Applied Intelligence (2024) 54:6085–6107
https://doi.org/10.1007/s10489-024-05482-2

Semantic- and relation-based graph neural network for knowledge graph completion

Xinlu Li[1] · **Yujie Tian**[1] · **Shengwei Ji**[1]

Accepted: 20 April 2024 / Published online: 6 May 2024

Abstract
Knowledge graph completion (KGC) refines missing entities, relationships, or attributes from a knowledge graph, which is significant for referral systems, biological informatics, and search engines. As an effective KGC approach, a graph neural network (GNN) learns to aggregate information from neighboring nodes by iteratively passing messages between them. However, the semantic and relational information contained in knowledge graphs is rarely used in the existing GNN-based approaches for KGC (i.e., only structure information is used). Hence, a semantic- and relation-based GNN (SR-GNN) model, which combines the semantic similarity information between neighboring entities and the relational features of knowledge graphs, is proposed. First, we develop an entity semantic aggregation module that learns semantic similarity information among neighboring entities connected to the same central entity via an RNN. Second, we propose a relational aggregation module that captures the different semantics among different types of relations through a GRU. This enables the model to better comprehend semantic relationships and be applied to KGC tasks requiring relationship embedding vectors. Extensive studies conducted on the FB15k-237, WN18RR, WN18 and YAGO3-10 datasets reveal that, when compared to 17 baseline models, the SR-GNN exhibits state-of-the-art performance in terms of the MRR and H@n metrics. Significantly, the MRR metric improves by 10.2% on the FB15K-237 dataset and by 4.2% on the WN18RR dataset over those of the rival models.

Keywords Knowledge graphs · Knowledge graph completion · Knowledge graph embedding · Recurrent neural network

1 Introduction

A knowledge graph (KG) is a graphical structure that describes entities, concepts, and the relationships among them [1]. KGs are usually constructed through manual annotation or automatic extraction from established databases such as Wikidata [2], DBPedia [3], and Freebase [4]. In KGs, fact triples are a representations of knowledge, where each fact triple consists of a subject entity $\boldsymbol{h}$, an object entity $\boldsymbol{t}$, and a relation $\boldsymbol{r}$ [5]. These triples are used to represent and store knowledge using precisely networked structures. However, these data are unable to capture all the information that is present in the world of reality, and most KGs still suffer from content incompleteness after being built. The incompleteness of a KG can be attributed to two main factors. First, a KG may have an information gap due to its inability to fully cover the entire real world. Second, the information in a KG might not be completely correct due to factors such as changes over time or errors introduced during the KG construction process. KGC improves the completeness and accuracy of a KG by predicting and inferring missing triple information. KGC accomplishes this task by automatically creating low-dimensional embedding representations of relationships and nodes from established fact triples. This process not only reduces the costs required to build a KG but also helps improve its completeness and accuracy.

KGC refers to inferring missing entities, attributes, or relationships from an incomplete KG. Acquiring a more comprehensive and accurate description of the target KG through techniques such as data mining and machine learning is key to achieving this goal. To date, several knowledge graph embedding (KGE) approaches have been developed for KGC. These embedding approaches adopt scoring algo-

✉ Shengwei Ji
jisw@hfuu.edu.cn

Xinlu Li
xinlu.li@hfuu.edu.cn

Yujie Tian
tianyujie@stu.hfuu.edu.cn

[1] School of Artificial Intelligence and Big Data, Hefei University, Hefei 230000, China

rithms to assess the plausibility of triples. They employ a variety of transformation functions to map relationships and nodes to low-dimensional vector spaces [6]. The KGE models used to address KGC tasks can be divided into five categories: transfer-based models [7–10], tensor decomposition-based models [11–13], traditional deep learning-based models [14–16], additional textual information-based models [17–19], and graph neural network (GNN)-based models [20–22]. Among them, GNN-based graph embedding models overcome the limitations of convolutional neural networks (CNNs) and can be applied to more data types than only Euclidean data. GNNs have also achieved excellent performances on non-Euclidean data such as KGs, and they can be used to capture neighborhood information by aggregating data from graph structures. Among them, the R-GCN [20] employs a framework paradigm, namely, the encoder-decoder framework, for for using the GNN family of models to complete link prediction tasks. Models based on GNNs are frequently adopted as global KG encoders to record graph structure data. Models such as ConvE [16], and InteractE [15] are often used as model decoders for prediction scoring. For the past several years, GNNs has been extensively used in KGC scenarios due to its excellent graph-structured data modeling capabilities.

However, most KGE models [20, 21] focus only on nodes associated with the information of the central node during the message passing process. They ignore semantic associations, i.e., common semantic similarity features among the entities that are not connected by any explicit relationships. Specifically, the neighboring entities $t_1 \sim t_N$ connected to the head entity $\boldsymbol{h}$ have some shared semantic similarity features, even though they are not connected by any explicit relationships. For example, TransE [7] models triples as vector translation expressions, such that $\boldsymbol{h}_u + \boldsymbol{h}_r = \boldsymbol{h}_v$. If the head entity, relation, and tail entity satisfy this addition operation, the triple is considered valid. RESCAL [11] models triple as $\boldsymbol{h}_u^{\mathrm{T}} \boldsymbol{h}_r \boldsymbol{h}_v = \boldsymbol{T}_{urv}$, computing the semantic similarity between the head entity, relation, and tail entity. ConvE [16] concatenates the head entity and relation, uses a CNN to re-encode the concatenation matrix, and finally computes the similarity with the tail entity. The R-GCN [20], CompGCN [21], and other embedding models based on GNNs aggregate the feature information of entities according to the topology of the input graph. These five types of embedding models all use relations as bridges to construct semantic information between entities. However, they all overlook the most direct way of obtaining semantic representations between entities: obtaining them from neighboring entities.

As depicted in Fig. 1(1), the triple (*Yeoh Choo Kheng, performance, Everything Everywhere All at Once*) means that *Yeoh* starred in the movie *Everything Everywhere All at Once*. The triple (*Yeoh Choo Kheng, award, Academy Awards*) means that *Yeoh Choo Kheng* won an Oscar. As shown in Fig. 1(2), in the traditional entity aggregation method, only the semantics of the *Everything Everywhere All at Once* entity can be passed to the central entity *Ke Huy-Quan*, while the information of the *Academy Awards* entity cannot be passed in this hop. As depicted in Fig. 1(3), in contrast to the traditional methods, we believe that the movie *Everything Everywhere All at Once* and the *Academy Awards* should have some semantic similarity as a pair of neighboring entities of *Yeoh*. Additionally, the *Academy Awards* entity should contain part of the information processed by *Everything Everywhere All at Once*, i.e., "a person who has won a movie award must have acted in a movie". Then, in the triple (*Ke Huy-Quan, performance, Everything Everywhere All at Once*), the central entity *Ke Huy-Quan* not only captures the semantic feature information of *Everything Everywhere All at Once* but also capture parts of the semantic information of the *Academy Awards*. Finally, in the query set (*Ke Huy-Quan, award*, ?) of the alternative entity set (*Golden Bear Award, Academy Awards, Chinatown*), the scoring function will give a higher score to the *Academy Awards* entity.

In addition, most KGE models [23, 24] aim at updating entity embeddings but neglect to include relationships in the updating process. RotatE [9] utilizes rotation in a complex space as a substitute for the vector addition operation of TransE in Euclidean space. The R-GCN [20] models relations as embeddings to facilitate feature propagation among entities. SACN [19] constructs a weight matrix based on different relation types and learns entity representations according to the importance levels of relations. Such methods focus on learning entity embedding features, implicitly incorporating relations into the learning process (keeping the relation features unchanged). This approach overlooks the explicit representations of relations, i.e., incorporating relation feature vectors into the same updating process as that of the entities. As depicted in Fig. 1(1), for different head entities, the same relation *performance* likely points to the same tail entity *Everything Everywhere All at Once*. Explicit relation aggregation operations update simultaneously both relation features and entity features. This effectively enables the relation features to learn semantic information about the tail entities. Furthermore, different types of relations, such as *award* and *performance* point to different tail entities *Academy Award* and *Everything Everywhere All at Once*, respectively.

To address these issues, this paper investigates how to fuse the semantic information of neighboring entities, how to aggregate relationship features, and how to use these features to accomplish the KGC task. Figure 2 is a graphic abstract of this paper. First, this paper draws on the distributed semantic assumption from natural language processing, namely, that words in the same context usually have similar semantics [25–29]. Semantic similarity is measured using the neighbor similarity of the same entities. An RNN is used to learn

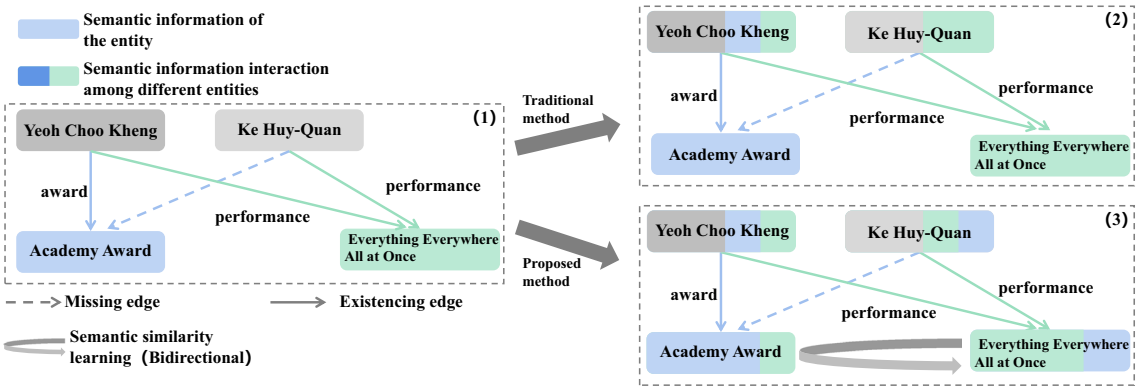

Fig. 1 Example of a fact triple in KG. Using different colors to indicate the semantics of different entities. If there is a same color between two entities, it means that the entities have similar semantics to each other. Figure 1(1) depicts the initial state of the fact triad; Fig. 1(2) depicts the learning of entity representations using traditional methods; and Fig. 1(3) shows the capture of semantic information between neighboring entities using semantic similarity learning methods (the proposed method)

the dependencies between entities and capture the semantic similarities between neighboring entities associated with the same central entity. Then, a GNN is used to pass the neighbor information to the central entities according to different relationship types. This process enhances the semantic interactions among the neighboring entities under the same central entity. In addition, this paper demonstrates how different types of relational embeddings can be input into a GRU to assist the model in providing a better understanding of the semantic differences between different relations. Moreover, these relational embeddings can also be used in other KGC tasks that require relational embedding to increase the robustness and accuracy of the model. In summary, this paper proposes a semantic and relational GNN framework, a SR-GNN, that can learn the semantic similarity information among neighboring entities and aggregate relationship features. A GNN-based model is used as an the encoder to obtain a more comprehensive knowledge representation. Finally, ConvE [16] is selected as the decoder to perform KGC tasks. The contributions made in this work are summarized in following sentences:

- An entity semantic aggregation module that learns semantic similarity information among the neighboring entities connected to the same central entity via an RNN is proposed.

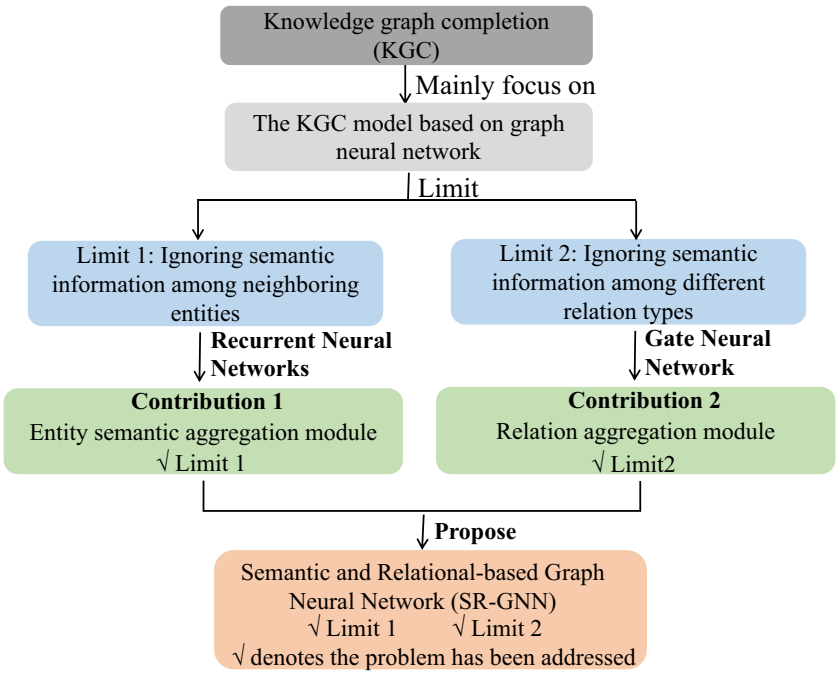

Fig. 2 Red denotes the proposed model, and blue and green boxes stand in for limitations and contributions, respectively, in the study of the relationship between motivation and contribution

- A relational aggregation module that captures the semantic differences among various types of relations through a GRU is proposed.
- A semantic- and relation-based GNN (SR-GNN) that incorporates entity semantic information, structural information, and graph relation information is proposed.
- Extensive experiments conducted on multiple datasets show that the SR-GNN achieves excellent performance in comparisons with 17 baseline models in terms of several metrics. Significantly, the MRR metric improves by 10.2% on the FB15K-237 dataset and by 4.2% on the WN18RR dataset over those of the the rival models.

