SPAC: Sparse Partitioning and Adaptive Core Tensor Pruning Model for Knowledge Graph Completion

Chuhong Yang[1], **Bin Li**[1,2*], **Nan Wu**[1,2]

[1]Beijing Institute of Technology, Beijing, China
[2]Yangtze Delta Region Academy of Beijing Institute of Technology, Jiaxing, Zhejiang, China
{3120230733, binli, wunan}@bit.edu.cn

Abstract

Tensor decomposition (TD) models are promising solutions for knowledge graph completion due to their simple structures but powerful representation capacities. These TD models typically adopt Tucker decomposition with a structured core tensor. Some models with a sparse core tensor, such as DistMult and ComplEx, are too simple and thus limit the interaction between embedding components, while other models with a dense core tensor are too complex and may lead to significant overfitting. To address these issues, we propose a new TD model called SPAC (Sparse Partitioning and Adaptive Core tensor pruning). Specifically, SPAC captures coarse and fine-grained semantic information using a hybrid core tensor, where auxiliary cores are used to model sparse interactions and main cores for dense interactions. Moreover, SPAC introduces a gating mechanism to control the output of intermediate variables, enhancing the interaction between different partition groups. Furthermore, SPAC employs an adaptive pruning approach to dynamically adjust the shape of the core tensor. The proposed TD model enhances expressive capacity and reduces the number of parameters in the core tensor. Experiments are conducted on datasets FB15k-237, WN18RR, and YAGO3-10. The results demonstrate that SPAC outperforms state-of-the-art tensor decomposition models, including MEIM and Tucker models. A series of ablation studies show that the gating mechanism and adaptive pruning strategy in SPAC are crucial for the performance improvement.

Introduction

Knowledge Graphs (KGs) have emerged as a compelling structure for representing structured and semi-structured data across a wide range of domains. They provide a flexible and intuitive way to model entities and their interrelationships, making them an essential tool for various applications such as semantic search (Cha et al. 2023), recommendation systems (Yang et al. 2023), and natural language processing (Zhang et al. 2019b). Constructing and maintaining a large-scale KG can be challenging due to the dynamic nature of data and the complexity of relationships. Moreover, large KGs often contain missing or incomplete information, which can hinder the performance of downstream applications. Therefore, Knowledge Graph Completion (KGC), which aims to address this issue by inferring missing relationships or entities, has attracted significant attention in recent years.

In KGC tasks, Tensor Decomposition (TD) models demonstrate substantial competitiveness. These models conceptualize a knowledge graph as a three-dimensional tensor and treat KGC as a tensor completion problem. These TD models can be viewed as shallow neural networks. Despite the simplicity, they achieve impressive performance. The RESCAL model (Nickel et al. 2011) interprets the specific relationships between head and tail entities as unique weight matrices, facilitating interactions between their embedding vectors. DistMult (Yang et al. 2014) simplifies the relationship embedding in RESCAL into diagonal weight matrices. ComplEx (Trouillon et al. 2016) extends DistMult's real embeddings into the complex space, enhancing its capacity to handle asymmetric relationships. Subsequent models like SimplE and QuatE (Kazemi and Poole 2018; Zhang et al. 2019a) have further simplified and expanded upon ComplEx, and have notable performance. In 2019, the author of (Balažević, Allen, and Hospedales 2019) applied Tucker decomposition to KGC tasks and illustrated that the aforementioned models can be regarded as special cases of Tucker decomposition with a sparse core tensor.

