight matrix for all relationship types as in GCN.

Building upon GCN, considering the heterogeneity of the graph, we perform information aggregation for different edge types between nodes. This is expressed in Equation (4).

$$h_i^{(l+1)} = \sigma\left(\sum_{r\in R}\sum_{j\in N_i^r}\frac{1}{c_{i,r}}W_r^{(l)}h_j^{(l)} + W_0^{(l)}h_i^{(l)}\right) \tag{4}$$

Where $h_i^{(l+1)}$ represents the embedding of node i in the $l+1$-th layer, σ is the activation function, R is the set of relationships, N_i^r is the set of neighboring nodes of node i under relationship r, $c_{i,r}$ is a normalization constant, $W_r^{(l)}$ is the weight matrix for relationship r. Since the information of the nodes themselves also needs to be considered, we introduce self-loops by treating them as a special type of edge. The weight matrix for self-loops is denoted as $W_0^{(l)}$.

In order to reduce the complexity of the model and improve its generalization performance, the weight matrix is diagonalized, transforming the complex weight matrix into a block diagonal matrix, thereby reducing the number of parameters and improving the generalization performance of the model. As shown in Equation (5): the size of the $Q_{br}^{(l)}$ matrix is $(d^{(l+1)}/B) \times (d^{(l)}/B)$. The block diagonalized weight matrix greatly reduces the size of the weight matrix parameters through a series of matrix additions, and the calculation process is also simplified.

$$W_{\mathrm{r}}^{(l)} = \bigoplus_{b=1}^{B} Q_{br}^{(l)} = \operatorname{diag}\left(Q_{1r}^{(l)}, \dots, Q_{Br}^{(l)}\right) \tag{5}$$

This study introduces the Node-level Across Relation Attention mechanism [18]. Before calculating the attention mechanism of nodes, for the similarity between node i and node j under a specific relationship r, the weight matrix corresponding to relationship r is multiplied by node i and node j respectively and spliced, and then multiplied by a trainable attention vector. Finally, the similarity between node i and node j under relationship r is obtained through an activation function. As shown in Equation (6): e_{ij}^r represents the similarity between node i and node j under relationship r. $\vec{a}^r$ is a trainable attention vector.， $\overrightarrow{W_r}$ is the weight matrix under relationship r.， $h_i^{(l)}$ represents the embedding vector of node i in the l-th layer.， $\oplus$ represents the concatenation operation of vectors.

$$\mathrm{e}_{ij}^r = \mathrm{ReLU}\left(\vec{a}^r\left[\overrightarrow{\mathrm{W}}_{\mathrm{r}}\mathrm{h}_{\mathrm{i}}^{(\mathrm{l})} \oplus \overrightarrow{\mathrm{W}}_{\mathrm{r}}\mathrm{h}_{\mathrm{j}}^{(\mathrm{l})}\right]\right) \tag{6}$$

Based on formula (6), during the iterative process of the graph neural network, the similarity of neighboring nodes under all relationships is calculated for each node in the knowledge graph, and then the attention of the neighboring nodes is calculated using the similarity. The information aggregation method of each layer of nodes is shown in formula (7). The final embedding representations of nodes and relationships for the entire network are as follows.

$$\mathrm{h}_{\mathrm{i}}^{(l+1)} = \sigma\left(\sum_{r\in R}\sum_{j\in N_i^r} \frac{\exp\left(\mathrm{ReLU}\left(\vec{a}^r\left[\overrightarrow{\mathrm{W}}_{\mathrm{r}}\mathrm{h}_{\mathrm{i}}^{(\mathrm{l})} \oplus \overrightarrow{\mathrm{W}}_{\mathrm{r}}\mathrm{h}_{j}^{(\mathrm{l})}\right]\right)\right)}{\sum_{k\in N_i^r}\exp\left(\mathrm{ReLU}\left(\vec{a}^r\left[\overrightarrow{\mathrm{W}}_{\mathrm{r}}\mathrm{h}_{\mathrm{i}}^{(\mathrm{l})} \oplus \overrightarrow{\mathrm{W}}_{\mathrm{r}}\mathrm{h}_{k}^{(\mathrm{l})}\right]\right)\right)} W_r^{(l)} h_j^{(l)} + W_0^{(l)} h_i^{(l)}\right) \tag{7}$$

3.2 Data preprocessing based on the Apriori association rule mining approach

The Apriori algorithm adopts a "level-wise search" strategy to determine frequent itemsets by scanning the dataset multiple times. This strategy allows for efficient and rapid discovery of frequent itemsets and association rules. By utilizing the Apriori algorithm to mine potential relationships and using them as input for graph neural networks, the accuracy and efficiency of knowledge reasoning can be improved. Additionally, the Apriori algorithm is interpretable, providing explanations and understanding of the results of knowledge reasoning.

