Knowledge-Based Systems 284 (2024) 111253

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

SDFormer: A shallow-to-deep feature interaction for knowledge graph embedding

Duantengchuan Li [a,1], Tao Xia [a], Jing Wang [b], Fobo Shi [c,1], Qi Zhang [d,*], Bing Li [a,e,*], Yu Xiong [f]

[a] School of Computer Science, Wuhan University, Wuhan 430072, China
[b] School of Automation, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
[c] National Engineering Research Center for E-Learning, Central China Normal University, Wuhan 430079, China
[d] School of Information Management, Central China Normal University, Wuhan 430079, China
[e] Hubei Luojia Laboratory, Wuhan 430079, China
[f] Research Center for Artificial Intelligence and Smart Education, Chongqing University of Posts and Telecommunications, Chongqing 400065, China

ARTICLE INFO

Keywords:
Link prediction
Knowledge graph embedding
Shallow interaction
Deep interaction
Attention mechanism
Vector tokenization

ABSTRACT

Inferring missing information from current facts in a knowledge graph (KG) is the target of the link prediction task. Currently, existing methods embed the entities and relations of KG as a whole into a low-dimensional vector space. Nonetheless, they ignore the multi-level interactions (shallow interactions, deep interactions) among the finer-grained sub-features of entities and relations. To overcome these limitations, we present a shallow-to-deep feature interaction for knowledge graph embedding (SDFormer). It takes into account the interpretability of sub-feature tokens of entities and relations and learns shallow-to-deep interaction information between entities and relations at a more fine-grained level. Specifically, entity and relation vectors are decomposed into sub-features to represent multi-dimensional information. Then, a shallow-to-deep feature interaction method is designed to capture multi-level interactions between entities and relations. This process enriches the feature representation by modeling the interaction between sub-features. Finally, a 1-X scoring function is utilized to calculate the score of each knowledge triplet. The experimental results on several benchmark datasets show that SDFormer obtains competitive performance results and more efficient training efficiency on other comparative models and because of the shallow-to-deep feature interaction between entities and relations.

1. Introduction

Freebase [1] and WordNet [2] are two examples of knowledge graphs, which are structured representations of facts with nodes designating entities and edges designating relations between them. KGs are widely adopted in different scenarios, like question answering systems, recommendation systems [3,4], and web search. Nonetheless, there are usually a large number of absent links in existing knowledge graphs. The link prediction task [5–7] is to infer the missing knowledge triplets from the existing knowledge graph to alleviate the drawbacks in the existing KGs.

At present, massive methods for link prediction tasks already exist. These approaches can in turn be divided into three branches: (1) translational distance methods, which calculates the translational distance between subject entity and object entity vectors via relation vector, such as [8–11], (2) semantic matching (or tensor decomposition) methods, which determines the plausibility of a knowledge triplet by matching the latent semantic information between entities and relations [12–14], and (3) neural network-based methods, which utilizes embedding-independent parameters to store the learned knowledge, so that they can enrich feature representations, like [15–17]. More recently, there are a lot of other views which employ Transformer [18] for the task of link prediction. For instance, KG-BERT [19] evaluates the confidence of the triplet by fine-tuning the pre-trained language model. In addition to this, StAR [20] and Ruleformer [21] also have employed Transformer in a different way for Knowledge Graph Embedding (KGE).

The above models map entities and relations into a low-dimensional space to generate a single static embedding, and learn how entities and relations interact mutually via simple vector splicing and convolutional networks. As a result, they ignore the multi-level interactions among finer-grained features. As shown in Fig. 1, to accurately predict the triplet $(Curry, Occupation, ?)$, we first initialize the subject entity and relation vectors. For the subject entity Curry, each dimension of its

* Corresponding author.
E-mail addresses: dtclee1222@gmail.com (D. Li), qi.zhang.3519@gmail.com (Q. Zhang), bingli@whu.edu.cn (B. Li).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.knosys.2023.111253
Received 23 February 2023; Received in revised form 13 November 2023; Accepted 30 November 2023
Available online 13 December 2023

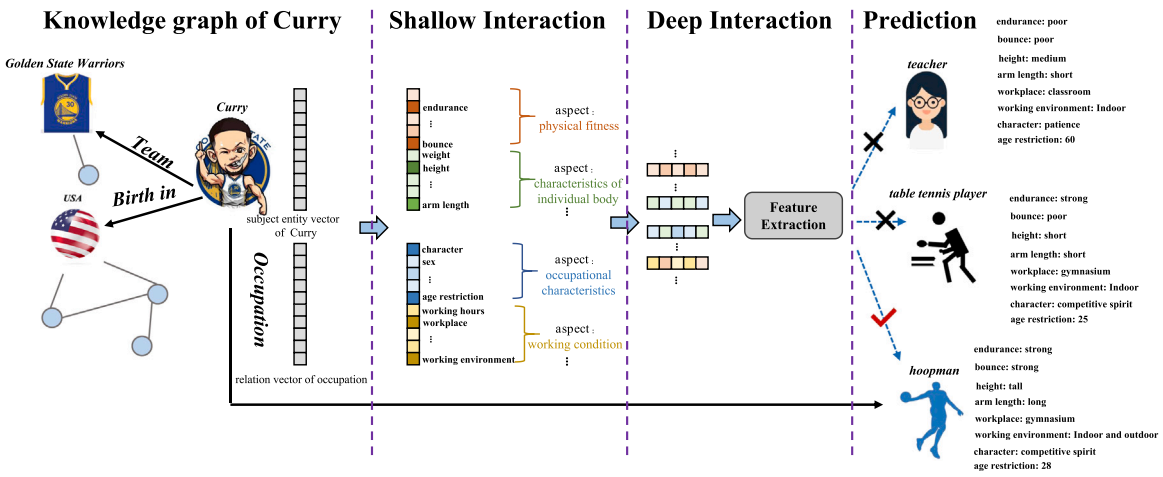

Fig. 1. An example of shallow and deep interactions between entity and relation.

features represents a certain attribute feature of Curry. Intuitively, every entity can be characterized from multiple perspectives or aspects. Therefore, it is natural to think that the subject entity and relation vectors can be divided into several groups or aspects, i.e., multiple aspects, as shown in the feature vectors in the figure, as shown in the Shallow Interaction phase in Fig. 1. Therefore, intuitively, when predicting Curry's profession, features such as endurance and bounce in physical fitness, as well as height and arm length in characteristics of individual body, and character and age restriction in occupational characteristics, as well as workplace and working environment in working condition, are the most useful for determining Curry's profession. For example, if Curry has strong "endurance" and "bounce", and is tall in "height", it would be assumed that Curry could be a volleyball, basketball, or high jump athlete. But when the relational attribute "workplace" is a stadium, and the "working environment" can be indoors or outdoors, it leans more towards being a basketball athlete. Therefore, the interaction of features based on different aspects in entities and relations is key to obtaining multi-angle feature representations.

Considering the feature interactions between entities and relations from different levels, we propose a shallow-to-deep feature interaction model for knowledge graph embedding (SDFormer). It breaks up the embedding vectors of entities and relations into finer-grained sub-features, and learns both shallow and deep interactions between entities and relations. SDFormer is composed of three main modules: (1) Vector Tokenization: an entity or relation vector is decomposed into finer-grained sub-features by setting the token size. (2) A shallow-to-deep feature interaction module: a multi-head self-attention mechanism for shallow interaction and a multi-scale convolutional neural network for deep interaction information between entities and relations. (3) Score prediction: The updated feature vector is projected into the embedding space of entities, and the sigmoid function is performed to obtain the scores of all candidate entities and the one with top score is chosen as the predicted object entity. Experiments and ablation studies show that our model can learn both shallow and deep interaction information between entities and relations, thereby achieving competitive performance results on benchmark datasets and a smaller parameter size for practical knowledge graph modeling.

The research contributions of our work are as follows:

- We propose a shallow-to-deep feature interaction for knowledge graph embedding (SDFormer), which employs a multi-head self-attention mechanism for shallow interaction and a multi-scale convolution network for deep interaction.
- We propose a vector tokenization method to decompose entities and relations embeddings into sub-features, which guarantees the interactivity between sub-features of entities and relations.
- Extensive experiments have demonstrated that on most benchmark datasets, SDFormer achieves competitive prediction results and superior training efficiency with a smaller parameter size compared to convolution-based and Transformer-based KGE models.

2. Related work

Employing models based on KGE to infer absent facts in KGs has been widely explored in last decades. Current KGE models are practically divided into *translational distance models*, *semantic matching models*, and *neural network-based models*.

2.1. Translational distance models

The earliest translational distance-based KGE model can be traced back to TransE [8]. Given a triplet (e_s, r, e_o), TransE attempts to measure the translational distance between the subject entity vector $\mathbf{e}_s$ and the object entity vector $\mathbf{e}_o$ with the relation vector $\mathbf{r}$. TransE is a plain, swift and effective model for large-scale knowledge graphs. Nonetheless, it only works for 1-to-1 relations and does not handle reasoning about complex relations effectively, like 1-to-N relations or N-to-N relations.

To alleviate the dilemma, TransH [9] proposes that each entity should behave differently for different relations. Therefore, a hyperplane is defined for each relation so that the same entity has different feature representations in different relations. RotatE [22] defines each relation as a rotation from a source entity to a target entity in complex vector space. Nevertheless, the features learned by the translational distance models are not sufficient in contrast to the deep, multi-layer models, thus making it difficult to achieve large improvements with complex conditions.

2.2. Semantic matching models

Considering that translational distance models are difficult to express complex relations effectively, the researchers match the potential semantic information in the vector space to calculate the reasonableness of the triplet. RESCAL [12] is a classical model using semantic matching method, which converts entities into vectors and relations into matrices, and acquires the internal interactions of triplets through a bi-linear scoring function. DistMult [13] streamlines the relation matrix of RESCAL by limiting it to a diagonal matrix. HolE [23] presents both entities and relations as vectors in space, combining the representational strength of RESCAL with the simplicity and effectiveness of DistMult. It defines a circular association operation where subject and object entities interact, and the result of the operation is matched with

Table 1
The main notations employed in this paper.