2 Related work

This section focuses on two research areas connected to this work: KGE models and the RNN series. A KGE model is used to accomplish the KGC task, and an RNN is used to learn the semantic similarities among entities and the semantic differences among various kinds of relationships. The symbols that are used often in the following sections are listed in Table 2.

2.1 Knowledge graph embedding

To date, various KGE methods have been proposed for KGC tasks. These KGE methods rely on various functions to transfer relationships and entities to low-dimensional vector spaces. They also define scoring functions to assess the probabilities of valid triples [30–32]. As shown in Fig. 3, KGE models can be divided into five categories.

(1) Translation distance-based models were first proposed by Bordes et al. [7]., who considered that if the triad composition assumption holds, the embeddings of subject entities h_u, object entities $\boldsymbol{h}_v$, and relations $\boldsymbol{h}_r$ satisfy $\boldsymbol{h}_u + \boldsymbol{h}_r \approx \boldsymbol{h}_v$. In contrast to TransE [7], RotatE [9] projects a vector into a complex space and uses a rotation operation to represent the relational information thereby encoding nodes and relations. Additionally, PaiRE [10] uses pairs of relation vectors to automatically adapt to the margin γ in the employed loss function to fit complex relations and is also able to encode different relation patterns. The transfer-based KGE models mentioned above can be quickly applied to large KG data due to their ease of operation and high learning efficiency. However, such models do not present the semantic features of the modeled entities and relationships, ultimately confusing the semantic representations of the entities.

(2) Tensor decomposition-based models consider a KG to be capable of being modeled by a three-dimensional tensor. The input KG is scored using semantic similarity to enable the representation of nodes and relations. RESCAL [11] is a matrix-based tensor decomposition method in which a higher-order tensor is decomposed into the product of numerous second-order matrices. During the learning process, RESCAL tries to learn an embedding vector by minimizing the reconstruction error of entity-relationship-entity triples. DistMult [12] models multirelational graphs using a bilinear neural network. It calculates the likelihood that triples will hold by matching the potential semantics of relationships and entities in the vector space. ComplEx [13] utilizes an asymmetric scoring function to represent how relationships and entities are represented in complex space. The above algorithms model a whole KG as a tensor, fully exploiting the expressive power of the KG. However, since a tensor representation that is too large reduces the efficiency of these models, subsequent models of this type will inevitably have to solve the efficiency issue first.

(3) Traditional deep learning-based models learn the KG triple feature representations through deep neural networks, such as CNNs. ConvE [16] splices the head entity and relationship vectors and performs feature extraction via a CNN to model the triples. Traditional deep neural networks have achieved excellent performances on some graph datasets due to their powerful learning capabilities. However, this type of

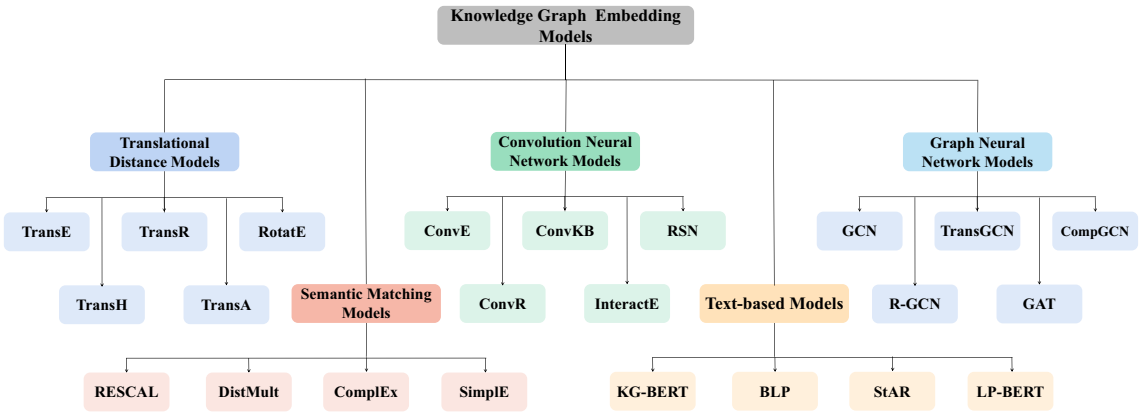

Fig. 3 Knowledge graph embedding model. Five color schemes are used to represent each of the five KGE types

modeling method lacks interpretability and it is difficult to make changes to the unique structural characteristics of graph data.

(4) Models based on additional textual information, including the names of relations and entities or textual descriptions of the relations and entities, can acquire semantic information about relationships and entities. KG-BERT [18] uses BERT [33] as a pretraining model and introduces text sequence descriptions of the observed relations and entities to capture semantic information about them. The BLP [17] model also uses BERT to learn the textual information of nodes and the structural information of KGs using the TransE model. MTL-KGC [34] is an effective multitask learning method that solves the problem that KG-BERT cannot adequately learn relationship information in KGs. StAR [19] combines graph structure coding with text coding. It comprehensively uses structure and text information to perform KGC tasks. KGE algorithms based on additional textual information incorporate textual descriptions of entities and relationships into the learning process, which helps capture rich semantic information. However, such methods require the introduction of additional textual information to pretrain the relation and entity vectors, thus increasing the impact of external information on the experimental results.

(5) Models based on GNNs enable the aggregation of entity features through graph structures. The R-GCN [20] differs from the traditional GCN in that it considers different relationship types and node aggregation between different relationship types. It associates a weight matrix with each type of relationship and performs weighted aggregation on the neighboring nodes with various relationship forms to obtain a richer node representation. SACN [24] is an encoder-decoder convolutional network for modeling entities and relationships that is sensitive to graph structures. It extracts entity features and then inputs them to its decoder to ensure that the entities satisfy triad constraints. Specifically, SACN sets weight matrices for different relation types and aggregates neighbor information according to the importance levels of different relationships. However, the R-GCN and SACN only set up a relation embedding matrix without adding the embedded representations of relations to the update process. CompGCN [21] ignores the critical role of semantic information between the neighboring entities in the information aggregation process, although multiple combinatorial operations involving entities and relations have been proposed. Li et al. [35] presented a neighborhood reordering model with relational constraints for the KGC task. They automatically generated specific constraints as extra type characteristics to augment the entity representations rather than relying on manual labels. However, this approach also ignores the influence of the semantic information among neighboring entities on the entity aggregation process. Table 1 shows how the SR-GNN compares with other existing approaches in terms of entity and relationship aggregation.

Based on the above analysis, we propose learning strategies for neighboring entities and relationships. 1) **A neighboring entity similarity learning strategy**. In previous methods, the semantic information between neighboring entities was often overlooked because they focused on the structure of the input KG. However, neighboring entities present close semantic association because they have a common central entity. 2) **An explicit relationship update strategy**. Previous algorithms usually treated relations as kinds of bridges to propagate information between entities. However, they neglected the relations that carry rich semantic representations between entities. Such relational representations should be learned and updated. By combining the above two strategies, the overall performance of the proposed model can be greatly improved.

2.2 Recurrent neural networks

RNNs [36] are used in situations where the inputs are dependent and sequential, i.e., where a prior input is related to the subsequent input. The fundamental characteristic of an RNN is that its output is a representation of the prior input and a function of the current data point. Given input data embeddings $\boldsymbol{X} = \{\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \ldots, \mathbf{X}_t\}$, an RNN reads the data

Table 1 The proposed model is compared to other GNN-based methods

Models	Entity Aggregation	Relationship Aggregation	Structural Features	Semantically Similar Features
GCN (Thomas N. et al.)	✓	✓	✓	✘
R-GCN (Schlichtkrull et al.)	✓	✘	✓	✘
SACN (Shang et al.)	✓	✘	✓	✘
CompGCN (Vashishth et al.)	✓	✓	✓	✘
SR-GNN (Proposed Model)	✓	✓	✓	✓

The models are assessed from four angles: whether there is entity aggregation; whether there is relationship aggregation; whether graph structure features are used; and whether node semantic features are used, respectively

sequentially. The following equation outputs the state that is hidden $\boldsymbol{H}^{(t)}$ at each time step:

$$\begin{aligned}\boldsymbol{H}^{(t)} &= RNN\left(\boldsymbol{X}^{(t)} \cdot \boldsymbol{H}^{(t-1)}\right) = \sigma\left(\boldsymbol{W}_{XH}\boldsymbol{X}^{(t)}\right.\\ &\quad \left.+\boldsymbol{W}_{HH}\boldsymbol{H}^{(t-1)} + \boldsymbol{b}_H\right),\\ \boldsymbol{Y}^{(t)} &= \sigma\left(\boldsymbol{W}_{HY}\boldsymbol{H}^{(t)} + \boldsymbol{b}_Y\right), \end{aligned} \tag{1}$$

where $\boldsymbol{X}^{(t)}$ is the input data vector at moments t, the hidden state of the data embedding at time $t-1$ is $\boldsymbol{H}^{(t-1)}$, and $\boldsymbol{Y}^{(t)}$ is the output vector at moments t. The bias parameter is $\boldsymbol{b}$, and the value of the weight matrix is $\boldsymbol{W}$. The semantic association between the current node and the next node is captured in this paper using an RNN to determine the semantic information between nodes. In addition, since RNNs are prone to gradient explosion or disappearance while training, other neural networks can be used to handle such problems, such as long short-term memory (LSTM) networks [37] and gated recurrent units (GRUs) [38]. LSTM alleviates the problems faced by RNN models during training through gating mechanisms and determines which important data in the given sequence need to be retained and which data need to be deleted through these gating mechanisms. A GRU is modified from LSTM by simplifying it into a reset gate and an update gate with the following equation:

$$\begin{aligned}\boldsymbol{X}^{(t)} &= \sigma\left(\boldsymbol{W}_{XR}\boldsymbol{X}^{(t)} + \boldsymbol{W}_{HR}\boldsymbol{H}^{(t-1)} + \boldsymbol{b}_R\right),\\ \boldsymbol{Z}^{(t)} &= \sigma\left(\boldsymbol{W}_{XZ}\boldsymbol{X}^{(t)} + \boldsymbol{W}_{HZ}\boldsymbol{H}^{(t-1)} + \boldsymbol{b}_Z\right), \end{aligned} \tag{2}$$

where $\boldsymbol{R}_t$ represents the reset gate and $\boldsymbol{Z}_t$ represents the update gate. By controlling the input embedding and the hidden state at the previous instant, the reset gate determines whether to forget the information contained in the hidden status. The hidden states of the prior and current input embeddings are controlled by the update gate, which determines whether to update them to the hidden state of the candidate. Utilizing the reset gate allows one to preserve historical information for updating the contents of new memories, which is calculated by the following expression:

$$\tilde{\boldsymbol{H}}^{(t)} = \tanh\left(\boldsymbol{W}_{X\tilde{H}}\boldsymbol{X}^{(t)} + \boldsymbol{R}^{(t)} \odot \boldsymbol{W}_{H\tilde{H}}\boldsymbol{H}^{(t-1)} + \boldsymbol{b}_{\tilde{H}}\right), \tag{3}$$

$\tilde{\boldsymbol{H}}^{(t)}$ represents the state of the current memory information, and the Hadamard product is $\odot$. Finally, the model computes the final hidden state information $\boldsymbol{H}^{(t)}$, which preserves the data from the current cell and passes it on to the following cell. To complete this procedure, we must utilize the update gate $\boldsymbol{Z}_t$, which determines the current memory content $\tilde{\boldsymbol{H}}^{(t)}$ and the information to be captured from the previous moment $\boldsymbol{H}^{(t-1)}$, as shown in the following equation:

$$\begin{aligned}\boldsymbol{H}^{(t)} &= \left(1 - \boldsymbol{Z}^{(t)}\right) * \boldsymbol{H}^{(t-1)} + \boldsymbol{Z}^{(t)} * \tilde{\boldsymbol{H}}^{(t)},\\ \boldsymbol{Y}^{(t)} &= \sigma\left(\boldsymbol{W}_{HY}\boldsymbol{H}^{(t)} + \boldsymbol{b}_Y\right), \end{aligned} \tag{4}$$

and GRU maintains the same effect as that of LSTM while simplifying the LSTM structure [39].