As shown in **Fig. 1**, the Apriori algorithm can be used to mine frequent itemsets from a knowledge graph, treating the frequent itemsets as a collection of edges for relationship inference. Specifically, the output of the Apriori algorithm is used as input for the graph neural network. The graph neural network filters and ranks these rules, treating the association rules as edges in the graph and the items in the association rules as nodes. The graph neural network then learns the relationships between these nodes. This transforms all the rules generated by the Apriori algorithm into a graph, which can be analyzed and processed using the graph neural network.

In this study, training subgraphs will be obtained using uniform random sampling [19], and data preprocessing will be performed on the training subgraphs using positive and negative sampling [20]. During the training process, positive examples (i.e., existing edges or nodes) and negative examples (i.e., non-existing edges or nodes) are randomly sampled from the graph to construct the training and validation sets.

In summary, the data preprocessing method based on the Apriori association rule mining approach effectively reduces noise in the training graph. Moreover, when dealing with large-scale graph datasets, this method significantly reduces computational and storage costs, thereby improving training efficiency.

3.3 Model Optimization

The loss function for training the final model is given as follows.

$$\mathrm{L} = -\sum_{(\mathrm{i,j})\in \mathrm{E}} \log\, \sigma\big(\tilde{\mathrm{y}}_{(\mathrm{i,j})}\big) - \sum_{(\mathrm{i,j})\in E} \log\, \sigma\big(\tilde{\mathrm{y}}_{(\mathrm{i,j})}\big) \tag{8}$$

Where E represents the set of positive sample edges.， $\tilde{E}$ represents the set of negative sample edges. $\tilde{y}_{(i,j)}$ represents the predicted probability of an edge between node i and node j. It can be calculated using Equation (9).

$$\tilde{y}_{(i,j)} = \frac{1}{k}\sum_{t=1}^{k} \sigma\big(h_i^{(L)} \cdot r_t^{(L)} h_j^{(L)}\big) \tag{9}$$

Where k represents the number of negative samples that are sampled. $h_i^{(L)}$ and $h_j^{(L)}$ represents the embedding vectors of node i and node j in the last layer. $r_t^{(L)}$ represents the weight vector of relation type r_t in the L-th layer. σ represents the sigmoid function. The sigmoid function is used to map the predicted values to a range between 0 and 1. The first term of the loss function is the cross-entropy loss for positive samples, indicating that the lower the model predicts the probability of positive sample edges, the lower the loss. The second term is the cross-entropy loss for negative samples, indicating that the lower the model predicts the probability of negative sample edges not existing, the lower the loss. By minimizing the loss function, the model can make more accurate predictions of the likelihood of edge existence between nodes.

4 Comparative experiments and analysis

In this section, we conducted extensive experimental evaluations. We compared our proposed method with three other state-of-the-art graph neural network models for knowledge reasoning in knowledge graphs to verify the performance of our method. In addition, we set different support and confidence values for the data sets used to explore the impact and significance of different parameter settings on association rule mining.

4.1 Dataset

We conducted comparative experimental studies using the FB15K-237 [21] and WN18 datasets. FB15K-237 is a commonly used knowledge graph dataset, which is a subset of the FB15K dataset. FB15K-237 contains 15,000 entities, 237 relationships, and 310,116 triples. The WN18 dataset is a classic dataset for knowledge graph link prediction tasks. The training set of WN18 contains 141,442 triples, and the validation and test sets each contain 5,000 triples. The triples in the training, validation, and test sets are mutually exclusive.

4.2 Evaluation metrics

We used two evaluation metrics: MRR and $Hits@N$. MRR [22] is a good evaluation metric because it considers not only whether the predicted results are correct but also the ranking of the predicted results. $Hits@N$ [23] is used to measure whether the algorithm can correctly predict the correct entities or relations in the test set among the top N candidate entities or relations.

