



# Enhancing knowledge graph embedding with structure and semantic features

Yalin Wang<sup>1</sup> · Yubin Peng<sup>1</sup> · Jingyu Guo<sup>1</sup>

Accepted: 2 February 2024 / Published online: 15 February 2024

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2024

## Abstract

Knowledge graph embedding converts knowledge graphs based on symbolic representations into low-dimensional vectors. Effective knowledge graph embedding methods are key to ensuring downstream tasks. Some studies have shown significant performance differences among various knowledge graph embedding models on different datasets. They attribute this issue to the insufficient representation ability of the models. However, what representation ability knowledge graph embedding models possess is still unknown. Therefore, this paper first selects three representative models for analysis: translation and rotation models in distance models, and the Bert model in neural network models. Based on the analysis results, it can be concluded that the translation model focuses on clustering features, the rotation model focuses on hierarchy features, and the Bert model focuses on word co-occurrence features. This paper categorize clustering and hierarchy as structure features, and word co-occurrence as semantic features. Furthermore, a model that solely focuses on a single feature will lead to a lack of accuracy and generality, making it challenging for the model to be applicable to modern large-scale knowledge graphs. Therefore, this paper proposes an ensemble model with structure and semantic features for knowledge graph embedding. Specifically, the ensemble model includes a structure part and a semantic part. The structure part consists of three models: translation, rotation and cross. Translation and rotation models serve as basic feature extraction, while the cross model enhances the interaction between them. The semantic part is built based on Bert and integrated with the structure part after fine-tuning. In addition, this paper also introduces a frequency model to mitigate the training imbalance caused by differences in entity frequencies. Finally, we verify the effectiveness of the model through link prediction. Experiments show that the ensemble model has achieved improvement on FB15k-237 and YAGO3-10, and also has good performance on WN18RR, proving the effectiveness of the model.

**Keywords** Knowledge graph embedding · Ensemble learning · Link prediction · Distance model · Neural network model

## 1 Introduction

Knowledge graphs are a structured representation of knowledge, typically consisting of a series of triples  $(h, r, t)$ , where  $h$  represents the head entity,  $r$  represents the relation, and  $t$  represents the tail entity. Embedding is the process of mapping data to a lower-dimensional space, aiming to preserve the

original features of the data and improve the computational efficiency of computers in processing the data. In machine learning, embeddings are widely used in fields such as natural language processing, image processing, and recommendation systems to better handle and analyze data. Knowledge graph embedding [1] is the process of embedding entities and relations of a knowledge graph into a lower-dimensional space. As shown in Fig. 1, initially, vectors are allocated to all entities and relations, forming a vector set. Subsequently, for a given triple (Mount Everest, instance of, Mountain), its corresponding vectors are queried. These vectors are input into a model. The model's output is also a vector which represents the knowledge features of this triple. Finally, a score function is used to transform the vector into a scalar, which represents the score of this triple. A lower score indicates that the triple is more correct. After training, we select the common down-

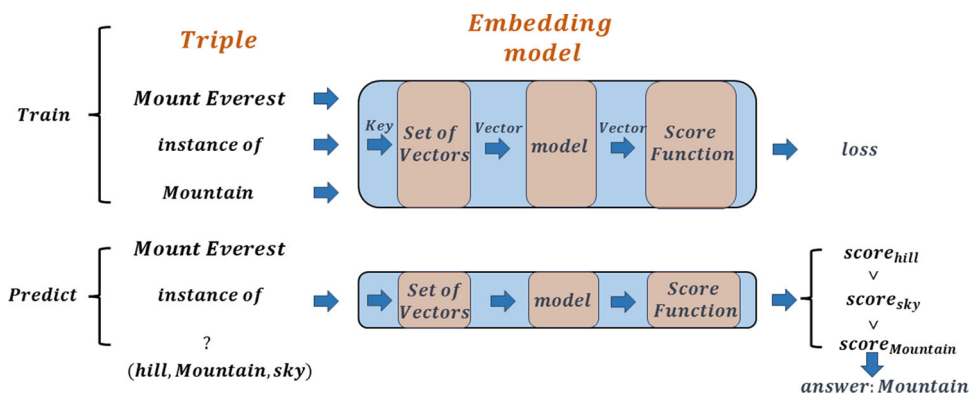
✉ Jingyu Guo  
csugjy@csu.edu.cn

Yalin Wang  
ylwang@csu.edu.cn

Yubin Peng  
ybpeng@csu.edu.cn

<sup>1</sup> School of Automation, Central South University, Hunan 410083, China

**Fig. 1 Knowledge graph embedding and link prediction.** loss is a scalar, and in link prediction, it represents the score for each entity



stream task of link prediction [2] in knowledge graphs for validation. Link prediction refers to the task of predicting the correct tail entity (or head entity) among candidate entities when given the head entity (or tail entity) and a specific relation. In Fig. 1, three candidate entities (hill, Mountain, sky) are initially presented. Subsequently, the scores for these three entities are computed using the trained vector set and the model. Finally, the candidate entity with the lowest score is selected as the prediction result.

Current popular knowledge graph embedding methods can be broadly categorized into two types: distance models and neural network models. For distance models, they represent triples using vector operations. Based on the operation methods, distance models can be further categorized into translation models and rotation models. Translation models are based on vector addition, while rotation models are based on vector multiplication. For neural network models, they represent triples using neural networks. Differing from distance models that directly define vector operations, neural network models perform calculations between vectors using neural networks. Previous research has shown that these models and their improvements tend to perform differently on various benchmarks. For instance, transE performs better on FB15k-237 but poorly on datasets like WN18RR and YAGO3-10. RotateE shows significant improvements on WN18RR, YAGO3-10 and FB15k-237. The neural network models (e.g., Bert) further improve performance on WN18RR but do not have a significant impact on FB15k-237. In response to this problem, some studies suggest that it is caused by the limited representational capacity of the model. Therefore, they incorporate additional models into the existing ones to enhance the performance. However, the fundamental problem of what these models can effectively represent remains unclear, which limits both further model improvements and the selection of different models for practical applications.

Inspired by HAKE [3], we think that what distance models can represent is a structure features that reflect the collective features of entities. Therefore, this paper first analyzes the

features of rotation and translation models in distance models. The results indicate that translation models can extract clustering features, while rotation models can extract hierarchical features. For neural network models, we discover through experimental analysis that an important feature they can extract is co-occurrence of words.

Furthermore, integrating these three types of models together may potentially enhance the model’s accuracy and applicability. Inspired by this idea, this paper adopts an ensemble learning approach to combine these three types of models, resulting in a unified ensemble model. Since cluster features and hierarchical features are related to geometric structural features, while word co-occurrence features are related to content, the design of the ensemble model is divided into two parts: structure part and semantic part. The structure part consists of an ensemble of translation and rotation models. Additionally, we find that introducing cross-terms similar to those in multivariate linear regression improves model performance. The semantic part of the model uses a bi-encoder architecture based on Bert. When different models are integrated, there is an issue of mismatched output magnitudes, leading to some models dominating with large outputs while others are weakened with smaller outputs. To address this problem, we redesign the score functions for both the structure and semantic parts. For the structure part, we adjust the norm selection for its score function and determine the optimal choice through grid search. For the semantic part, we introduce the sigmoid function to control the range of the output values.

During the training process, a common issue arises concerning the high-frequency occurrence of certain samples, where some entities and relations appear frequently in the triples, resulting in them being trained more frequently. Therefore, this paper introduces the frequency model to address the problem and conducts an analysis within the “Frequency Part”.

Finally, this paper validates the ensemble model on commonly used benchmarks and achieves competitive results. Particularly, compared to using individual models like

transE and rotateE, the ensemble model exhibits significant improvements. In summary, this paper's contributions are as follows: (1) **Explaining the primary knowledge features extracted by translation models, rotation models, and neural network models (Bert).** (2) **Combining the features of various models to train a more powerful representation model using ensemble learning.** (3) **Introducing the frequency model to alleviate the issue of imbalanced learning between high-frequency and low-frequency samples.**

## 2 Related work

Roughly speaking, knowledge graph embedding models can be divided into two categories: distance models and neural network models. Our work involves the two types of models mentioned above. More specifically, the structure part shares similarities with transE and rotateE. The semantic part utilizes Bert, a neural network model.