Notations	Explanations
$\mathcal{G}$	Knowledge Graph
(e_s, r, e_o)	Triplet
$\mathbf{e}_s, \mathbf{r}, \mathbf{e}_o \in \mathbb{R}^d$	Embedded semantic vector of subject entity, relation, and object entity separately
$\mathcal{E}$, $\mathcal{R}$, $\mathcal{T}$, $\mathcal{T}'$, $\mathcal{T}_{test}$	Set of entities, relations, triplets, negative triplets, and test triplets
$\lvert\mathcal{E}\rvert$, $\lvert\mathcal{R}\rvert$, $\lvert\mathcal{T}_{test}\rvert$	Number of entities, relations and test triplets
$\mathbf{E}, \mathbf{R}$	Embedding matrices of entities and relations
$\mathbf{M}, \mathbf{M}'$	The input matrix after embedding and reshaping, and feature matrix after shallow interaction
$\mathbf{U} \in \mathbb{R}^d$	Semantic interaction information vector of head entity and relation
SI	*Shallow Interaction* module
$\mathbf{w}_i^i, \mathbf{v}_i$	The ith convolution kernel and feature map respectively
W	Learnable parameter matrix
$\varphi(e_h, r, e_o)$	Score function of SDFormer
Θ	All the parameters of SDFormer
$\mathcal{L}$	Loss function

the representation of the relation to compute the triplet score. Based on the framework of DistMult, ComplEx [14] introduces complex-valued embedding, in which entities and relations are embedded into the complex space rather than the real-valued space. ANALOGY [24] improves RESCAL to reflect the inference characteristics of entities and relations more accurately. PairRE [25] is equipped with paired relationship vectors for each relation in order to handle complex relations and various relation patterns simultaneously.

Nonetheless, the semantic matching models tend to be less applicable for large KGs, because they need more embedding dimensionality of KGs to enhance their expressiveness. As a consequence, these models tend to be easier to overfit, which is often a tough issue to solve when the knowledge graphs contain a significant number of entities and relations.

2.3. Neural network-based models

Nowadays, researchers are focusing more on putting deep learning to work for link prediction. R-GCN [26] is designed for processing extremely multi-relational KGs with Graph Neural Networks (GCN) [16], which utilizes convolutional operations to capture local information in the knowledge graph. ConvE [15] performs a two-dimensional convolution of the embedding vector to predict the missing links. Its architecture contains a convolutional layer, a projection layer and a matrix multiplication for the final prediction. InteractE [17] suggests that semantic interaction information is limited in ConvE, thus they propose to use multiple feature combination approaches and circular convolution operations for obtaining better interaction information. REP-OTE [27] proposes an information aggregation method for graph neural networks based on relational embedding.

Inspired by the great improvement of Transformer [18] in different fields, some Transformer-based KGE models have also been proposed for link prediction. These transformer-based KGE models can be mainly classified into two categories: (1) Pure transformer: only the transformer architecture is improved to learn entity and relation embeddings in KG. StAR [20] was the first to attempt the link prediction task with the transformer, a method that injects structural features into the entity and relation embeddings obtained through the transformer encoder. Ruleformer [21] joins rule learning and transformer to mine subgraph context information in KG to achieve link prediction. (2) Variants of transformer: these models introduce the description text of entities as auxiliary information, and use variants of transformer to learn the textual descriptions of entities. KG-BERT [19] is the first to evaluate the confidence of a triplet by fine-tuning the pre-trained language model. BLP-TransE [28] incorporates text as auxiliary information into the learned entity representation through the link prediction task. SimKGC [29] utilizes the pre-trained language model based on contrastive learning to enhance prediction accuracy. Inspired by [28], RAILD [30] learns about unseen entities and relations using relational features in the link prediction task. SDFormer belongs to the Pure transformer type of Transformer-based Model.

However, majority of the Transformer-based models mentioned above are built using pre-trained language models. Transformer has demonstrated excellent global feature interaction aggregation and capabilities in many works. In many different disciplines, its self-attention mechanism is extensively employed. Taking into account that the interaction between entities and relations is not only mutual but also self-interactive, our work further improves the interaction between entities and relations by multi-head self-attention mechanism to make entities and relations interact themselves, and by convolution to capture information of cross-interaction between entities and relations.

3. Methodology

3.1. Problem formulation

Knowledge Graph is normally described as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E} = (e_1, e_2, \cdots, e_{|\mathcal{E}|})$ represents the set of entities and $\mathcal{R} = (r_1, r_2, \cdots, r_{|\mathcal{R}|})$ represents the set of relations. $|\mathcal{E}|$ and $|\mathcal{R}|$ denote the total number of entities and relations, respectively. $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ expresses knowledge triplets set and each triplet can be represented as (e_s, r, e_o). Normally we use bold letters $\mathbf{e}_s$, $\mathbf{r}$, $\mathbf{e}_o \in \mathbb{R}^d$ to express the d-dimensional embeddings of e_s, r, and e_o, respectively.

Link Prediction. For a triplet $(e_s, r, ?)$, given the subject entities and relations, the goal of link prediction is to select the most suitable one from the candidate entities as the object entities. To calculate the credibility of the predicted triplet, a score function is generally defined for the triplet. Then an optimization problem is solved for maximizing the plausibility of all true triplets $\mathcal{T}$ in the KG. Finally, the entity and relation embeddings could be well learned. Table 1 lists the main notations employed throughout this article.

3.2. Outline of SDFormer

Considering that Transformer has limited extraction of interaction information for subject entities and relations while convolutional neural networks have a powerful receptive field, we combine the advantages of both to propose the SDFormer. SDFormer consists of three components, namely, vector tokenization, feature interaction, and score prediction. For a triplet (e_s, r, e_o), we first index to the vector representations of subject entity e_s and relation r and divide them into t tokens individually. For obtaining more detailed interaction information among sub-features of entities and relations, a multi-head self-attention mechanism is utilized for shallow interactions. Further, multi-scale convolutional network is utilized to obtain the deep interaction between entities and relations. Then the semantic vector between entity and relation is generated through a hidden layer. Finally, the semantic vector is matched with all candidate entities embeddings to judge whether the triplet is proper. The general architecture of SDFormer is illustrated in Fig. 2

3.2.1. Vector tokenization

Both entities and relations are composed of multiple fine-grained sub-features. Take *Stephen Curry* as an example, it can be further decomposed into sub-features such as "height", "weight" "age" and so on. In order to decompose entities and relations vectors into finer-grained sub-features, we are inspired by tokens in Vision Transformer [31] to tokenize the entity and relation vectors. The specific steps are as follows.

For a knowledge graph $\mathcal{G}$, we initialize all entities $\mathcal{E}$ and relations $\mathcal{R}$ as the d-dimension embeddings. In terms of a specific triplet (e_s, r, e_o),

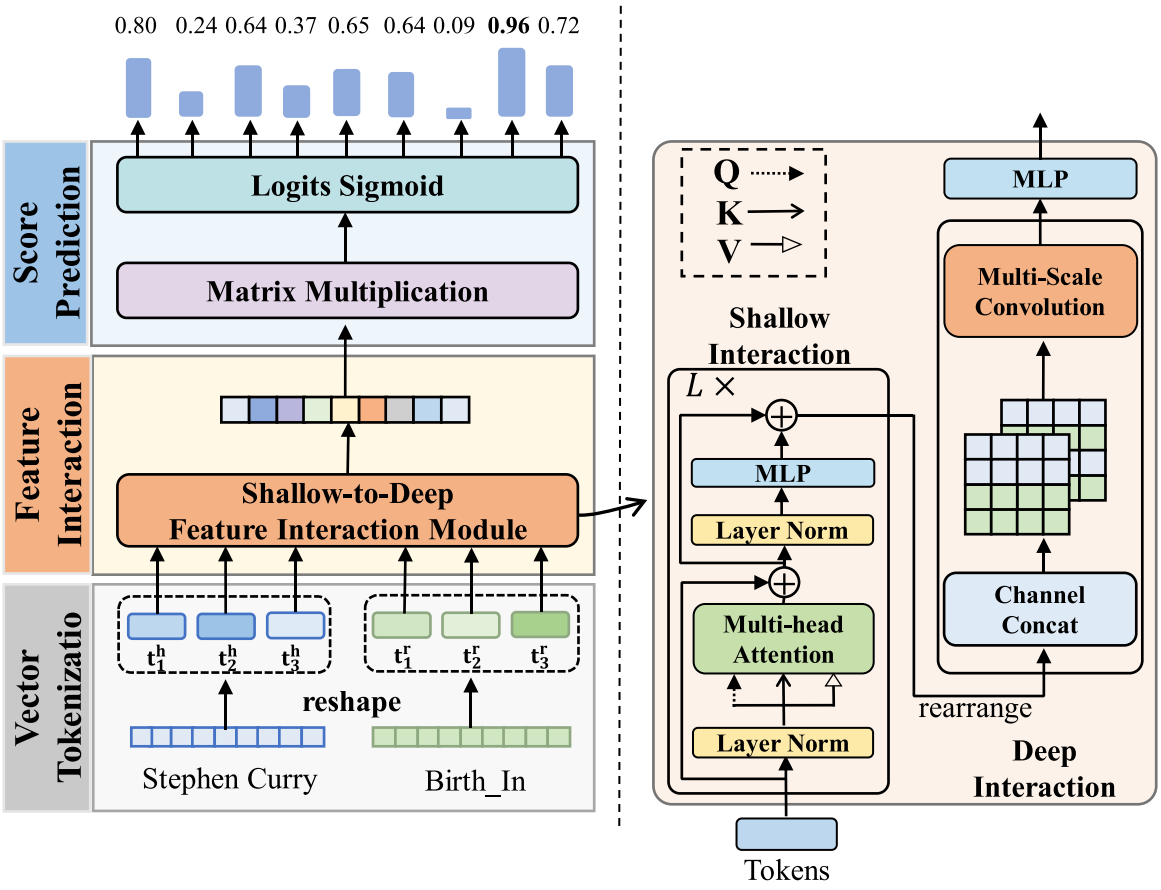

Fig. 2. The general architecture of SDFormer. The left part is the three parts of this model, namely vector tokenization, feature interaction and score prediction.

the embedding representations of subject entity and relation can be indexed through the embedding matrices respectively as $\mathbf{e}_s, \mathbf{r} \in \mathbb{R}^d$. First, the token size is set as t, and token splitting is performed for entities and relations, respectively. Then, the embedddings of entities and relations are transformed into two set of tokens, and layer normalization [32] is employed to normalize sub-features, separately. Finally, the sub-feature tokens of entities and relations are spliced as the input matrix $\mathbf{M}$ of the subsequent model.

$$\mathbf{M} = \left[ln(\overline{\mathbf{e}_s}) \oplus ln(\overline{\mathbf{r}})\right] \in \mathbb{R}^{2\times t\times d_m}, \tag{1}$$

where $\oplus$ denotes the concatenation operation, and $\overline{\mathbf{e}_s}, \overline{\mathbf{r}}$ denote the vectors of the subject entity and relation are sliced into t tokens respectively and $t \times d_m = d$. We use VT to represent the above steps.