In summary, this paper proposes a semantic- and relation-based graph neural network (SR-GNN) model, which combines the semantic similarity information between neighboring entities and the explicit relationship features of the input KG. First, this paper utilizes an RNN to mine the close semantic associations between neighboring entities. Second, relationships serve as crucial bridges connecting the head and tail entities, bearing rich semantic representations of the entities. Therefore, such relationship representations need to be learned and updated. In this paper, we incorporate relations into aggregation operations, and utilizing a GRU allows use to better capture the semantic differences between various types of relations. This enables the model to gain a deeper understanding of the semantics between the relations and facilitates its application in a variety of KGC tasks. The specific modeling methods employed herein are detailed in Section 3.

3 Model description

The SR-GNN model is proposed in this section. The SR-GNN architecture is first mentioned in Section 3.1. Then, Sections 3.2-3.4 describe how the encoder part of the SR-GNN is modeled. These steps include entity semantic aggregation, entity structure aggregation, and explicit relationship aggregation. Finally, the decoder of the model is discussed in Section 3.5, and Section 3.6 analyses the strengths and weaknesses of the SR-GNN. The overall architecture is shown in Fig. 4.

3.1 Model framework

This section describes a KGC framework that fuses the semantic and structural information of entities and explicitly aggregates graph relationship information. The model adopts the encoder-decoder architecture. Algorithm 1 shows the flow of the model algorithm. Steps 1-6 concern the encoder, and steps 7-12 address the decoder. Figure 4 depicts the constructed modeling framework. The model learns entity and relationship embedding representations in the encoder through three modules: entity semantic aggregation, entity structural aggregation, and explicit relationship aggregation modules. The model merges the final embedding representa-

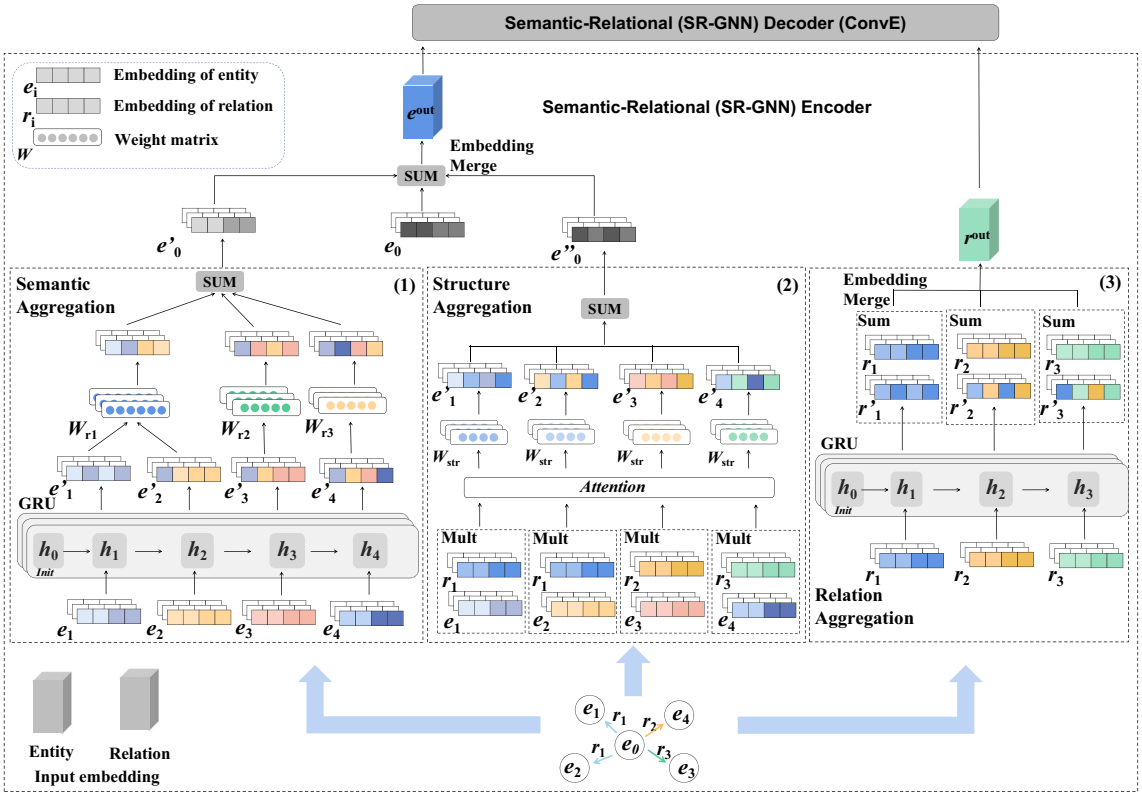

Fig. 4 The SR-GNN model uses an encoder-decoder structure. Within the encoder, the model models entity relationships at three levels: the entity semantic aggregation module, the entity structural aggregation module, and the explicit relationship aggregation module. In addition, the entities and relational embeddings are transferred to the decoder to complete the KGC task

tions of the entities in the semantic and structural aggregation stages. As shown in steps 3-6 of Algorithm 1, these merged entity embeddings and relationship representations serve as the final outputs of the encoder. As shown in steps 7 and 8 of Algorithm 1, the model uses the output of the decoder as the original input for the decoder to complete the KGC task. The encoder part contains three components.

(1) An entity semantic aggregation module that captures the semantic similarity information between neighboring entities using an RNN. As shown in Fig. 4(1) and steps 1-4 of Algorithmic 2, this module uses an RNN to learn semantic similarity information between the neighborhood entities under the same central entity. It can effectively learn the dependencies among the neighboring entities, enabling the acquisition of the semantic information between adjacent nodes. Subsequently, this module transmits the acquired neighborhood information to the central entity based on different relationship types.

(2) An entity structure information aggregation module that captures the structural information of the input KG using a GNN. As depicted in Fig. 4(2) and steps 5-7 of Algorithmic 2, this module uses a GNN to learn the combined embeddings of relationships and entities. This enables the central node to capture information on relationships and entities during the information aggregation phase, enhancing the formed structural representation by incorporating relational information.

(3) An explicit relational aggregation modules. As shown in Fig. 4(3) and steps 8-10 of Algorithmic 2, incorporating relational embeddings into the aggregation process effectively enhances the relational representations and improves the ability of the model to understand the semantics between different types of relations. Finally, ConvE is chosen as the decoder to merge the final embedding representations of the entities in the semantic and structural aggregation stages to form the original inputs of the decoder. Additionally, the relationship representations serve as the original inputs for the decoder to complete the KGC task. Next, these model components are described in detail (Table 2).

3.2 Entity semantic aggregation

In the semantic aggregation module, the model aggregates the semantic information among all neighboring entities connected to the same entity. First, the semantic information among the neighboring entities is learned using an RNN, which is well suited for learning long-term dependencies in

Table 2 Table of specialized terms

Acronyms	Explication
KGE	Knowledge Graph Embedding
KGC	Knowledge Graph Completion
GNN	Graph Neural Network
RNN	Recurrent Neural Network
GRU	Gated Recurrent Unit
SR-GNN	Semantic and Relational-based Graph Neural Network
Notation	Explanation
$\boldsymbol{G}$	Knowledge graphs
$\boldsymbol{K}$	Number of GNN layers
$\boldsymbol{E}, \boldsymbol{R}, \boldsymbol{T}$	Sets of entities, relation-types, and triplets
$\boldsymbol{h}_r^1$ / $\boldsymbol{h}_u^1$	Initial entity/relation-type features
$\boldsymbol{h}_u, \boldsymbol{h}_r, \boldsymbol{h}_v$	Embedding of subject entity, relation-type, and object entity
$\boldsymbol{e}_u, \boldsymbol{r}, \boldsymbol{e}_v$	Subject entity, relation, and object entity
$\boldsymbol{r}_i$	Embedding of the corresponding relationship of the i-th node
$\boldsymbol{\alpha}_{uv}$	Aggregation attention
$\boldsymbol{m}_u$	The aggregated representation of those neighbors
$\boldsymbol{N}_u$	Relation-path based neighbors
$\boldsymbol{W}$	Transformation weight matrix
$\boldsymbol{X}$	Input feature vectors of RNN
$\boldsymbol{H}$	Hidden layer feature of RNN
$\boldsymbol{R}$	Reset gate feature of GRU
$\boldsymbol{Z}$	Update gate feature of GRU
Matrix	Dimension
$\boldsymbol{W}_r$	The dimension of the weight matrix for the relationship type is $\mathcal{R}^{d\times d}$
$\boldsymbol{W}_{str}$	The weight matrix dimension of the structure aggregation module is $\mathcal{R}^{d\times d}$
$\boldsymbol{W}_{rel}$	The weight matrix dimension of the relation aggregation module is $\mathcal{R}^{d\times d}$
$\boldsymbol{m}_u^{sem}$	The semantic level entity feature dimensions are $\mathcal{R}^d$
$\boldsymbol{m}_u^{str}$	The struture level entity feature dimensions are $\mathcal{R}^d$
$\boldsymbol{m}_r$	The explicit relation feature dimensions are $\mathcal{R}^d$

data. The RNN learns the dependencies between the current node and the nodes at the following moment. It selectively passes the hidden semantic features of the previous entity to the next entity for the purpose of learning the semantic similarity between the entities. By doing so, the module can learn information concerning the semantic similarities between different nodes in the graph. In addition, a relation-specific transformation is introduced, setting a weight matrix $\boldsymbol{W}_r \in R^{d\times d}$ for each type of relationship and aggreging the nodes with the same type of relationship via a simple linear

Algorithm 1 The SR-GNN Process.

Input: Training set $S = \{(h, r, t) \in T\}$, entities and relation-type. sets $\forall \boldsymbol{e} \in E$ and $\forall \boldsymbol{r} \in R$. Embedding dim h_{dim}. Number of GNN-layers $\{L \mid L = (1, 2, \ldots, K-1)\}$.

initialize $\boldsymbol{h}_u^1 \leftarrow$ get param for each entity $\boldsymbol{e} \in E$
  $\boldsymbol{h}_r^1 \leftarrow$ get param for each entity $\boldsymbol{r} \in R$
for $L = 1, 2, \ldots, K-1$ **do**
  $\boldsymbol{h}_u^L \leftarrow \boldsymbol{h}_u^L + \left(\boldsymbol{m}_u^{sem}\right)^L + \left(\boldsymbol{m}_u^{str}\right)^L$
  $\boldsymbol{h}_r^L \leftarrow \boldsymbol{h}_r^L + \boldsymbol{m}_r^L$
end for
$\boldsymbol{r} \leftarrow \boldsymbol{h}_r^K$ // the K-layer relational vector acts as the encoder output vector(decoder input vector)
$\boldsymbol{h}$ *or* $\boldsymbol{t} \leftarrow \boldsymbol{h}_u^K$ // the K-layer entity vector acts as the encoder output vector(decoder input vector)
for $(h, r, t) \in T$ **do**
  $\psi_\tau(\boldsymbol{h}, \boldsymbol{t}) \leftarrow \text{ConvE}(\boldsymbol{h}, \boldsymbol{r})$
  $\mathbf{p} \leftarrow \text{Sigmoid}\,(\psi_r(\boldsymbol{h}, \boldsymbol{t}))$
end for
Update embeddings w.r.t $-\frac{1}{N}\sum_i (t_i \cdot \log(p_i) + (1-t_i)\cdot \log(1-p_i))$

layer:

$$\left(\boldsymbol{m}_u^{sem}\right)^{L+1} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{(h_v,h_r)\in\mathcal{N}_u}\frac{1}{c_{i,r}}\boldsymbol{W}_r^{(L)}RNN(\boldsymbol{h}_v^L)\right), \tag{5}$$

$\boldsymbol{m}_u^{sem} \in R^d$ is the representation of an entity at the semantic level, where R^d is the hidden dimension. $\boldsymbol{h}_v \in R^d$ is the embedding representation of the tail entity. $(\boldsymbol{h}_v, \boldsymbol{h}_r) \in \mathcal{N}_u$ represents the neighbor index set of the head entity produced under different relationship types $\boldsymbol{r} \in \mathcal{R}$. The normalization constant $c_{i,r}$ is problem-specific and can be predetermined or learned. In addition, the excessive algorithmic complexity that may be induced by excessive relationship weight matrices is solved by introducing the basis vector decomposition method as follows:

$$\boldsymbol{W}_r^L = \sum_{b=1}^{B}\boldsymbol{a}_{rb}^L\boldsymbol{V}_b^L, \tag{6}$$

where $\boldsymbol{a}_{rb}^L \in R^d$ is a learnable weight representation based on a specific relationship. $\boldsymbol{V}_b$ is a set of learnable base vectors.

3.3 Entity structure aggregation

In the structural aggregation module, the model incorporates the representations of adjacent entities and relationships, $\boldsymbol{m}_u^{str} \in R^d$ is a representation of the entities at the structural level, and $(\boldsymbol{h}_v, \boldsymbol{r}_v) \in \mathcal{N}_u$ denotes the set of relationships and tail nodes connected to the head node $\boldsymbol{h}_u$, where $Mult\,(\boldsymbol{h}_v, \boldsymbol{r}_v)$ is the combined multiplicative embedding of

Algorithm 2 Learning entity semantic aggregation and relation representations.