### 2.1 Distance model

The distance model is mainly based on the idea of geometric translation and rotation, where entities and relationships exist in vector form. TransE [4] is the earliest distance model which represents the relationship as a translation vector between head and tail entities. It defines the score function as  $f = \|h + r - t\|_{1/2}$ , where  $h, r, t \in R^n$ . So it is a translation model. However, transE suffers from the problem of not being able to learn 1-N, N-1 and N-N relationships. Therefore, transE needs to be improved to accommodate these various patterns. RotateE [5] is the earliest distance model which represents the relationship as a rotation vector between head and tail entities. It defines the score functions as  $f = \|h \circ r - t\|_{1/2}$ , where  $h, r, t \in R^n$ . It can fit a variety of relational patterns, such as 1-N, N-1 and self-reflexive, which not only solve the problems in translation models well, but also represent more patterns.

Subsequent research regard these two models as fundamental models in the distance model. GTransE improves the regularization and negative sampling techniques of the transE model [6]. RotatPRH [7] combines the ideas of rotateE and transE. To emphasize the significant role of entities, transP [8] introduces an entity space. It also aims to solve one-to-many and many-to-one problems. To further capture the transitive relations between entities, Rot-Pro [9] introduces a projection mechanism to model transitivity in knowledge graphs. To enhance representational capabilities, QuatE [10] embeds triples into a complex space. Similarly, AttH [11] employs a hyperbolic space with a larger capacity for knowledge graph embedding. Due to the lack of consideration for entity types in the above model, ETF [12] add type infor-

mation constraints to the base distance model. Similarly, relation-constraint model [13] introduces relationship constraints to the base distance model, and make the knowledge more accurate.

### 2.2 Neural network model

Neural network models have been applied to knowledge graph embeddings due to their powerful representational capabilities. Based on the type of neural network architecture, they can be broadly classified into two main categories: CNN-based neural network models and Transformer-based neural network models.

ConvE [14] is a CNN-based neural network model. It reshapes the vectors of head entities and relations into a 2D matrix and then undergoes learning through a CNN network. Finally, it generates predicted tail entity vectors. Inspired by ConvE, ConvNE [15] considers relationships with globally consistent dimensions.

The Bert [16] model is a Transformer-based neural network model. Leveraging Bert's outstanding performance in NLP tasks, KG-Bert [17] fine-tunes the Bert model to predict triples in knowledge graphs. However, it performs poorly in low-dimensional spaces. Therefore, SAttLE [18] improves this issue through self-attention mechanisms. To extract richer relationship representations, RFAN [19] utilizes a multi-head attention mechanism to capture complex relationship features. To constrain the learning pattern of the network, some models [20] incorporate structural constraints into neural network models. Compared to CNN-based neural network models, Transformer-based neural network models have gained broader applications.

In addition, there are also some studies focusing on other aspects. For example, RLPath [21] applies reinforcement learning for knowledge graph embeddings. DynamicKG [22] explores dynamic knowledge graphs.

Overall, both distance models and neural network models have performed well on certain benchmarks. However, they still exhibit fundamental differences in their representation capabilities for knowledge.

## 3 Features every model has learned

This section analyzes the features that distance models and neural network models can represent. Specifically, for distance models, we choose the classical translation model and rotation model. For semantic models, we choose the classical Bert model.

### 3.1 Features of translation model

The translation model defines the score functions as  $f = \|h + r - t\|_{1/2}$ , where  $h$  represents the vector for the head

entity,  $r$  represents the vector for the relationship, and  $t$  represents the vector for the tail entity. After training, the score for each correct triple is minimized as much as possible. Ideally, the gradient is 0 when learning to a steady state. Since entities and relationships exist in multiple triples, the stability condition becomes that the gradient sum of all triples associated with a particular entity or relationship in the knowledge graph is 0. Suppose that for the relation  $r$ , all the triples related to  $r$  in the training set are taken out and their score functions are summed. Then the total score expression is:

$$\sum_i f_i = \sum_i \|h_i + r - t_i\|_{1/2}$$

Taking the derivative of the above expression, the equation that  $r$  satisfies when the training converges to the equilibrium state can be introduced as:

$$\sum_i t_i - \sum_i h_i = Nr$$

From the above equation, it can be seen that after training by gradient descent, the relation is a vector pointing from the centroid of the set of tail entities to the centroid of the set of head entities. The results are shown in Fig. 2. To make the score function of each triple as small as possible, it is necessary to make the tail entity as close as possible to the centroid of the set of tail entities and the head entity as close as possible to the centroid of the set of head entities. The ideal case is when all head entities are at the same point and all tail entities are at the same point. In general,  $r$  responds to the translation relationship between two centroids, and the entity embeddings learned by the translation model have distinct clustering features.

### 3.2 Features of rotation model

The rotation model defines the score functions as  $f = \|h \circ r - t\|_{1/2}$ , and use the same learning approach as the translation model. Therefore, in the same way, all the triples

in the training set associated with the relation  $r$  are taken and summed over the score function. Calculate the derivative of this expression.

$$r \circ \sum_i h_i = \sum_i t_i$$

The summation term in the above equation can still be considered as two clustering centroids. Let  $\bar{h} = \sum_i h_i$ ,  $\bar{t} = \sum_i t_i$ , and then take the  $L_1$ -norm.

$$\|r \circ \bar{h}\|_1 = \|\bar{t}\|_1$$

To correlate  $r \circ \bar{h}$  with dot product, the dimension of  $r \circ \bar{h}$  is divided into positive and negative terms, then  $\|r \circ \bar{h}\|_1 = \|(r \circ \bar{h})^+\|_1 + \|(r \circ \bar{h})^-\|_1$ . To facilitate the analysis, consider the special case  $\|r \circ \bar{h}\|_1 = \|(r \circ \bar{h})^+\|_1$  ( $h$  and  $r$  have the same sign for each dimension). Since each dimension of the vector is positive, the above equation can be changed to  $|r \circ \bar{h}|_1 = \|r\|_2 \|\bar{h}\|_2 \cos \vartheta$ , where  $\vartheta$  is the angle between  $\bar{h}$  and  $r$ . From the equivalence between 1-norm and 2-norm we can get:

$$\|\bar{t}\|_2 \leq \|\bar{t}\|_1 \leq \sqrt{n} \|\bar{t}\|_2$$

$$\|\bar{t}\|_2 \leq \|r\|_2 \|\bar{h}\|_2 \cos \vartheta \leq \sqrt{n} \|\bar{t}\|_2$$

$$\frac{\|r\|_2 \cos \vartheta}{\sqrt{n}} \|\bar{h}\|_2 \leq \|\bar{t}\|_2 \leq \|r\|_2 \cos \vartheta \|\bar{h}\|_2$$

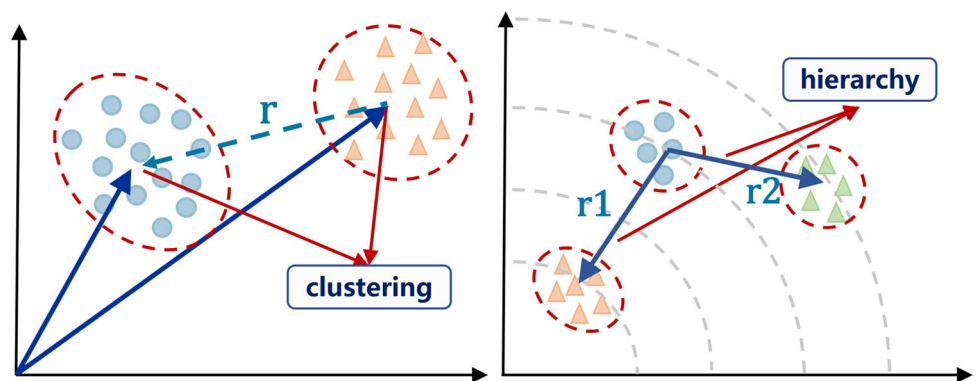
$\|r\|_2 \cos \vartheta$  reflects the hierarchical relationship between the head entity and the tail entity. The tail entity is in the interval from  $\|r\|_2 \|\bar{h}\|_2 \cos \vartheta / \sqrt{n}$  to  $\|r\|_2 \|\bar{h}\|_2 \cos \vartheta$ .