From the perspective of core tensor structure, models like DistMult and ComplEx, while interpretable to some extent, suffer from performance issues due to the excessive sparsity of the core tensor, which limits effective interaction between embedding dimensions. Conversely, Tucker's dense core tensor structure leads to excessive interactions, but significantly reduces training efficiency and increases the risk of overfitting. MEI and MEIM (Tran and Takasu 2020; Nghiep Tran and Takasu 2022) model the core tensor with a diagonally blocked sparse structure, striking a balance between effectiveness and efficiency. These developments motivate us toward exploring a sparse and better dynamically adaptive core tensor structure, starting from a semi-sparse structure and pruning unnecessary parts to achieve a streamlined model. From an efficiency standpoint, pruning along the diagonal blocks is beneficial. Instead of completely discarding the pruned dimensions, we replace some parts of the dense interactions with sparse interactions, which can capture general semantic information. Meanwhile, dense interactions can capture precise semantic information. Further-

*The corresponding author.

more, considering that embedding features corresponding to different diagonal blocks are independent and lack interaction, introducing additional layers to facilitate parameter updates is beneficial. NePTuNe (Sonkar, Katiyar, and Baraniuk 2022) is an insightful model that incorporates nonlinear activation layers into the linear TD model, enhancing its expressive capacity. This insight inspires us to seek an additional network that can be integrated into the original network without significantly altering the intermediates, allowing for interactions between different embedding groups and introducing additional nonlinearity. In this regard, a gating layer is an excellent choice. It enables interaction among independent embedding dimensions and adds dynamic nonlinearity to the linear model, thus enhancing its expressive power. To sum up, we propose a new model, namely SPAC (**S**parse **P**artitioning and **A**daptive **C**ore tensor pruning), which utilizes sparse partitioning with gating mechanism and adaptive pruning to enhance the model's performance. The main contributions of this paper can be summarized as follows:

- We propose a SPAC model, which adopts a sparse block diagonal tensor with main cores and auxiliary cores filling its hyperdiagonals. These two types of cores capture semantic information of varying granularity, and thus enhance the model's performance.
- A linear gating mechanism is introduced in SPAC to control intermediate outputs. This generates the interaction between partition groups and incorporates a non-linear transformation for the model, which enhances the network's expressive power.
- To dynamically adjust the shape of the core tensor, we employ an adaptive pruning approach during training. Specifically, SPAC evaluates the importance and diversity of each dimension in intermediate outputs and prunes the corresponding parts of the core tensor.
- SPAC achieves state-of-the-art performance on the FB15k-237, WN18RR, and YAGO3-10 datasets across multiple metrics, and has a smaller number of parameters compared to other typical models. Experimental results further demonstrate that the introduced gating mechanism and adaptive pruning strategy can be seamlessly integrated into other TD models, and significantly improve the model's performance.

Preliminaries

A Knowledge Graph (KG) can be mathematically represented as a directed, labeled graph $\mathcal{T} = \{(e_i, r_k, e_j)\} \subset \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where $\mathcal{E}$ and $\mathcal{R}$ denote the entity set and relation set, respectively. Each triple (e_i, r_k, e_j) represents a fact that the head entity e_i is related to the tail entity e_j by relation r_k. The Knowledge Graph Completion (KGC) task is to infer missing relationships or entities within a KG. Mathematically, this task can be solved by predicting the likelihood of a missing link $(e_i, r_k, ?)$ or $(?, r_k, e_j)$, and this task can be formalized as learning a scoring function $\phi : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \to \mathbb{R}$, where $\phi(e_i, r_k, e_j)$ estimates the plausibility of the triple (e_i, r_k, e_j). A high score indicates a high likelihood that the triple is correct.

A KG $\mathcal{T} \subset \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ can be regarded as a third-order tensor $\mathcal{X}_\mathcal{T} \in \{0, 1\}^{|\mathcal{E}| \times |\mathcal{R}| \times |\mathcal{E}|}$, where $|\cdot|$ denotes the number of elements in a set. Specifically, the element $\mathcal{X}_\mathcal{T}[i, j, k]$ is 1 if the triple (e_i, r_k, e_j) exists in the $\mathcal{T}$, and 0 otherwise.

