Computing and Informatics, Vol. 44, 2025, 717–744, doi: 10.31577/cai_2025_3_717

KNOWLEDGE GRAPH REPRESENTATION LEARNING BY TEXT ENCODING AND GRAPH STRUCTURE

Song Li*, Chengyu Zhong, Liping Zhang

School of Computer Science and Technology
Harbin University of Science and Technology
150080 Harbin, China
e-mail: lisongbeifen@163.com

Abstract. Knowledge graph representation learning aims to embed entities and relationships into low-dimensional space through knowledge graph embedding methods. Because knowledge graphs are incomplete, it is often necessary to complete the knowledge graph through representation learning methods. With the development of pre-trained language models, more and more research applies them to the field of knowledge graph representation learning, using the powerful semantic representation capabilities of pre-trained language models to improve the performance of knowledge graph embedding. Most of the existing methods make use of the semantic information of the triple text but do not fully consider the structural information of the triple and the graph structure information of the knowledge graph. The triple structure reflects the semantic information and relationship pattern of the triple, and the graph structure reflects the surrounding entity's semantic features. To address the above issues, this paper proposes a knowledge graph representation learning method named PREGSE, which is based on pre-trained language models and integrates graph structure information. Firstly, pre-trained language models are employed to encode triplets through text encoding, obtaining vectors for the triplets. Secondly, a graph attention network is utilized to learn various local graph structure information. Lastly, a multi-task learning strategy is applied to simultaneously learn triplet structure information and semantic information. We trained our model on the FB15k-237 and WN18RR datasets, and the results show that on the FB15k-237 dataset, our model improved the MRR metric by 27 % and the Hits@10 metric by 8 % compared to the StAR model. The experiments show that our model can further improve the performance of knowledge graph representation learning.

* Corresponding author

Keywords: Knowledge graph, knowledge graph completion, knowledge graph representation learning, language model, link prediction

Mathematics Subject Classification 2010: 68-T30

1 INTRODUCTION

Knowledge Graph [1] (KG) is a type of directed graph with multiple relations, composed of nodes and edges. The nodes represent entities, such as "Leonardo DiCaprio" and "Titanic", while edges represent the relationships between entities, such as "acted in". Knowledge Graphs store information in a network-like structure, where each piece of knowledge is represented in the form of a triple, including a head entity, a relationship, and a tail entity. Knowledge Graphs have gained widespread applications in various fields, including recommendation systems [2], question answering systems [3], and knowledge inference [4], as an emerging knowledge storage structure. In numerous domains, large-scale Knowledge Graphs like Freebase [5], DBpedia [6], and YAGO [7] have been successfully created and applied in various practical scenarios.

Knowledge graph representation learning [8, 9, 10, 11], also known as knowledge graph embedding (KGE), is a technique that employs machine learning methods to map entities and relationships in a knowledge graph to a low-dimensional vector space. Due to the sparse nature of information within knowledge graphs, Knowledge Graph Completion (KGC) has emerged as a significant research topic in this field. KGC involves utilizing existing knowledge and structure within a KG to infer new knowledge through a series of algorithms and models, thereby continuously enhancing the completeness of the knowledge graph. KGE is a subtask of KGC, where KGE models assess the plausibility of given triples and subsequently add correct triples to the knowledge graph. Specifically, KGE models first map entities and relations into a unified low-dimensional vector space, with each entity and relation corresponding to a low-dimensional vector. The completion of the knowledge graph is then performed by calculating the plausibility of the triples based on their vector representations.

Early methods in knowledge graph representation learning were predominantly centered around the TransE [12] model and its variants. These models assess the plausibility of triplets through a defined scoring function. While these methods are relatively simple and applicable to large knowledge graphs, they only take into account the information within the triplets. Additionally, these models exhibit weak scalability; when encountering entities or relationships not present in the training set, they struggle to accurately represent vectors. Pre-trained language models (PLM), trained on extensive textual data, capture general features of language and inherently contain rich semantic information. Utilizing pre-trained language models as encoders can effectively enhance the scalability of knowledge graph representation

learning models. With the introduction of the Transformer architecture and the development of pre-trained language models, models such as BERT [13], GPT [14], and BART [15] have achieved significant success in the field of natural language processing. In various domains, the introduction of pre-trained language models has become increasingly common to achieve improved results. KG-BERT [16] was the first to apply the BERT model to the task of knowledge graph completion and achieved superior results on the MR metric. However, KG-BERT's scoring function is overly simplistic, lacking modeling of the triplet structure itself. This limitation prevents it from learning triplet structure information, making KG-BERT highly dependent on text representation. When the textual expressions of different entities are similar, KG-BERT faces challenges in correctly identifying the correct triplets. Additionally, during the training process, KG-BERT does not reuse the embedding vectors of triplets, resulting in a very slow inference speed.

Typically, knowledge graphs can be analyzed from two perspectives: the triplet perspective and the graph structure perspective. The triplets in a knowledge graph primarily reflect the semantic information and relationship patterns of the entities involved. Traditional knowledge graph representation learning methods mainly model the triplet structure and semantic information. When analyzed from the graph structure perspective, a knowledge graph can be viewed as a directed graph with multiple relations, where the characteristics of an entity are often influenced by the surrounding nodes. Graph Neural Networks (GNN) have gained significant attention in recent years for their outstanding performance in modeling graph-structured data. GNN possess remarkable capabilities in information aggregation and propagation, enabling efficient handling of graph structure features. They can effectively learn semantic relationships and latent information between nodes in a graph. Due to the inherent graph structure of knowledge graphs, utilizing GNN for knowledge graph processing has significant advantages. GNN can effectively capture node information, relationships between nodes, and structural information within knowledge graphs. Traditional knowledge graph embedding methods merely embed triplets into a low-dimensional space to calculate their plausibility, neglecting the utilization of graph structure information. In knowledge graphs, neighbouring nodes often have close semantic relationships with the target entity. Integrating information from neighbouring nodes can enhance the embedding vector of the target entity, better capturing the semantic features of its surrounding entities. In knowledge graphs, the local graph structure of a node can reveal the implicit semantic information associated with that node. Current representation learning methods based on Graph Neural Networks mainly focus on integrating information from neighbouring nodes but often overlook the structural information expressed by different graph structures.

Based on the above analysis, it is evident that incorporating a Pre-trained Language Model can enhance the model's scalability. However, most current PLM-based approaches tend to overlook the structural information of triples, leading to poor performance when handling similar texts. Therefore, we introduce PLM while simultaneously employing a scoring function based on translation models to score triples,

thereby capturing the structural information of the triples. On the other hand, existing Graph Neural Network-based methods often focus on integrating information from neighbouring nodes, neglecting the implicit structural information within different local structures. To address this, we define two distinct local structures to learn the implicit structural information embedded within them.

In this paper, we propose a representation learning method named PREGSE, which integrates both semantic and graph structure information. The method is designed for knowledge graph completion tasks. The main contributions of this paper are as follows:

- Addressing the scalability issues of traditional representation learning methods and the performance shortcomings of KG-BERT, we leverage the powerful semantic learning capabilities of pre-trained language models. We employ the BERT model to map entities and relationships in the knowledge graph to vector representations, facilitating the learning of deep semantic connections between entities and relationships. This enhancement contributes to an overall improvement in the quality of knowledge graph embedding representations.
- To address the issue of current language model-based representation learning methods neglecting graph structure information, we define two distinct subgraph structures: loop-structure and star-structure. Subsequently, we introduce a triplet-based attention mechanism to capture various graph structure features within the knowledge graph.
- To expedite model convergence and enhance performance, we propose a multi-task learning strategy to assist in the model training process. Training results on several benchmark datasets indicate that our approach achieves state-of-the-art results in both link prediction and triplet classification tasks.