The core idea of MRR is to consider each test triple (h, r, t) and find the true tail entity t among all possible tail entities t' prime, and calculating the reciprocal of its score ranking. In the end, the mean reciprocal rank of all test triples is computed as the MRR score of the model. The MRR score ranges from 0 to 1, where a value closer to 1 indicates that the model's predicted results are ranked higher, indicating better prediction performance.

$Hits@N$ quantifies the prediction accuracy of a model in knowledge graph link prediction tasks. Given a test set T, for each triple $(h, r, t) \in T$, the algorithm predicts N sorted candidate entities. The indicator function returns 1 if the condition in the parentheses is true and 0 otherwise. Specifically, when the correct answer t is among the top N predicted entities, the function value is 1; otherwise, it is 0. When calculating $Hits@N$, if there is no correct answer in all candidate entities for a triple, that triple is not included in the $Hits@N$ calculation. A higher hit rate indicates that the model can better predict the correct entities and relations.

In association rule mining research, we use two evaluation metrics: support and confidence. Support reflects the frequency and probability of data appearing in the entire dataset. Confidence represents the probability of the occurrence of one data item given the presence of another data item, or in other words, conditional probability.

4.3 Implementation details

In this experiment, the proposed model is compared with Graph Convolutional Neural Network (GCN), Graph Attention Network (GAT), and Relational Graph Convolutional Network (R-GCN) on two datasets, FB15K-237 and WN18. The final set of hyperparameters used in the experiments are as follows: the model was trained for 6000 iterations, the dropout rate was set to 0.2, the learning rate was set to 0.01, the output dimension of the hidden layers was set to 500, the number of weight matrix block diagonal decompositions was set to 100. During the evaluation of the model, the size of the subgraph splits was set to 500. The positive sample extraction ratio was set to 0.5, and for each positive sample, 10 negative samples were generated.

Additionally, in terms of association rule mining, experiments were conducted on the FB15K-237 dataset by setting the confidence to 0.8 and varying the support with finer-grained values of 0.14, 0.15, 0.16, 0.17, and 0.18. The model used was trained on subgraphs of size 80,000, and its performance metrics were MRR: 0.275, $Hits@1$: 0.181, $Hits@3$: 0.307, and $Hits@10$: 0.464. The dataset consists of 14,541 entities, 237 relations, and 272,115 triples. After computing the scores for the triples to be evaluated, triples with a rank of 1 or a score greater than 0.999 were selected as the final inferred triples that meet the requirements.

In the WN18 dataset, the confidence was set to 0.8, and compared to FB15K-237, the support was varied with coarser-grained values of 0.35, 0.4, 0.45, 0.5, and 0.55. To make a fair comparison with FB15K-237, the model used was also trained on subgraphs of size 80,000, with performance metrics of MRR: 0.905, $Hits@1$: 0.877, $Hits@3$: 0.932, and $Hits@10$: 0.942. The dataset consists of 40,943 entities, 18 relations, and 141,442 triples.

4.4 Experimental results

The experimental results of comparing our model with three other neural network models are presented in **Table 1**, **Table 2**, and **Fig. 2**.

Table 1. Summarizes the experimental results on the FB15K-237 dataset. Higher values for MRR and $Hits@N$ indicate better performance.

FB15K -237	The training subgraph size is 30,000				The training subgraph size is 80,000			
	MRR	$Hits@1$	$Hits@3$	$Hits@10$	MRR	$Hits@1$	$Hits@3$	$Hits@10$
Ours	**0.246**	**0.158**	**0.268**	**0.425**	**0.275**	**0.181**	**0.307**	**0.464**
GCN	0.167	0.087	0.193	0.321	0.191	0.099	0.228	0.369
GAT	0.214	0.128	0.237	0.386	0.250	0.161	0.270	0.441
R-GCN	0.241	0.140	0.253	0.409	0.265	0.176	0.286	0.454

In the FB15K-237 dataset, when the training subgraph size is 30,000, this research model outperforms the GCN, GAT, and R-GCN models in all metrics, but the relative differences are small. However, when the training subgraph size is 80,000, the research model achieves an MRR score of 0.275, higher than the GCN, GAT, and R-GCN models which scored 0.191, 0.250, and 0.265, respectively. Compared to the GCN model, the MRR is improved by approximately 40%. Compared to the GAT model, the MRR is improved by approximately 10%. Compared to the R-GCN model, the MRR is improved by approximately 3.8%. Additionally, metrics such as $Hits@1$, $Hits@3$, and $Hits@10$ also show significant improvements.