Tucker form: Given a tensor $\mathcal{X}_\mathcal{T}$ which represents the knowledge graph $\mathcal{T}$, Tucker decomposition method factorizes $\mathcal{X}_\mathcal{T}$ by a core tensor $\mathcal{G} \in \mathbb{R}^{d_e \times d_r \times d_e}$ and two factors $\mathbf{U}^{(e)} \in \mathbb{R}^{d_e \times |\mathcal{E}|}$ and $\mathbf{U}^{(r)} \in \mathbb{R}^{d_r \times |\mathcal{R}|}$, which are matrices composed of the embeddings of entities and relationships. Note that d_e and d_r are the embedding dimensions of entities and relations, respectively. Let $\sigma(\cdot)$ denotes the sigmoid function, the Tucker form can be expressed as

$$\mathcal{X}_\mathcal{T} \approx \sigma(\mathcal{G} \times_1 \mathbf{U}^{(e)} \times_2 \mathbf{U}^{(r)} \times_3 \mathbf{U}^{(e)}), \tag{1}$$

where $\times_i$ denotes the mode-i product of a tensor and a matrix along the i-th dimension. Let $\boldsymbol{e}_i$, $\boldsymbol{e}_j \in \mathbb{R}^{d_e}$, $\boldsymbol{r}_k \in \mathbb{R}^{d_r}$ denote the embeddings of entities e_i, e_j and relation r_k, respectively. For a given triple $t = (e_i, r_k, e_j)$, the scoring function $\phi(e_i, r_k, e_j)$ can be calculated by

$$\phi(e_i, r_k, e_j) \tag{2}$$

$$= \sigma(\sum_{a=1}^{d_e}\sum_{b=1}^{d_r}\sum_{c=1}^{d_e}\mathcal{G}[a, b, c]\mathbf{U}^{(e)}[a, i]\mathbf{U}^{(r)}[b, k]\mathbf{U}^{(e)}[c, j]) \tag{3}$$

$$= \sigma(\mathcal{G} \times_1 \boldsymbol{e}_i \times_2 \boldsymbol{r}_k \times_3 \boldsymbol{e}_j) \approx \mathcal{X}_\mathcal{T}[i, k, j]. \tag{4}$$

In many TD models, their core tensors have sparse structures. For example, the core tensor of CP model $\tilde{\mathcal{G}}_{CP}$ can be regarded as a third-order tensor with all the superdiagonals being 1 and all the other elements being 0. The core tensor of DistMult $\tilde{\mathcal{G}}_{DM}$ can be regarded as a third-order tensor with all the superdiagonals being learnable parameters and all the other elements being 0. For MEI and MEIM, the core tensor $\tilde{\mathcal{G}}_M$ is a block diagonal tensor, where each block is a dense tensor filled with learnable parameters.

SPAC Model

Models like CP, DistMult, SimplE, and ComplEx utilize very simple core tensors, which limit the performance of the model, while Tucker itself uses dense core tensor, which is prone to overfitting and high computational cost. MEI and MEIM make a compromise by using a block-diagonal core tensor. Notice that the core tensor can be further sparsified and the model performance can be improved by dynamic pruning and introducing additional networks. To this end, we propose a SPAC model, which partitions the core tensor into a sparse block diagonal tensor and introduces a gating mechanism to control the output of intermediate variables and employs an adaptive mechanism to prune the core tensor during training. The overall architecture of SPAC model is shown in Figure 1.

Sparse partitioning

Given a triplet $t = (e_i, r_k, e_j)$ and its corresponding d-dimensional embedding representations (assuming $d_e = d_r = d$) $\boldsymbol{e}_i, \boldsymbol{r}_k, \boldsymbol{e}_j \in \mathbb{R}^d$, we divide the sparse core tensor's diagonal into p sufficiently interactive groups of size c and q independent interaction units such that $d = pc + q$.