3.2.2. Feature interaction

A multi-head self-attention mechanism is employed to compute the correlation between sub-features in entities and relations to simulate the interaction between sub-features. The shallow interaction is now completed, forming a new token embedding that includes various sub-features. On this basis, the updated sub-features are merged upwards into "aspect features" and rearranged with relations. The merged features undergo multi-scale convolution operations to extract multiple relations and complete the deep interaction between entities and relations.

Shallow Interaction (SI): Sub-features in entities and relations interact with each other. To enrich the representation of each sub-feature token, a multi-head self-attention mechanism is used to extract shallow interactions between sub-features in entities and relations.

For each triplet $(e_s, r, ?)$, we generate the query $\mathbf{M}^Q \in \mathbb{R}^{2\times t\times d_m}$, the key $\mathbf{M}^K \in \mathbb{R}^{2\times t\times d_m}$, and the value $\mathbf{M}^V \in \mathbb{R}^{2\times t\times d_m}$ by performing three different projection on the input matrix $\mathbf{M}$ with matrices $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V \in \mathbb{R}^{d_m\times d_m}$. Take the single-head attention as an example, the attention matrix $\mathbf{A} \in \mathbb{R}^{2t\times 2t}$ is calculated as:

$$\mathbf{A} = \mathrm{softmax}(\frac{\mathbf{M}^Q \mathbf{M}^{K^T}}{\sqrt{d_m}}), \tag{2}$$

Attention matrix $\mathbf{A}$ shows the attention weight of sub-features to the other sub-features in an entity or relation. For example, attention scores will be higher for sub-feature "height" with sub-features "weight" and "shoe size" and lower with sub-feature "skin color". Later the attention matrix is multiplied with the value matrix $\mathbf{M}^V$ to obtain the output $\mathbf{H} \in \mathbb{R}^{2t\times d_m}$ of the attention module block as:

$$\mathbf{H} = \mathbf{A}\mathbf{M}^V, \tag{3}$$

To obtain multi-dimensional shallow interactions, multi-head self-attention (MultiH) is a good choice for obtaining more expressive representations. Multiple linear projections are conducted to generate multiple sets of queries, keys and values with various sets of parameters. Then, self-attention mechanism is applied to each set of query, key, and value. Finally, all value matrices are spliced and linearly reconstructed. The formula is expressed as follows:

$$\mathrm{MultiH}(\mathbf{M}^Q, \mathbf{M}^K, \mathbf{M}^V) = [\mathbf{H}_1, \mathbf{H}_2, \cdots \mathbf{H}_h]\mathbf{W}_O, \tag{4}$$

where

$$\mathbf{H}_i = \mathrm{softmax}\left(\frac{\mathbf{M}_i^Q \mathbf{M}_i^{K^T}}{\sqrt{d_k}}\right)\mathbf{M}_i^V, \tag{5}$$

In the above equation, $\mathbf{M}_i^Q = \mathbf{M}^Q\mathbf{W}_i^Q, \mathbf{M}_i^K = \mathbf{M}^K\mathbf{W}_i^K$ and $\mathbf{M}_i^V = \mathbf{M}^V\mathbf{W}_i^V$, where $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V \in \mathbb{R}^{d_m\times d_k}$, and $\mathbf{W}^O \in \mathbb{R}^{h\times d_k\times d_m}$ are projection matrices. h and d_k denote the number of heads and the dimension of each head, respectively.

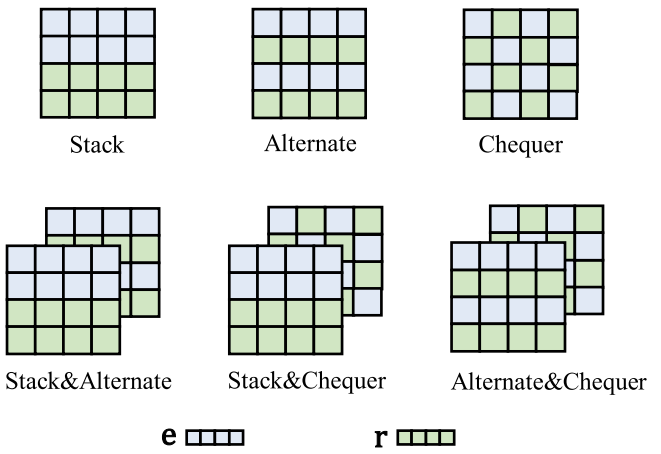

Fig. 3. Six different token splicing methods utilized in this paper. Here, **e** and **r** represent the Token of entitie and relation after shallow interaction, correspondingly.

Just like the normal Transformer model, a feed-forward network is conducted for generating the eventual result $\mathbf{M}'$ as:

$$\mathbf{M}' = FFN(\text{MultiH}(\mathbf{M}^Q, \mathbf{M}^K, \mathbf{M}^V)), \tag{6}$$

where

$$FFN(x) = f(x\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2, \tag{7}$$

where f denotes the activate function GELU [33], and $\mathbf{W}_1 \in \mathbb{R}^{d_m \times d_h}, b_1 \in \mathbb{R}^{d_h}, \mathbf{W}_2 \in \mathbb{R}^{d_h \times d_m}$ and $b_2 \in \mathbb{R}^{d_m}$ are learnable parameters.

The above-mentioned multi-head attention mechanism and feed-forward network (FFN) constitute a single layer of *Shallow Interaction* (*SI*) module. For the sake of fully mining the interaction information between entities and relational sub-features, SDFormer implements a L-layer *Shallow Interaction* module. The value of L will be discussed in the experiment section.

Deep Interaction (DI): To induce deep integration of the aspect features of entities and relations, six different splicing methods are adopted to rearrange and combine the feature vectors of entities and relations, i.e. Stack [15], Alternate [17], Chequer [17], Stack&Alternate, Stack&Chequer and Alternate&Chequer. We give the definition below and the formalized stitching is shown in Fig. 3.

- **Stack**: The tokens of entities and relations are directly connected to spell a matrix.
- **Alternate**: The tokens of entities and relations are sorted by index, forming a staggered patchwork of tokens.
- **Chequer**: The tokens are arranged such that no two adjacent cells are occupied by components of the same embedding.
- **Stack&Alternate**: The tokens of entities and relations are stitched by Stack and Alternate through a picture-like multi-channel format.
- **Stack&Chequer**: The same way to stitch Stack and Chequer.
- **Alternate&Chequer**: The same way to stitch Alternate and Chequer.

There are often complex relations in the KGs, and the same entity corresponds to different facts under different relations. Take apple as an example, when referring to food, it is a fruit, and when referring to a product, it could be a cell phone, computer, etc. Considering that entities exhibit different semantic features under specific relations, we employed multi-scale convolutional kernels to extract different aspects of features to better extract complex relations in the knowledge graph.

As shown in Fig. 4, we refine the feature extraction with several distinct convolution kernels $(\mathbf{w}_r^1, \mathbf{w}_r^2 \cdots \mathbf{w}_r^n)$ with various size to obtain various feature maps$(\mathbf{v}_1, \mathbf{v}_2 \cdots \mathbf{v}_n)$. Specifically we use kernels $\left[\mathbf{w}_r^1 \in \mathbb{R}^{1\times 9}, \mathbf{w}_r^2 \in \mathbb{R}^{2\times 4}, \mathbf{w}_r^3 \in \mathbb{R}^{3\times 3}, \cdots\right]$ to capture multi-level interaction information between sub-features vectors of entities and relations. The mathematical formulation is as follows.

$$\mathbf{v}_n(i,j) = f(\mathbf{M}' * \mathbf{w}_r^n) = f(\sum_{a=1}^{2\times t}\sum_{b=1}^{d_m} \mathbf{M}'(i+a, j+b)\,\mathbf{w}_r^n(a,b)), \tag{8}$$

where $*$ denotes the convolution operation, and $f(x)$ represents the rectified linear units (RELU) [34]. Then all feature maps are flattened. Later a linear transformer parameterized by the matrix $\mathbf{W}$ is performed to project the flattened feature maps into a d-dimension vector space to gain the semantic interaction vector $\mathbf{U} \in \mathbb{R}^d$ as follows:

$$\mathbf{U} = f(\text{dense}(\mathbf{v}_1, \mathbf{v}_2, \cdots \mathbf{v}_n)\mathbf{W}), \tag{9}$$

where $\text{dense}(\cdot)$ denotes the flattening operation, $\mathbf{W}$ is the learnable parameter matrix which maps the interaction features between entities and relations into the entity space. The vector $\mathbf{U}$ holds information about the shallow-to-deep interaction between subject entity e_s and relation r.