Table 2. Summarizes the experimental results on the WN18 dataset. Higher values for MRR and $Hits@N$ indicate better performance.

WN18	The training subgraph size is 30,000				The training subgraph size is 80,000			
	MRR	$Hits@1$	$Hits@3$	$Hits@10$	MRR	$Hits@1$	$Hits@3$	$Hits@10$
Ours	**0.847**	**0.795**	**0.891**	**0.929**	**0.905**	**0.877**	**0.932**	**0.942**
GCN	0.369	0.222	0.448	0.655	0.485	0.300	0.610	0.838
GAT	0.440	0.322	0.498	0.673	0.722	0.596	0.824	0.925
R-GCN	0.807	0.728	0.849	0.902	0.838	0.800	0.866	0.933

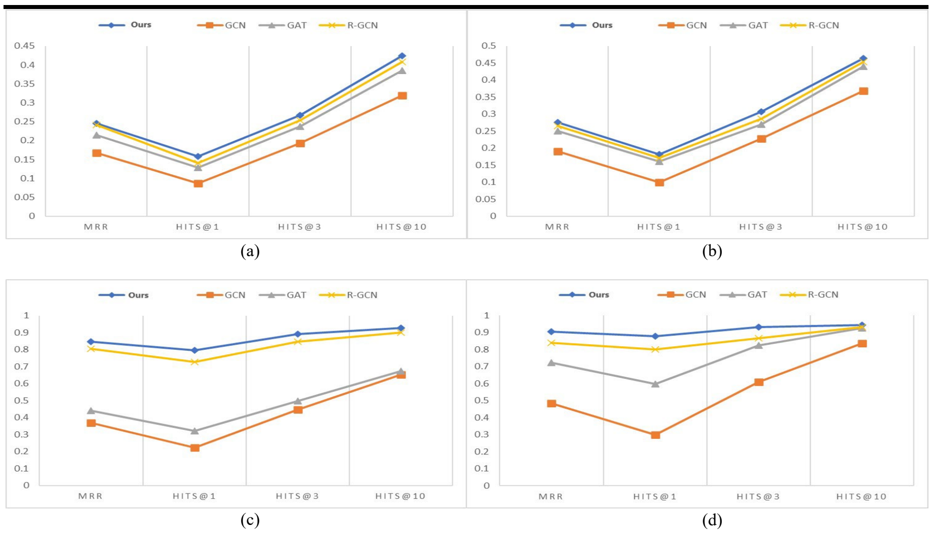


Fig. 2. (a) Comparative analysis of various models on the FB15K-237 dataset (training subgraph size: 30,000) (b) Comparative analysis of various models on the FB15K-237 dataset (training subgraph size: 80,000) (c) Comparative analysis of various models on the WN18 dataset (training subgraph size: 30,000) (d) Comparative analysis of various models on the WN18 dataset (training subgraph size: 80,000)

In the WN18 dataset, when the training subgraph size is 30,000, this research model performs outstandingly well, with an MRR score of 0.847, significantly higher than the GCN and GAT models. When the training subgraph size is 80,000, the research model achieves an MRR score of 0.905, which is significantly higher than the GCN, GAT, and R-GCN models that scored 0.485, 0.722, and 0.838, respectively. Compared to the GCN model, the MRR is improved by approximately 86%. Compared to the GAT model, the MRR is improved by approximately 25%. Compared to the R-GCN model, the MRR is improved by approximately 8%. Additionally, in terms of hit rates, when the training subgraph size is 80,000, the research model outperforms the other three models significantly in $Hits@1$ and $Hits@3$ on the WN18 dataset, while being comparable to GAT and R-GCN in $Hits@10$.

Overall, this research model has achieved better performance compared to link prediction models based on GCN, GAT, and R-GCN in the task of knowledge graph link prediction. It demonstrates good robustness and generalization capability. The experimental results of this study confirm the effectiveness and superiority of the proposed model in the task of knowledge graph link prediction.

4.5 Comparative experimental results on incorporating Apriori association rule mining

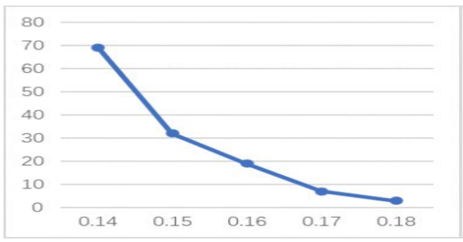
After introducing the Apriori association rule mining approach, the impact of different support and confidence values on the discovered association rules and their influence on the final results is presented in **Table 3**, **Table 4** and **Fig. 3**. The following is an analysis report comparing the results.