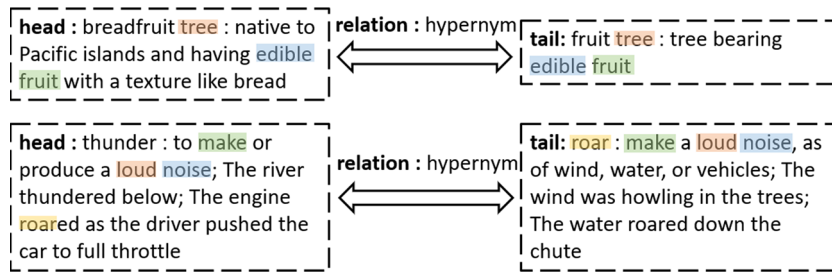
$$\frac{\|r\|_2 \cos \vartheta}{\sqrt{n}} \geq 1 \rightarrow \|\bar{t}\| \geq \|\bar{h}\|$$

$$\|r\|_2 \cos \vartheta \leq 1 \rightarrow \|\bar{t}\| \leq \|\bar{h}\|$$

If  $\|r\|_2 \cos \vartheta \in [1, \sqrt{n}]$ ,  $\bar{h}$  and  $\bar{t}$  can be considered to be at the same level. If  $\|r\|_2 \cos \vartheta \in [0, 1]$ ,  $\bar{t}$  is closer to the origin, so it is at a lower level. If  $\|r\|_2 \cos \vartheta \in [\sqrt{n}, \infty]$ ,  $\bar{t}$  is far to the origin, so it is at a higher level. The results are shown in Fig. 2.

**Fig. 2 Structural features.** The left figure represents the clustering features of entities extracted by the translation model, and the associated entities are in the same area; the right figure represents the hierarchical features of entities extracted by the rotation model, and the associated entities are in the same circle





**Fig. 3 Semantic features.** The semantic model for learning word relationships can capture the co-occurrence of words between sentences. The left side of the figure shows the triple head entity and its description information, while the right side shows the triple tail entity and

its description information. Some triple entities themselves have the same words. The co-occurrence of words between sentences is more prominent after adding description information

### 3.3 Features of bert model

The Bert model learns the relationships between words and sentences through a self attention mechanism. This paper explores the contribution of each word to prediction accuracy by excluding some specified words. Finally, we found that the Bert model can extract co-occurrence between words which is shown in Fig. 3. This property is a feature that is completely unlearned by structure models. The relevant experiments are in the "Semantic Part Analysis".

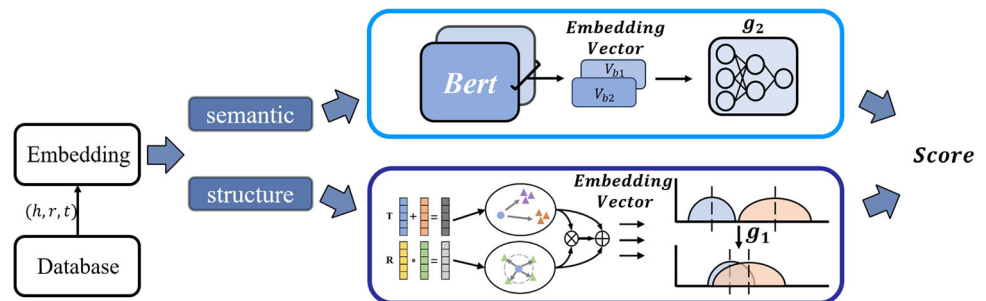
## 4 Ensemble model with structure and semantic

This section describes the design of the structure part and the semantic part, and the way of training the ensemble model. The ensemble model framework is shown in Fig. 4.

### 4.1 Structure part

First, we analyze the three models in the structure part: translation, rotation, and cross. Then, we analyze the selection of the score function.

**Fig. 4 Ensemble model.** The model consists of a structure part and a semantic part. The entire process is divided into two steps. Firstly, it retrieves triples from the dataset and queries their corresponding vectors. Then, it calculates scores through the model



### 4.1.1 Combine translation and rotation model

The structure part consists of translation and rotation models. The translation model is denoted as  $T$  and the rotation model is denoted as  $R$ . The embeddings for the translation model are denoted as  $h_m, r_m$  and  $t_m$ , while  $h_n, r_n$  and  $t_n$  represent the embeddings for the rotation model. The score function is denoted as  $g_1$ . First, calculate the feature vectors  $V_m$  and  $V_n$  for the triplets based on the translation and rotation model. The specific expressions are as follows:

$$V_m = h_m + r_m - t_m$$

$$V_n = h_n \circ r_n - t_n$$

Then, we calculate the scores for both the translation and rotation models, and the specific expressions are as follows:

$$T = g_1(V_m)$$

$$R = g_1(V_n)$$

Finally, the two parts are weighted and summed. The expression is as follows:

$$M = (\omega_1 T + \omega_2 R) \quad \omega_1, \omega_2 \in \mathbb{R}_+^k$$



### 4.1.2 Cross model

This paper is inspired by the crossover term in the multiple linear regression equation. To avoid the inability to interact information between models, the cross module of two structure models are added at the end of the model. It enables the models to interact with each other, and at the same time extends the simple linear summation to nonlinear summation to improve the representation ability of the models. The specific expression for the intersection term is:

$$C = T \cdot R$$

Combining the modules, the expression for the structure part is:

$$M = (\omega_1 T + \omega_2 R + \omega_3 C) \quad \omega_1, \omega_2, \omega_3 \in \mathbb{R}_+^k$$

### 4.1.3 Score function for structure

Structure models typically use the L2 norm to calculate the scoring function. However, for different structure models, the score distribution calculated using the L2 norm varies, as shown in Fig. 5. This can result in different contributions to the final score from each model.

To solve this problem, this paper introduces scaling factors  $p$  and  $q$  to control the local and global calculation of scores, and tries to make the mean values of the two types of models as close as possible. Take rotation model as an example. Calculate the derivative of the head entity  $h_n$ :

$$\frac{dR}{dh_n} = g'_1 V'_n$$

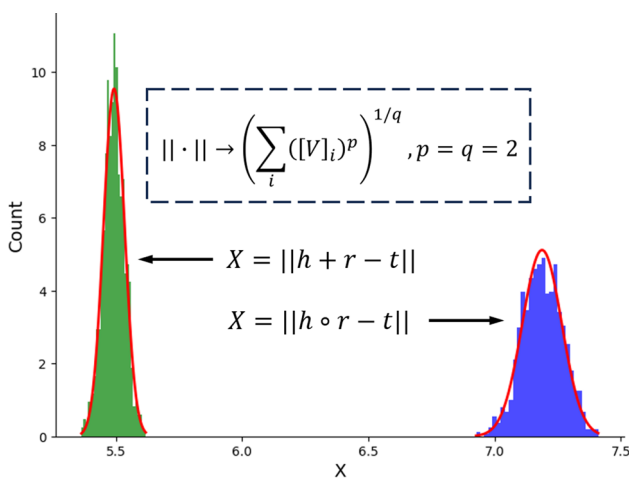


Fig. 5 The score distribution for the T and R structure models. Count represents the number of scores in that interval, and X represents the scores calculated by the model. The scoring function utilizes the L2 norm

$V_n$  determined by the design of the model, so it is fixed.  $g$  is determined by the score function and can be designed as needed. The general expression of the score function can be expressed as:

$$g_1 = \left( \sum_i ([V_n]_i)^p \right)^{1/q}$$

where  $[V_n]_i$  denotes the  $i$ -th dimension of the vector  $V$ , and  $p, q$  are adjustable coefficients.

First, analyze it as a whole by letting  $x = \sum_i ([V_n]_i)^p$ , so that  $g_1 = x^q$ . By changing  $q$ , the gradient will have a different form. When  $q = 2$ , it can prevent excessively large score values. Additionally, due to the inverse proportionality between gradient values and  $x$ , data with high scores update slowly, while data with low scores update quickly. This helps prevent extreme differentiation between high and low score values. When  $q = 1$ , the score grows linearly with  $x$ . The gradient is constant, and the constraint on the features is loose. Overall, different values of  $q$  can change the form of the gradient and thus affect the learning process.