2 LITERATURE

Knowledge graph representation learning is a crucial method for knowledge graph completion. Based on different underlying principles, models can be categorized into four types: translation models, tensor decomposition models, neural network-based methods, and language model-based methods. Translation models and tensor decomposition models are earlier approaches, both of which compute the score of a triple by performing operations on the entities and relations within the triple. The main idea behind translation models is that the relation can transform the head entity vector into the tail entity vector, while tensor decomposition models typically employ multiplicative operators to establish interactions between entity and relation embedding vectors. As a result, many scholars refer to translation models as additive models and tensor decomposition models as multiplicative models. Neural network-based methods introduce neural networks such as convolutional neural networks and graph neural networks into knowledge graph representation learning, leveraging the powerful feature capture capabilities of neural networks to learn the representations

of entities and relations. Language model-based methods have emerged with the rise of large language models, leading to research that incorporates language models into knowledge graph embedding tasks. This approach takes full advantage of the strong learning capabilities of language models, as well as the rich semantic information and knowledge acquired by the models from upstream tasks.

2.1 Translation-Based Models

The translation model is inspired by word vectors and mainly consists of TransE and its derivative models. TransE considers each triple (h, r, t) as a directed transfer in the embedding space from the head entity to the tail entity. By mapping the head entity, tail entity, and relation to a low-dimensional vector space, and modeling the triple using an energy function $h + r \approx t$. The energy function indicates that when a triple holds, the sum of the head entity vector and the relation vector should be close to the tail entity vector. TransE has lower complexity and can be applied to large knowledge graphs. Although TransE performs well in handling one-to-one relationships, it faces challenges in distinguishing entities on the side with multiple entities for more complex one-to-many or many-to-many relationships, as the vectors of entities on the side with more entities become very close and are difficult to differentiate. TransM [17] improves upon TransE by introducing an additional weight matrix, enabling it to handle complex relationship mappings while maintaining a parameter complexity similar to TransE. TransH [18], built upon TransE, introduces the concept of hyperplanes. Each relation corresponds to the normal vector of a hyperplane, and by using these normal vectors, the embedding vectors of the head and tail entities are projected onto the respective hyperplanes. This enables TransH to handle cases involving reflexive and complex relationship mappings. TransR [19] argues that both TransE and TransH embed entities and relations into the same semantic space. However, an entity may have multiple aspects, and each relation may focus on different aspects. In TransR, each relation corresponds to a relation space, which includes a relation matrix M_r and an embedding vector r for this relation space. The embedding vectors of the head and tail entities are mapped into the relation space through the relation matrix. TransD [20] is an improvement based on TransR. In TransR, during the projection, entities are mapped through the mapping matrix corresponding to the relation. However, entities connected by a relation typically have various types and attributes, and the diversity of entities should be considered during the projection process. On the other hand, in TransR, the mapping operations for the head and tail entities are obtained through matrix multiplication, resulting in a problem of excessive parameters and computational complexity. In TransD, each entity and relation are associated with two vectors: an embedding vector and a projection vector. The embedding vector represents the meaning of the entity or relation, while the projection vector indicates how to embed the entity into the vector space of the relation. This way, the projection matrices M_{rh} and M_{rt} can be jointly determined by both the relation and the entity. Moreover, matrix operations can be replaced by vector operations, addressing

the issue of excessive parameters and computational complexity. RotatE [21] views the transformation from the head entity to the tail entity as a rotation operation in the complex plane space. This approach can capture more relational features, such as direction, symmetry, antisymmetry, and combination. Compared to other translation-based models, RotatE achieves the best performance in link prediction tasks.

2.2 Tensor Decomposition Models

The tensor decomposition model treats the knowledge graph as a three-dimensional tensor, where each third-order tensor corresponds to a triple, and the tensor values represent the likelihood of the triple's existence. RESCAL [22], also known as a bilinear model, considers a triple (h, r, t) fundamentally as binary relational data. RESCAL employs a three-dimensional tensor factorization method to model this binary relationship. If there is a relationship between entities, the corresponding point in the three-dimensional tensor is 1; otherwise, it is 0. DistMult [23] builds upon RESCAL by constraining the relation matrix M_r to a diagonal matrix to reduce complexity. DistMult has the same scalability as TransE and exhibits better performance in link prediction tasks. HolE [24] proposes a holographic embedding method that utilizes circular correlation operations for entity vector interactions. Similar to DistMult, HolE simplifies the computational complexity of RESCAL. However, since circular correlation operations are asymmetric, HolE can model non-symmetric relationships.

2.3 Neural Network-Based Methods

In recent years, there have been many attempts in knowledge graph representation learning to achieve better performance by introducing neural networks, such as convolutional neural networks and graph neural networks. ConvE [25] introduces a convolutional neural network, reshaping the head and tail entities into a two-dimensional matrix, concatenating them, and then inputting them into a CNN for computation. ConvKB [26] merges the embedding vectors of the triples into a $k \times 3$ matrix and then inputs the matrix into a convolutional neural network for computation. ConvR [27] reshapes the embedding vector h of the head entity into a matrix as the input to the convolutional layer. The embedding vector r of the relation is then split and reshaped into a set of convolutional kernels. This convolution operation allows the kernels to interact with each region of the input matrix, capturing the interactions between the head entity and the relation effectively.

R-GCN [28] introduces an Encoder-Decoder framework, associating a weight matrix with each relationship and using Graph Convolutional Networks (GCN) to aggregate neighbor node information. CompGCN [29] considers relationship embeddings, aggregating entities and relationships in the Encoder phase using a compositional approach, and decoding the triples in the Decoder phase using methods like TransE, TransH, or ConvE. MRGAT [30] integrates attention mechanisms in both

entity-level and relationship-level aggregation processes, hierarchically learning entity and relationship embeddings. GATFCN [31] uses the GCN model as an encoder to fuse graph structural information and then decodes the information encoded by GCN using tensor decomposition.

2.4 Language Model-Based Methods

KG-BERT fine-tunes BERT through triple classification. Initially, the triple is concatenated into a sentence and input into the BERT model. The output from the [CLS] position in BERT's results is used as the sequential representation of the triple, projected into the scoring function space. Compared to existing models, KG-BERT achieves relatively good results but still falls short of RotatE, and KG-BERT's inference efficiency is lower than RotatE. Addressing the issues of the explosion of combinations in the KG-BERT model and the inability to learn structured knowledge, StAR divides each triple into two parts. It utilizes a Siamese-style text encoder to encode the relation and head entity together and the tail entity separately. The final representation of the triple is calculated interactively, considering the reasonableness of the encoding. Additionally, since the triple is divided into two parts, graph embedding methods can also be employed to model the triple. Compared to KG-BERT, StAR's entity embeddings are reusable and can learn structured knowledge effectively.

3 METHODOLOGY

This section introduces detailed information on the PREGSE model, which mainly consists of three modules: the triplet encoding module based on pre-trained language models, the graph structure feature fusion module, and the multi-task learning strategy module, where the triplet encoding module is responsible for encoding triplets into initial embedding vectors, the graph structure feature module is responsible for integrating various graph structure information, and the multi-task learning strategy module acts as a decoder using structural methods to model triplets, while also accelerating the convergence speed of the model. The overall architecture of the model is shown in Figure 1. First, BERT is used as an encoder to encode the entities and relations in the triples into embedding vectors, where the orange represents the embedding vector of the relation, and the blue represents the embedding vectors of the head and tail entities. The embedding vectors of the entities are then fed into the graph structure feature module, where two types of local structural information are integrated. Finally, the resulting entity embedding vectors and the relation embedding vectors are input into the translation model-based decoder.