Table 3. Knowledge inference results on the FB15K-237 dataset, where a higher ratio of inference results to the number of triples to be evaluated indicates better performance.

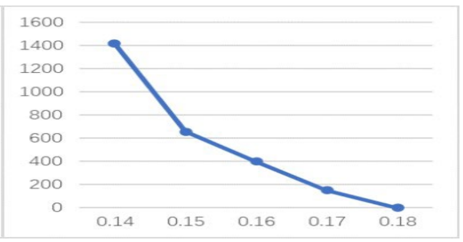
Support	Relation pairs	Triples to be evaluated	Inference results	Inference results / Triples to be evaluated
0.14	69	16353	1417	0.086
0.15	32	7584	656	0.086
0.16	19	4503	398	0.088
0.17	7	1659	149	0.090
0.18	3	711	0	0

Table 4. Knowledge inference results on the WN18 dataset, where a higher ratio of inference results to the number of triples to be evaluated indicates better performance.

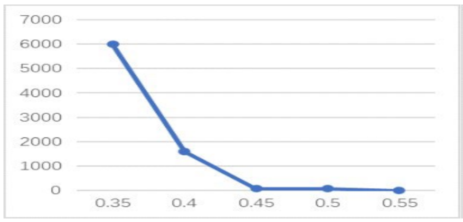
Support	Relation pairs	Triples to be evaluated	Inference results	Inference results / Triples to be evaluated
0.35	5990	107820	1174	0.011
0.4	1589	28602	575	0.020
0.45	72	1296	40	0.031
0.5	72	1296	40	0.031
0.55	3	54	0	0



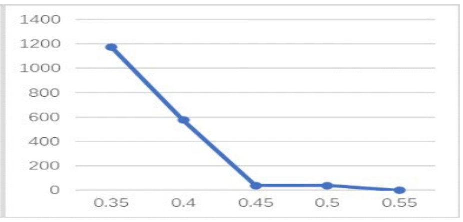
(a)



(b)



(c)



(d)

Fig. 3. (a) Comparison of the number of relation pairs (FB15K-237) (b) Comparison of inference results (FB15K-237) (c) Comparison of the number of relation pairs (WN18) (d) Comparison of inference results (WN18)

Firstly, the above results prove that setting different degrees of support can filter the noisy triples before knowledge inference to different degrees and improve the efficiency of knowledge inference. Meanwhile, in terms of the ratio of inference results and to-be-evaluated triples, the ratio shows an increasing trend with the increase of support degree on both datasets, which indicates that the quality of to-be-evaluated triples after data preprocessing using association rule mining is getting higher and higher in addition to filtering out most of the noises, resulting in a higher and higher ratio of the final results inferred using the model.

Secondly, it is worth noting that in FB15K-237, as the support increases, the ratio between the inferred results and the evaluated triplets increases from 8.6% to 9%. In WN18, the ratio of inferred results increases from 1.1% to 3.1%. Overall, although the support threshold set in FB15K-237 is much lower than that in WN18, the proportion of inferred results is higher in FB15K-237. This is because WN18 has a higher information density compared to FB15K-237, resulting in a relatively lower proportion of knowledge that can be inferred. However, with the increase in support, the proportion of inferred results in WN18 increases faster than in FB15K-237, rising from 1.1% to 3.1%, an increase of nearly 3 times. The analysis in this study suggests that the higher support threshold and larger numerical interval of 0.5 in WN18 compared to 0.1 in FB15K-237 contribute to a more significant improvement in data quality through association rule mining. As a result, the overall quality of the evaluated triplets increases, leading to a larger increase in the proportion of inferred results.

In summary, the experimental results of this study have well verified the effectiveness of incorporating the Apriori association rule mining algorithm in improving the quality and efficiency of the inference results on both datasets, but it is important to note that the number of mined relationship pairs starts to decrease dramatically when the support is increased to a certain degree, so a balance between the support and the inference results needs to be found to obtain optimal performance of the inference model in practical applications.