Second, analyze the coefficients  $p$  in  $x$ . When  $p = 2$ , the gradient is positively proportional to  $[V_n]_i$ , which will keep the  $[V_n]_i$  from being too large (because the data in each dimension is less than 1, it becomes even smaller after squaring). This will result in a more concentrated distribution of each dimension on the vector space. When  $p = 1$ , the gradient is constant, so the learning rate is not influenced by  $[V_n]_i$ . This will not constrain the distribution of each dimension on

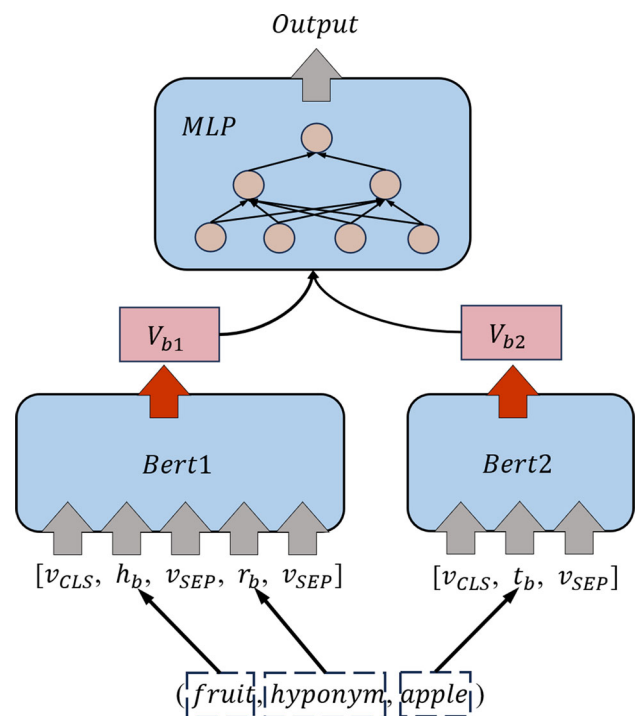


Fig. 6 Semantic Part The model is primarily composed of Bert and MLP

the vector space. There is no advantage or disadvantage to the different values of  $p$ . It is a matter of which one is more applicable to the model.

### 4.2 Semantic part

This section provides a detailed overview of the overall process of the semantic part, as shown in Fig. 6. Previous studies have shown that, for prediction tasks, utilizing the bi-encoder architecture yields the best results. Therefore, this paper adopts the same design approach.

#### 4.2.1 Bert model

The bi-encoder architecture refers to utilizing one Bert [16] encoding for  $(h, r)$  and another Bert encoding for  $t$ , with the parameters between the two Berts not shared.

Taking the triple  $(h, r, t)$  as an example. First, we extract embedding vectors  $h_b, r_b,$  and  $t_b$  from the sets of entity vectors and relation vectors. Then, we combine  $h_b$  and  $r_b$  together as the input  $X$  for Bert1. The advantage of using a single Bert encoder to jointly encode  $h$  and  $r$  is to enable information interaction between them. This ensures that the embedding vectors for the same entity under different relations are distinct. According to the input structure required by Bert, the combination method for  $h_b$  and  $r_b$  is as follows:

$$X = [v_{CLS}, h_b, v_{SEP}, r_b, v_{SEP}]$$

where  $X$  represents the input for Bert1,  $v_{CLS}$  represents the embedding vector of the sentence's start symbol,  $v_{SEP}$  represents the embedding vector of the sentence's end symbol,  $h_b$  represents the embedding vector of the head entity  $h$ , and  $r_b$  represents the embedding vector of the relation  $r$ .  $v_{CLS}$  and  $v_{SEP}$  are both special symbols inherent to Bert and do not require user input.  $V_{b1}$  and  $V_{b2}$  represent the triple feature vector after passing through the Bert model.

Similarly, the second Bert encoder is used to embed  $t$ . The only difference is that when encoding  $t$ ,  $r$  is not introduced. If two Bert models embed relation at the same time, the contribution of relation to the model to discriminate entity will be reduced, which is not conducive to effective learning of knowledge. The input for Bert2 is denoted as  $Y$ .

$$Y = [v_{CLS}, t_b, v_{SEP}]$$

where  $t_b$  represents the embedding vector of the tail entity  $t$ . Therefore, the semantic part produces two feature vectors for the triple. The expressions are as follows:

$$V_{b1}, V_{b2} = BiEncoder(X, Y)$$

where  $BiEncoder$  represents the semantic model composed of Bert1 and Bert2,  $V_{b1}$  represents the output of Bert1, and  $V_{b2}$  represents the output of Bert2.

Finally, the score computation for a triple by the Bert model can be represented by the following equation:

$$S = g_2(V_{b1}, V_{b2})$$

#### 4.2.2 Score function for semantic

This paper employs an MLP as the score function to transform the output vectors of Bert into scores.  $V_{b1}$  and  $V_{b2}$  are passed into an MLP network. The output of the MLP network represents the probability that  $X$  and  $Y$  can form a triple.

$$g_2 = \sigma(MLP(V_{b1}, V_{b2}))$$

where  $V_{b1}$  and  $V_{b2}$  are the outputs of the Bert1 and Bert2.  $\sigma$  represents the sigmoid function.

---

#### Algorithm 1 Training the ensemble model.

---

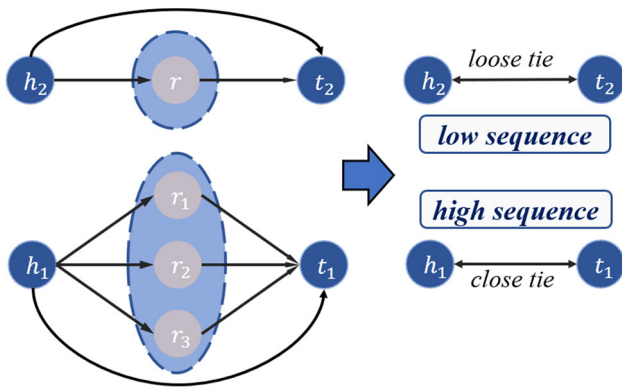
**Require:** structure part:  $h_{type}, r_{type}, t_{type}(type = m, n)$ ; semantic part:  $h_b, r_b, t_b$ ; frequency part:  $h_v, t_v$   
**for** every epoch **do**  
    Randomly sample one positive triple:  $Pos$   
    Randomly sample  $n$  negative triples:  $Neg_i$   
    // ----- StructurePart -----  
     $T = g_1(model(h_m, r_m, t_m))$  ▷  
     $model : h_m + r_m - t_m, \quad g_1 : (\sum_i (\cdot)^p)^{1/q}$   
     $R = g_1(model(h_n, r_n, t_n))$  ▷  
     $model : h_n + r_n - t_n, \quad g_1 : (\sum_i (\cdot)^p)^{1/q}$   
     $C = T \cdot R$   
    // ----- SemanticPart -----  
     $h_b, r_b \rightarrow X, \quad t_b \rightarrow Y$   
     $S = g_2(model(X, Y))$  ▷  
     $model : BiEncoder(X, Y), \quad g_2 : \sigma(MLP(\cdot))$   
    // ----- FrequencyPart -----  
     $\beta = g_f(model(h_v, t_v))$  ▷  $model : h_v - t_v, \quad g_f : \|\sigma(\cdot)\|$   
    // ----- Loss -----  
     $d_r = \beta \cdot sum(T, R, C, S)$   
     $L_+ = -\log \sigma(\gamma - d_r(Pos))$   
     $L_- = -\sum_i^n k \log \sigma(d_r(Neg_i) - \gamma)$   
     $L_{+Bert} = CE(S(Pos), 1)$   
     $L_{-Bert} = \sum_i^n CE(S(Neg_i), 0)$   
     $Loss = L_+ + L_- + L_{+Bert} + L_{-Bert}$   
**end for**  
**return**  $h_{type}, r_{type}, t_{type}(type = m, n), h_b, r_b, t_b, h_v, t_v$

---

### 4.3 Frequency part

#### 4.3.1 Frequency model

The frequency of knowledge pairs is helpful for link prediction, and higher frequency means more likely to be correct. However, during the training process, knowledge pairs with



**Fig. 7 Frequency part.**  $h_1$  and  $t_1$  have three types of relationships in the training set, while  $h_2$  and  $t_2$  have only one type of relationship. This will cause  $h_1$  and  $t_1$  to appear more frequently during the training process, and the training process is unbalanced. (The same problem exists for entities with one-to-many or many-to-one relationships)