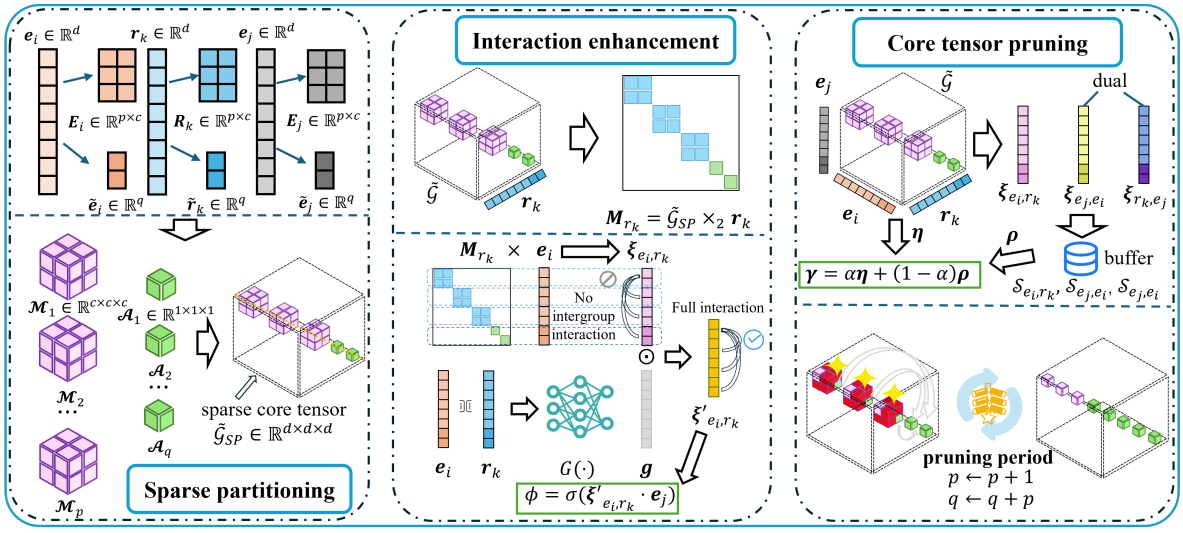


Figure 1: **The overall architecture of SPAC model.** SPAC model consists of three components: sparse partitioning, intermediate output gating, and adaptive pruning.

Therefore, the first pc dimensions of the embedding vector $\boldsymbol{e}_i$ can be reshaped into a matrix $\boldsymbol{E}_i \in \mathbb{R}^{p\times c}$, and the last q dimensions can be denoted by a vector $\tilde{\boldsymbol{e}}_i \in \mathbb{R}^q$. The same operation can be applied to $\boldsymbol{r}_k$ and $\boldsymbol{e}_j$ to obtain $\boldsymbol{R}_k \in \mathbb{R}^{p\times c}$, $\tilde{\boldsymbol{e}}_k \in \mathbb{R}^q$ and $\boldsymbol{E}_j \in \mathbb{R}^{p\times c}$, $\tilde{\boldsymbol{e}}_j \in \mathbb{R}^q$, respectively. The sparse core tensor $\tilde{\mathcal{G}}_{SP}$ can be partitioned into p main cores $\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_p \in \mathbb{R}^{c\times c\times c}$ and q auxiliary cores $\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_q \in \mathbb{R}^{1\times 1\times 1}$, with corresponding weights $a_1, a_2, \cdots, a_q \in \mathbb{R}$. Therefore, the form of the scoring function $\phi(e_i, r_k, e_j)$ of SPAC can be formulated as

$$\phi(e_i, r_k, e_j) = \sigma(\tilde{\mathcal{G}}_{SP} \times_1 \boldsymbol{e}_i \times_2 \boldsymbol{r}_k \times_3 \boldsymbol{e}_j) \tag{5}$$

$$= \sigma(\sum_{l=1}^{p} \mathcal{M}_l \times_1 \boldsymbol{E}_i[l,:] \times_2 \boldsymbol{R}_k[l,:] \times_3 \boldsymbol{E}_j[l,:] + \sum_{l=1}^{q} a_l \tilde{\boldsymbol{e}}_i[l]\tilde{\boldsymbol{r}}_k[l]\tilde{\boldsymbol{e}}_j[l]), \tag{6}$$