3.1 Notation

The set of triplets in the knowledge graph is denoted as G, and the set of negative triplets is denoted as G'. The set of all entities is represented as E, and the set of all

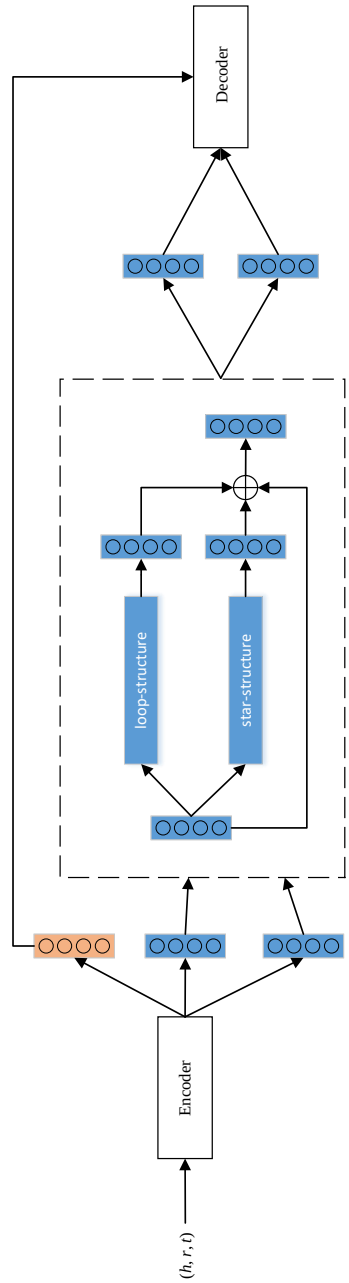


Figure 1. Overall architectural design of the PREGSE

relations is represented as R. Each triplet is represented as (h, r, t), where $h, r \in E$, and $r \in R$. Each negative triplet is represented as (h', r', t'). The embedding vectors for triplets are represented as (e^h, e^r, e^t). Given an entity v, its set of neighbouring nodes is represented as $N(v)$, and the set of triplets formed by entity v and its neighbouring nodes $N(v)$ is represented as G_v. The representation of entity v as a node in the graph is denoted as e_v, and the relationship between entities is denoted as r_i.

3.2 Triplet Encoding with Pre-Trained Language Models

In this section, the primary objective is to map the triplets from the knowledge graph to their corresponding text representations. Subsequently, we employ pre-trained language models to map the entities and relations of the triplets to initial embedding vectors.

During training, KG-BERT requires each triple to be input into the BERT model, and then the hidden state output by BERT is used to determine whether the triple is correct. In the inference process, repeatedly occurring entities and relations are calculated multiple times without effective reuse, leading to the problem of combinatorial explosion. In this paper, we use BERT to encode the text of the triplets. Each part of the BERT output is processed individually to obtain the corresponding vector representations. We then utilize the obtained embedding vectors for structured modeling, thereby enhancing the performance of the model.

As shown in Figure 2, we use BERT for text encoding by concatenating the text representations of the head entity, relation, and tail entity of each factual triplet into a token sequence. The token sequence for the head entity is denoted as $T^h = (t_1^h, t_2^h, \ldots, t_n^h)$, and similarly, the token sequence for the relation is $T^r = (t_1^r, t_2^r, \ldots, t_n^r)$, and the token sequence for the tail entity is $T^t = (t_1^t, t_2^t, \ldots, t_n^t)$. We separate the head entity, tail entity, and relation using [CLS] and [SEP] tokens to construct an input sequence compatible with the BERT model, with the specific format: [CLS]T^h[SEP]T^r[SEP]T^t[SEP], where [CLS] and [SEP] are token markers for model input.

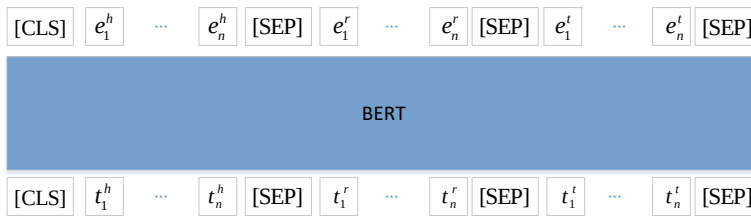


Figure 2. Triples are encoded as primitive embedding vector schematics via BERT

We use the output corresponding to the position of [CLS] as the sequence representation for the entire triplet. The output corresponding to the positions of input

tokens is used as the sequence representation for the head entity, relation, and tail entity in the triplet. Subsequently, each part of the BERT output is projected into the corresponding embedding vectors through average pooling. We denote the obtained embedding vectors for the head entity, relation, and tail entity as e^h, e^r, and e^t, respectively.

3.3 Graph Structure Feature Fusion

In this section, we provide detailed information on the method for integrating graph structure features. The knowledge graph is a network composed of a large number of factual triplets, and different structures in the graph contain distinct semantic information. As shown in Figure 3, we define two types of graph structures: the loop-structure and the star-structure.

Loop-Structure: A loop-structure refers to a configuration where all nodes are connected in a circular manner. Starting from the original node, moving along the edges, and eventually returning to the original node forms a loop-structure. This structure contains inferential information about triplets. For example, through triplets like (Leonardo DiCaprio, father, George DiCaprio) and (George DiCaprio, spouse, Eileen), one can infer the triplet (Leonardo DiCaprio, mother, Eileen). In loop-structure, the number of hops represents the number of triplets involved in the inference.

Star-Structure: A star-structure indicates that all nodes in the graph are directly connected to a central node. In a knowledge graph, a set of triplets sharing the same head entity or tail entity can form a star-structure. By integrating information from neighbouring nodes, the target entity can better understand the contextual semantics. The neighbouring nodes provide contextual information around the target entity.

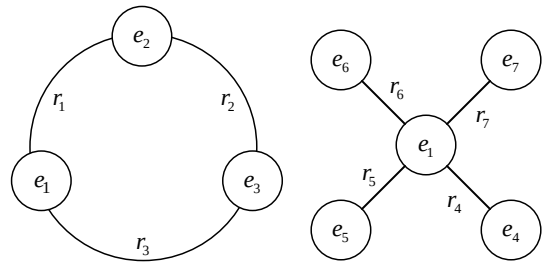


Figure 3. Graph structures

In the PREGSE model, we integrate graph structure information separately from loop-structure and star-structure using Graph Attention Networks (GAT). A typical GAT consists of multiple layers, where each layer aggregates information from

neighbouring nodes and updates the nodes. The final layer's output node vector is the result of multiple layers of propagation, encompassing a broader range of neighbor information. However, this introduces high computational complexity. In this paper, we consider only first-order neighbors and use a single-layer GAT to aggregate graph structure information, aiming to reduce the model's computational complexity.

3.3.1 Attention Mechanism Based on Triplets

We design an attention mechanism to enable entities to more effectively integrate graph structure information. For a given entity v in the knowledge graph, the set of neighbouring entities $N(v)$ can form a triplet set G_v. Since the vector representations of each triplet in G_v can express different and varying amounts of information, the attention coefficients when fusing neighbor node information for entity v are often highly correlated with the vector representations of the triplets. For example, in the triplets (Leonardo DiCaprio, profession, actor) and (Leonardo DiCaprio, acted in, Titanic), the second triplet contains more information, not only about acting in a movie but also implicitly implies information about the profession.

In summary, when calculating the attention weight coefficients between two entities, consideration should be given to the information in the embedding vectors of triplets. We concatenate the embedding vectors of the head entity, tail entity, and relation, followed by a linear transformation to represent the embedding vector of the triplet. Finally, we use softmax for normalization. The following formula is used to compute the attention weight coefficients between entities:

$$e^{(h,r,t)} = W_1 \cdot \text{concat}\left(e^h, e^r, e^t\right), \tag{1}$$

$$\alpha = \sigma\left(\text{LeakyRELU}\left(e^{(h,r,t)}\right)\right), \tag{2}$$

where e^h, e^r, e^t represent the embedding vectors of the head entity, relation, and tail entity in the triplet, respectively. $e^{(h,r,t)}$ represents the embedding vector of the triplet. The operation concat denotes concatenation, $W_1 \in \mathbb{R}^{d\times 3d}$ is the linear transformation matrix, LeakyRELU represents the activation function, and σ represents the softmax function.