Input: embedding of subject entity h_u , embedding of relation-type h_r.

for $L = 1, 2, \ldots, K-1$ **do**
  $\boldsymbol{h}_v^L \longleftarrow RNN\left(\boldsymbol{h}_v^L\right)$
  $\left(\boldsymbol{m}_u^{\text{sem}}\right)^{L+1} \leftarrow \sum\left(\text{Linear}\left(\boldsymbol{h}_v^L\right)\right)$ // Node aggregation by edge type
end for
for $L = 1, 2, \ldots, K-1$ **do**
  $\left(\boldsymbol{m}_u^{str}\right)^{L+1} \leftarrow \sum\left((Attention)\boldsymbol{W}_{str}^L \text{Mult}\left(\boldsymbol{h}_v^L, \boldsymbol{r}_v^L\right)\right)$
end for
for $L = 1, 2, \ldots, K-1$ **do**
  $\boldsymbol{m}_r^{L+1} \leftarrow Sigmoid\left(\boldsymbol{W}_{rel}^L\left(GRU\left(\boldsymbol{h}_r^L\right)\right)\right)$
end for
return $\boldsymbol{m}_u^{sem}, \boldsymbol{m}_u^{str}, \boldsymbol{m}_r$

the tail node and the relationships. $\boldsymbol{W}_{str} \in R^{d\times d}$ is the associated weight matrix.

$$\left(\boldsymbol{m}_u^{str}\right)^{L+1} = \sigma\left(\sum_{(h_v,r_v)\in\mathcal{N}_u}\left(\boldsymbol{\alpha}_{uv}^{str}\right)^L \boldsymbol{W}_{str}^L Mult\left(\boldsymbol{h}_v^L, \boldsymbol{r}_v^L\right)\right). \tag{7}$$

The attention weight $\boldsymbol{\alpha}_{uv}$ is calculated as follows:

$$\left(\boldsymbol{\alpha}_{uv}^{tri}\right)^L = \frac{\exp\left(Mult\left(\boldsymbol{h}_v^L, \boldsymbol{r}_v^L\right)^T \boldsymbol{h}_u^L\right)}{\sum_{(h_k,r_k)\in\mathcal{N}_u}\exp\left(Mult\left(\boldsymbol{h}_k^L, \boldsymbol{r}_k^L\right)^T \boldsymbol{h}_u^L\right)}. \tag{8}$$

3.4 Explicit relationship aggregation

The (9) inputs the relational representation $\boldsymbol{h}_r \in R^d$ through a GRU to determine the updated relationship embedding representation:

$$\boldsymbol{m}_r^{L+1} = \sigma\left(\boldsymbol{W}_{rel}^L\left(GRU\left(\boldsymbol{h}_r^L\right)\right)\right), \tag{9}$$

$\boldsymbol{m}_r \in R^d$ is the updated relationship representation, where d is the hidden dimension. $\boldsymbol{h}_r^L$ denotes the embedding of the relation in layer L, $GRU(\bullet)$ is the gated recurrent neural network-based transfer process, and the detailed GRU procedure is shown in (2)-(4).

3.5 Scoring functions

To implement the KGC task, this paper uses a GNN-based encoder for computing entity and relationship representations. In addition, it uses ConvE as the decoder to compute a score for each triad, as shown in (10), which represents the ConvE scoring function:

$$\psi_r\left(\boldsymbol{h}_u, \boldsymbol{h}_v\right) = f\left(\text{vec}\left(f\left(\left[\overline{\boldsymbol{h}_u}; \overline{\boldsymbol{h}_r}\right] * \omega\right)\right)\boldsymbol{W}\right)\boldsymbol{h}_v, \tag{10}$$

$$p = \sigma\left(\psi_r\left(\boldsymbol{h}_u, \boldsymbol{h}_v\right)\right), \tag{11}$$

where $\boldsymbol{h}_u \in R^d$ and $\boldsymbol{h}_v \in R^d$ denote the embeddings of the subject entity and the object entity derived from the encoder output, respectively. $\boldsymbol{h}_r$ denotes the relational embeddings of the fact triples. The convolution procedure and the convolution kernel are represented by the symbols * and ω, respectively. vec indicates an operation that projects the feature map into a vector representation, and $\boldsymbol{W}$ is a mapping matrix that maps the splicing vectors of the relations and head entities into entity embeddings. The sigmoid function is utilized to train the model parameters of the scoring function (8), and to facilitate the algorithmic description, in this paper, (12) is simplified as follows:

$$\begin{aligned}\psi_r(\boldsymbol{h}, \boldsymbol{t}) &= \text{ConvE}(\boldsymbol{h}, \boldsymbol{r}),\\ p &= \text{Sigmoid}\left(\psi_r(\boldsymbol{h}, \boldsymbol{t})\right).\end{aligned} \tag{12}$$

With the above scoring function, it is possible to determine whether the given fact triples are true or false and minimize the following binary cross-entropy loss:

$$L(p, t) = -\frac{1}{N}\sum_i\left(t_i \cdot \log\left(p_i\right) + \left(1 - t_i\right) \cdot \log\left(1 - p_i\right)\right), \tag{13}$$

where t_i represents a label and $|N|$ is the total number of factual triads. When a triplet is true, t_i is 1; otherwise, t_i is 0. p_i is the score of the ith triplet.

3.6 Analysis of SR-GNN

According to the detailed description of the model, the SR-GNN can achieve the following goals: 1) The SR-GNN deeply explores the close semantic associations between neighboring entities, capturing the most direct semantic information contained in graph data. By integrating the semantic and structural features of graph data, the SR-GNN effectively addresses the problem of mixed entity information caused by the single norm employed in traditional embedding algorithms. This significantly improves the inferential explanatory power of the model. 2) Explicitly incorporating relationships into the graph aggregation process enhances the sources of inference information and strengthens the semantic associations between entities. This approach addresses the semantic feature losses and decreased inference accuracies of traditional GNN algorithms due to their lack of relationship

features modeling. However, the SR-GNN still has the following limitations.

Algorithm complexity The SR-GNN aggregates the entity and relationship features from an entire set of graph data. The temporal complexity of this approach is heavily influenced by the utilized dataset. When using larger datasets, longer training times may be observed. Specifically, K represents the number of layers in the model, d is the embedding dimensionality, B represents the basis, and $|R|$ denotes the total number of relationship types in the input graph. The semantic aggregation module aggregates the entity semantic information produced under different types of relationships using basis vector decomposition, which incurs a complexity level of $\mathcal{O}\left(\mathcal{B}Kd^2 + \mathcal{B}K|\mathcal{R}|\right)$. The structural aggregation module and explicit relationship aggregation module use a single GNN to aggregate graph information, requiring a complexity level of $\mathcal{O}\left(Kd^2\right)$. Therefore, the complexity of the SR-GNN is $\mathcal{O}\left(\mathcal{B}Kd^2 + \mathcal{B}K|\mathcal{R}| + 2Kd^2\right)$.

Verbalized expression This paper adopts a simple combination method for neighboring nodes, directly concatenating entity embedding features. This combination method may weaken the semantic correlations between entities to some extent. As shown in Fig. 1, directly concatenating the *Academy Award* and *Everything Everywhere All at Once* entities and feeding them into the GRU weakens some semantic correlations. Conversely, combining them into a complete sentence (e.g., The actor from "Everything Everywhere All at Once" won an *Academy Award* for their vivid performance.) would greatly enhance the role of semantics in the model reasoning process.

4 Experiments

In this section, various experiments are conducted to answer the following questions for validating the contributions of the model:

- RQ1: Compared with other baseline KGE methods, how does the SR-GNN perform (Section 4.2)?
- RQ2: How do the different modules (i.e., the node semantic aggregation, structural aggregation and explicit relationship aggregation modules) affect the SR-GNN (Section 4.3)?
- RQ3: Does the use of different RNNs (an RNN, LSTM, and a GRU) have an impact on the entity semantic extraction process (Section 4.4)?
- RQ4: How do the hyperparameters affect the resulting model performance (Section 4.5)?

4.1 Setup

Datasets In this work, the SR-GNN model is tested on four datasets, i.e., FB15k-237, WN18RR, WN18 and YAGO3-10. FB15k-237 consists of real-world named entities and their relationships. WN18RR and WN18 consist of English phrases and their corresponding semantic relationships. YAGO3-10 describes the attributes of people, such as their citizenship statuses, genders, and occupations. These datasets are acquired from the real world, contain rich semantic information, and are suitable for a model such as the SR-GNN that explores semantics. The statistical information concerning these datasets is summarized in Table 3. The four datasets mentioned above are KG datasets that are commonly used for evaluating KGC models. When creating experimental setups for these datasets, the same division ratio as that used in the original datasets is commonly employed. A subset of the Freebase dataset, namely, FB15k-237, contains 14,541 entities and 237 relations. This dataset is divided into 272,115 training triples, 17,535 validation triples, and 20,466 tests triples. A subset of WordNet, namely, WN18RR, includes 40,943 entities and 11 relationships. The dataset is divided into 86,835 training triples, 3,034 validation triples, and 3,134 test triples. Similarly, WN18 is a subset of WordNet that contains the lexical relations between words. It includes 40,943 entities and 18 relationships. The dataset is divided into 141,442 training triples, 5,000 validation triples, and 5,000 test triples. A subset of YAGO3, namely, YAGO3-10, contains 123,182 entities and 37 relations. The dataset is divided into 1,079,040 training triples, 5,000 validation triples, and 5,000 test triples.

Evaluation indicators To evaluate the SR-GNN method, the following classic complementary rank-based KG evaluation metrics are used in this paper.

(1) MRR: This metric sorts the energy of each correct triple in the entire KG. Its values are obtained by averaging the inverse of the sorted ordinal numbers. The formula for calculating the MRR is as follows:

$$\mathrm{MRR} = \frac{1}{|S|}\sum_{i=1}^{|S|}\frac{1}{\mathrm{rank}_i} = \frac{1}{|S|}\left(\frac{1}{\mathrm{rank}_1} + \frac{1}{\mathrm{rank}_2} + \ldots + \frac{1}{\mathrm{rank}_{|S|}}\right). \tag{14}$$

Table 3 Statistics for the dataset used in this paper

Datasets	Entities	Relations	Train	Valid	Test
FB15K-237	14,541	237	272,115	17,535	20,466
WN18RR	40,943	11	86,835	3,034	3,134
WN18	40,943	18	141,442	5,000	5,000
YAGO3-10	123,182	37	1,079,040	5,000	5,000

(2) H@n: This metric sorts the energy levels of the correct triples in the KG. It represents the percentage of triads with ordinal numbers that are less than n after sorting. The definition of this measure is as follows:

$$\text{HITS } @n = \frac{1}{|S|}\sum_{i=1}^{|S|}\mathbb{I}\left(\text{ran}\, k_i \leqslant n\right). \tag{15}$$

Baseline To enhance the credibility of our experiment, we choose multiple models from each category for comparison rather than solely comparing our method with a single type of KGC algorithm. This approach ensures that our experimental results are comprehensive and reliable. The SR-GNN is compared with 17 baselines based on different methods, which are briefly described as follows.

- TransE [7]: TransE is a classic translation-based model that encodes node and relationship vectors by projecting the nodes and relationships into the representation space and using translation representation relationship information in the Euclidean space.
- RotatE [9]: RotatE projects the representation vector into the complex space and uses a rotation operation to represent the relational information for encoding nodes and relationships.
- PaiRE [10]: PaiRE uses pairs of relation vectors to automatically adapt to the margin γ in the employed loss function to fit complex relations and is also able to encode different relation patterns.
- DistMult [12]: DistMult models multirelational graphs using a bilinear neural network and calculates the likelihood that triple facts will hold by matching the vector space semantics of relationships and entities.
- ComplEx [13]: ComplEx utilizes an asymmetric scoring function to represent how relationships and entities are represented in complex space.
- ConvE [16]: ConvE splices the triad head entity of a triple and relationship vectors and performs feature extraction via a CNN to model the triad.
- TuckER [14]: TuckER performs Tucker decomposition on two-dimensional tensors that represent triplet facts.
- InteractE [15]: This approach captures more interactions between nodes and relationships through feature displacement, splicing operations, and cyclic convolution.
- KG-BERT [18]: Using the descriptions of entities and relationships as inputs, the KGC task is completed by fine-tuning the BERT model.
- MTL-KGC [34]: MTL-KGC is an effective multitask learning method for overcoming the problem that KG-BERT fails to fully learn the relational information contained in a KG.
- StAR [19]: This method combines graph structure coding with text coding to comprehensively use structure and text information to complete KGC tasks.
- R-GCN [20]: The R-GCN uses a GCN to process the impacts of many kinds of relationships on the nodes in a graph.
- SACN [24]: This convolutional network is sensitive to end-to-end graph structures, and its encoder is a WGCN, which takes node structures, node attributes, and relationship types as inputs. The decoder is Conv-TransE, which makes it is possible to retain the translation attributes among the nodes and relationship embeddings.
- CompGCN [21]: CompGCN jointly learns nodes and relationship embeddings in a multirelationship graph.
- LTE [40]: This approach investigates the role of GCNs in KGC. The authors proposed a straightforward yet effective framework called LTEKGE, which combines existing KGE models with linearly transformed entity embeddings.
- (Yu Li et al. 2023) [35]: This paper presented a neighborhood reordering model with relational constraints for KGC tasks and automatically generated specific constraints as extra type characteristics to augment entity representations rather than relying on manually labeled labels.
- GreenKGC [41]: This article introduced GreenKGC, a novel KGC method for low-dimensional KGs. The method includes three modules-representation learning, feature screening and decision learning modules-and achieves better results than those of other low-dimensional methods under 32-dimensional conditions by combining feature screening and decision learning.