where $(\cdot)[l,:]$, $(\cdot)[l]$ denote the l-th row of a matrix and the l-th element of a vector, respectively. This scoring function can be viewed as the sum of the results of p Tucker decompositions and the result of a CP decomposition. Specifically, the sparse core tensor $\tilde{\mathcal{G}}_{SP}$ can be regarded as a block diagonal tensor with p main cores and q auxiliary cores. The auxiliary cores only interact within the same dimension, and are uncorrelated to other dimensions. Conversely, the main cores generates dense interactions, where all dimensions within the same group engage in multiplication once. Under this design, it is expected that the sparse interactions produced by the auxiliary core could capture vague, general semantic information, while the dense interactions generated by the main cores could capture refined semantic associations.

Intermediate output gating

Note that given a triplet $t = (e_i, r_k, e_j)$, $\forall l \neq m \in \{1, 2, \cdots, p\}, u \neq v \in \{1, 2, \cdots, q\}$, we have $\frac{\partial^2 \phi(e_i,r_k,e_j)}{\partial \boldsymbol{E}_i[l,:]\partial \boldsymbol{E}_i[m,:]} \equiv \boldsymbol{0}$, $\frac{\partial^2 \phi(e_i,r_k,e_j)}{\partial \tilde{\boldsymbol{e}}_i[u]\partial \tilde{\boldsymbol{e}}_i[v]} \equiv \boldsymbol{0}$, and $\frac{\partial^2 \phi(e_i,r_k,e_j)}{\partial \boldsymbol{E}_i[l,:]\partial \tilde{\boldsymbol{e}}_i[u]} \equiv \boldsymbol{0}$. This indicates that the gradient information cannot be shared among different groups (i.e., two different rows of matrix $\boldsymbol{E}_i, \boldsymbol{R}_k, \boldsymbol{E}_j$). In other words, there is no interaction between two different embedding groups, which may lead to a performance degradation. To enhance the interactions between groups, we introduce a gating mechanism to control the output of intermediate variables.

Intermediate output: When calculating the score function of a triplet $t = (e_i, r_k, e_j)$, SPAC first calculates the mode product of the core tensor $\tilde{\mathcal{G}}_{SP}$ and the relation embedding $\boldsymbol{r}_k$ along the second dimension, whose result is a relation-aware transformation matrix $\boldsymbol{M}_{r_k} \in \mathbb{R}^{d\times d}$. Then, the relation-aware transformation matrix $\boldsymbol{M}_{r_k}$ is multiplied by the entity embedding $\boldsymbol{e}_i$ to obtain the intermediate output $\boldsymbol{\xi}_{r_k,e_i} \in \mathbb{R}^d$. Finally, we calculate the dot product of the intermediate output $\boldsymbol{\xi}_{r_k,e_i}$ and the tail entity embedding $\boldsymbol{e}_j$, and apply the sigmoid function to obtain the score of the triplet $\phi(e_i, r_k, e_j)$. The process of calculating the intermediate variable can be expressed as

$$\boldsymbol{\xi}_{r_k,e_i} = \boldsymbol{M}_{r_k}\boldsymbol{e}_i = (\tilde{\mathcal{G}}_{SP} \times_2 \boldsymbol{r}_k)\boldsymbol{e}_i. \tag{7}$$

Since $\boldsymbol{M}_{r_k}$ is a block diagonal matrix, the dimensions of the intermediate variable $\boldsymbol{\xi}_{r_k,e_i}$ can be divided into $p+q$ groups, akin to the embedding vectors. For convenience, we denote the corresponding l-th main group of intermediate outputs as $\boldsymbol{\Xi}_{r_k,e_i}[l,:]$ and the auxiliary groups as $\tilde{\boldsymbol{\xi}}_{r_k,e_i}$.