For a given entity v in the knowledge graph, we analyze its local graph structure through both star-structure and loop-structure. The star-structure can be considered as a set of triplets formed by the entity v and its neighbouring entities. The loop-structure can be viewed as comprising two triplets: one directly connecting entity v to the target entity and another formed by v traversing a series of triplets in an inferential path to reach the target entity. A_{v_i,v_j} indicates the connection relationship between two entities, with a value of 1 when there is a connection and 0 when there is none:

$$A_{v_i,v_j} = \begin{cases} 1, & v_i = v_j \text{ or } v_j \in N_g(vi), \\ 0, & \text{otherwise}, \end{cases} \tag{3}$$

where v_i, v_j represent entities. Value $N_g(v_i)$ represents the set of nodes adjacent to entity v_i under the condition of satisfying the graph structure.

3.3.2 Fusion of Loop-Structural Information

Representation of loop-structure starts from the initial node, progresses along relationships and entities, and eventually returns to the initial node. This structure includes two relation paths: the first path involves multi-hop relationships starting from the head entity, passing through a series of relationships and entities, and reaching the tail entity; the second path is a one-hop relationship directly from the head entity to the tail entity, representing the triplets in the knowledge graph. As illustrated in Figure 3, there are two paths starting from e_1 and reaching e_3: (e_1,r_1,e_2,r_2,e_3) and (e_1,r_3,e_3). These two paths share a considerable degree of semantic relationships.

Learning various representations of paths between two entity nodes can enhance the model's predictive ability for triplets. For multi-hop relational paths, embedding representations of these paths can be obtained through a combinatorial approach, treating relationships and entities along the path as a sequence of words. Current pre-trained language models often struggle to effectively capture the meaning of such word sequences. This limitation arises because pre-trained language models typically rely on the compositionality of words, and the combination of different words may deviate significantly from the meaning of each individual word, lacking compositionality. Therefore, we employ Long Short-Term Memory (LSTM) Recurrent Neural Networks to combine multi-hop paths, obtaining the embedded representation of the combined relationships. Subsequently, the embedded representation of the combined path is used as the final relationship embedding vector.

The process of identifying loop-structures in a knowledge graph is a breadth-first search process, where the results are represented using a relational path. The processing of the relational path is shown in Algorithm 1. The input to the algorithm includes the entity node e and the set of neighbouring nodes N for all nodes, while the output is a list of identified relational paths. In steps 2 and 3, two nodes e_2 and e_3 are selected from $N(e)$, respectively. Steps 4 and 5 determine whether e_3 is a neighbouring node of e_2. If it is, the path is added to the path list.

We represent the relationship paths in the form of text sequences, replacing the triplet relationship in BERT's input with the text of the relationship path. Taking the loop-structure in Figure 3 as an example, the token sequence for the relationship path is

$$T^{lr} = \text{concat}(T^{r_1}, T^{e_2}, T^{r_2}), \tag{4}$$

where T^{r_1}, T^{e_2} and T^{r_2} represent sequences composed of tokens corresponding to entities and relationships in the relationship path.

When using T^{lr} as the relationship sequence, the input representation corresponding to the BERT model is

$$[CLS]T^h[SEP]T^{lr}[SEP]T^t[SEP]. \tag{5}$$

Algorithm 1 Algorithm for Processing of Loop-Structure Paths

Input: Entity e, Neighbor Node Set N.
Output: Path list $list_{path}$.

```
list_path ← []
for each e_2 in N(e) do
    for each e_3 in N(e) do
        if e_3 in N(e_2) then
            list_path.add([e_2, r_2, e_3, r_3])
        end if
    end for
end for
return list_path
```

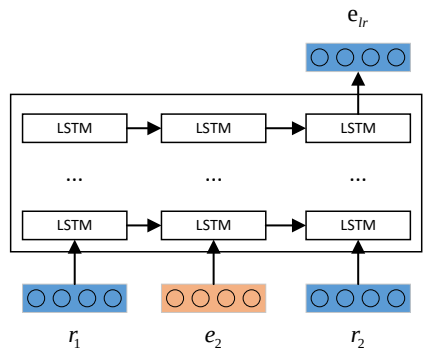


Figure 4. Relational path embedding

As shown in Figure 4, after obtaining the sequential representations of the parts composed of r_1, e_2 and r_2 in the relationship path, we use a unidirectional LSTM to combine the relationship path. Sequential representations corresponding to entities and relationships on the path are sequentially used as inputs to the LSTM. After passing through multiple layers of LSTM, we select the output of the last LSTM unit as the embedding vector for the entire relationship path. The calculation process of the path embedding vector is represented by the following formula:

$$e^{cr} = \mathrm{LSTM}\left(e^{r_1}, e^{e_2}, e^{r_2}\right), \tag{6}$$

where e^{r_1}, e^{e_2} and e^{r_2} represent the embedding vectors of entities and relationships on the relationship path, respectively.

For a given entity e_i, we aggregate a set of triplets, where the path embedding serves as the triplet relationship embedding, using a multi-head attention mechanism. This aggregation allows the target entity to incorporate information from

relationship paths within the loop-structure:

$$e_c^h = \frac{1}{M}\left(\sum_{K=1}^{M}\sigma\left(\sum_{e_j\in N_c(e_i)}\alpha_{ij}^k\cdot\left(W_2\cdot\text{concat}\left(e^{e_i},e^{r_{cij}},e^{e_j}\right)\right)\right)\right), \tag{7}$$

where M represents the number of attention heads, σ denotes the softmax function, e^{e_i} and e^{e_j} are the embedding vectors for the head and tail entities, respectively. $e^{r_{cij}}$ represents the embedding vector for the relationship path or relationship between entities e_i and e_j, $W_2 \in \mathbb{R}^{d\times 3d}$ is the weight matrix, $N_c(e_i)$ represents the set of neighbouring nodes that can form a loop-structure with e_i and α_{ij}^k denotes the attention weight computed based on the triplet.

3.3.3 Fusion of Star-Structural Information

Star-structure is a common graph structure in knowledge graphs, where the relationships between a node (head entity) and its neighbouring nodes (tail entities) can be considered as a star-structure, the star-structure is shown in Figure 3.

Star-structure can provide contextual semantic information for the target entity. To better understand the meaning of the target entity, it is essential to consider this contextual information and integrate it into the embedding representation. We can view the star-structure as a set of triplets composed of nodes and their neighbouring nodes.

For a given target entity e_i, we employ a multi-head attention mechanism with weighted summation to aggregate features from neighbouring nodes:

$$e_c^h = \frac{1}{M}\left(\sum_{K=1}^{M}\sigma\left(\sum_{e_j\in N_c(e_i)}\alpha_{ij}^k\cdot\left(W_3\cdot\text{concat}\left(e^{e_i},e^{r_{ij}},e^{e_j}\right)\right)\right)\right), \tag{8}$$

where M represents the number of attention heads, σ denotes the softmax function, e^{e_i} and e^{e_j} are the embedding vectors for the head and tail entities, respectively. $e^{r_{ij}}$ represents the embedding vector for the relationship between entities e_i and e_j, $W_3 \in \mathbb{R}^{d\times 3d}$ is the weight matrix, $N_s(e_i)$ represents the set of neighbouring nodes that can form a star-structure with e_i and α_{ij}^k denotes the attention weight computed based on the triplet.

3.3.4 Entity Feature Fusion

For the head entity h in a triplet (h, r, t) in the knowledge graph, two embedding vectors, each fused with different graph structure information, can be obtained using the methods described above. By integrating these two embedding vectors with learnable weight parameters and adding the initial vector to avoid losing the entity's original information, the final embedding vector for entity h is calculated as follows:

$$e_f^h = a_c e_c^h + a_s e_s^h + a_h e_h, \tag{9}$$

where a_c, a_s and a_h represent learnable weight parameters, e_h represents the initial embedding vector for the entity. e_h^c and e_h^s represent the final embedding vectors fused with loop and star-structural information, respectively. Similarly, the final embedding vector e_f^t for the tail entity in the triplet can be obtained.