4.2 Overall Results and Analysis (RQ1)

This section offers a thorough assessment and analysis of the SR-GNN, analyzing the MRR and H@n metrics obtained on FB15k-237 and WN18RR datasets. Table 4 summarizes the comparative results produced by the SR-GNN and the existing KGE models, presenting the experimental results obtained using RNNs and GRUs in semantic aggregation models. The best results are bolded, while the second-best results are underlined. This paper uses a GRU to construct the semantic aggregation model of the SR-GNN for reasons that are explained in detail in Section 4.4. The visualization results are shown in Fig. 5, where the SR-GNN exhibits excellent performance compared to that of the rival models. The following conclusions are obtained from the experimental results.

Table 4 shows that the SR-GNN achieves state-of-the-art or competitive results in comparisons with the 17 rival models. On the FB15K-237 dataset, the SR-GNN obtains optimal results in terms of three metrics and competitive results with

Table 4 A summary of the FB15k-237 and WN18RR results

Models	FB15K-237				WN18RR			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
Translational Distance								
TransE†	.279	.198	.376	.441	.243	.043	.441	.532
RotatE	.338	.241	.375	.533	.476	.428	.492	.571
PaiRE	.351	.256	.387	.544	–	–	–	–
Tensor Decomposition								
DistMult†	.281	.199	.301	.446	.444	.412	.470	.504
ComplEx†	.278	.194	.297	.450	.449	.409	.469	.530
ConvE	.325	.237	.356	.501	.430	.400	.440	.520
TuckER†	.358	.266	.394	.544	.470	.443	.482	.526
InteractE	.354	.263	–	.535	.463	.430	–	.528
Text-based								
KG-BERT†	–	–	–	.420	.216	.041	.302	.524
MTL-KGC	.267	.172	.298	.458	.331	.203	.383	.597
StAR	.296	.205	.322	.482	.401	.243	.491	**.709**
GNN-based								
R-GCN	.249	.151	.264	.417	–	–	–	–
SACN	.350	.260	.390	.540	.470	.430	.480	.540
CompGCN	.355	.264	.390	.535	.479	.443	.494	.546
LTE	.355	.264	.389	.535	.472	.437	.485	.544
(Li et al. 2023)	.361	**.270**	.397	.543	.463	.424	.480	.540
GreenKGC	.345	.265	.369	.507	.411	.367	.430	.491
Proposed								
SR-GNN (GRU)	**.361**	.269	**.399**	**.546**	**.490**	**.456**	**.504**	.556
SR-GNN (RNN)	.360	.267	.398	.543	**.490**	.455	.504	.558

Optimal indicators are bolded, and sub-optimal indicators are underlined. † indicates we reproduced results using (Wang et al.2021a) [19], and other results from published papers. Compared with 17 baseline models, our model achieves state-of-the-art or competitive results

respect to one metric. On the WN18RR dataset, the SR-GNN obtains optimal MRR, H@1, and H@3 results and competitive H@10 results. The MRR metric is the most important metric for evaluating the performance of the tested algorithms, and the SR-GNN achieves better results than those of all rival models on both datasets. This demonstrates the importance of the entity semantic aggregation module and the explicit relationship aggregation module proposed in this paper for entity and relationship coding, respectively. In addition, the SR-GNN achieves the best performance in terms of

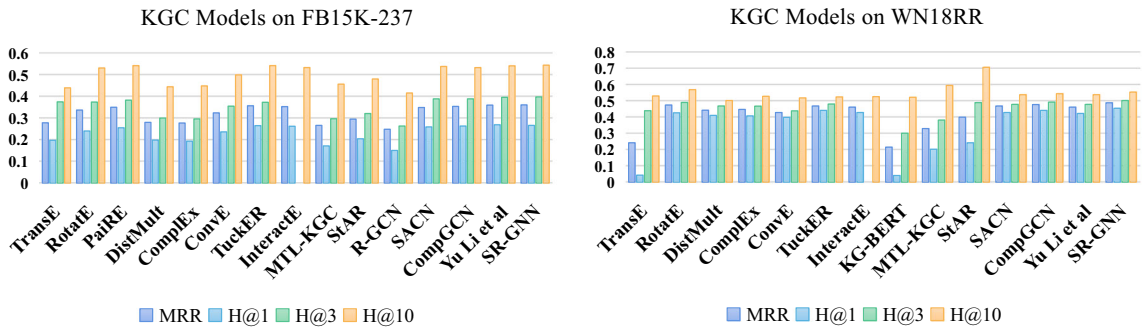

Fig. 5 MRR and Hits@n metrics on FB15K-237, WN18RR datasets. In comparison experiments between SR-GNN and 17 baseline models on FB15K-237 and WN18RR datasets, the experiments show the excellent performance of SR-GNN

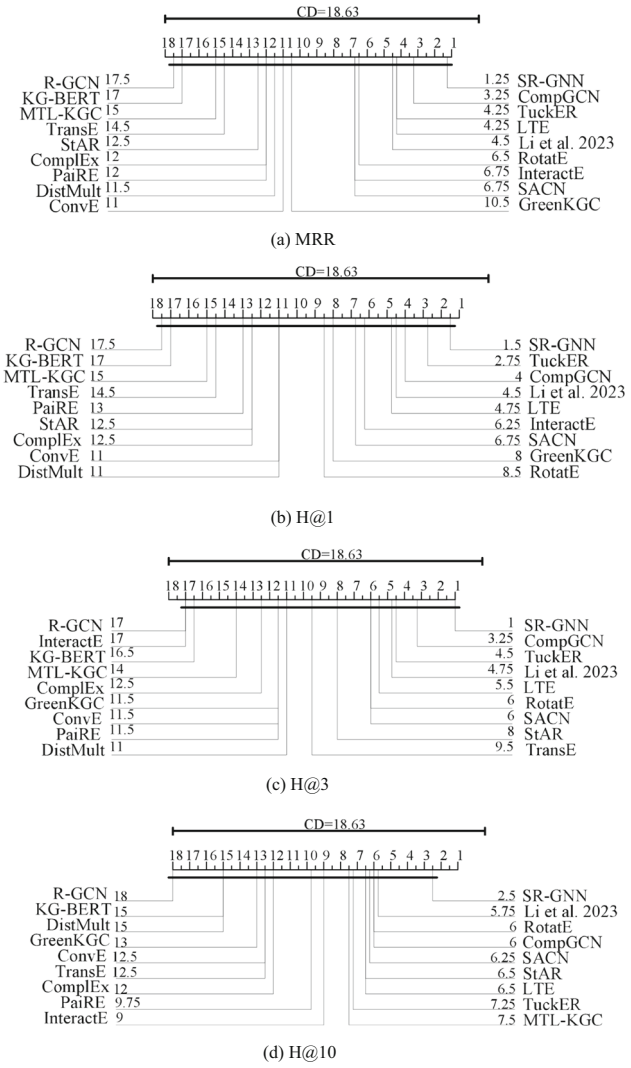

Fig. 6 The Nemenyi test results for SR-GNN performance differences. Average rankings for each algorithm are labeled along the axis (lower rankings on the right). Comparison results under 17 baselines show that SR-GNN significantly outperforms all rival models

the MRR, H@1, and H@3 metrics on the WN18RR dataset. This indicates that the SR-GNN has greater predictive power for higher-ranked entities.

The experimental data show that the SR-GNN provides more significant improvements on the WN18RR dataset than

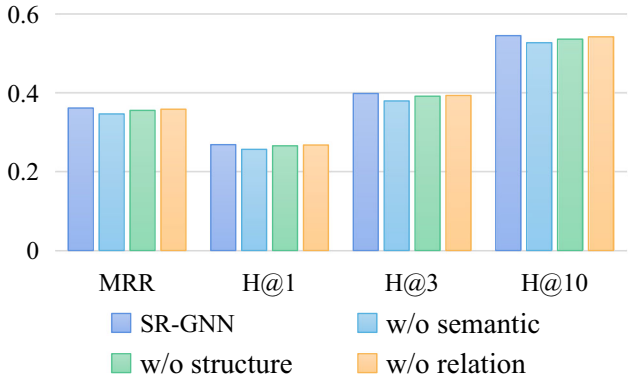

Fig. 7 Experiments with ablation in the FB15K-237 dataset. The horizontal coordinates indicate the four evaluation indicators. Different colors indicate different ablation modules

on the FB15K-237 dataset. The reason for this difference is the degree difference between the datasets. Compared to the FB15K-237 dataset, WN18RR has only 11 relations, which indicates that each head entity connects more tail entities; i.e., its degree is larger. This type of dataset is more suitable for the SR-GNN to model the semantic links between neighboring entities. Because no explicit relational connections are present between the neighboring entities, all such potential semantic associations are not detected by the previously developed models.

The Nemenyi test [42] is performed to compare the overall performance differences among the SR-GNN and the 17 rival models. The average rankings of each algorithm are labeled along the axis (lower rankings are shown on the right). A significant difference between two methods is considered to exist if the difference between their mean ranks reaches a significant critical difference (CD). As shown in Fig. 6, the statistical comparison with the 17 rival models indicates that the SR-GNN has the best performance.

4.3 Ablation experiments (RQ2)

To verify the efficacy of several modules, we conduct ablation experiments on the SR-GNN by analyzing the MRR and H@n metrics obtained on the FB15K-237 and WN18RR datasets. The advantages of the SR-GNN model are ana-

Table 5 Ablation experiments with SR-GNN

Models	FB15K-237				WN18RR			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
SR-GNN	**.361**	**.269**	**.399**	**.546**	**.490**	**.456**	**.504**	**.556**
w/o semantic	.347	.257	.380	.528	.485	.448	.499	.553
w/o structure	.356	.266	.392	.537	.481	.448	.496	.545
w/o relation	.359	.268	.394	.543	.486	.451	.502	.555

Where w/o refers to deleting the associated module from the SR-GNN
The bold values are the best values for each of those measures

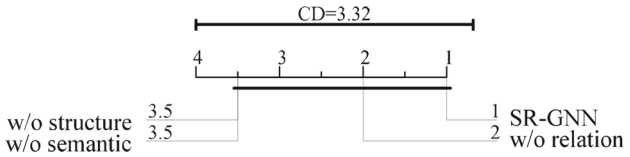

Fig. 8 Critical difference analysis. Significant performance differences between SR-GNN and different ablation modules were analyzed under the **MRR** metric

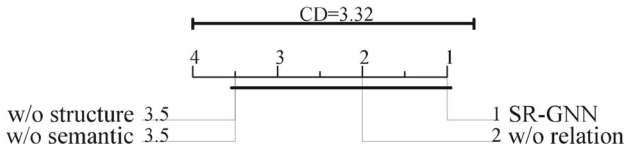

Fig. 10 Critical difference analysis. Significant performance differences between SR-GNN and different ablation modules were analyzed under the **H@3** metric

lyzed from four perspectives: 1) the overall performance of the SR-GNN; 2) the semantic aggregation module; 3) the structural aggregation module; and 4) the explicit relation aggregation module. The specific information is shown in Table 5 and Fig. 7.

Overall Performance of the SR-GNN The Nemenyi test was performed to compare the performance differences between SR-GNN and the other three ablation modules. A significant difference between the two methods is considered to exist if the difference between the mean ranks of the two methods reaches a significant critical difference (CD). Average rankings for each module are labeled along the axis (lower rankings on the right). As shown in Figs. 8, 9, 10 and 11, the statistical comparison with the other three ablation modules indicates that SR-GNN has the best performance.

Semantic Aggregation Module (w/o semantic) The w/o semantic part shows that when the semantic entity aggregation module is removed from the model, and the ablation results significantly decrease. The MRR and H@10 values decrease to 0.347 (a decrease of 4.03%) and 0.528 (a decrease of 3.4%), respectively, on FB15K-237. The experimental results demonstrate that employing a GRU to capture the semantic similarity information between neighboring entities in the semantic aggregation module is effective. By integrating the similar semantics between entities, the information possessed by neighboring entities under different relationships can be more efficiently passed to the central entity.