3.2.3. Score prediction

To calculate the plausibility score of the triplet, the semantic interaction vector $\mathbf{U}$ will be utilized to do matrix multiplication with the object entities embeddings. The score function $\varphi(e_s, r, e_o)$ of the SDFormer is described as follows:

$$\varphi(e_s, r, e_o) = \sigma(\mathbf{U} \cdot \mathbf{e}_o + b) \in (0, 1), \tag{10}$$

where b is the bias item, and $\sigma(x) = 1/(1 + \exp(-x))$ is the logistic sigmoid function to calculate the probabilistic score to judge whether the triplet (e_s, r, e_o) is proper or not. With equations (1)–(10), the score function is defined in detail as follows:

$$\begin{aligned}&\varphi(e_s, r, e_o) = \\ &\sigma(f(\text{dense}(f(SI([ln(\overline{\mathbf{e}_s}) \oplus ln(\overline{\mathbf{r}})]) * \mathbf{w}_r^i)) \cdot \mathbf{W}) \cdot \mathbf{e}_o + b),\end{aligned} \tag{11}$$

To increase computational efficiency and optimize the model, SDFormer utilizes a 1-N scoring technique in its scoring module, which is called the KvsAll Strategy [35]. Concretely, unlike the other models that calculate the score for each knowledge triplet (e_s, r, e_o) (1-1 scoring), SDFormer employs the 1-X scoring skill [15], i.e., the acquired semantic vector is matched with the embedding matrix $\mathbf{E}$ of all entities at the same time. Compared to 1-1 scoring, 1-N scoring improves computational efficiency and lifts the accuracy. We applied a similar method, 1-X scoring, for training. Concretely, for each object entity in triplet, we negatively sampled X (X < N) entities as the candidate entities for training.

3.3. Training and optimization

SDFormer will optimize the maximum likelihood function based on the knowledge graph $\mathcal{G}$ and all parameters of SDFormer, defined as follows:

$$\max\ p(\mathcal{G}|\Theta), \tag{12}$$

where Θ denotes the whole parameters of SDFormer, and contains the embeddings of entities and relations, parameters in *Shallow Interaction* module and *Deep Interaction* module. In this paper, we expect the confidence score $\varphi(e_s, r, e_o)$ of the correct triplet (e_s, r, e_o) to be equal to 1 and the opposite to be 0. Hence, the likelihood function is defined as the Bernoulli distributions.

$$p(\mathcal{G}|\Theta) = \prod_{(e_s,r,e_o)\in\mathcal{T}\cup\mathcal{T}'} (\varphi(e_s, r, e_o))^y (1 - \varphi(e_s, r, e_o))^{1-y}, \tag{13}$$

in which

$$y = \begin{cases} 1 & for \quad (e_s, r, e_o) \in \mathcal{T} \\ 0 & for \quad (e_s, r, e_o) \in \mathcal{T}' \end{cases}, \tag{14}$$

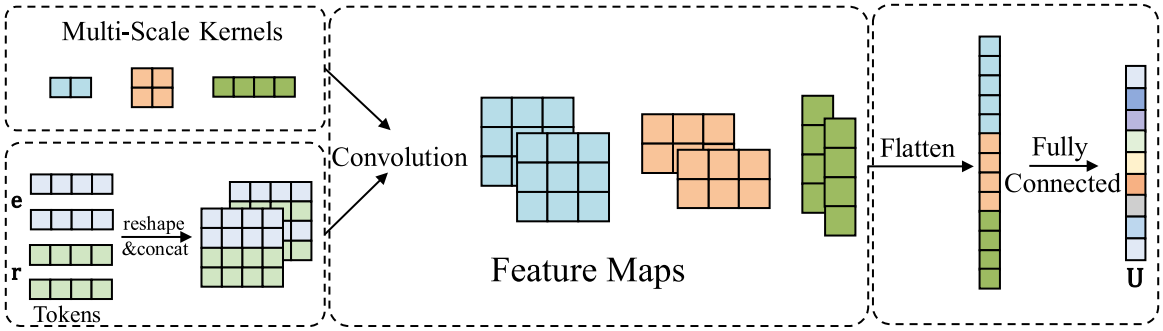

Fig. 4. The specific process of multi-scale feature interaction. Several convolutional kernels at different scales are used in SDFormer to extract feature representations of multiple relations, and then perform flattening and fully connected operations.

where $\mathcal{T}'$ is the set of negatively sampled incorrect triplets, which is generated by random substitution of the subject or object entities in the correct triplet while $\mathcal{T}$ is the truth triplet. Based on Eqs. (11)–(14), the loss function of SDFormer is summarized as follows:

$$\min\mathcal{L} = -\log\, p(\mathcal{G}|\Theta) = -\sum_{(e_s,r,e_o)\in\{\mathcal{T}\cup\mathcal{T}'\}} (y\log\varphi(e_s,r,e_o) + (1-y)\log(1-\varphi(e_s,r,e_o))), \tag{15}$$

Algorithm 1 The training process of SDFormer

Input: Knowledge Graph $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$;
batch size b;
learning rate α;
number of negative samples n;
Output: The triplet prediction probability $\hat{y}(\cdot)$;

```
E ← ℰ, R ← ℛ;   // Randomly initialize the embedding matrix of entities and relations.
loop
  T_batch ← sample(T, b);   // Randomly sample b triplets as training set.
  for (e_s, r, e_o) ∈ T_batch do
    (e_s, r, e_o) → (e_s, r, e'_o);   // Random generate n negative triplets.
    { e_s  ← (e_s, E)
      r    ← (r, R)
      e_o  ← (e_o, E)
      e_o' ← (e'_o, E) ;   // Look up the embeddings.
    ŷ(e_s, r, e_o) ← φ(e_s, r, e_o);   // Probability of correct triplets.
    ŷ(e_s, r, e'_o) ← φ(e_s, r, e'_o);   // Probability of incorrect triplets.
  end for
  With α, update embeddings matrix E and R and other parameters with Equation (15);
end loop.
```

Algorithm 1 shows the training progress of our model. Next, we will present the experimental implementation and analysis.

4. Experiments and analysis

4.1. Datasets

We perform adequate experiments for link prediction task on the following benchmark datasets: FB15k, WN18, FB15k-237, and WN18RR. As noted by ConvE [15], there are many reversible triplets in the test sets of WN18 and FB15k, resulting in high accuracy. Therefore, to eliminate the problem of false correctness due to test set leakage, WN18RR and FB15k-237 were re-built through removing the reversible triplets in WN18 and FB15k. Statistical information on these datasets is listed in Table 2.

Table 2
Statistical details of the datasets for knowledge graph.

Dataset	Entities	Relations	Triplets		
			Train	Valid	Test
FB15k	14.951	1,345	483,142	50,000	59,071
WN18	40,943	18	141,442	5,000	5,000
FB15k-237	14,541	237	272,115	17,535	20,466
WN18RR	40,943	11	86,835	3,034	3,134

(1) *FB15k* [8] is extracted from Freebase [1], which consists of 14,951 entities and 1,345 relations. It contains numerous facts of the true world, such as movies, actors, sports and sports teams, etc.
(2) *WN18* [8] is withdrawn from WordNet [2], which contains 40,943 entities and 18 relations. Entities of WN18 represent the word meanings, and the relations define lingual relations between entities.
(3) *FB15k-237* [36] is the subset of FB15k which removed all inverse relations. It includes 14,541 entities and 237 different relations.
(4) *WN18RR* [15] is the subset of WN18 which deleted reversible relations. It contains 40,943 entities with 11 different relations.

4.2. Evaluation metrics

For a knowledge graph $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, let $\mathcal{T}_{test} = \{x_1, x_2, \cdots x_{|\mathcal{T}_{test}|}\}$ represents the test set. Similar to TransE [8], for the ith test triplet x_i, we create its entire negative triplets $\widetilde{x}_i^s \notin \mathcal{T}$ (resp. $\widetilde{x}_i^o \notin \mathcal{T}$) by substituting its subject or object entity with any other entities in KG. Next, we observe whether SDFormer achieves a greater score of x_i and a worse score of false triplets. We calculate the score of the triplet via score function $\varphi(e_s, r, e_o)$ and the scores of all triplets after negative sampling. With these scores, we can calculate the ranking of the correct triplet among all candidate triplets. The calculation procedure for predicting the subject and object entities is shown below:

$$\begin{aligned} \text{rank}_i^s &= 1 + \textstyle\sum_{\widetilde{x}_i^s \notin \mathcal{T}_{test}} I[\varphi(x_i) < \varphi(\widetilde{x}_i^s)] \\ \text{rank}_i^o &= 1 + \textstyle\sum_{\widetilde{x}_i^o \notin \mathcal{T}_{test}} I[\varphi(x_i) < \varphi(\widetilde{x}_i^o)] \end{aligned}, \tag{16}$$

where $I[C]$ is the indicator function, which returns 1 when the condition C is true, or else gives 0. In our experiments, we adopt several common metrics, which conclude Mean Rank (MR), Mean Reciprocal Rank (MRR), and Hits@k (for k=1, 3, and 10) metrics to assess the precision of the predicted triplet. The mathematical formulas are defined as follows:

$$\text{MR}: \frac{1}{2|\mathcal{T}_{test}|} \sum_{x_i \in \mathcal{T}_{test}} (\text{rank}_i^s + \text{rank}_i^o), \tag{17}$$

$$\text{MRR}: \frac{1}{2|\mathcal{T}_{test}|} \sum_{x_i \in \mathcal{T}_{test}} \left(\frac{1}{\text{rank}_i^s} + \frac{1}{\text{rank}_i^o}\right), \tag{18}$$

Table 3
The combinations of parameters on different datasets. #tokens denotes the number of tokens.