3.4 Multi-Task Learning Strategy

In this section, we propose two tasks to assist model training through a multi-task feature learning approach, thereby improving the learning efficiency of the model. The knowledge representation learning method based on pre-trained language models leverages rich semantic information learned during the pre-training process, making it more convenient to extend to new entities and relationships without being affected by the incompleteness of the knowledge graph. However, this approach overlooks the learning of structural information in the triplet structure, which is crucial in traditional knowledge graph embedding tasks. To address this, we design a multi-task learning strategy for knowledge graph representation learning tasks to jointly learn semantic and structural information from triplets. These tasks share the output vectors from BERT.

3.4.1 Triplet Structure Prediction

The embedding method based on triplet structure learns the structural information of triplets by embedding entities and relationships into a low-dimensional space. However, this approach has limited scalability and may not be applicable to new entities and relationships. Additionally, it could be weakened in predictive ability due to the incompleteness of the knowledge graph.

On the other hand, embedding methods based on pre-trained language models exhibit strong scalability and can be applied to previously unseen entities and relationships. However, they may lack effectiveness in learning triplet structures, potentially leading to issues such as missing structural information and entity ambiguity. Therefore, introducing structural information into the model is necessary.

Inspired by RotatE, we consider the process of mapping the head entity to the tail entity through the relationship in a triplet as a rotation from the head entity to the tail entity in the complex vector space. This approach is capable of modeling various relationship patterns, including symmetric, asymmetric, inversion and composition. For a given triplet (h, r, t), the distance formula is given by:

$$d(h, r, t) = h \circ r - t, \tag{10}$$

where $\circ$ denotes the Hadamard product.

For a given triplet (h, r, t), firstly, we obtain the embedding vectors for each part of the triplet through triplet encoding and feature fusion. Next, we split the head entity, relationship, and tail entity into real and imaginary parts. Finally, we calculate the triplet score using the distance formula.

The specific algorithm is shown in Algorithm 2. The input to the algorithm is the embedding vectors corresponding to the triplet (h, r, t). Steps 2 and 3 calculate the real and imaginary parts based on the embedding vectors of the head and tail entities. while steps 4 and 5 compute the real and imaginary parts of the relationship using sine and cosine functions. Steps 6 and 7 involve calculating scores for the real and imaginary parts through rotation. Finally, steps 8 merge the scoresof the real and imaginary parts to compute the final score.

After calculating the score for a triplet according to the algorithm, we use a loss function similar to cross-entropy to compute the loss:

$$L_1 = -\sum_{t \in G \cup G-} \left(y_t \log \sigma\left(\gamma - s_t\right) + \left(1 - y_t\right) \log \sigma\left(s_t' - \gamma\right)\right), \tag{11}$$

where γ is a fixed margin used to prevent overfitting, σ represents the sigmoid function, s_t is the score for a positive triplet, s_t' is the score for a negative triplet, $y_t \in \{0, 1\}$ indicates the correctness of the triplet.

Algorithm 2 Algorithm for Triple Structure Score

Input: Triple embedding vectors (e_h, e_r, e_t).
Output: Triple score s_t.

1: $pi \leftarrow 3.141592653589$
2: $h_{re}, h_{im} \leftarrow$ split e_h into real and imaginary parts
3: $t_{re}, t_{im} \leftarrow$ split e_t into real and imaginary parts
4: $r_{re} \leftarrow \cos(r)$
5: $r_{im} \leftarrow \sin(r)$
6: $s_{re} \leftarrow r_{re} * t_{re} + r_{im} * t_{im} - h_{re}$
7: $s_{im} \leftarrow r_{re} * t_{im} - r_{im} * t_{re} - h_{im}$
8: $s_t \leftarrow$ norm and sum (s_{re}, s_{im})
9: **return** s_t

3.4.2 Triplet Semantic Prediction

We observed that using a triplet semantic prediction classifier during the training phase can help the BERT model converge faster and achieve better performance. We extract the hidden state e^c corresponding to the [CLS] position in the output sequence of BERT's last layer, considering it as the semantic representation of the entire triplet sequence. Subsequently, we calculate the triplet loss through the triplet classification task. Initially, we introduce a weight matrix to linearly transform e^c, then compute the score for the triplet, and finally use the cross-entropy loss function to calculate the loss:

$$s_r = \sigma\left(W_4 e^c\right), \tag{12}$$

$$L_2 = -\sum_{t \in G \cup G'} \left(y_t \log\left(s_r\right) + \left(1 - y_t\right) \log\left(s_r'\right)\right), \tag{13}$$

where s_r is the score for a positive triplet, s'_r is the score for a negative triplet, $W_4 \in \mathbb{R}^{2\times H}$ represents the weight matrix, σ represents the sigmoid function and $y_r \in \{0, 1\}$ indicates the correctness of the triplet.

3.4.3 Model Training

During the training process, negative examples are typically generated using negative sampling. Specifically, given a triplet (h, r, t), negative triplets are generated by randomly replacing either the head entity or tail entity. The set of negative triplets is represented as:

$$\begin{aligned} G' =& \{(h', r, t) | h' \in E \wedge h' \neq h \wedge (h', r, t) \notin G\} \\ & \cup \{(h, r, t') | t' \in E \wedge t' \neq t \wedge (h, r, t') \notin G\}. \end{aligned} \tag{14}$$

We combine the scores from both tasks to calculate the final score for better training effectiveness. This is because the triplet semantic prediction task directly operates on the output of BERT, which helps the model overcome issues such as gradient vanishing or excessive smoothing, leading to improved performance. The final loss for the model is computed based on the losses obtained from the two tasks as described earlier:

$$L = \lambda L_1 + (1 - \lambda) L_2, \tag{15}$$

where $\lambda \in (0, 1)$ represents the weight of the two losses. When $\lambda = 1$, it indicates that only the structural embedding loss is used, and when $\lambda = 0$, it indicates that only the BERT module is used.

4 EXPERIMENTS

In this section, we evaluate the model using both the link prediction task and the triplet classification task.

4.1 Datasets

We conducted benchmark tests on four datasets to evaluate the performance of our approach (Table 1).

FB15k-237: Due to the presence of a large number of reversible triples in FB15k, there is a risk of leakage between the training and test sets. FB15k-237 is a dataset created from FB15k for link prediction, containing 14,541 entities and 237 relations, which can avoid the leakage issue present in FB15k.

FB13: FB13 is a subset of Freebase, containing 75,043 entities and 13 relations. These entities cover a diverse range of topics, including people, places, events, and works. The 13 relations are among the most representative and widely used in Freebase.

WN18RR: WN18RR is a dataset created from WN18 for the link prediction task, containing 40 943 entities and 11 relations. It ensures that there is no leakage between the training and test sets.

WN11: WN11 is a dataset specifically designed for knowledge graph research, containing 38 696 entities and 11 relations. WN11 is a subset of WordNet and has been refined to facilitate research on specific tasks such as knowledge graph construction, reasoning, and link prediction.

Dataset	Entity	Relation	Train	Dev	Test
FB15k-237	14 541	237	272 115	17 535	20 466
FB13	75 043	13	316 232	5 908	23 733
WN18RR	40 943	11	86 835	3 034	3 134
WN11	38 696	11	112 581	2 609	10 544

Table 1. The summary statistics of the datasets used in the experiments

4.2 Evaluation Metrics

This paper uses three performance metrics for evaluating knowledge graph completion:

Hits@N: Hits@N is a metric used to evaluate the performance of knowledge graph completion models. It focuses on whether the model can correctly find the true entity, relationship, or attribute within the top N candidates. Specifically, for each test sample, the model's predictions are ranked, and it is checked whether the true value is within the top N predictions. If yes, the count is 1; otherwise, the count is 0. Finally, the counts for all test samples are summed and divided by the total number of test samples to obtain the value of Hits@N. Typically, a higher Hits@N value indicates that the model more accurately predicts the true value within the top N candidates.