Structural Aggregation Module (w/o structure) The second ablation experiment, i.e., w/o structure, shows that when the entity structure aggregation module of the SR-GNN is removed, the outcomes of the ablation model decrease. The MRR and H@10 decrease to 0.356 (a decrease of 1.4%) and 0.537 (a decrease of 1.6%), respectively, on FB15K-237. The experimental results show that it is important to use a GNN to learn the combined embedding of relations and entities so that the central node can capture information about them. The structural representation of graph data is effectively enhanced by introducing relational information.

Explicit Relation Aggregation Module (w/o relation). In the third ablation experiment (without relation), we remove the explicit relationship aggregation module from the SR-GNN. To ensure the completeness of the algorithm, we use a relationship vector that is randomly initialized and not updated. The outcomes of the ablation model decrease, and the MRR and H@10 metrics decline to 0.359 (a decrease of 0.5%) and 0.543 (a decrease of 0.5%), respectively. The experimental results demonstrate that explicitly updating the relation embeddings to enhance the feature representations of the relations is beneficial. This improvement aids the model in understanding the semantic differences between different types of relations.

4.4 Impact of using different models (an RNN, LSTM, and a GRU) on entity semantic extraction (RQ3)

This section analyses the metrics obtained on the FB15K-237 dataset, where RNN variants are employed to capture the semantic similarity information between neighboring entities; the tested methods are assessed in terms of how they affect the performance of the resulting model.

As illustrated in Table 6, compared with that of the GRU model, the MRR of the RNN model slightly decreases to 0.360 (a decrease of 0.1%) compared to using GRU. The H@10 results of the model using RNN is reduced to 0.543 (a decrease of 0.3%) compared to that obtained using the GRU. The H@3 result of the model using LSTM is reduced to

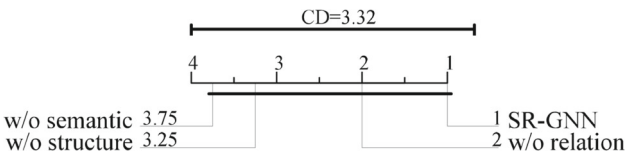

Fig. 9 Critical difference analysis. Significant performance differences between SR-GNN and different ablation modules were analyzed under the **H@1** metric

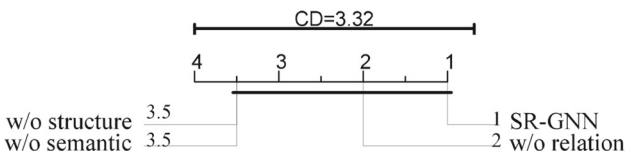

Fig. 11 Critical difference analysis. Significant performance differences between SR-GNN and different ablation modules were analyzed under the **H@10** metric

Table 6 Semantic information between neighboring entities is captured using different models (a RNN, LSTM, and a GRU)

Models	FB15K-237			
	MRR	H@1	H@3	H@10
SR-GRU	.361	.269	.399	.546
lSR-RNN	.360	.267	.398	.543
3lSR-LSTM	.360	.268	.395	.543

0.395 (a decrease of 0.4%) compared to that obtained using the GRU.

All three types of RNN models, namely, the RNN, LSTM, and GRU, exhibit the ability to learn dependencies among the current and next entities. They selectively transfer hidden semantic features from the previous entities to the subsequent entities, thereby facilitating the learning of the semantic similarities between neighboring entities. However, in this study, the GRU is ultimately selected for semantic information extraction for two primary reasons: 1) the GRU has a better semantic transfer effect than do the RNN and LSTM; and 2) the GRU possesses fewer parameters and higher operational efficiency than the LSTM.

The excellent performance achieved by the SR-GRU in terms of all the metrics demonstrates the importance of using the GRU to aggregate the entity semantics. Particularly, the excellent performance achieved by the SR-GRU with respect to the H@n metric indicates its superior predictive ability for entities.

4.5 Impact of hyperparameters (RQ4)

As illustrated in Tables 7 and 8, this section analyzes the effects of the hyperparameters on the SR-GNN. The effects of different hyperparameters on the model are experimentally determined on the FB15K-237 and WN18RR datasets.

Batch Size (b). Fig. 12 and Fig. 13 illustrate the effects of different batch sizes b on the model. The SR-GNN achieves the best results on FB15K-237 when b is 1024, and it achieves the best results on WN18RR when b is 256. On FB15K-237, a batch size of 1024 yields the best outcome, which is 0.1% greater than that produced when the batch size is 512 and 0.1% greater than that obtained when the batch size is 2048. The evaluations of the experiments suggest that the hyperparameters specified in this study are optimal.

Embedding Dimensionality (d). Fig. 14 and Fig. 15 illustrate the effect of different embedding dimensions d on the model. When d is 300, the SR-GNN achieves optimal results on FB15K-237 and WN18RR. Specifically, compared to an embedding dimensionality of 300, setting the dimensions to 200 and 400 results in decreases 0.343 (a decrease of 1.8%) and 0.360 (a reduction of 0.1%) on FB15K-237.

Table 7 The results of using various hyperparameter settings

Models	FB15K-237			
	MRR	H@1	H@3	H@10
Embedding Dimension				
SR-GNN(d = 300)	**.361**	**.269**	**.399**	**.546**
SR-GNN(d = 200)	.343	.254	.376	.519
SR-GNN(d = 400)	.360	.267	.397	.546
GNN Layers				
SR-GNN(K = 2)	**.361**	**.269**	**.399**	**.546**
SR-GNN(K = 3)	.357	.265	.393	.539
RNN Layers				
SR-GNN(L = 2)	**.361**	**.269**	**.399**	**.546**
SR-GNN(L = 1)	.360	.268	.398	.545
SR-GNN(L = 3)	.360	.267	.397	.545
Batch Size				
SR-GNN(b = 1024)	**.361**	**.269**	**.399**	**.546**
SR-GNN(b = 512)	.360	.266	.397	.546
SR-GNN(b = 2048)	.360	.269	.398	.544

According to the findings of the experiment, the SR-GNN performs best in the following parameters: b = 1024, d = 300, K = 2, and L = 2
The bold values are the best values for each of those measures

GNN Layers (K). The effects of different numbers of GNN K on the model are evaluated. The SR-GNN achieves the best results on FB15K-237 when K is 2, and it achieves the best results on WN18RR when K is 1. The reason for this

Table 8 The results of using various hyperparameter settings

Models	WN18RR			
	MRR	H@1	H@3	H@10
Embedding Dimension				
SR-GNN(d = 300)	**.490**	**.456**	**.504**	**.556**
SR-GNN(d = 200)	.479	.441	.493	.555
SR-GNN(d = 400)	.482	.447	.496	.549
GNN Layers				
SR-GNN(K = 1)	**.490**	**.456**	**.504**	**.556**
SR-GNN(K = 2)	.293	.211	.336	.445
RNN Layers				
SR-GNN(L = 2)	**.490**	**.456**	**.504**	**.556**
SR-GNN(L = 1)	.485	.447	.502	.554
SR-GNN(L = 3)	.485	.449	.501	.555
Batch Size				
SR-GNN(b = 256)	**.490**	**.456**	**.504**	**.556**
SR-GNN(b = 128)	.487	.451	.504	.554
SR-GNN(b = 512)	.485	.449	.499	.556

According to the findings of the experiment, the SR-GNN performs best in the following parameters: b = 256, d = 300, K = 1, and L = 2
The bold values are the best values for each of those measures

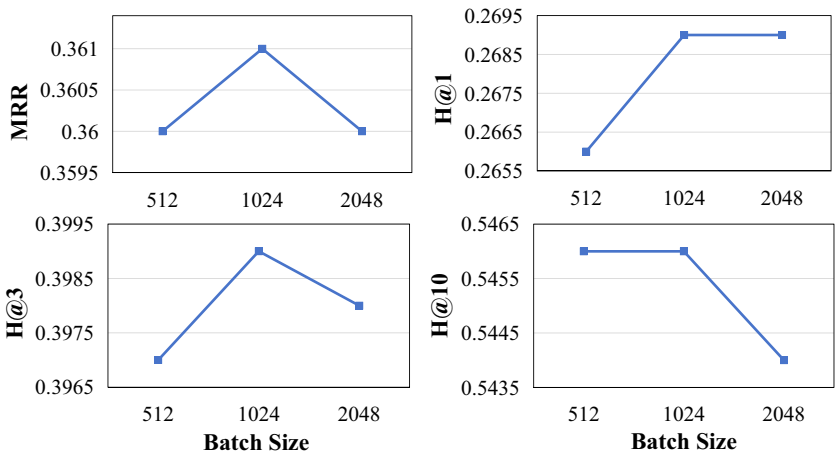

Fig. 12 Impact of different batch size b on SR-GNN performance on the FB15K-237 dataset

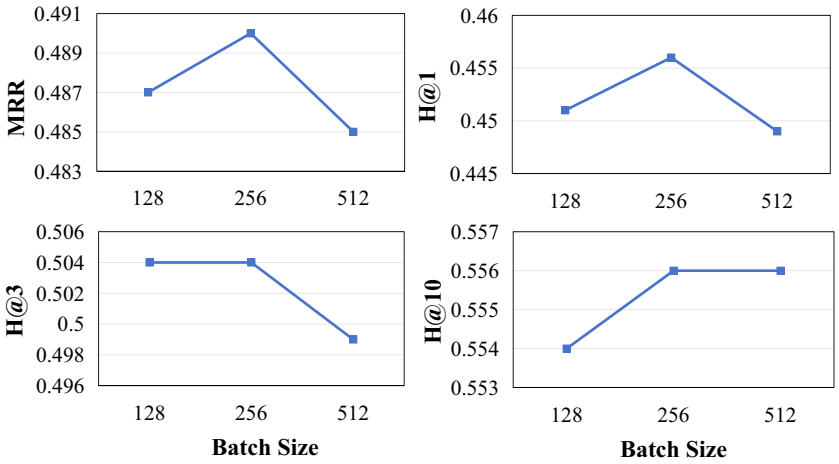

Fig. 13 Impact of different batch size b on SR-GNN performance on the WN18RR dataset

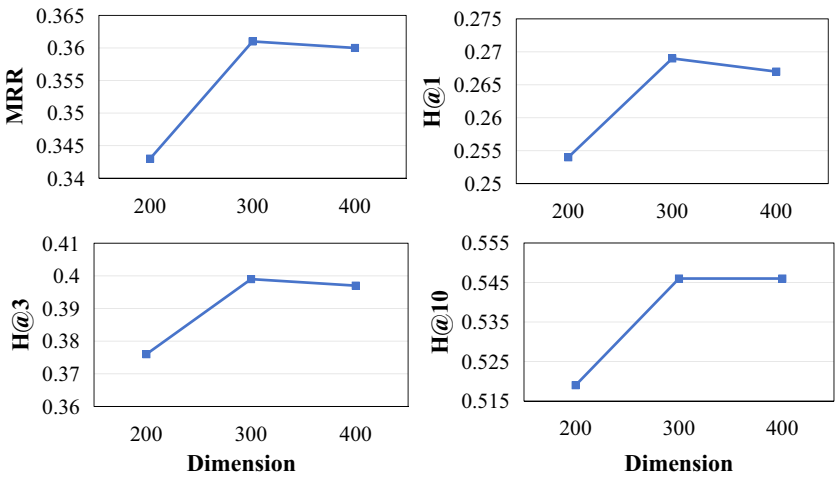

Fig. 14 Impact of different dimension d on SR-GNN performance on the FB15K-237 dataset

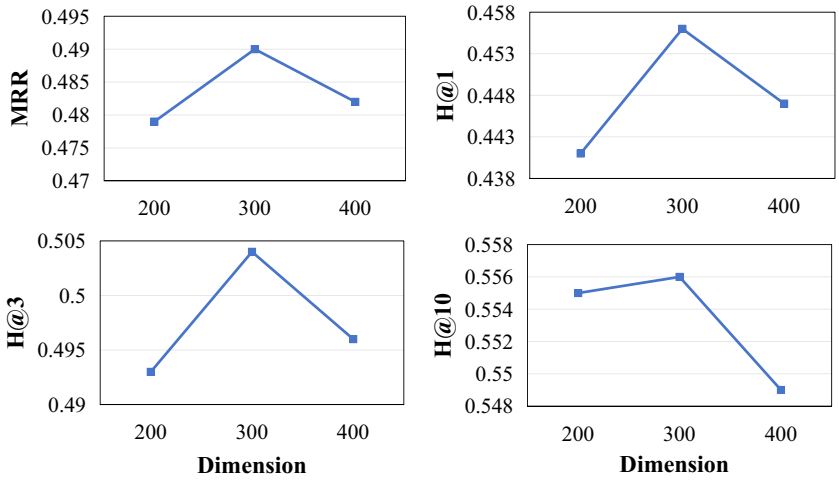

Fig. 15 Impact of different dimension d on SR-GNN performance on the WN18RR dataset

difference is the size of the dataset. Larger datasets require more aggregation layers. On the FB15K-237 dataset, the performance of the SR-GNN decreases to 0.357 when the value of K increases from 2 to 3, which is 0.4% lower than the performance achieved when K is 2.