Gating mechanism: The main groups has more frequent inter-dimensional interactions compared to the auxiliary groups, resulting in a magnitude difference between the intermediate variables of the main groups and those of the auxiliary groups. This disparity hinders training stability. To address this issue, we apply batch normalization to

the intermediate variable $\boldsymbol{\xi}_{r_k,e_i}$, and then employs a gating mechanism to enhance inter-group interactions, combining input vectors $\boldsymbol{e_i}$ and $\boldsymbol{r_k}$ through a linear transformation and sigmoid activation in the gating network G. This produces a gating vector, which is then multiplied by the intermediate variable $\boldsymbol{\xi}_{r_k,e_i}$, yielding the gated intermediate output

$$\boldsymbol{\xi}'_{r_k,e_i} = G([\boldsymbol{e}_i^\top, \boldsymbol{r}_k^\top]^\top) \odot \mathrm{BN}[(\tilde{\mathcal{G}}_{SP} \times_2 \boldsymbol{r}_k)\boldsymbol{e}_i], \quad (8)$$

where $\odot$ denotes the element-wise product, and $\mathrm{BN}(\cdot)$ denotes the batch normalization operation. The output $\boldsymbol{\xi}'_{r_k,e_i}$ after the gating mechanism can be viewed as a non-linear transformation of $\boldsymbol{e}_i$ that is related to $\boldsymbol{r}_k$, which further enhances the network's expressive power. Mathematically, replacing $\boldsymbol{\xi}_{r_k,e_i}$ by $\boldsymbol{\xi}'_{r_k,e_i}$ as the new intermediate output, we have

$$\frac{\partial^2 \phi(e_i, r_k, e_j)}{\partial \boldsymbol{E}_i[l,:] \partial \boldsymbol{E}_i[m,:]} \equiv \mathcal{F}_{l,m}(\boldsymbol{e}_i, \boldsymbol{r}_k), \quad (9)$$

$$\frac{\partial^2 \phi(e_i, r_k, e_j)}{\partial \tilde{\boldsymbol{e}}_i[u] \partial \tilde{\boldsymbol{e}}_i[v]} \equiv \mathcal{H}_{u,v}(\boldsymbol{e}_i, \boldsymbol{r}_k), \quad (10)$$

$$\frac{\partial^2 \phi(e_i, r_k, e_j)}{\partial \boldsymbol{E}_i[l,:] \partial \tilde{\boldsymbol{e}}_i[u]} \equiv \mathcal{I}_{l,v}(\boldsymbol{e}_i, \boldsymbol{r}_k), \quad (11)$$

where $\mathcal{F}_{l,m}(\boldsymbol{e}_i, \boldsymbol{r}_k)$, $\mathcal{H}_{u,v}(\boldsymbol{e}_i, \boldsymbol{r}_k)$, $\mathcal{I}_{l,v}(\boldsymbol{e}_i, \boldsymbol{r}_k)$ are non-linear functions corresponding to the original embeddings $\boldsymbol{e}_i, \boldsymbol{r}_k$. The dimensions of the intermediate variable therefore become correlated during the parameter update process. With the above manipulations, the revised form of the scoring function $\phi(e_i, r_k, e_j)$ can be rewritten as

$$\phi(e_i, r_k, e_j) = \sigma\Big(\sum_{l=1}^{p} \boldsymbol{\Xi}'_{r_k,e_i}[l,:] \boldsymbol{E}_j[l,:]^\top + \sum_{l=1}^{q} \tilde{\boldsymbol{\xi}}'_{r_k,e_i}[l] \tilde{\boldsymbol{e}}_j[l]\Big). \quad (12)$$