MR (Mean Rank): MR is a metric used to measure the performance of knowledge graph completion models. It represents the average rank of the entities, relationships, or attributes predicted by the model for each test sample among all possible candidates in the test dataset. Specifically, for each test sample, the model's predictions are ranked, and the average rank across all test samples is computed. A lower MR value is better, as it indicates that the model's predictions are closer to the true values.

MRR (Mean Reciprocal Rank): MRR is a metric used to evaluate the ranking performance of a model, commonly used in information retrieval tasks. For each query, MRR considers the reciprocal rank of the highest-ranked correct answer returned by the model. The calculation involves finding the position of the correct answer in the ranking for each query and taking the reciprocal.

Finally, the average of the reciprocals for all queries is computed, resulting in the MRR value. The MRR value ranges between 0 and 1, where a value closer to 1 indicates better model performance.

4.3 Parameter Settings

The model is implemented based on PyTorch and the Transformers library. BERT-base is used as the base model, with a batch size of 16, a learning rate of 3×10^{-5}, and a linear decay rate set to 0.01. The number of training epochs is set to 5, and the fixed margin $\gamma \in \{6, 9, 12, 24\}$.

4.4 Link Prediction

The link prediction task involves predicting the missing entity in a given triplet (h, r, t) when the head entity is missing (denoted as $(?, r, t)$), or predicting the missing entity when the tail entity is missing (denoted as $(h, r, ?)$). The model is then used to calculate scores for entities in the candidate set as potential missing entities. We use three common evaluation metrics, MR, MRR and Hits@10, to assess the effectiveness of the model in link prediction.

	FB15k-237			WN18RR		
Method	Hits@10	MR	MRR	Hits@10	MR	MRR
TransE*	0.465	347	0.294	0.501	3 384	0.226
DisMult*	0.419	254	0.241	0.490	5 110	0.430
ConvE	0.491	246	0.316	0.480	5 277	0.460
ConvKB*	0.517	257	0.396	0.525	2 554	0.248
RotatE	0.533	177	0.338	0.571	3 340	0.476
KG-BERT	0.420	153	–	0.524	97	–
CompGCN	0.535	197	0.355	0.546	3 533	0.479
BLP-TransE	0.363	–	0.195	0.580	–	0.285
StAR	0.482	117	0.296	0.709	51	0.401
MRGAT	0.539	–	0.355	0.544	–	0.481
PREGSE	0.525	120	0.410	0.714	54	0.440

Table 2. Link prediction results on FB15k-237 and WN18RR

We use PREGSE for the link prediction task on the FB15k-237 and WN18RR datasets. Table 2 presents the results of link prediction on these two datasets. The [*] results are taken from the ConvKB paper, while the results of other baseline models are taken from the original papers. Based on the experimental results, the following conclusions can be drawn:

- Compared to other models, our proposed PREGSE model achieves better results in most metrics, indicating stronger representation learning capabilities on knowledge graphs. It can be observed that our model exhibits more significant

improvements in the Hits@10 metric on the FB15k-237 dataset. This is because the FB15k-237 dataset has more relationship types and fewer entities, leading to a more complex graph with richer structural information. Our model can better learn the graph structure information in such scenarios. The improvement on WN18RR is relatively lower because of its lower graph complexity and relatively similar structures, resulting in fewer graph structure information learned by the model.

- Compared to early translation models and tensor decomposition models, methods based on pre-trained language models show better performance. This is because knowledge graph representation learning methods based on translation models are limited by training only on the triplet structure, making it challenging to distinguish different relationships adequately. In contrast, pre-trained language models can learn rich semantic information from the text, allowing them to exclude many unreasonable triplets. For example, for two triplets (Jack, born in, London) and (Jack, graduated from, London), using the TransE model would likely result in embeddings for the relationships "born in" and "graduated from" being closer in the vector space projection, potentially leading to incorrect results when predicting similar triplets.
- Compared to the KG-BERT model, PREGSE shows a 10.5 % improvement in the Hits@11 metric on the FB15k-237 dataset. On the WN18RR dataset, PREGSE achieves a 19 % improvement in the Hits@10 metric. This is because, relative to the KG-BERT model, PREGSE model calculates the loss using structured methods, giving due consideration to triplet structure information and achieving better results in terms of Hits@N metrics.
- Compared to CompGCN and MRGAT, PREGSE achieves a higher MRR metric on the FB15k-237 dataset. The MRR improvement on the WN18RR dataset is lower due to the limited local structural information in WN18RR, which hinders the full learning of implicit information from different local structures. However, there is still a significant improvement in the Hits@10 metric, as the introduction of the PLM effectively mitigates the lack of local structural information.

4.5 Triplet Classification

The triplet classification task aims to determine whether a given factual triplet is correct, typically treated as a binary classification problem. The decision rule is based on a scoring function with a specific threshold. The structured embedding methods calculate the triplet score, and if the score is below the threshold, the factual triplet is deemed correct; otherwise, it is considered incorrect. We conducted the triplet classification task on two datasets, WN11 and FB13. Table 3 presents the results of the triplet classification task, and the following conclusions can be drawn:

- Compared to traditional knowledge graph embedding models, using text encoding methods yields better results, with a more significant improvement on the WN11 dataset. This is because pre-trained models have already learned rich semantic information in upstream tasks, providing better support for model reasoning.
- Compared to the KG-BERT model, our approach produces slightly better results. This is because the scoring function of the PREGSE model can better learn semantic and structural information, providing enhanced capability in handling complex semantic relationships.

Model	WN11	FB13	Avg
TransE	75.9	81.5	78.7
TransD	86.4	89.1	87.8
DistMult	87.1	86.2	86.7
ConvKB	87.6	88.8	88.2
KG-BERT	93.5	90.4	91.9
PREGSE	93.2	91.6	92.4

Table 3. Triple classification accuracy for different embedding methods

4.6 The Effectiveness of Graph Structure Feature Fusion

In this section, we conduct ablation experiments to assess the impact of integrating local graph structure features on the model's performance. Specifically, on the FB15k-237 dataset, we individually remove either loop-structure or star-structure, retaining the other for training. We then compare these scenarios with the method that does not utilize graph structure embeddings. The experimental results are presented in Figure 5, where PREGSE represents the original model, PREGSE-S denotes retaining star structures, PREGSE-L denotes retaining loop-structure, and PREGSE- signifies removing both types of graph structures.

The results indicate that when removing either star-structure or loop-structure, the model's performance is inferior to the original model. This suggests that simultaneously integrating loop-structure and star-structure leads to better results. The experimental outcomes with integrated loop-structure or star-structure are notably superior to these without integrated graph structures, indicating that both types of graph structures contribute positively. Moreover, when using only star-structure, the experimental results outperform those using only loop-structure, suggesting that incorporating information from entity neighbors plays a more crucial role.

4.7 Model Complexity

Traditional translation-based methods typically have fewer parameters and are less complex. Knowledge graph representation learning methods based on PLM can