GRU Layers (L). Fig. 16 and Fig. 17 illustrate the effects of different numbers of GRU layers L on the model. When L is 2, the SR-GNN achieves optimal results on FB15K-237 and WN18RR. When the number of GRU layer is set to 2, the model achieves a result of 0.361, compared to results of 0.360 (a decrease of 0.1%) and 0.360 (a reduction of 0.1%) produced when the number of layers is set to 1 and 3, respectively.

4.6 Convergence and stability experiments

In this subsection, we analyze the convergence and stability of the SR-GNN algorithm.

Figure 18 represents the convergence trends of the four models with respect to the MRR, H@1, H@3, and H@10 evaluation metrics. Three GNN-based KGC models are selected as the baselines for comparison purposes. The performance of the SR-GNN is best under all numbers of epochs. In addition, compared with the other three models, the SR-GNN converges in approximately 100 epochs and is the fastest converging model.

In addition, we conduct an in-depth experimental study on FB15K-237 and WN18RR to evaluate the algorithmic stability of the SR-GNN. To ensure the reliability of our results, we perform 5 repetitions of the experiments on each dataset. In these experiments, we use four different evaluation metrics to measure the performance of the model and demonstrate the stability of each evaluation metric through the mean deviation measure. The mean deviation represents the degree of stability of each metric over 5 experimental repetitions. The experimental results are shown in Table 9.

4.7 Generalizability experiment

In this subsection, we apply the SR-GNN to two datasets, WN18 and YAGO3-10, to evaluate the generalizability and

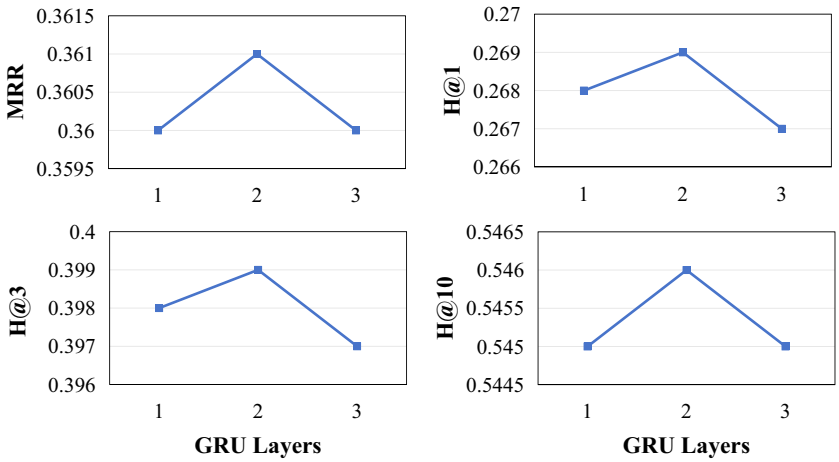

Fig. 16 Impact of different GRU layers on SR-GNN performance on the FB15K-237 dataset

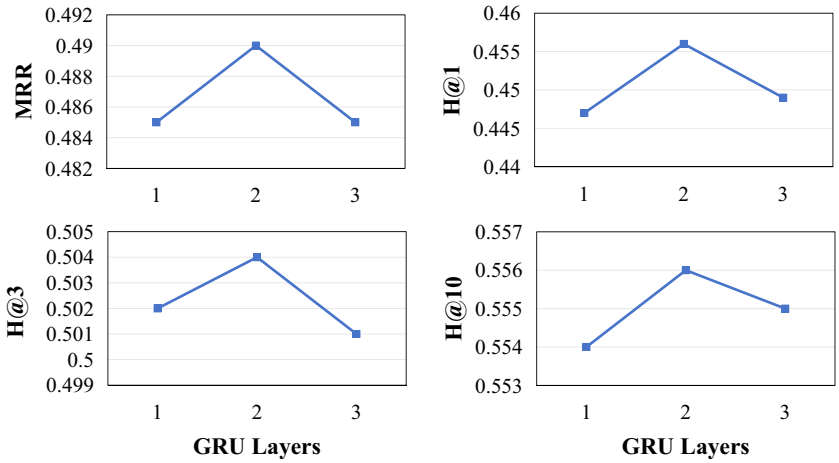

Fig. 17 Impact of different GRU layers on SR-GNN performance on the WN18RR dataset

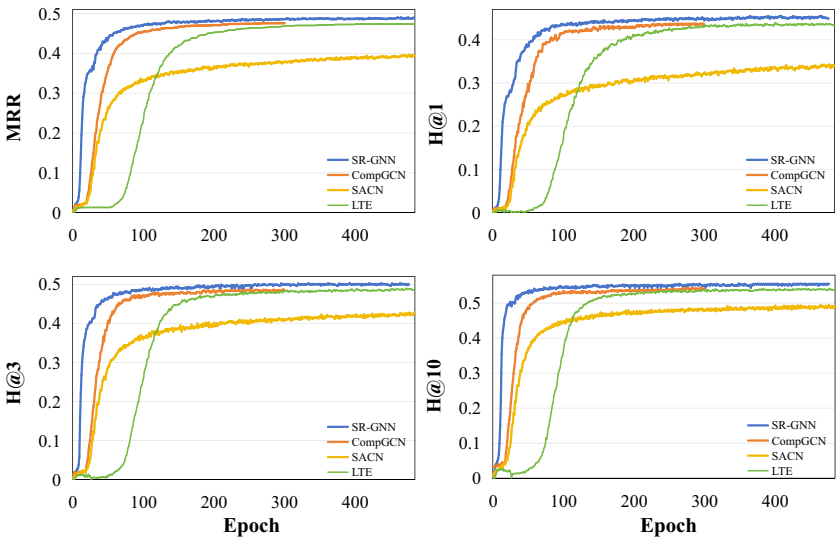

Fig. 18 Convergence analysis of SR-GNN, CompGCN, SACN and LTE algorithms on WN18RR dataset

Table 9 Stability analysis of SR-GNN on the FB15K-237 and WN18RR datasets

Models	FB15K-237			
	MRR	H@1	H@3	H@10
CompGCN	.351 ± .002	.260 ± .001	.385 ± .001	.532 ± .002
SACN	.338 ± .003	.249 ± .003	.372 ± .002	.511 ± .004
LTE	.353 ± .001	.263 ± .002	.383 ± .001	.535 ± .001
SR-GNN	**.361 ± .003**	**.299 ± .002**	**.399 ± .002**	**.546 ± .003**
	WN18RR			
CompGCN	.474 ± .004	.437 ± .005	.487 ± .003	.544 ± .003
SACN	.415 ± .002	.370 ± .002	.437 ± .001	.504 ± .003
LTE	.473 ± .002	.437 ± .003	.488 ± .003	.543 ± .002
SR-GNN	**.490 ± .001**	**.456 ± .001**	**.504 ± .002**	**.556 ± .002**

The bold values are the best values for each of those measures

Table 10 SR-GNN comparison results on the WN18 dataset

Models	WN18			
	MRR	H@1	H@3	H@10
TransE*	.821	.713	.930	.955
TransH*	.823	.721	.929	.954
TransD*	.822	.722	.926	.956
DistMult*	.810	.694	.922	.949
ConvE*	.942	.935	.947	.955
ComplEx**	.813	.701	.921	.943
R-GCN**	.819	.697	.929	.964
CompGCN	.936	.924	.946	.957
SR-GNN	**.947**	**.938**	**.955**	**.961**

Results marked ** are taken from Peng et al. [43]. Results marked * are taken from Michael Schlichtkrull et al. [20]
The bold values are the best values for each of those measures

robustness of the model to other datasets. The dataset details are shown in Table 3.

Tables 10 and 11 present the generalizability and robustness exhibited by the SR-GNN on the WN18 and YAGO3-10 datasets. The experimental results show that the SR-GNN achieves significant results in terms of all four metrics.

These results demonstrate the applicability and robustness of the SR-GNN model on different datasets. By incorporating extensive data, the performance of the model and its ability to handle real-world scenarios can be better assessed.

4.8 Interpretability experiment

In this subsection, we illustrate the significant contributions made by semantics and relationships in KGC tasks by directly extracting multiple real cases from the FB15K-237 dataset. The specific cases are shown in Tables 12 and 13, allowing for a more intuitive demonstration. We discuss the results from the semantic and relation perspectives separately.

(1) Semantic Level. Table 12 presents specific examples of the semantic similarity aggregation process. A high level of semantic similarity is observed between the neighboring entities. By fusing the semantic information among the neighbors and adopting the topological structure of the input graph, semantic information can be transmitted to the central entities effectively enhancing the graph reasoning capabilities of the model. Specifically, for the incomplete triple (*Atlanta*, *city with dogs/dog breed*, ?) → *German Shepherd Dog*, if the model observes from the training set that the dog breeds in Houston are *German Shepherd Dogs* and *Labrador Retrievers*, then these two breeds possess high semantic similarity. In this scenario, if the model observes that the dog breed in *Atlanta* is a *Labrador Retriever*, it aids the model in inferring that another dog breed in *Atlanta* is a *German Shepherd Dog*, even if the model has never seen this information in the training set. Additionally, the other two cases demonstrate that the semantic similarity aggregation process remains beneficial for model inference purposes even when the relationship types between the data are different. For the reader's convenience, Fig. 19 shows a visualization of the semantic similarity process.

(2) Relation Level. Table 13 presents specific examples demonstrating the relationship aggregation process. Entities at the tail ends of relationships of the same type exhibit greater similarity. Incorporating relationships into the aggregation process facilitates the model inference procedure. Specifically, for the incomplete triple (*Sandra Bernhard*, *profession*, ?) → *Actor-GB*, if the model observes from multiple training set data that being an actor is a common *profession* in the UK, this helps the model infer that Sandra Bernhard's *profession* is also an actor in the UK.

By incorporating the semantic and relationship aggregation modules into the SR-GNN and utilizing the GNN to aggregate the entity and relationship information of the input graph, the model can more accurately infer the tail entities.

Table 11 SR-GNN comparison results on the YAGO3-10 dataset

Models	YAGO3-10			
	MRR	H@1	H@3	H@10
TransE*	.501	.406	–	.674
RotatE*	.495	.402	.550	.670
InteractE*	.541	.462	–	.687
DistMult*	.501	.413	–	.661
ConvE*	.488	.399	–	.658
GreenKGC	.453	.361	.509	.629
SR-GNN	**.543**	**.439**	**.603**	**.743**

Results marked * are taken from Tran et al. [44]
The bold values are the best values for each of those measures

4.9 Efficiency analysis of the SR-GNN algorithm

This section compares the training time of the SR-GNN with those of different rival models on the FB15K-237 and WN18RR datasets. Based on the experimental results (Table 14), the following conclusions can be drawn.

1) The SR-GNN outperforms different competitive models in terms of all four metrics. This shows that the SR-GNN is a more powerful KGC algorithm than the competing approaches. 2) On the WN18RR dataset, the SR-GNN can produce the best metrics among all the models in a shorter amount of time, which suggests that our model has a faster convergence rate. 3) The time overhead of the SR-GNN is similar to those of the competitive models on the FB15K-237 dataset. However, the SR-GNN possesses better performance metrics.