Model	Dataset	*Batch size*	*lr*	emb_{drop}	$feat_{drop}$	hid_{drop}	*#tokens*	*#heads*	*#layers*	*ls*
SDFormer–	FB15k-237	128	0.0005	0.3	0.4	0.5	4	4	4	0.2
	WN18RR	128	0.0005	0.3	0.4	0.5	2	8	4	0.2
	FB15k	128	0.001	0.2	0.3	0.4	4	4	4	0.2
	WN18	256	0.00125	0.3	0.2	0.3	4	4	4	0.2
SDFormer	FB15k-237	128	0.0005	0.3	0.4	0.5	4	4	4	0.2
	WN18RR	256	0.00125	0.3	0.1	0.5	2	2	4	0.2
	FB15k	256	0.001	0.2	0.2	0.2	8	4	4	0.2
	WN18	256	0.00125	0.2	0.3	0.3	4	4	4	0.2

and

$$\text{Hits@}k : \frac{1}{2|\mathcal{T}_{test}|} \sum_{x_i \in \mathcal{T}_{test}} I[\text{rank}_i^s \le k] + I[\text{rank}_i^o \le k], \tag{19}$$

For all test triplets, MRR is the average inverse rank, while Hits@k refers to the average percentage of triplets ranked less than or equal to k in the link prediction. Usually, higher scores of MRR and Hits@k indicate greater effect.

4.3. Experimental implementation

To better train SDFormer, our model utilizes Xavier [37] technique to initialize all embedding parameters, and dropout [38] is used in different stages to regularize the model, including the feature maps, multi-head self-attention mechanism, and hidden layers after each linear fully connected network. We also adapt batch normalization [39] after each layer to regularize and improve the speed of convergence. Label smoothing [40] is employed to avoid overfitting and make model more generalized, while Adam [41] optimizer is selected to decrease the loss of the model, which is a swift and computing efficiently approach to optimize gradient-based models.

For the experiments of SDFormer and its single-scale version SDFormer–, the hyperparameters were chosen through grid search on the validation set to select an advisable configuration. The range of gird search hyperparameters was as follows: entity and relation embedding size range [128, 256, 512], feature map dropout range [0.2, 0.3, 0.4], hidden layer dropout range [0.3, 0.4, 0.5], learning rate range [5e-4, 1e-3, 1.25e−3], and label smoothing range [0, 0.1, 0.2], batch size range [64, 128, 256]. Further, we experimented with different layers of *SI*, 2, 3, and 4 layers respectively. We used the number of heads of 2, 4, 8, 16, 32, and 64 in the *SI* module to validate the effect of different number of heads. Also, we have done different experiments for the numbers of tokens fed into the *SI* module, which are 2, 4, 8 and 16. A special point is that we tried three different transformations in the *DI* module, namely Stack, Alternate and Stack&Alternate.

For the combinations of parameters on different datasets we present in Table 3. We realized SDFormer and duplicated other baselines using python library PyTorch [42].

4.4. Comparison models

To verify the effectiveness of SDFormer, some baseline models, which are also designed for link prediction task, are introduced. The general descriptions of them are as follows:

- *TransE* [8] maps entities and relations into vector space and minimizes the distance between entities and relations.
- *DistMult* [13] utilizes matrix multiplication to represent the relations between entities and relations.
- *ComplEx* [14] is a KGE model on basic of complex numbers, which can capture complex relations between entities and relations.
- *R-GCN* [26] is a relational model based on graph convolutional networks for extracting features from structured knowledge graphs.
- *ConvE* [15] is a typical KGE model using convolutional neural networks, which can infer missing links in KGs.
- *TorusE* [43] uses a circular-structured indexing structure to store and retrieve entities and relations in KGs.
- *CrossE* [44] initially learns to capture the bidirectional interactions between entities and relations.
- *RotatE* [22] defines each relation as a rotation from a source entity to a target entity in complex vector space.
- *PairRE* [25] is equipped with paired relationship vectors for each relationship in order to handle complex relationships and various relation patterns simultaneously.
- *BLP-TransE* [28] incorporates text as auxiliary information into the learned entity representation through the link prediction task.
- *StAR* [20] attempts to inject structural features into the entity and relationship embeddings obtained through the transformer encoder.
- *REP-OTE* [27] proposes an information aggregation method for graph neural networks based on relational embedding.
- *ComplexGCN* [45] adopts a graph convolutional neural network with complex domain convolution to capture knowledge representations of entities and relations.
- *Ruleformer* [21] joins rule learning and transformer to mine subgraph context information in KG to achieve knowledge reasoning.
- *SimKGC* [29] utilizes the pre-trained language model based on contrastive learning to enhance prediction accuracy.
- *RAILD* [30] learns about unseen entities and relationships using relational features in the link prediction task.
- *SDFormer–*: The single-scale version of SDFormer only use single-scale convolutional kernels in *Deep Interaction* module.
- *SDFormer*: The proposed shallow-to-deep feature interaction model for link prediction, which consists of three components, namely Vector Tokenization, Feature Interaction, and Score Prediction.

4.5. Link prediction results

In this section, we will analyze the overall performance of SDFormer on the link prediction task. The best result is in **bold** and the second-best result is underlined.

To verify the validity of SDFormer, we have compared it with several baseline models. The overall results are presented in Tables 4 and 5. For FB15K-237 and WN18RR, SDFormer is far ahead in all evaluation metrics compared to the best results in other approaches. Since our model almost builds on ConvE [15], we specifically compare against it and find that both SDFormer– and SDFormer outperform ConvE on all metrics for all the four benchmark datasets. On FB15k-237 and WN18RR, compared with ConvE, SDFormer has 9.5% and 6.5% relative improvement in MRR, 11.4% and 6.3% relative promotion in Hits@1, and 8.0% and 1.5% relative lift in Hits@10. It proves that SDFormer is effective. The reason for SDFormer's underperformance on FB15k compared to WN18 could be attributed to the uneven data distribution in the training set. We found that, unlike WN18, in the FB15k training and testing sets, the number of 1-to-N and N-to −1 relationships differs drastically, almost by two orders of magnitude, as also mentioned in [46]. This led us to naturally speculate whether such data imbalance caused SDFormer's underperformance on FB15k. In addition, given the data leakage problem on the FB15k and WN18

Table 4
Comparison results with existing models for link prediction on FB15k-237 and WN18RR.

Model	FB15k-237					WN18RR				
	MR	MRR	Hits			MR	MRR	Hits		
			@1	@3	@10			@1	@3	@10
Traditional Models										
TransE [8]	357	0.294	–	–	0.465	**3384**	0.226	–	–	0.501
DistMult [13]	254	0.241	0.155	0.263	0.419	5110	0.430	0.390	0.440	0.490
ComplEx [14]	339	0.247	0.158	0.275	0.428	5261	0.440	0.410	0.460	0.510
R-GCN [26]	–	0.249	0.151	0.264	0.417	–	0.226	0.157	0.269	0.376
ConvE [15]	244	0.325	0.237	0.356	0.501	4187	0.430	0.400	0.440	0.520
TorusE [43]	–	0.316	0.217	0.335	0.484	–	0.452	0.422	0.464	0.512
CrossE [44]	–	0.299	0.211	0.331	0.474	–	–	–	–	–
RotatE [22]	–	0.333	0.240	0.368	0.522	–	0.478	0.439	0.494	0.553
PairRE [25]	–	0.351	0.256	0.387	0.544	–	0.454	0.411	0.469	0.548
ComplexGCN [45]	–	0.338	0.245	0.371	0.524	–	0.455	0.423	0.468	0.516
REP-OTE [27]	–	0.354	0.262	0.388	0.540	–	0.488	0.439	0.505	0.588
Transformer-based Models										
StAR [20]	–	0.296	0.205	0.322	0.482	–	0.401	0.243	0.491	0.709
BLP-TransE [28]	–	0.195	0.113	0.213	0.363	–	0.285	0.135	0.361	0.580
Ruleformer [21]	–	0.342	0.255	0.374	0.513	–	0.452	0.417	0.465	0.530
RAILD [30]	–	0.216	0.127	0.241	0.397	–	0.291	0.177	0.390	0.609
SimKGC [29]	–	0.333	0.246	0.362	0.510	–	**0.671**	**0.585**	**0.731**	**0.817**
SDFormer–(*ours*)	192	0.354	0.262	0.389	0.537	3772	0.449	0.415	0.465	0.513
SDFormer(*ours*)	**185**	**0.356**	**0.264**	**0.390**	**0.541**	3633	0.458	0.425	0.471	0.528