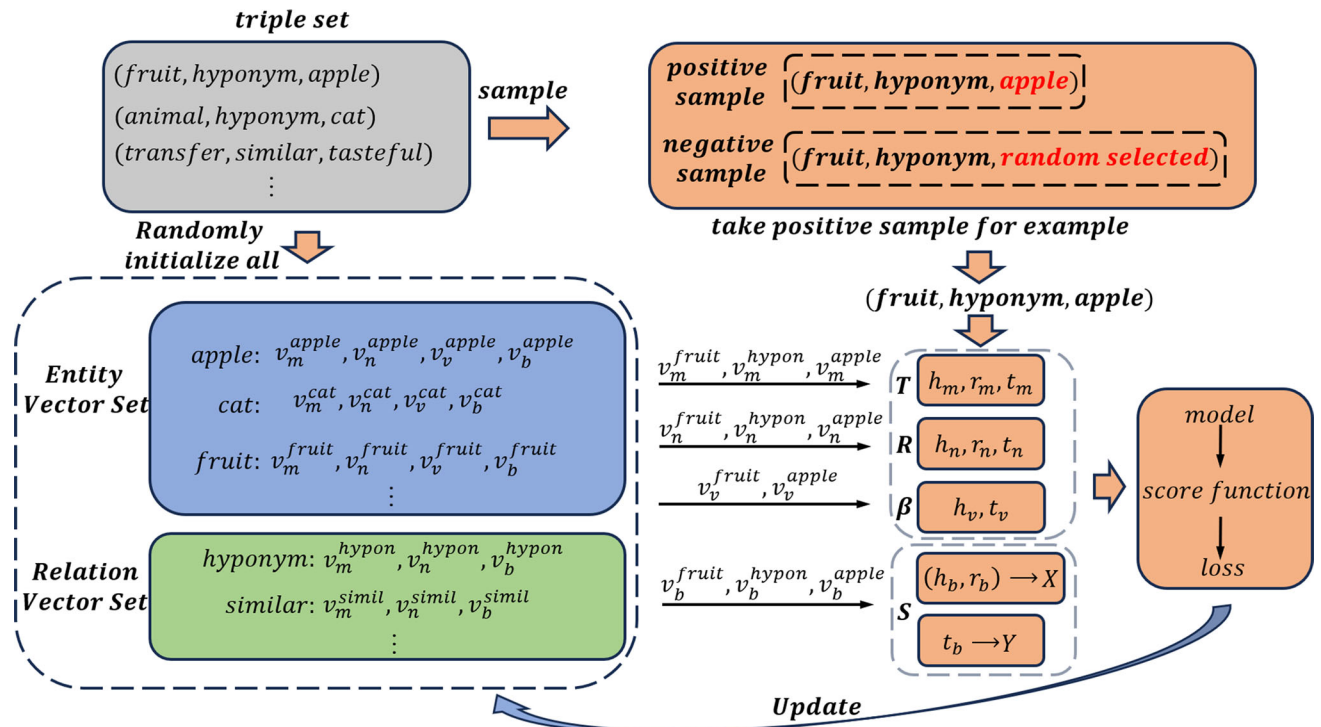
high frequency occurrences will have more iterations and thus learn the distribution adapted to themselves faster, while knowledge pairs with low frequency occurrences will have the opposite. The difference between high and low frequency will destroy the balances of the training process. Therefore, the purpose of introducing frequency features in this paper is both to be introduced as features to enhance the model

representation ability and to balance the differences in the training process due to different frequencies. As for the frequency, most of the triples appear only once in the training set, so we used the binary knowledge pairs which also containing local knowledge. The binary knowledge pair contains  $(h, r)$ ,  $(t, r)$ , and  $(h, t)$ . Because relations and entities are not equivalent concepts, the binary knowledge pairs are specific to  $(h, t)$  which is shown in Fig. 7.

The frequency of the binary knowledge pair  $(h, t)$  is not directly calculated using the number of occurrences in the training set. On the one hand, different triples contain the same  $(h, t)$  binary pair, this represents a potential connection between pieces of knowledge. On the other hand, the frequency information calculated in this way is less flexible and is highly influenced by the training process. So we make the binary frequency features learnable by designing the frequency model so that it can be used as both weights to influence the training process and features to influence the prediction. Inspired by the SE [23] model, we introduce the distance to reflect the frequency. The frequency model is:

$$\beta = g_f(h_v - t_v)$$

where  $h_v$  and  $t_v$  represent the embedding vectors of the frequency model.



**Fig. 8 The training process.**  $m, n$  represent input vectors for the structure part,  $b$  represents input vectors for the semantic part, and  $v$  represents input vectors for the frequency part



### 4.3.2 Score function for frequency

The scoring computation in the frequency model directly employs the L2 norm (normalized through a sigmoid function first), and the specific expression is as follows.

$$g_f = \|\sigma(\cdot)\|$$

From the above equation, it can be seen that the binary frequency is positively proportional to the distance between  $h_v$  and  $t_v$ .  $\sigma$  is used to constrain the distance values and the middle interval can quickly separate binary knowledge pairs with different frequencies. When the distance is greater (or less) than a certain value, the difference in frequency will not bring great changes, and then the main reliance is on clustering and hierarchical features to predict.

### 4.4 Training method

The training process is depicted in Fig. 8. For all entities and relations, randomly initialize all required vectors (with dimensions  $D$ ), forming the sets of entity vectors and relation vectors. Firstly, randomly select a positive sample, such as (fruit, hyponym, apple), from the set of triples. Randomly replace the apple with other entities  $n$  times to generate  $n$  negative samples. Secondly, retrieve all vectors related to fruit and apple from the entity vector set, and retrieve all vectors related to hyponym from the relation vector set. Then, calculate the score based on the vector's corresponding model and score function. Finally, calculate the loss based on the scores and update the vectors.

For a clearer representation of the entire process, we present the training flow in pseudocode, as shown in Algorithm 1. Firstly, sample positive and negative examples. Then, calculate  $T$ ,  $R$ ,  $C$ ,  $S$ , and  $\beta$  using structure and semantic models respectively. Finally, compute the loss functions  $L_+$  and  $L_{-Bert}$  for positive samples, and  $L_-$  and  $L_{+Bert}$  for negative samples, and sum them up to obtain the final loss.

Additionally, to select the optimal parameters  $p$  and  $q$ , we employed a grid search approach. Specifically,  $p$  and  $q$  are searched in the range  $[1 : 2]$  with a step size of 0.5. Then repeat Algorithm 1 based on the searched values of  $p$  and  $q$ . Finally, the optimal combinations of  $p$  and  $q$  is determined separately for  $T$  and  $R$ .

### 4.5 Loss function

To train the model, we use a self-adversarial multi-negative sampling strategy (Sun et al. 2019). This method makes the difference between positive and negative samples more obvious by means of multiple negative sampling, thus making it

easier to distinguish between positive and negative samples. The specific expression of the loss function is as follows:

$$d_r = \beta \cdot \text{sum}(T, R, C, S)$$

$$L_+ = -\log \sigma(\gamma - d_r(h, r, t))$$

$$L_- = -\sum_i^n k \log \sigma(d_r(h'_i, r'_i, t'_i) - \gamma)$$

where  $k = \frac{\exp(\alpha d_r(h'_i, r'_i, t'_i))}{\sum_i \exp(\alpha d_r(h'_i, r'_i, t'_i))}$ ,  $k$  is the weighting coefficient for multiple negative sampling,  $(h, r, t)$  is a positive sample,  $(h'_i, r'_i, t'_i)$  is a negative sample,  $n$  is the number of negative samples, and  $\alpha$  is the sampling temperature coefficient.

Meanwhile, since a binary classification approach is employed in the semantic section, a binary classification loss function is further introduced for the output of the semantic part.

$$L_{+Bert} = CE(S(h, r, t), 1)$$

$$L_{-Bert} = \sum_i^n CE(S(h'_i, r'_i, t'_i), 0)$$

where  $(h, r, t)$  is a positive sample,  $(h'_i, r'_i, t'_i)$  is a negative sample.

## 5 Experimental setup and analysis

In this section we will expand on the following aspects. First we present the details of the experimental setup. Then we show the effects of the proposed model on three data sets. Finally, we conduct a comprehensive analysis of the ensemble model in five parts. (1) The Structure Part Analysis section investigates the optimal scoring function selection for the T and R models. Then, We perform the ablation experiment on the structure part to validate the effectiveness of each part. (2) The Semantic Part Analysis section conducts the ablation experiment to validate the effectiveness of semantic. A set of comparative experiments is then designed to validate the features of Bert in extracting word co-occurrence. (3) The Frequency Part Analysis section verifies whether the model learned frequency features. (4) The Complementarity Of Structure And Semantic section involved experiments on the complementarity of structure and semantics to explore the possibility of discarding certain part. (5) The Complexity Analysis section provides an analysis of the model's complexity.