Table 12 The case of semantic similarity aggregation processes on the FB15K-237 dataset

Incomplete Triple	Train Set Cases
(Atlanta, city with dogs/dog breed, ?) → German Shepherd Dog	
	(Houston, city with dogs/dog breed, German Shepherd Dog
	(Houston, city with dogs/dog breed, Labrador Retriever)
	(Atlanta, city with dogs/dog breed, Labrador Retriever)
(William Shatner, actor/performance, ?) → DodgeBall: A True Underdog Story	(Vince Vaughn, award/nominated for, DodgeBall: A True Underdog Story)
	(Vince Vaughn, marriage/type of union, Marriage)
	(William Shatner, marriage/type of union, Marriage)
(David J, nationality, ?) → United Kingdom	(Ballet Shoes, country of origin, United Kingdom)
	(Ballet Shoes, common/webpage, Official Website)
	(David J, common/webpage, Official Website)

The first column represents the triple to be completed, while the second column represents the training examples observed by the SR-GNN

Table 13 The case of relation aggregation processes on the FB15K-237 dataset

Incomplete Triple	Train Set Cases
(Blues-rock, artists, ?) → John Lennon	(Experimental rock, artists, John Lennon)
	(Art rock, artists, John Lennon)
	(Soft rock, artists, John Lennon)
(Sandra Bernhard, profession, ?) → Actor-GB	(Masako Nozawa, profession, Actor-GB)
	(Cliff Richard, profession, Actor-GB)
	(Corey Feldman, profession, Actor-GB)
(Vinod Khanna, languages, ?) → Punjabi language	(Raj Kapoor, languages, Punjabi language)
	(Sunny Deol, languages, Punjabi language)
	(Raj Kapoor, languages, Punjabi language)

The first column represents the triple to be completed, while the second column represents the training examples observed by the SR-GNN

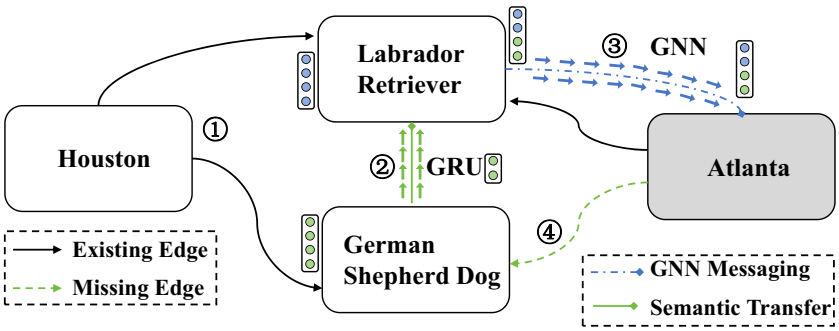

Fig. 19 Semantic propagation process. ① Based on the graph structure, selectively choose neighboring entities with similar semantic meanings; ② Utilize GRU to transfer entity semantic information, facilitating semantic interaction between the two entities; ③ Leverage GNN to propagate semantic features to the central entity; ④ Predict tail entities based on learned semantics

Table 14 Comparison of training duration

Models	FB15K-237					
	MRR	H@1	H@3	H@10	Training Time	Epoch
CompGCN	.355	.264	.390	.535	1.45×10^4	300
SACN	.350	.260	.390	.540	7.65×10^4	1000
LTE	.355	.264	.389	.535	8.04×10^3	500
GreenKGC	.345	.265	.369	.507	3.54×10^4	2000
SR-GNN	**.356**	**.269**	**.494**	**.545**	$\mathbf{2.33 \times 10^4}$	**300**
WN18RR						
CompGCN	.479	.443	.494	.546	1.16×10^4	300
SACN	.470	.430	.480	.540	2.36×10^4	1000
LTE	.472	.437	.480	.540	1.53×10^4	500
GreenKGC	.411	.367	.430	.491	–	–
SR-GNN	**.480**	**.455**	**.494**	**.547**	$\mathbf{6.86 \times 10^3}$	**200**

Time required for SR-GNN to outperform the best metrics of rival models on the FB15K-237 and WN18RR datasets
The bold values are the best values for each of those measures

5 Conclusion

This paper studies KGE models from the perspectives of the semantic information contained in neighborhood entities and explicit relation aggregation. The results show that both proposed entity aggregation modules, namely, the entity semantic aggregation module and the entity structure aggregation module, play significant roles in fusing entity semantic and graph structure information, respectively. Simultaneously, when performing relational aggregation, a GRU learns relationship embedding representations and applies them to link prediction tasks. Based on these findings, we create a novel SR-GNN model that consistently performs better than the competing methods on various datasets. More models may be employed in the future to represent more precise KGE models by capturing the semantic information between entities or relationships.

Additionally, the SR-GNN is adept at efficiently handling large-scale KG data. Learning the semantic similarities between entities and the differences among their relationships enables the inference and analysis of complex relationship networks. 1) In e-commerce, the SR-GNN can enhance user experiences and platform revenue by providing more accurate personalized recommendations through user behavior and item attribute learning. 2) In the health care sector, the SR-GNN can analyse the relationships between drugs by considering factors such as their chemical structures, action mechanisms, and side effects, thereby enhancing the understanding of drug interactions and improving treatment outcomes.

Acknowledgements This work is partly supported by the National Natural Science Foundation of China under grants 62306100 and 62176085, and the Natural Science Research Project of Anhui Educational Committee under grant 2023AH052180. The authors are equally grateful to the Hefei University Arithmetic Platform for providing support.

Author Contributions Xinlu Li: Conceptualization, Methodology, Supervision. Yujie Tian: Software, Conducting experiments, Writing - Original draft preparation. Shengwei Ji: Reviewing, Investigation and Editing.

Availability of data and materials The datasets and materials used during this study are available by following the links in the text.

Code Availability The Python code can be obtained by contacting the author (Yujie Tian).

Declarations

Conflicts of interest The authors declare that they have no conflict of interest.

References

1. Yu M, Jiang T, Yu J, Zhao M, Guo J, Yang M, Yu R, Li X (2023) Sepake: a structure-enhanced and position-aware knowledge embedding framework for knowledge graph completion. Appl Intell 53(20):23113–23123
2. Ta HT, Rahman ABS, Majumder N, Hussain A, Najjar L, Howard N, Poria S, Gelbukh A (2023) Wikides: A wikipedia-based dataset for generating short descriptions from paragraphs. Inf Fusion 90:265–282
3. Formica A, Taglino F (2023) Semantic relatedness in dbpedia: A comparative and experimental assessment. Inf Sci 621:474–505

4. Zeb A, Saif S, Chen J, Yu JJ, Jiang Q, Zhang D (2024) Cope: Composition-based poincaré embeddings for link prediction in knowledge graphs. Inf Sci 662:120197
5. Wang J, Li W, Liu W, Wang C, Jin Q (2023) Enabling inductive knowledge graph completion via structure-aware attention network. Appl Intell 53(21):25003–25027
6. Jiang D, Wang R, Yang J, Xue L (2021) Kernel multi-attention neural network for knowledge graph embedding. Know-Based Syst 107188
7. Bordes A, Usunier N, Garcia-Duran A, Weston J, Yakhnenko O (2013) Translating embeddings for modeling multi-relational data. Proceedings of advances in neural information processing systems (NIPS), pp 2787–2795
8. Bai L, Li N, Li G, Zhang Z, Zhu L (2024) Embedding-based entity alignment of cross-lingual temporal knowledge graphs. Neural Netw 172:106143
9. Sun Z, Deng Z, Nie J, Tang J (2019) Rotate: Knowledge graph embedding by relational rotation in complex space. In: 7th International conference on learning representations (ICLR)
10. Chao L, He J, Wang T, Chu W (2021) Pairre: Knowledge graph embeddings via paired relation vectors. Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (ACL-IJCNLP), pp 4360–4369
11. Nickel M, Tresp V, Kriegel H (2011) A three-way model for collective learning on multi-relational data. Proceedings of the 28th international conference on machine learning (ICML), pp 809–816
12. Yang B, Yih SW-t, He X, Gao J, Deng L (2015) Embedding entities and relations for learning and inference in knowledge bases. In: Proceedings of the international conference on learning representations (ICLR)
13. Trouillon T, Welbl J, Riedel S, Gaussier É, Bouchard G (2016) Complex embeddings for simple link prediction. Proceedings of the 33nd international conference on machine learning (ICML), pp 2071–2080
14. Balazevic I, Allen C, Hospedales TM (2019) Tucker: Tensor factorization for knowledge graph completion. Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP), pp 5184–5193
15. Vashishth S, Sanyal S, Nitin V, Agrawal N, Talukdar PP (2020) Interacte: Improving convolution-based knowledge graph embeddings by increasing feature interactions. Proceedings of the AAAI conference on artificial intelligence (AAAI), pp 3009–3016
16. Dettmers T, Minervini P, Stenetorp P, Riedel S (2018) Convolutional 2d knowledge graph embeddings. Proceedings of the thirty-second AAAI conference on artificial intelligence (AAAI), pp 1811–1818
17. Daza D, Cochez M, Groth P (2021) Inductive entity representations from text via link prediction. Proceedings of the web conference 2021 (WWW), pp 798–808
18. Yao L, Mao C, Luo Y (2019) Kg-bert: Bert for knowledge graph completion. arXiv preprint arXiv:1909.03193
19. Wang B, Shen T, Long G, Zhou T, Wang Y, Chang Y (2021) Structure-augmented text representation learning for efficient knowledge graph completion. Proceedings of the web conference 2021 (WWW), pp 1737–1748
20. Schlichtkrull MS, Kipf TN, Bloem P, Berg R, Titov I, Welling M (2018) Modeling relational data with graph convolutional networks. 15th International conference on extended semantic web conference (ESWC), pp 593–607
21. Vashishth S, Sanyal S, Nitin V, Talukdar PP (2020) Composition-based multi-relational graph convolutional networks. In: 8th International conference on learning representations (ICLR)
22. Li R, Cao Y, Zhu Q, Bi G, Fang F, Liu Y, Li Q (2022) How does knowledge graph embedding extrapolate to unseen data: A semantic evidence view. Proceedings of the AAAI conference on artificial intelligence (AAAI), pp 5781–5791
23. Velickovic P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y (2018) Graph attention networks. In: 6th International conference on learning representations (ICLR)
24. Shang C, Tang Y, Huang J, Bi J, He X, Zhou B (2019) End-to-end structure-aware convolutional networks for knowledge base completion. Proceedings of the AAAI conference on artificial intelligence (AAAI), pp 3060–3067
25. Wang H, Yang J, Yang LT, Gao Y, Ding J, Zhou X, Liu H (2024) Mvtucker: Multi-view knowledge graphs representation learning based on tensor tucker model. Inform Fusion 106:102249
26. Deng W, Li K, Zhao H (2023) A flight arrival time prediction method based on cluster clustering-based modular with deep neural network. IEEE Trans Intell Trans Syst pp 1–10
27. Zhao H, Wu Y, Deng W (2023) An interpretable dynamic inference system based on fuzzy broad learning. IEEE Trans Instrum Meas 72:1–12
28. Zhao H, Liu H, Jin Y, Dang X, Deng W (2021) Feature extraction for data-driven remaining useful life prediction of rolling bearings. IEEE Trans Instrum Meas 70:1–10
29. Zhao H, Liu J, Chen H, Chen J, Li Y, Xu J, Deng W (2023) Intelligent diagnosis using continuous wavelet transform and gauss convolutional deep belief network. IEEE Trans Reliab 72(2):692–702
30. Yang S, Zhang W, Tang R, Zhang M, Huang Z (2022) Approximate inferring with confidence predicting based on uncertain knowledge graph embedding. Infor Sci pp 679–690
31. Wang J, Li W, Liu W, Wang C, Jin Q (2023) Enabling inductive knowledge graph completion via structure-aware attention network. Appl Intell 53(21):25003–25027
32. Zhang D, Rong Z, Xue C, Li G (2024) Simre: Simple contrastive learning with soft logical rule for knowledge graph embedding. Inf Sci 661:120069
33. Devlin J, Chang M, Lee K, Toutanova K (2019) BERT: pre-training of deep bidirectional transformers for language understanding. Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: human language technologies, (NAACL-HLT), pp 4171–4186
34. Kim B, Hong T, Ko Y, Seo J (2020) Multi-task learning for knowledge graph completion with pre-trained language models. Proceedings of the 28th international conference on computational linguistics (COLING), pp 1737–1743
35. Li Y, Hu B, Liu J, Chen Y, Xu J (2023) A neighborhood re-ranking model with relation constraint for knowledge graph completion. IEEE/ACM Transactions on Audio, Speech, and Language Processing, pp 411–425
36. Paul GR, Preethi J (2024) A novel breast cancer detection system using SDM-WHO-RNN classifier with LS-CED segmentation. Exp Syst Appl 238(Part C), p 121781
37. Oyewola DO, Akinwunmi SA, Omotehinwa TO (2024) Deep lstm and lstm-attention q-learning based reinforcement learning in oil and gas sector prediction. Knowl-Based Syst 284:111290
38. Akilan T, Baalamurugan K (2024) Automated weather forecasting and field monitoring using gru-cnn model along with iot to support precision agriculture. Exp Syst Appl p 123468
39. Cho K, Merrienboer B, Gülçehre Ç, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation. Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp 1724–1734

40. Zhang Z, Wang J, Ye J, Wu F (2022) Rethinking graph convolutional networks in knowledge graph completion. In: The web conference 2022
41. Wang Y, Ge X, Wang B, Kuo C-J (2023) Greenkgc: A lightweight knowledge graph completion method. In: Proceedings of the 61st annual meeting of the association for computational linguistics (ACL), pp 10596–10613
42. Ji S, Bu C, Li L, Wu X (2023) Localtgep: A lightweight edge partitioner for time-varying graph. IEEE Trans Emerg Top Comput pp 1–12
43. Peng Y, Zhang J (2020) Lineare: Simple but powerful knowledge graph embedding for link prediction. In: 20th IEEE International conference on data mining, ICDM 2020, Sorrento, Italy, pp 422–431. Accessed 17–20 Nov 2020
44. Tran HN, Takasu A (2022) MEIM: Multi-partition Embedding Interaction Beyond Block Term Format for Efficient and Expressive Link Prediction. In: Proceedings of the thirty-first international joint conference on artificial intelligence (IJCAI), pp 2262–2269