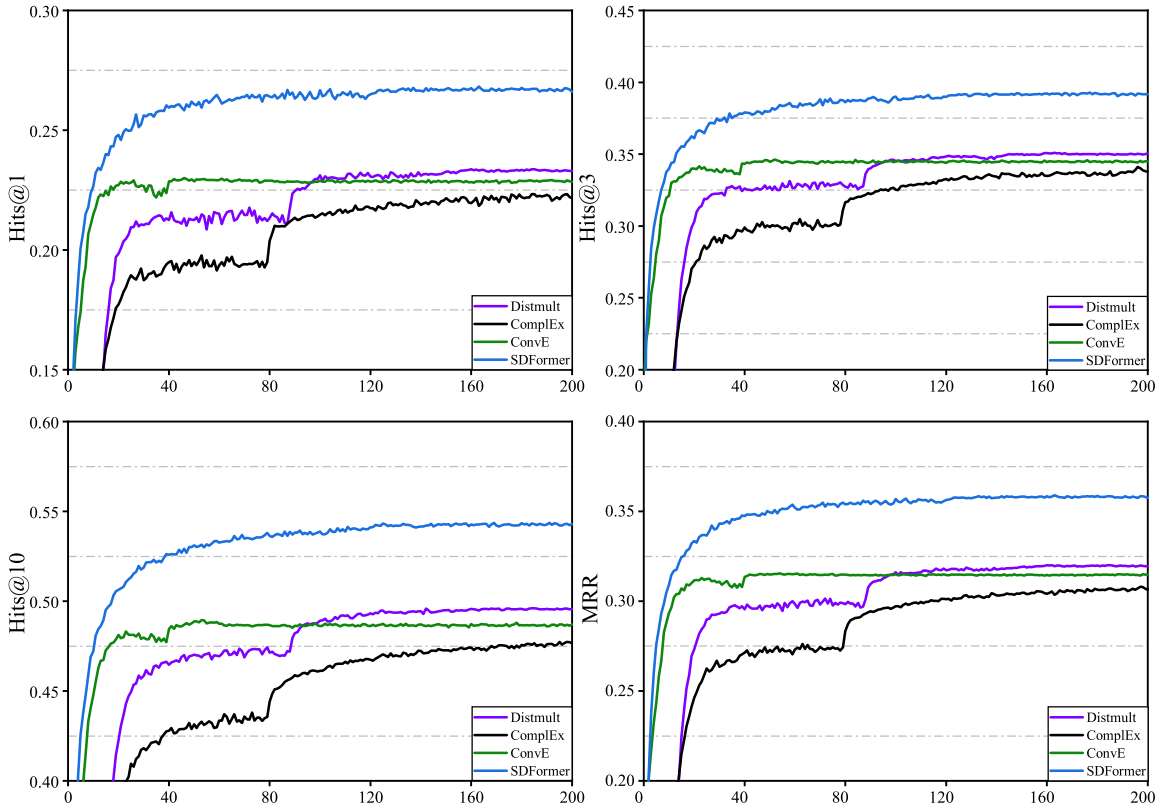

Fig. 5. Validation performance of SDFormer, ConvE, DistMult, ComplEx on FB15k-237.

datasets, usually some simple rule-based methods can achieve good results. For the sake of experimental fairness, the researchers preferred to use FB15k-237 and WN18RR datasets to avoid data leakage.

Additionally, Table 4 shows that on the FB15k-237 and WN18RR datasets, SDFormer outperforms similar Transformer-based Models. Even compared to knowledge graph embedding models that use textual descriptions as auxiliary information, which SDFormer does not have (such as StAR [20], BLP-TransE [28]), SDFormer's performance on FB15k-237 is highly competitive. The reason for SDFormer's slightly inferior performance on certain metrics in the WN18RR dataset is that WN18RR itself is a language-type knowledge graph, and the introduction of textual descriptions of entities can significantly enhance the model's performance. However, most real-world knowledge graphs, like FB15k-237, which contain world knowledge, are mainly modeled on their structural information alone, as the introduction of textual description information would increase the training burden of the model. Therefore, we believe that the SDFormer model has stronger universality.

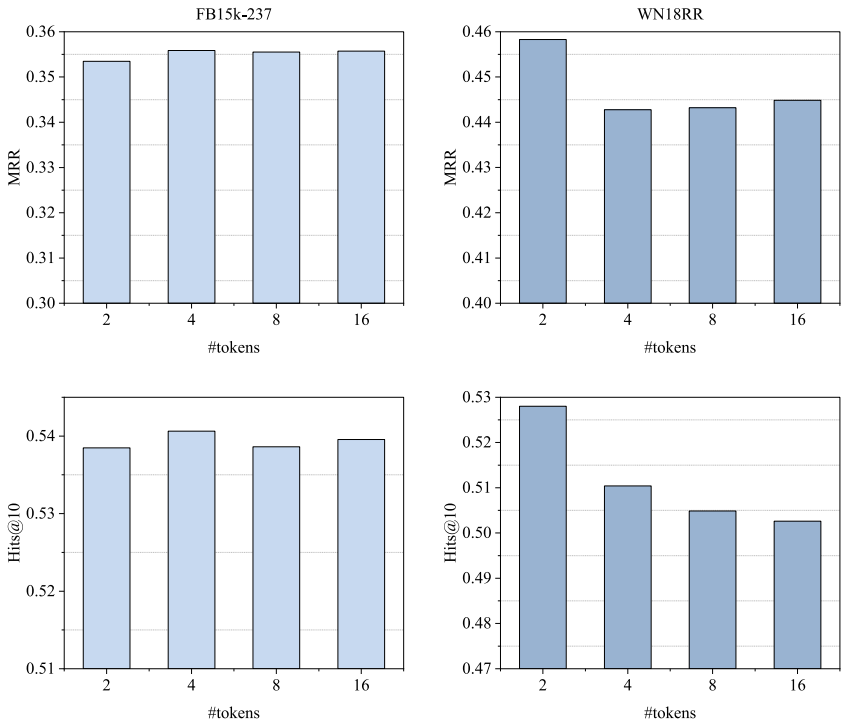

Fig. 6. Results of SDFormer on FB15k-237 and WN18RR datasets with different numbers of tokens.

Table 5
Comparison results with existing models for link prediction on FB15k and WN18.

Model	FB15k				WN18			
	MRR	Hits			MRR	Hits		
		@1	@3	@10		@1	@3	@10
TransE [8]	0.380	0.231	0.472	0.641	0.454	0.089	0.823	0.934
DistMult [13]	0.654	0.546	0.733	0.824	0.822	0.728	0.914	0.936
ComplEx [14]	0.692	0.599	0.759	0.840	0.941	0.936	0.936	0.947
R-GCN [26]	0.696	0.601	0.760	**0.842**	0.814	0.686	0.928	0.955
ConvE [15]	0.657	0.558	0.723	0.831	0.943	0.935	0.946	0.956
TorusE [43]	**0.733**	**0.674**	0.771	0.832	0.947	0.943	0.950	0.954
CrossE [44]	0.728	0.634	**0.802**	–	0.830	0.741	0.931	–
SDFormer–	0.688	0.628	0.725	0.794	0.946	0.939	0.950	0.956
SDFormer	0.692	0.628	0.732	0.802	**0.948**	**0.944**	**0.951**	**0.957**

Fig. 5 shows the performance of SDFormer and others on the train set of FB15-237. It is clearly observed that SDFormer outperforms the other models in all metrics throughout the whole training process. SDFormer has a greater increasing speed of all metrics than the others while it has a lower converging speed. The lower rate of convergence is due to the complexity of SDFormer.

In addition, we have compared the parameter sizes of Traditional Models and Transformer-based Models in Table 6. The scale of parameters in SDFormer has a clear advantage over both traditional models and Transformer-based models, with parameter sizes of 9.11M and 16.30M on FB15k-237 and WN18RR datasets respectively, smaller than traditional structure-based models and significantly smaller than those transformer-based models. The smaller parameters of SDFormer also ensure that the model can be applied to training on large-scale knowledge graph datasets, greatly improving the efficiency of model training and having certain practical value.

In general, SDFormer makes great progress in modeling expressive KGE for link prediction and efficient knowledge graph data modeling. On the basis of its shallow-to-deep architecture, SDFormer learns multi-level interactions among sub-features between entities and relations in KGs and obtained momentous enhancements on all evaluation metrics.

4.6. Ablation study

4.6.1. Effectiveness of each module in SDFormer

To analyze the validation of *Vector Tokenization* (VT) module, *Shallow Interaction* (SI) module and, *Deep Interaction* (DI) module in SDFormer, it is necessary to carry out an ablation study. The results of this section are shown in Table 7.

Effectiveness of VT: To demonstrate the role of tokenization, we take the embeddings of entity and relation as a whole instead of tokenizing them. The experimental results indicated that *Vector Tokenization* could improve the performance of the model. This is because after slicing into multiple tokens, we can learn the interactions between sub-features more deeply and thus obtain richer feature interactions.

Effectiveness of SI: Without *Shallow Interaction* module, the scores of evaluation metrics were much lower on the WN18RR dataset, but less on the FB15k-237 dataset. Since there are only 11 relations in the WN18RR dataset, each relation will correspond to a large number of entities. In this case, the extraction of sub-features by *Shallow Interaction* module becomes more important.

Effectiveness of DI: We observe that the scores of all metrics drop rapidly when we drop the *Deep Interaction* module. Concretely, w/o DI declines the MRR by 4.0% and Hits@10 by 3.0% on FB15k-237. On the WN18RR dataset, w/o DI declines more the MRR by 7.2% and Hits@10 by 5.0%. We believe that the *Shallow Interaction* module performs more of an aggregation of aspect features and lacks interaction between entities and relations as a whole. In this case, the interaction information we acquire is limited. Thus the deep module in SDFormer allows for further interaction of the aggregated aspect features. Hence, the role of shallow interactions is improved.

Considering the powerful perceptual field of convolutional networks, it can perceive the feature interactions between entities and relations more globally. Therefore, it is easy to understand why deep modules work better than shallow ones.

Experimentally, the SDFormer, as a shallow to deep Feature Interaction model, proves to be able to amplify the two modules very well. This also demonstrates that our proposed multi-head self-attention and convolutional neural networks are able to achieve shallow to deep feature interaction effectively.

Table 6
Comparison results with Traditional and Transformer-based models for model parameter on FB15k-237 and WN18RR.