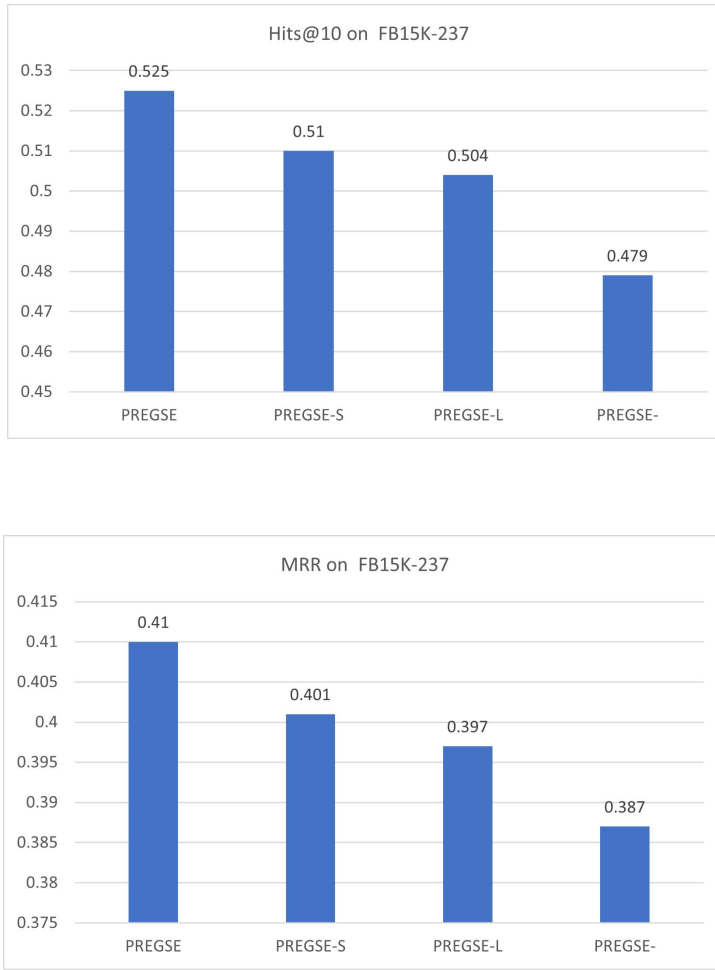


Figure 5. Experimental results on the FB15K-237 dataset

leverage the semantic information learned during the pre-training phase of the model to obtain better vector representations, but this also leads to larger parameter sizes. We analyzed the complexity and parameter scale of current models, with results shown in Table 4. Here, n denotes the number of triples, n_e denotes the number of entities, n_r denotes the number of relations, n_p denotes the number of relational paths, d_e denotes the dimensionality of entities, d_r denotes the dimensionality of relations, and M denotes the number of attention heads. As shown in Table 4, it

can be observed that the PREGSE method has lower spatial complexity, similar to TransE, but it has higher computational complexity due to the inclusion of multiple modules. In terms of model parameter scale, the introduction of the PLM results in a larger parameter scale for PREGSE compared to other traditional representation learning methods. According to research in BERT for Link Prediction (BLP) [32], initializing entities with only BERT can reduce the complexity of model training while fully utilizing semantic information.

Model	Space Complexity	Time Complexity
TransE	$O(n_e d_e + n_r d_r)$	$O(n)$
TransR	$O(n_e d_e + n_r(1 + d_e)d_r)$	$O(2d_e d_r n)$
DistMult	$O((n_e d_e + n_r d_r)^2)$	$O(nd_e d_r)$
ConvE	$O(n_e d_e + n_r d_r)$	$O(nd_e)$
PREGSE	$O(n_e d_e + (n_r + n_p)d_r)$	$O((n + n_p)d_e d_r)$

Table 4. Time and space complexity of models used in the experiments

4.8 Further Analyses

Compared to existing baseline models, the PREGSE model demonstrates stronger robustness and generalization capabilities when handling input data under various conditions. First, when the input entities or relations are absent from the training set, as indicated by the research in StAR, PLM-based models can still encode entities or relations into relatively appropriate embedding vectors due to the rich semantic information learned by the PLM in upstream tasks. This gives PLM-based models a more powerful capacity for extension compared to other types of models. Second, when structural information is missing, PLM can better compensate for this issue. Finally, in KG-BERT, only BERT is used for link prediction tasks, which can lead to difficulties in distinguishing between entities or relations with similar textual semantics. Research in LASS [33] shows that introducing the scoring function from the translation model can effectively address this problem.

5 CONCLUSIONS

In this paper, we propose a representation learning method, Pretrained Language Model-based Graph Structure Enhanced Embedding (PREGSE), which incorporates text encoding and graph structure information for improved knowledge graph completion tasks.

Firstly, we utilize a pretrained language model (BERT) to map entities and relations in the knowledge graph into vector representations, leveraging the deep semantic information learned by the pretrained language model on large-scale text data. Secondly, we introduce a graph attention network to capture various graph structure information among entities. Finally, employing a multi-task learning strategy,

we structurally model triplets to accelerate the model's convergence. Experimental results on FB15k-237, FB13, WN18RR, and WN11 datasets demonstrate that our approach enhances the understanding of deep semantic connections between entities. It effectively considers both semantic and structural information, resulting in improved accuracy and effectiveness in knowledge graph completion. Future research directions include incorporating more types of graph structures to further enhance the performance of knowledge graph representation learning.

6 DECLARATIONS

This research was supported by the National Natural Science Foundation of China under Grant No. 62072136, the Natural Science Foundation of Heilongjiang Province No. LH2023F031, the National Key R&D Program of China under Grant No. 2020YFB1710200.

REFERENCES

[1] Wang, M.—Wang, H.—Li, B.—Zhao, X.—Wang, X.: Survey on Key Technologies of New Generation Knowledge Graph. Journal of Computer Research and Development, Vol. 59, 2022, No. 9, pp. 1947–1965 (in Chinese).

[2] Chen, J.—Yu, J.—Yang, X.: A Feature Extraction Based Recommender Algorithm Fusing Semantic Analysis. Journal of Computer Research and Development, Vol. 57, 2020, No. 3, pp. 562–575 (in Chinese).

[3] Qiao, S.—Yang, G.—Yu, Y.—Han, N.—Qin, X.—Qu, L.—Ran, L.—Li, H.: QA-Kgnet: Language Model-Driven Knowledge Graph Question-Answering Model. Journal of Software, Vol. 34, 2023, No. 10, pp. 4584–4600, doi: 10.13328/j.cnki.jos.006882 (in Chinese).

[4] Hou, Z.—Jin, X.—Chen, J.—Guan, S.—Wang, Y.—Cheng, X.: Survey of Interpretable Reasoning on Knowledge Graphs. Journal of Software, Vol. 33, 2022, No. 12, pp. 4644–4667, doi: 10.13328/j.cnki.jos.006522 (in Chinese).

[5] Bollacker, K.—Evans, C.—Paritosh, P.—Sturge, T.—Taylor, J.: Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD '08), 2008, pp. 1247–1249, doi: 10.1145/1376616.1376746.

[6] Auer, S.—Bizer, C.—Kobilarov, G.—Lehmann, J.—Cyganiak, R.—Ives, Z.: DBpedia: A Nucleus for a Web of Open Data. In: Aberer, K., Choi, K. S., Noy, N. et al. (Eds.): The Semantic Web (ISWC 2007, ASWC 2007). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 4825, 2007, pp. 722–735, doi: 10.1007/978-3-540-76298-0_52.

[7] Suchanek, F. M.—Kasneci, G.—Weikum, G.: Yago: A Core of Semantic Knowledge. Proceedings of the 16th International Conference on World Wide Web (WWW '07), ACM, 2007, pp. 697–706, doi: 10.1145/1242572.1242667.

[8] Yang, D.—He, T.—Wang, H.—Wang, J.: Survey on Knowledge Graph Embedding Learning. Journal of Software, Vol. 33, No. 9, pp. 3370–3390 (in Chinese).

[9] Li, S.—Shu, S.—Hao, X.—Hao, Z.: Knowledge Representation Learning Method Integrating Textual Description and Hierarchical Type. Journal of Zhejiang University (Engineering Science), Vol. 57, 2023, No. 5, pp. 911–920 (in Chinese).

[10] Shu, S.—Li, S.—Hao, X.—Zhang, L.: Knowledge Graph Embedding Technology: A Review. Journal of Frontiers of Computer Science and Technology, Vol. 15, 2021, No. 11, pp. 2048–2062 (in Chinese).

[11] Du, X.—Liu, M.—Shen, L.—Peng, X.: Survey on Representation Learning Methods of Knowledge Graph for Link Prediction. Journal of Software, Vol. 35, 2024, No. 1, pp. 87–117, doi: 10.13328/j.cnki.jos.006902 (in Chinese).

[12] Bordes, A.—Usunier, N.—Garcia-Duran, A.—Weston, J.—Yakhnenko, O.: Translating Embeddings for Modeling Multi-Relational Data. In: Burges, C. J., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K. Q. (Eds.): Advances in Neural Information Processing Systems 26 (NIPS 2013). Curran Associates, Inc., 2013, pp. 2787–2795, https://proceedings.neurips.cc/paper_files/paper/2013/file/1cecc7a77928ca8133fa24680a88d2f9-Paper.pdf.