5 Conclusions

This paper focuses on knowledge graphs and knowledge reasoning and proposes an improved approach for multi-relational knowledge graph link prediction by integrating a graph neural network with a cross-relation attention mechanism and the Apriori association rule mining algorithm. The approach includes the knowledge graph input, node information aggregation based on node-level cross-relation attention mechanism, training sampling and loss function, DistMult decoder, and the process of integrating association rule mining. Finally, comparative experiments and analysis are conducted on the FB15k-237 and WN18 datasets to validate the effectiveness of the proposed algorithm.

In the future, it would be beneficial to consider intra-relation attention mechanisms for node relationships [24]. The distribution of importance among nodes within the same relationship is equally important, and it is essential to make information propagation among nodes more efficient within the same relationship. These two attention mechanisms are not mutually exclusive in knowledge graph reasoning, so they can be considered simultaneously for further exploration. Additionally, further validation of the algorithm's complexity and effectiveness can be carried out.

References

1. Steiner, T., Verborgh, R., Troncy, R., Gabarro, J., Van~de Walle, R.: Adding realtime coverage to the google knowledge graph. In: 11th International Semantic Web Conference (ISWC 2012). vol.~914, pp. 65--68. Citeseer (2012)
2. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE transactions on neural networks 20. 1, 61--80 (2008)
3. Schlichtkrull, M., Kipf, T.N., Bloem, P., Van Den~Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3--7, 2018, proceedings 15. pp. 593--607. Springer (2018)
4. Nathani, D., Chauhan, J., Sharma, C., Kaul, M.: Learning attention-based embeddings for relation prediction in knowledge graphs. arXiv preprint arXiv:1906. 01195 (2019)
5. Fox, M., Poole, D.: Proceedings of the twenty-fourth AAAI conference on artificial intelligence. In: AAAI 2010 (2010)
6. Suda, M., Weidenbach, C., Wischnewski, P.: On the saturation of YAGO. In: Automated Reasoning: 5th International Joint Conference, IJCAR 2010, Edinburgh, UK, July 16-19, 2010. Proceedings 5. pp. 441--456. Springer (2010)
7. L Bühmann, Lehmann J.: Pattern based knowledge base enrichment. In: The Semantic Web--ISWC 2013: 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I 12. pp. 33--48. Springer (2013)

8. Jiang, S., Lowd, D., Dou, D.: Learning to refine an automatically extracted knowledge base using markov logic. In: 2012 IEEE 12th International Conference on Data Mining. pp. 912--917. IEEE (2012)
9. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. Advances in neural information processing systems 26 (2013)
10. Nickel, M., Tresp, V., Kriegel, H.P., et al.: A three-way model for collective learning on multi-relational data. In: Icml. vol.~11, pp. 3104482—3104584 (2011)
11. Yang, B., Yih, W. t., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. arXiv preprint arXiv:1412.6575 (2014)
12. Socher, R., Chen, D., Manning, C.D., Ng, A.: Reasoning with neural tensor networks for knowledge base completion. Advances in neural information processing systems 26 (2013)
13. Shi, B., Weninger, T.: ProjE: Embedding projection for knowledge graph completion. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol.~31 (2017)
14. Agrawal R., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of the 1993 ACM SIGMOD international conference on Management of data. pp. 207--216 (1993)
15. Schlichtkrull, M., Kipf, T.N., Bloem, P., Van Den~Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3--7, 2018, proceedings 15. pp. 593--607. Springer (2018)
16. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
17. Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., Yu, P.S.: Heterogeneous graph attention network. In: The world wide web conference. pp. 2022—2032 (2019)
18. Velikovi P., Cucurull G., Casanova A., et al.: Graph Attention Networks. *Stat* 1050. 20, 10--48550. (2017)
19. Chen, J., Zhu, J., Song, L.: Stochastic training of graph convolutional networks with variance reduction (2018)
20. Mikolov T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
21. Toutanova, K., Chen, D.: Observed versus latent features for knowledge base and text inference. In: Proceedings of the 3rd workshop on continuous vector space models and their compositionality. pp. 57--66 (2015)
22. Voorhees, E.M., Tice, D.M., et al.: The trec-8 question answering track evaluation. In: TREC. vol.~1999, p.~82 (1999)
23. Ricardo, B.Y., Berthier, R.N.: Modern information retrieval: the concepts and technology behind search. New Jersey, USA: Addi-son-Wesley Professional (2011)
24. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Advances in neural information processing systems 30 (2017)