**Table 1** Statistics of datasets

Dataset	#E	#R	#TR	#TE
WN18RR	40,493	11	86,835	3,134
FB15k-237	14,541	237	272,115	20,466
YAGO3-10	123,182	37	1,079,040	5,000

The symbols #E and #R denote the number of entities and relations, respectively. #TR and #TE denote the size of train set and test set, respectively

### 5.1 Experimental setup

We analyze our model on three commonly used datasets: WN18RR(Toutanova and Chen 2015), FB15K-237(Dettmers et al.2018), YAGO3-10(Mahdisoltani, Biega and Suchanek 2013). Details of these datasets are summarized in Table 1.

WN18RR, FB15K-237 and YAGO3-10 are subsets intercepted from the three data sets WN18, FB15K and YAGO3. Since there is a large overlap between the test and training sets in the WN18, FB15K and YAGO3 datasets, simple models can also achieve good results. So we use WN18RR, FB15K-237 and YAGO3-10 as baseline models.

Test setup: we use the same test as Bordes et al. (2013). For the test triple  $(h, r, t)$ , we replace its head and tail entities with the entities in the candidate set to form a series of triples respectively. Then the score value of each triple is calculated and ranked. The main metrics we use are H@K, MR and MRR. H@K is the percentage of correct triples ranked in the top K. MR is the mean value of the correct triple ranking. MRR is the mean of the inverse of the correct triple ranking. We use the “Filtered” setting to calculate the results to avoid the impact of 1-N, N-1 and N-N problems.

Training setup: we use Adam (Kingma and Ba 2015) as the optimizer. The search range for the optimal values of  $p$  and  $q$  is 0-2, and the search step is 0.5.  $D$  (vector dimension) is set to 768.

**Table 2** Link prediction results on FB15k-237, YAGO3-10 and WN18RR(H@1, H@3, H@10)

	FB15K-237			YAGO3-10			WN18RR		
	H@1	H@3	H@10	H@1	H@3	H@10	H@1	H@3	H@10
TransE	—	—	0.465	—	—	—	0.025	0.132	0.501
RotatE	0.241	0.375	0.533	0.402	0.55	0.67	0.428	0.492	0.571
Rot-Pro	0.246	0.383	0.537	0.443	0.596	0.699	0.397	0.482	0.577
HAKE	0.25	0.381	0.542	0.462	0.596	0.694	0.427	0.516	0.582
QuatE	0.221	0.342	0.495	—	—	—	0.436	0.5	0.564
AttH	0.252	0.384	0.54	0.31	0.437	0.566	0.443	0.482	0.526
ConvE	0.237	0.356	0.501	0.35	0.49	0.62	0.4	0.44	0.52
ConvKB	—	—	0.517	—	—	—	—	—	0.525
KG-BERT	—	—	0.42	—	—	—	0.041	0.302	0.524
SAttLE	0.266	0.394	0.541	—	—	—	0.442	0.49	0.54
Ensemble	0.259	0.387	0.537	0.477	0.61	0.701	0.424	0.506	0.582

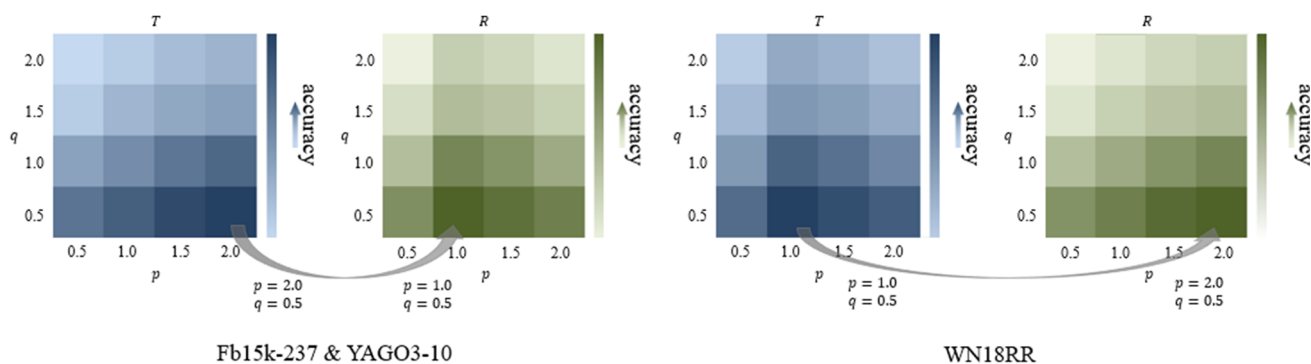
**Table 3** Link prediction results on FB15k-237, YAGO3-10 and WN18RR(MR, MRR)

	FB15K-237		YAGO3-10		WN18RR	
	MR	MRR	MR	MRR	MR	MRR
TransE	323	0.294	—	—	2300	0.226
RotatE	177	0.338	—	0.495	3340	0.476
Rot-Pro	201	0.344	—	—	2815	0.457
HAKE	—	0.346	—	0.545	—	0.497
QuatE	176	0.311	—	—	3472	0.481
AttH	—	0.324	—	0.397	—	0.466
ConvE	245	0.325	—	0.44	4464	0.43
ConvKB	257	0.396	—	—	2554	0.248
KG-BERT	153	—	—	—	97	0.216
SAttLE	—	0.373	—	—	—	0.503
Ensemble	174	0.353	—	0.558	3123	0.466

### 5.2 Main results

We compare the proposed model with a series of baseline models. Since the ensemble model primarily incorporates transE, rotatE, and Bert, we place special emphasis on analyzing the improvements achieved by the ensemble model in comparison to these three individual models. The results are shown in Tables 2 and 3.

For the FB15K-237 dataset, both transE and rotateE perform well. So it can be determined that both clustering and hierarchicality are represented on this dataset. Compared to rotateE, it is 1.8% higher on H@1, 1.36% higher on H@3 and 0.84% higher on H@10. Compared to HAKE, which also uses both translation and rotation models, the ensemble model is 0.9% higher on H@1, 0.6% higher on H@3. This shows that the ensemble model is effective. Based on the MR and MRR metrics, it can be inferred that the ensemble model pays more attention to overall ranking. Because the MR met-



**Fig. 9 Optimal parameters.** The left figure represents the same parameter settings for FB15k-237 and YAGO3-10 datasets. The right figure represents the parameter settings for the WN18RR dataset

ric focuses more on triples with lower rankings, while MRR is more concerned with triples ranked towards the top. Some models perform well in only one metric, such as ConvKB, while the ensemble model performs well in both metrics.

For YAGO3-10, the ensemble model achieved the best results across all metrics. Compared to rotateE, the ensemble model is 7.51% higher on H@1, 6.02% higher on H@3, 3.14% higher on H@10, and 0.063 higher on MRR. Compared to HAKE, the ensemble model is 1.5% higher on H@1, 1.5% higher on H@3, 0.7% higher on H@10, and 0.013 higher on MRR. Interestingly, YAGO3-10 is a large-scale dataset with a training set of one million, which means it has richer features compared to smaller datasets. The ensemble model performed better on larger datasets, demonstrating the advantage of the ensemble model in combining more features.

For the WN18RR, the ensemble model also perform well. Specifically, the transE model does not perform well on this dataset, but the ensemble model can leverage features from different models, thus still achieving competitive results. Therefore, for different datasets, the ensemble model exhibits greater adaptability.

### 5.3 Structure part analysis

#### 5.3.1 Optimal score function for structure part

We use a control variable approach. First, we randomly determine the parameters  $p$  and  $q$  in  $R$  and search for the parameters in  $T$ . Then fix the parameters in  $T$  and search the parameters in  $R$ .  $p$  and  $q$  parameters in the other terms are set to 2. Figure 9 shows the optimal  $p, q$  parameters searched on the FB15K-237, YAGO3-10 and WN18RR datasets. Among them, the FB15K-237 and YAGO3-10 datasets have the same parameter selection, where the  $T$  term is chosen to be 2-norm and the  $R$  term is chosen to be 1-norm. This indicates that the clustering feature constraint is tight while the hierarchical feature constraint is loose. In contrast, for the WN18RR

dataset, the  $T$  term is chosen to be 1-norm and the  $R$  term is chosen to be 2-norm. This indicates that the clustering feature constraint is loose while the hierarchical feature constraint is tight. In general, the generalized score function does optimize the results in terms of expressions.