Dataset	Model							
	Traditional Model				Transformer-based Model			
	TransE [8]	DistMult [13]	ComplEx [14]	RotatE [22]	BLP-TransE [28]	StAR [20]	Ruleformer [21]	SDFormer
FB15k-237	14.78M	29.56M	29.56M	29.32M	108.44M	355.37M	11.45M	**9.11M**
WN18RR	20.47M	40.95M	40.95M	40.95M	108.41M	355.37M	21.07M	**16.30M**

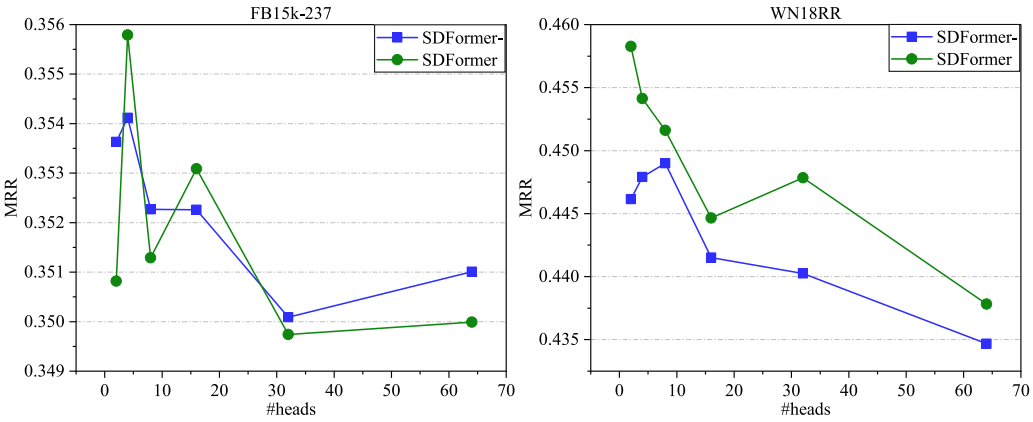

Fig. 7. The results of MRR for SDFormer– and SDFormer on FB15k-237 and WN18RR datasets. #heads denotes the number of attention head.

Table 7
Results of ablation study. w/o DI means SDFormer without *Deep Interaction* module.

Model	FB15k-237		WN18RR	
	MRR	hits@10	MRR	hits@10
SDFormer	**0.356**	**0.541**	**0.458**	**0.528**
w/o VT	0.352	0.536	0.454	0.507
w/o SI	0.348	0.530	0.432	0.493
w/o DI	0.342	0.525	0.425	0.502

4.6.2. Effect of the number of tokens

The number of tokens directly affects the sub-feature granularity of entities and relations. To investigate the effect of the number of tokens, we perform some correlative experiments on SDFormer in WN18RR and FB15k-237 datasets. Fig. 6 depicts the variation in the number of tokens, which is tuned from [1, 2, 4, 8, 16]. For the comprehensive evaluation metric MRR, we can observe that SDFormer achieves better performance when the token number $\#tokens = 4$ on the FB15k-237 dataset and $\#tokens = 2$ on the WN18RR dataset. The reason for setting a different number of tokens for different datasets may be that the structural complexity of knowledge graph datasets is different. Specifically, as shown in Table 2, there are only 11 relations in WN18RR, but there are 237 relations in FB15k-237. For more relations, entities need more combinations of sub-features to distinguish triplets of the same entity under different relations. However, it demonstrates that the finer-grained segmentation of entity and relation vectors is logical and resultful. Model effectiveness could be improved by setting an appropriate number of tokens.

4.6.3. Effect of multi-scale convolution

In order to explore the impact of multi-scale convolutional kernels, we conduct some comparison experiments on FB15k-237 and WN18RR datasets. The experimental results are shown in Table 9. We found that multi-scale convolutional kernels achieve optimal performance while reducing model parameters compared to simple stacking of single-scale convolutional kernels. Moreover, SDFormer obtains the best experimental results when the moderate filter sizes are $(1\times9, 2\times4, 3\times3)$. Smaller filter kernels restrict the degree of deep interaction between entities and relations, and larger filter kernels tend to result in model overfitting. The above results show that multi-scale filters can effectively capture the deep interactions between entities and relations, thus enriching feature representations.

4.6.4. Effect of negative sampling method

SDFormer employs random negative sampling during the negative sampling process. For the method of negative sampling, we referred to the works of [47,48]. From their experimental results, it is evident that among the methods like Random sampling, Corrupting positive instances, Typed Sampling, Relational Sampling, and Nearest Neighbor sampling, Random sampling is the most efficient. The idea of using semantically similar negative sampling and Nearest Neighbor sampling methods is similar, as they both require obtaining negative samples through the semantic embedding of entities, which intuitively can add difficult negative cases and increase the training difficulty of the model. However, on one hand, semantically similar negative sampling generates negative examples by calculating the similarity of word vectors, which can impose a significant burden on the model's training efficiency, contradicting our original intention of a simple and efficient knowledge representation model. On the other hand, our SDFormer method is based on embedding learning from the structural information of knowledge graphs, without introducing textual semantic information and only considering the structural information of entities and relations. Considering the above reasons, we chose Random sampling, which can effectively improve the efficiency of the SDFormer model. Regarding the quantity of Random sampling, after referencing related works [35,47,48], we adopted the two random negative sampling methods mentioned in [35]: 1vsAll and KvsAll.

We conducted Random sampling using the above two methods (1vsAll and KvsAll), after numerous experiments, found the optimal number of negative samples for SDFormer on different datasets, as shown in the Table 8. The experimental results show that under the 1vsAll approach, when the number of negative samples is set to 1000, our model achieves better performance on the WN18RR and FB15k237 datasets.

4.7. Parameter sensitivity analysis

Attention heads impact. Self-attention with different numbers of heads can extract multi-level interaction information between entities

Table 8
The effect of different negative sampling methods. $|\mathcal{E}|$ denotes the number of entities.

Random sampling		FB15k-237				WN18RR			
Type	# of negative examples	MRR	Hits			MRR	Hits		
			@1	@3	@10		@1	@3	@10
1vsAll	500	0.353	0.260	0.389	0.535	0.449	0.417	0.460	0.514
	1000	**0.356**	**0.264**	**0.390**	**0.541**	**0.458**	**0.425**	**0.471**	**0.528**
	1500	0.355	0.263	0.390	0.540	0.454	0.422	0.465	0.517
	2000	0.353	0.261	0.387	0.540	0.454	0.420	0.469	0.517
KvsAll	$\|\mathcal{E}\| - 1$	0.346	0.256	0.382	0.529	0.454	0.424	0.466	0.510

Table 9
Performance of different scale convolution kernels in SDFormer on the FB15k-237 and WN18RR datasets.

Method	Filter-type	FB15k-237		WN18RR	
		MRR	Hits@10	MRR	Hits@10
Single-Scale	1×5	0.350	0.536	0.449	0.513
	1×9	0.353	0.539	0.437	0.501
	2×4	0.354	0.443	0.455	0.508
	2×5	0.353	0.538	0.444	0.498
	3×3	0.350	0.538	0.445	0.510
	3×4	0.352	0.537	0.440	0.500
Multi-Scale	$\begin{bmatrix} 1 \times 5 \\ 2 \times 5 \\ 3 \times 3 \end{bmatrix}$	0.351	0.538	**0.458**	**0.528**
	$\begin{bmatrix} 1 \times 9 \\ 2 \times 4 \\ 3 \times 3 \end{bmatrix}$	**0.356**	**0.541**	**0.458**	0.520
	$\begin{bmatrix} 1 \times 9 \\ 2 \times 5 \\ 3 \times 4 \end{bmatrix}$	0.352	0.537	0.454	0.512

Table 10
Layer number analysis in Shallow Interaction modules on FB15k-237 and WN18RR datasets.

	#Layers	FB15k-237		WN18RR	
		MRR	Hits@10	MRR	Hits@10
Shallow Interaction Module	2	0.336	0.520	0.413	0.484
	3	0.339	0.525	0.419	0.491
	4	**0.342**	**0.525**	**0.425**	**0.502**

Table 11
Performance analysis of different splicing methods in Deep Interaction modules on FB15k-237 and WN18RR datasets.

	Splicing method	FB15k-237		WN18RR	
		MRR	Hits@10	MRR	Hits@10
Deep Interaction Module	Stack	0.339	0.521	0.413	0.489
	Alternate	0.344	0.525	0.432	0.493
	Chequer	0.281	0.426	0.413	0.447
	Stack&Alternate	**0.348**	**0.530**	**0.444**	**0.497**
	Stack&Chequer	0.342	0.526	0.439	0.488
	Alternate&Chequer	0.345	0.525	0.440	0.487

and relations. To analyze the influence of the number of heads on proposed model, some comparison experiments with attention heads ranging from [2, 4, 8, 16, 32, 64] are performed on FB15k-237 and WN18RR datasets. As shown in Fig. 7, for the FB15k-237 dataset, both SDFormer– and SDFormer achieve the greatest results when the number of heads is set as 4, while for the WN18RR dataset, the number of heads is equal to 2 and 8 on SDFormer and SDFormer–, respectively. Too many self-attention heads may bring more redundant features, while a smaller number of heads is more conducive to the hierarchy of feature interactions. Therefore, an appropriate number of attention heads is selected for different datasets in the *Shallow Interaction* module.

Layers in Shallow Interaction. Considering that too many layers will introduce too many parameters and increase the complexity of the model. There, we perform several experiments with the number of layers sampled from [2, 3, 4] to explore the effect of layers in the *Shallow Interaction* module. The results are shown in Table 10. The results show that as the number of layers in the *Shallow Interaction* module increases, the performance of SDFormer continues to improve. However, considering the performance and operating efficiency of the model, the number of layers in the *Shallow Interaction* module is set to 4.

Token splicing method in Deep Interaction. The splicing method between sub-features in entities and relations directly affects the interaction ability. To analyze the effects of three different token splicing methods, we conducted comparative experiments on the Stack, Alternate, and Stack&- Alternate methods on FB15k-237 and WN18RR datasets, respectively. According to the results in the following Table 11, the best performance is achieved when the Splicing Method uses Stack&Alternate. The main reason might be that there is a certain correlation between the features obtained from the Shallow Interaction output, and each output token represents a collection of features of a certain aspect of entities and relations. However, using the Chequer method disrupts the existing correlation between features, hence reducing the model's performance after implementing the Chequer mechanism. Moreover, our chosen Stack&Alternate method, introducing some randomness through the Alternate method, has been experimentally proven to enhance the model's generalizability and performance. Therefore, ultimately, we use the Stack&Alternate method for splicing methods.