[13] Devlin, J.—Chang, M. W.—Lee, K.—Toutanova, K.: BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. In: Burstein, J., Doran, C., Solorio, T. (Eds.): Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2019), Volume 1 (Long and Short Papers). ACL, 2019, pp. 4171–4186, doi: 10.18653/v1/N19-1423.

[14] Brown, T.—Mann, B.—Ryder, N.—Subbiah, M.—Kaplan, J. D. et al.: Language Models Are Few-Shot Learners. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., Lin, H. (Eds.): Advances in Neural Information Processing Systems 33 (NeurIPS 2020). Curran Associates, Inc., 2020, pp. 1877–1901, https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[15] Lewis, M.—Liu, Y.—Goyal, N.—Ghazvininejad, M.—Mohamed, A.—Levy, O.—Stoyanov, V.—Zettlemoyer, L.: BART: Denoising Sequence-to-Sequence Pre-Training for Natural Language Generation, Translation, and Comprehension. In: Jurafsky, D., Chai, J., Schluter, N., Tetreault, J. (Eds.): Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020). ACL, 2020, pp. 7871–7880, doi: 10.18653/v1/2020.acl-main.703.

[16] Yao, L.—Mao, C.—Luo, Y.: KG-BERT: BERT for Knowledge Graph Completion. CoRR, 2019, doi: 10.48550/arXiv.1909.03193.

[17] Fan, M.—Zhou, Q.—Chang, E.—Zheng, T. F.: Transition-Based Knowledge Graph Embedding with Relational Mapping Properties. Proceedings of the 28th Pacific Asia Conference on Language, Information and Computation (PACLIC 2014), 2014, pp. 328–337, https://aclanthology.org/Y14-1039.pdf.

[18] Wang, Z.—Zhang, J.—Feng, J.—Chen, Z.: Knowledge Graph Embedding by Translating on Hyperplanes. Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 28, 2014, No. 1, pp. 1112–1119, doi: 10.1609/aaai.v28i1.8870.

[19] Lin, Y.—Liu, Z.—Sun, M.—liu, Y.—Zhu, X.: Learning Entity and Relation Embeddings for Knowledge Graph Completion. Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 29, 2015, No. 1, pp. 2181–2187, doi: 10.1609/aaai.v29i1.9491.

[20] Ji, G.—He, S.—Xu, L.—Liu, K.—Zhao, J.: Knowledge Graph Embedding via Dynamic Mapping Matrix. In: Zong, C., Strube, M. (Eds.): Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2015). 2015, pp. 687–696, doi: 10.3115/v1/P15-1067.

[21] Sun, Z.—Deng, Z. H.—Nie, J. Y.—Tang, J.: RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. Proceedings of the International Conference on Learning Representations (ICLR 2019), 2018, doi: 10.48550/arXiv.1902.10197.

[22] Nickel, M.—Tresp, V.—Kriegel, H. P.: A Three-Way Model for Collective Learning on Multi-Relational Data. Proceedings of the 28th International Conference on Machine Learning (ICML 2011), 2011, pp. 809–816.

[23] Yang, B.—Yih, W.—He, X.—Gao, J.—Deng, L.: Embedding Entities and Relations for Learning and Inference in Knowledge Bases. Proceedings of the 3rd International Conference on Learning Representations (ICLR), 2015, doi: 10.48550/arXiv.1412.6575.

[24] Nickel, M.—Rosasco, L.—Poggio, T.: Holographic Embeddings of Knowledge Graphs. Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 30, 2016, No. 1, pp. 1955–1961, doi: 10.1609/aaai.v30i1.10314.

[25] Dettmers, T.—Minervini, P.—Stenetorp, P.—Riedel, S.: Convolutional 2D Knowledge Graph Embeddings. Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, 2018, No. 1, pp. 1811–1818, doi: 10.1609/aaai.v32i1.11573.

[26] Nguyen, D. Q.—Nguyen, T. D.—Nguyen, D. Q.—Phung, D.: A Novel Embedding Model for Knowledge Base Completion Based on Convolutional Neural Network. In: Walker, M., Ji, H., Stent, A. (Eds.): Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2018), Volume 2 (Short Papers). 2018, pp. 327–333, doi: 10.18653/v1/N18-2053.

[27] Jiang, X.—Wang, Q.—Wang, B.: Adaptive Convolution for Multi-Relational Learning. In: Burstein, J., Doran, C., Solorio, T. (Eds.): Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2019), Volume 1 (Long and Short Papers). 2019, pp. 978–987, doi: 10.18653/v1/N19-1103.

[28] Schlichtkrull, M.—Kipf, T. N.—Bloem, P.—van den Berg, R.—Titov, I.—Welling, M.: Modeling Relational Data with Graph Convolutional Networks. In: Gangemi, A., Navigli, R., Vidal, M. E. et al. (Eds.): The Semantic Web (ESWC 2018). Springer, Cham, Lecture Notes in Computer Science, Vol. 10843, 2018, pp. 593–607, doi: 10.1007/978-3-319-93417-4_38.

[29] Vashishth, S.—Sanyal, S.—Nitin, V.—Talukdar, P.: Composition-Based Multi-Relational Graph Convolutional Networks. Proceedings of the 8th In-

ternational Conference on Learning Representations (ICLR 2020), 2020, doi: 10.48550/arXiv.1911.03082.

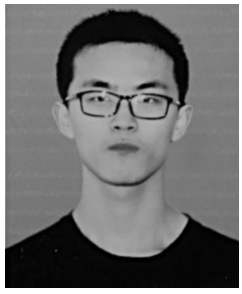
[30] Li, Z.—Zhao, Y.—Zhang, Y.—Zhang, Z.: Multi-Relational Graph Attention Networks for Knowledge Graph Completion. Knowledge-Based Systems, Vol. 251, 2022, Art. No. 109262, doi: 10.1016/j.knosys.2022.109262.

[31] Jin, Y.—Yang, L.: Graph-Aware Tensor Factorization Convolutional Network for Knowledge Graph Completion. International Journal of Machine Learning and Cybernetics, Vol. 15, 2024, No. 5, pp. 1755–1766, doi: 10.1007/s13042-023-01995-3.

[32] Daza, D.—Cochez, M.—Groth, P.: Inductive Entity Representations from Text via Link Prediction. Proceedings of the Web Conference 2021 (WWW '21), ACM, 2021, pp. 798–808, doi: 10.1145/3442381.3450141.

[33] Shen, J.—Wang, C.—Gong, L.—Song, D.: Joint Language Semantic and Structure Embedding for Knowledge Graph Completion. In: Calzolari, N., Huang, C. R., Kim, H., Pustejovsky, J. et al. (Eds.): Proceedings of the 29th International Conference on Computational Linguistics (COLING 2022). 2022, pp. 1965–1978, https://aclanthology.org/2022.coling-1.171.pdf.

Song Li received his Ph.D. degree in computer application technology from the Harbin University of Science and Technology, Harbin, China, in 2009. He is currently Professor with the College of Computer Science and Technology, Harbin University of Science and Technology. He is the author of two books, more than 60 articles. His research interests include spatial database, data mining, big data and information privacy protection, knowledge graph.



Chengyu Zhong received his B.Sc. degree in software engineering from the Harbin University of Science and Technology, in 2022. He is currently pursuing a M.Sc. degree in the Department of Computer Science and Technology, Harbin University of Science and Technology, China. His current research interests include knowledge graph and knowledge representation learning.

Liping Zhang received her M.Sc. degree in computer application technology from the Harbin University of Science and Technology, Harbin, China, in 2006. She is currently Professor with the College of Computer Science and Technology, Harbin University of Science and Technology. She is the author of two books, more than 40 articles. Her research interests include spatial database, data mining, knowledge graph, big data and information privacy protection.