#### 5.3.2 Ablation experiment

In this section, we take YAGO3-10 as an example. Training each part of the model separately and exploring the contribution of each part to the final accuracy. From Table 4, it can be seen that the direct combined learning of both  $T$  and  $R$  is not a significant improvement compared to the baseline model. And the introduction of the  $C$  term has significantly improved on YAGO3-10. Therefore the effective combination of features is quite important for prediction. In general, each term in the model contributes to the final prediction.

### 5.4 Semantic part analysis

We first conduct ablation experiments on the semantic part to analyze its impact on the results. We choose the FB15k-237 and WN18RR datasets, and the metrics Hit@1 and Hit@10. As shown in Table 5, the semantic part indeed brings a significant improvement to the WN18RR dataset, but the improvement on FB15k-237 is minimal. This is consistent with previous research findings.

**Table 4 Ablation Experiment On YAGO3-10,**  $T$  is the translation part(TransE in the baseline model),  $R$  is the rotation part(RotateE in the baseline model),  $C$  is the cross part, and  $P$  is the frequency part

T R C	MRR	H@1	H@3	H@10
✓	—	—	—	—
✓	0.495	0.402	0.550	0.670
✓ ✓	0.508	0.413	0.566	0.674
✓ ✓ ✓	0.552	0.473	0.603	0.695

**Table 5** Ablation experiment for semantic

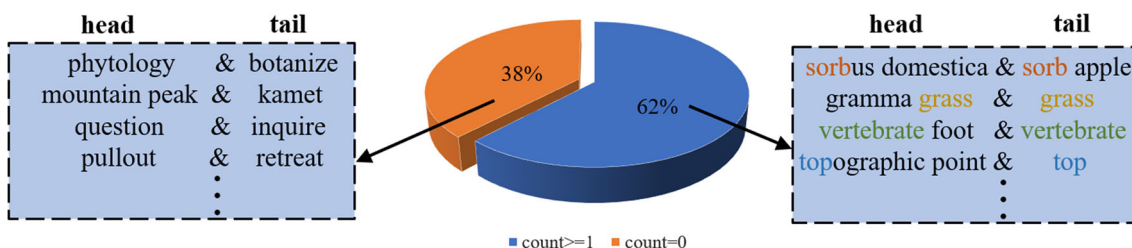
Semantic	FB15K-237		WN18RR	
	H@1	H@10	H@1	H@10
✓	0.259	0.537	0.424	0.582
×	0.257	0.536	0.412	0.561

✓ indicates that the ensemble model has semantic part. × indicates that the ensemble model does not have the semantic part

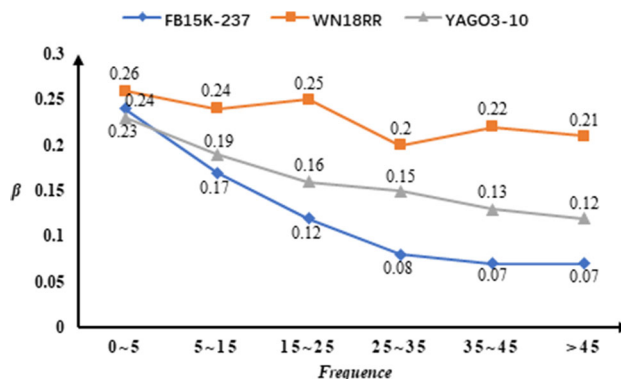
We further analyze the textual features of the two datasets. Since the WN18RR dataset is a word bank, entities may have the same word roots among themselves. Entities encoded through Bert contain the same information (word roots or the same categories). This property is referred to as word co-occurrence. Therefore, co-occurrence after word separation is helpful for WN18RR dataset. For the FB15K237 dataset, the entities contain a large number of names of people and places, which do not provide information about the same words (word roots or entity categories, etc.). Therefore, for the FB15K-237 dataset, simply encoding entity names by the Bert model cannot obtain valid information.

To verify this result, this paper designs the following comparison experiments: (i) encode only the entities. (ii) encode only the serial numbers (pre-generate a serial number for each entity, causing the textual information of the entities to be lost). For example, consider the original triple (fruit, hyponym, apple). First, introduce a serial number for each text, such as (fruit: 1, hyponym, apple: 2). Then, the entity part is removed, leaving only the serial number, such as (1, hyponym, 2).

On the original predicted results, due to the absence of entity textual information, there will inevitably be additional errors in predicting some triples. Excluding the results that were already predicted incorrectly in (i), we conducted a statistical analysis of the newly generated incorrect prediction results in (ii). From Fig. 10, it is evident that among the newly added incorrect predictions, 62% of the triples originally had entities with co-occurring words. The encoding of serial numbers resulted in the loss of this information. Therefore, we can see that the Bert model pays attention to



**Fig. 10** Entity co-occurrence of WN18RR dataset. count = 0 indicates no co-occurrence between the head and tail entities, while count ≥ 1 signifies co-occurrence



**Fig. 11** β-Frequency. Divide the frequency of entity occurrence in the training set into 6 intervals. Verify the relationship between β and the frequency interval on three datasets respectively

the co-occurrence of words, and the absence of words will result in a decline in Bert’s predictive ability.

### 5.5 Frequency part analysis

This section analyzes the relationship between the number of occurrences of knowledge pairs in the training set and the score of the frequency model(β), and detects whether the learning results of the frequency model are consistent with expectations.

The frequency model aims to learn the frequency features of knowledge pairs in the knowledge graph and reacts by the score of the model. To verify the law of frequency model learning, we divide the frequency of knowledge pairs into six intervals:0-5, 5-15, 15-25, 25-35, 35-45, ≥ 45. Compute the scores of all triples in each interval and then take the average. The relationship between frequency and the score is shown in Fig. 11. We can see that the score and frequency are inversely related. It means that the higher the frequency, the closer the distance between *h* and *t*. Among them, the inverse trend is most obvious in the FB15K-237 and YAGO3-10 datasets, Therefore the introduction of frequency is valid for both datasets. However, on the WN18RR dataset, the curve has some fluctuations. It means that the WN18RR dataset does not have a very distinct frequency profile.

## 5.6 Complementarity of structure and semantic

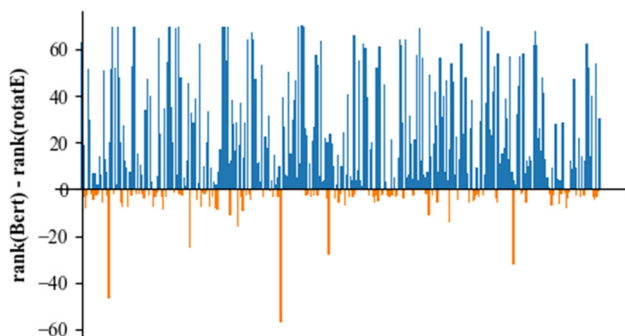
The structure part assigns vectors directly to each entity and relation, and it does not focus on each clause. Therefore, the same vocabulary after subsumption does not bring any help to the structure embedding. However, the structure part can focus on the overall distribution among the data (different structure designs focus on different aspects). Therefore, as long as similar structure features exist in the dataset, it is possible to achieve certain results without introducing additional knowledge.

The semantic part is embedded as a vector by the Bert model. It focuses on each subword, so that the connections that entities represent from words can be captured. However, when entities are abstracted (person names, etc.), it is difficult to capture the connections between entities in this way, and need to introduce more valid descriptive information.

For WN18RR, we choose the structure model rotateE(Rotation Model) and the semantic model Bert (with descriptive information) to embed the entities and relations of WN18RR, respectively, and then calculate the ranking of the test triples on the two models. As can be seen from Fig. 12, the Bert model achieves better results than the rotateE model on most of the triples. However, the rotateE model still achieves better results than the Bert model on a small number of triples, and even far better results than Bert on individual triples. This indicates that the semantic model cannot easily replace the holistic knowledge learned by the structure model.

## 5.7 Complexity analysis

In terms of time complexity analysis, the gradient descent algorithm has a time complexity of  $O(NCL)$ , where  $N$  is the batch size,  $C$  is the computation per single sample, and  $L$  is the number of iterations. For the structure part, the time complexity per single sample is  $O(d)$ , where  $d$  is the dimension of the embedding vectors, as it involves only norm calculations.