5. Conclusion and future work

This paper presents a novel shallow-to-deep feature interaction for knowledge graph embedding called SDFormer, which takes the multi-level interaction information among fine-grained sub-features between entities and relations into account. Concretely, the vectors of entity and relation are decomposed into multiple tokens to represent sub-features of different dimensions. On this basis, we design a shallow-to-deep feature interaction method. First, a multi-head self-attention is employed to compute the correlation between sub-features between entities and relations in the *Shallow Interaction* module. This process enriches the feature representations by weighting sub-features with multiple correlation weights. Then, we merge upwards from sub-features into "aspect features", which are rearranged with the sub-features of relations. The merged features undergo multi-scale convolution operations to extract multiple relationships in the *Deep Interaction* module. Experimental results show that SDFormer, which extracts multi-level interaction information, outperforms numerous baselines.

In the future, we are desired to incorporate textual description information of entities and relations as auxiliary information in the model to simulate the semantic embedding of entities (relations) in the real world. Besides, we further study on the impact of the quality and efficiency of negative sampling on KGE.

CRediT authorship contribution statement

Duantengchuan Li: Conceptualization, Methodology, Software, Supervision, Writing – original draft, Writing – review & editing. **Tao Xia:** Data curation, Software, Writing – original draft, Writing – review

& editing. **Jing Wang:** Conceptualization, Formal analysis, Writing – review & editing. **Fobo Shi:** Methodology, Software, Writing – review & editing. **Qi Zhang:** Conceptualization, Methodology, Supervision, Writing – original draft, Writing – review & editing. **Bing Li:** Conceptualization, Project administration, Supervision. **Yu Xiong:** Data curation, Writing – review & editing.

Declaration of competing interest

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.

Data availability

Data will be made available on request.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Nos. 62032016, 62377007), the Key Research and Development Program of Hubei Province, China (No. 2021BAA031).

References

[1] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor, Freebase: A collaboratively created graph database for structuring human knowledge, in: SIGMOD, Association for Computing Machinery, 2008, pp. 1247—1250.
[2] G.A. Miller, WordNet: A lexical database for english, 38, (11) Association for Computing Machinery, 1995, pp. 39—41,
[3] B. Wu, L. Zhong, L. Yao, Y. Ye, EAGCN: An efficient adaptive graph convolutional network for item recommendation in social internet of things, IEEE Internet Things J. 9 (17) (2022) 16386–16401.
[4] B. Wu, L. Zhong, H. Li, Y. Ye, Efficient complementary graph convolutional network without negative sampling for item recommendation, Knowl.-Based Syst. 256 (2022) 109758.
[5] Z. Li, Y. Zhao, Y. Zhang, Z. Zhang, Multi-relational graph attention networks for knowledge graph completion, Knowl.-Based Syst. 251 (2022) 109262.
[6] Z. Li, Q. Zhang, F. Zhu, D. Li, C. Zheng, Y. Zhang, Knowledge graph representation learning with simplifying hierarchical feature propagation, Inf. Process. Manage. 60 (4) (2023) 103348.
[7] J. Wang, Q. Zhang, F. Shi, D. Li, Y. Cai, J. Wang, B. Li, X. Wang, Z. Zhang, C. Zheng, Knowledge graph embedding model with attention-based high-low level features interaction convolutional network, Inf. Process. Manage. 60 (4) (2023) 103350.
[8] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: Advances in Neural Information Processing Systems, Vol. 26, 2013, pp. 2787–2795.
[9] Z. Wang, J. Zhang, J. Feng, Z. Chen, Knowledge graph embedding by translating on hyperplanes, 28, (1) 2014, pp. 1112–1119,
[10] Y. Lin, Z. Liu, M. Sun, Y. Liu, X. Zhu, Learning entity and relation embeddings for knowledge graph completion, in: AAAI, AAAI Press, 2015, pp. 2181—2187.
[11] G. Ji, S. He, L. Xu, K. Liu, J. Zhao, Knowledge Graph Embedding Via Dynamic Mapping Matrix, Association for Computational Linguistics, 2015, pp. 687–696.
[12] M. Nickel, V. Tresp, H.-P. Kriegel, A three-way model for collective learning on multi-relational data, in: ICML, Omni Press, 2011, pp. 809—816.
[13] B. Yang, W. tau Yih, X. He, J. Gao, L. Deng, Embedding entities and relations for learning and inference in knowledge bases, International Conference on Learning Representations, 2015.
[14] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, G. Bouchard, Complex embeddings for simple link prediction, in: ICML, JMLR.org, 2016, pp. 2071—2080.
[15] T. Dettmers, P. Minervini, P. Stenetorp, S. Riedel, Convolutional 2D knowledge graph embeddings, in: AAAI/IAAI/EAAI, AAAI Press, 2018.
[16] M. Defferrard, X. Bresson, P. Vandergheynst, NIPS, Curran Associates Inc., 2016, pp. 3844—3852.
[17] S. Vashishth, S. Sanyal, V. Nitin, N. Agrawal, P. Talukdar, InteractE: Improving convolution-based knowledge graph embeddings by increasing feature interactions, 34, (03) 2020, pp. 3009–3016,
[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, in: I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Attention is All you Need, Vol. 30, Curran Associates, Inc., 2017.
[19] L. Yao, C. Mao, Y. Luo, KG-BERT: BERT for knowledge graph completion, 2019.
[20] B. Wang, T. Shen, G. Long, T. Zhou, Y. Wang, Y. Chang, Structure-augmented text representation learning for efficient knowledge graph completion, in: Proceedings of the Web Conference 2021, 2021, pp. 1737–1748.
[21] Z. Xu, P. Ye, H. Chen, M. Zhao, H. Chen, W. Zhang, Ruleformer: Context-aware rule mining over knowledge graph, in: Proceedings of the 29th International Conference on Computational Linguistics, 2022, pp. 2551–2560.
[22] Z. Sun, Z.-H. Deng, J.-Y. Nie, J. Tang, Rotate: Knowledge graph embedding by relational rotation in complex space, in: International Conference on Learning Representations, 2019.
[23] M. Nickel, L. Rosasco, T. Poggio, Holographic embeddings of knowledge graphs, in: AAAI, AAAI Press, 2016, pp. 1955—1961.
[24] H. Liu, Y. Wu, Y. Yang, Analogical inference for multi-relational embeddings, in: ICML, JMLR.org, 2017, pp. 2168—2178.
[25] L. Chao, J. He, T. Wang, W. Chu, PairRE: Knowledge graph embeddings via paired relation vectors, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 2021, pp. 4360–4369.
[26] M. Schlichtkrull, T.N. Kipf, P. Bloem, R. van den Berg, I. Titov, M. Welling, Modeling relational data with graph convolutional networks, in: A. Gangemi, R. Navigli, M.-E. Vidal, P. Hitzler, R. Troncy, L. Hollink, A. Tordai, M. Alam (Eds.), Springer International Publishing, 2018, pp. 593–607.
[27] H. Wang, S. Dai, W. Su, H. Zhong, Z. Fang, Z. Huang, S. Feng, Z. Chen, Y. Sun, D. Yu, Simple and effective relation-based embedding propagation for knowledge representation learning, in: Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22, 2022, pp. 2755–2761.
[28] D. Daza, M. Cochez, P. Groth, Inductive entity representations from text via link prediction, in: Proceedings of the Web Conference 2021, 2021, pp. 798–808.
[29] L. Wang, W. Zhao, Z. Wei, J. Liu, SimKGC: Simple contrastive knowledge graph completion with pre-trained language models, in: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2022, pp. 4281–4294.
[30] G.A. Gesese, H. Sack, M. Alam, RAILD: Towards leveraging relation features for inductive link prediction in knowledge graphs, in: Proceedings of the 11th International Joint Conference on Knowledge Graphs, 2023, pp. 82–90.
[31] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby, An image is worth 16x16 words: Transformers for image recognition at scale, 2021.
[32] J.L. Ba, J.R. Kiros, G.E. Hinton, Layer normalization, 2016.
[33] D. Hendrycks, K. Gimpel, Gaussian error linear units (GELUs), 2016.
[34] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, Vol. 25, Curran Associates, Inc., 2012, pp. 1106–1114.
[35] D. Ruffinelli, S. Broscheit, R. Gemulla, You CAN teach an old dog new tricks! on training knowledge graph embeddings, in: International Conference on Learning Representations, 2020.
[36] K. Toutanova, D. Chen, Observed versus latent features for knowledge base and text inference, Association for Computational Linguistics, 2015, pp. 57–66.
[37] X. Glorot, Y. Bengio, in: Y.W. Teh, M. Titterington (Eds.), Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of Machine Learning Research, vol. 9, PMLR, 2010, pp. 249–256.
[38] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, 15, (56) 2014, pp. 1929–1958,
[39] S. Ioffe, C. Szegedy, in: F. Bach, D. Blei (Eds.), Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, PMLR, 2015, pp. 448–456.
[40] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the Inception Architecture for Computer Vision, IEEE Computer Society, 2016, pp. 2818–2826.
[41] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014.
[42] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch, 2017.
[43] T. Ebisu, R. Ichise, Toruse: Knowledge graph embedding on a Lie group, in: AAAI/IAAI/EAAI, AAAI Press, 2018.
[44] W. Zhang, B. Paudel, W. Zhang, A. Bernstein, H. Chen, Interaction embeddings for prediction and explanation in knowledge graphs, in: WSDM, Association for Computing Machinery, 2019, pp. 96—104.
[45] A. Zeb, S. Saif, J. Chen, A.U. Haq, Z. Gong, D. Zhang, Complex graph convolutional network for link prediction in knowledge graphs, Expert Syst. Appl. 200 (2022) 116796.
[46] K. Toutanova, D. Chen, Observed versus latent features for knowledge base and text inference, in: Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality, 2015, pp. 57–66.
[47] B. Kotnis, V. Nastase, Analysis of the impact of negative sampling on link prediction in knowledge graphs, 2018, arXiv:1708.06816.
[48] H. Kamigaito, K. Hayashi, Comprehensive analysis of negative sampling in knowledge graph representation learning, in: Proceedings of the 39th International Conference on Machine Learning, Vol. 162, 2022, pp. 10661–10675.