**Fig. 12** The predicted rankings obtained with the structure and semantic models, respectively, then subtracted(On WN18RR)

**Table 6** Training time

	Structure	Semantic
Training Gpus	1	4
Batch size	512	4
Iterations	20000	4
Training way	–	Fine-tuning
Time	2.5h	5h

In the semantic part, the computational load per single sample mainly depends on the self-attention mechanism within the Bert model. Generally, the time complexity of self-attention is  $O(kd^2)$ , where  $k$  is the number of model layers, and  $d$  is the dimension of the embedding vectors. Therefore, the overall time complexity for Algorithm 1 over one pass is approximately  $O(N(k^2d + 3d)L)$ , where  $3d$  represents the computations required for the translation model, rotation model, and interaction model within the structure part.

In terms of space complexity analysis, the space complexity of the structure part is mainly determined by the number of samples and the dimension of the embedding vectors for each sample. Thus, its space complexity is  $O(Vd)$ , where  $V$  is the size of the vocabulary (the number of entities and relations). The space complexity of the semantic part is primarily determined by the number of parameters. When considering only embedding vectors and self-attention matrices, its space complexity is  $O(Vd + kd^2)$ .

We have provided the parameter settings during training and the final runtime, as shown in Table 6. It can be observed that due to fine-tuning, the semantic part has smaller values for iterations and batch size. However, Bert, with its high time complexity, results in a longer runtime compared to the structure part. Algorithm 1 takes approximately 7 hours to complete.

## 6 Conclusion

To investigate what knowledge embedding models can represent, this paper conduct theoretical proofs. Building upon this foundation, we employ ensemble learning to integrate various models, aiming to achieve higher accuracy and broader applicability. Experimental results show that our proposed method outperforms existing approaches on most benchmark datasets and remains competitive on a few others. From our research, we can summarize several practical recommendations.

(1) If entities exhibit shared characteristics based on relationship descriptions, such as ([London, Oxford, Cambridge], city of, United Kingdom), introducing translation models capable of extracting clustering features is effective.



(2) If entities exhibit a clear hierarchical structure based on relationship descriptions, such as (fruit, hyponym, [apple, banana, grape]), (fruit, hypernym, [food, nature]), introducing rotation models capable of extracting hierarchical features is effective.

(3) If entities possess rich textual information, such as (Beethoven: enjoy playing music, profession, musician), incorporating neural networks like Bert can extract more information (e.g., music and musician).

In summary, selecting appropriate models based on their features can enhance the representational capacity of embedding models.

**Funding** This work was supported in part by the National Natural Science Foundation of China (NSFC) (92267205), the National Natural Science Foundation of China (NSFC) (U1911401), and in part by the Science And Technology Innovation Program of Hunan Province in China (2021RC4054).

**Availability of data and materials** The authors declare that the data and materials are reliable.

## Declarations

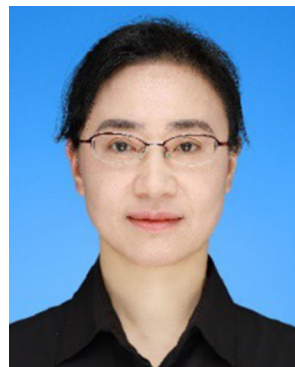
**Conflict of Interests** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References

- Wang Q, Mao Z, Wang B, Guo L (2017) Knowledge graph embedding: a survey of approaches and applications. *IEEE Trans Knowl Data Eng* 29(12):2724–2743
- Wang M, Qiu L, Wang X (2021) A survey on knowledge graph embeddings for link prediction. *Symmetry* 13(3):485
- Zhang Z, Cai J, Zhang Y, Wang J (2020) Learning hierarchy-aware knowledge graph embeddings for link prediction. *Proceedings of the AAAI conference on artificial intelligence* 34:3065–3072
- Bordes A, Usunier N, Garcia-Duran A, Weston J, Yakhnenko O (2013) Translating embeddings for modeling multi-relational data. *Adv Neural Inf Process Syst* 26
- Sun Z, Deng ZH, Nie JY, Tang J (2019) Rotate: Knowledge graph embedding by relational rotation in complex space. [arXiv:1902.10197](https://arxiv.org/abs/1902.10197)
- Ebisu T, Ichise R (2019) Generalized translation-based embedding of knowledge graph. *IEEE Trans Knowl Data Eng* 32(5):941–951
- Le T, Huynh N, Le B (2022) Knowledge graph embedding by projection and rotation on hyperplanes for link prediction. *Appl Intell* 1–25
- Ren F, Li J, Zhang H, Yang X (2020) Transp: a new knowledge graph embedding model by translating on positions. In: *2020 IEEE International Conference on Knowledge Graph (ICKG)*, pp 344–351. IEEE
- Song T, Luo J, Huang L (2021) Rot-pro: modeling transitivity by projection in knowledge graph embedding. *Adv Neural Inf Process Syst* 34:24695–24706
- Zhang S, Tay Y, Yao L, Liu Q (2019) Quaternion knowledge graph embeddings. *Adv Neural Inf Process Syst* 32
- Chami I, Wolf A, Juan DC, Sala F, Ravi S, Ré C (2020) Low-dimensional hyperbolic knowledge graph embeddings. [arXiv:2005.00545](https://arxiv.org/abs/2005.00545)
- Chen W, Zhao S, Zhang X (2023) Enhancing knowledge graph embedding with type-constraint features. *Appl Intell* 53(1):984–995
- Li M, Sun Z, Zhang S, Zhang W (2021) Enhancing knowledge graph embedding with relational constraints. *Neurocomputing* 429:77–88
- Dettmers T, Minervini P, Stenetorp P, Riedel S (2018) Convolutional 2d knowledge graph embeddings. In: *Proceedings of the AAAI conference on artificial intelligence*, vol 32
- Feng J, Wei Q, Cui J, Chen J (2022) Novel translation knowledge graph completion model based on 2d convolution. *Appl Intell* 52(3):3266–3275
- Devlin J, Chang MW, Lee K, Toutanova K (2018) Bert: pre-training of deep bidirectional transformers for language understanding. [arXiv:1810.04805](https://arxiv.org/abs/1810.04805)
- Yao L, Mao C, Luo Y (2019) Kg-bert: bert for knowledge graph completion. [arXiv:1909.03193](https://arxiv.org/abs/1909.03193)
- Baghersshahi P, Hosseini R, Moradi H (2023) Self-attention presents low-dimensional knowledge graph embeddings for link prediction. *Knowl-Based Syst* 260:110124
- Duan H, Liu P, Ding Q (2023) Rfan: relation-fused multi-head attention network for knowledge graph enhanced recommendation. *Appl Intell* 53(1):1068–1083
- Van DTT, Lee YK (2023) A similar structural and semantic integrated method for rdf entity embedding. *Appl Intell* 1–15
- Chen L, Cui J, Tang X, Qian Y, Li Y, Zhang Y (2022) Rlpath: a knowledge graph link prediction method using reinforcement learning based attentive relation path searching and representation learning. *Appl Intell* 1–12
- Fang Y, Wang H, Zhao L, Yu F, Wang C (2020) Dynamic knowledge graph based fake-review detection. *Appl Intell* 50:4281–4295
- Bordes A, Weston J, Collobert R, Bengio Y (2011) Learning structured embeddings of knowledge bases. *Proceedings of the AAAI conference on artificial intelligence* 25:301–306

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

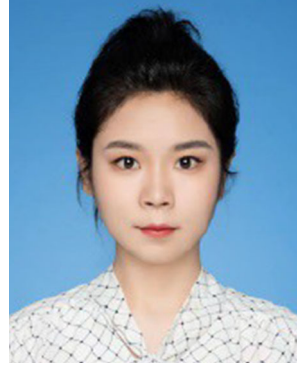


**Yalin Wang** received the B.Eng. degree in automation and the Ph.D. degree in control science and engineering from the Department of Control Science and Engineering, Central South University, Changsha, China, in 1995 and 2001, respectively. She is currently a Professor with the School of Automation, Central South University, Changsha, China. Her research interests include the modeling, optimization and control for complex industrial processes, intelligent control, and

process simulation.



**Yubin Peng** received the B.Eng. degree in Automation Engineering from the Department of Control Science and Engineering, Central South University, Changsha, China, in 2022. He is currently a graduate student in the Department of Automation at Central South University. His research interests include knowledge representation embedding, knowledge distillation, and image recognition.



**Jingyu Guo** received the B.Eng. degree in Automation Engineering from the Department of Control Science and Engineering, Central South University, Changsha, China, in 2019. She is currently pursuing the Ph.D degree with the School of Automation, Central South University, Changsha, China. Her research interests include knowledge representation embedding, knowledge distillation, and NLP.