Adaptive pruning

How to ensure a model's predictive capability while compressing the dimensions of tensor cores remains another important issue. The core tensors in traditional TD models are all static, which limits the adaptability of the model. In SPAC, we propose an adaptive pruning approach to dynamically adjust the shape of the core tensor during training. Specifically, given an embedding dimension d, we first initializes a core tensor shape, which consists of p main cores of shape c_0 and q_0 auxiliary cores, denoted as $\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_p \in \mathbb{R}^{c_0 \times c_0 \times c_0}$ and $\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_{q_0} \in \mathbb{R}^{1\times 1\times 1}$, respectively. During training, the importance and diversity of each dimension in the intermediate output are evaluated. Then, the corresponding parts of the core tensor with small combination evaluation values are pruned, and those positions will be replaced by auxiliary cores. For example, consider a sparse core tensor $\tilde{\mathcal{G}}_{SP} \in \mathbb{R}^{4\times 4\times 4}$ with $p_0 = 1$, $c_0 = 3$, and $q_0 = 1$. If the first dimension is pruned, the corresponding $\mathcal{M}_1 \in \mathbb{R}^{3\times 3\times 3}$ would be reshaped into a new core tensor $\mathcal{M}'_1 \in \mathbb{R}^{2\times 2\times 2}$ with $p = 1$, $c = 2$, and $q = 2$ by removing $\mathcal{M}_1[1,:,:], \mathcal{M}_1[:,1,:], \mathcal{M}_1[:,:,1]$, and a new auxiliary core $\mathcal{A}_2 \in \mathbb{R}^{1\times 1\times 1}$ will be added to the core tensor with a new initialization. Finally, the first dimension of all the embedding vector will be put after the last dimension.

Dimension importance evaluation: The importance of each dimension can be evaluated by the average Frobenius norm. Specifically, given a core tensor $\tilde{\mathcal{G}}_{SP}$ with p main cores $\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_p \in \mathbb{R}^{c\times c\times c}$, the importance of the u-th dimension $\boldsymbol{\eta}[u]$ can be calculated as:

$$\boldsymbol{\eta}[u] = \frac{1}{3c^2} \sum_{v=1}^{p} (\|\mathcal{M}_v[u,:,:]\|_F + \|\mathcal{M}_v[:,u,:]\|_F + \|\mathcal{M}_v[:,:,u]\|_F), \quad (13)$$

where $\|\cdot\|_F$ denotes the Frobenuis norm of a matrix. Equation (13) measures the importance of the u-th slice from three different dimensions of all the main cores.

Dimension diversity evaluation: It is expected that the main core tensor can capture the nuanced semantics of the data to enhance prediction accuracy. Therefore, dimensions with high diversity across different triplets are more likely to be remained in the main cores. Conversely, dimensions demonstrating less diversity are more inclined towards pruning. Recalling Equations (5) and (7), the u-th dimension of the v-th main group of the intermediate variable $\boldsymbol{\Xi}_{r_k,e_i}[v,u]$ can be calculated by

$$\boldsymbol{\Xi}_{r_k,e_i}[v,u] = \Big(\sum_{l=1}^{c} \boldsymbol{R}_k[v,l] \mathcal{M}_v[u,l,:]\Big) \boldsymbol{E}_i[v,:]^\top, \quad (14)$$

which is related to the u-th slice of all the main cores $\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_p$ on the first dimension. Similarly, for the other two dual intermediate variables $\boldsymbol{\xi}_{e_j,e_i}$ and $\boldsymbol{\xi}_{r_k,e_j}$, the u-th dimension of these intermediate variables can be calculated by

$$\boldsymbol{\Xi}_{e_j,e_i}[v,u] = \Big(\sum_{l=1}^{c} \boldsymbol{E}_j[v,l] \mathcal{M}_v[u,:,l]\Big) \boldsymbol{E}_i[v,:]^\top, \quad (15)$$

$$\boldsymbol{\Xi}_{r_k,e_j}[v,u] = \Big(\sum_{l=1}^{c} \boldsymbol{R}_k[v,l] \mathcal{M}_v[:,l,u]\Big) \boldsymbol{E}_j[v,:]^\top, \quad (16)$$

which are associated with the u-th slice of all the main cores on the second and third dimension, respectively. Regarding the corresponding tensors $\mathcal{S}^{(N)}_{r_k,e_i}, \mathcal{S}^{(N)}_{r_k,e_j}, \mathcal{S}^{(N)}_{e_j,e_i} \in \mathbb{R}^{N\times p\times c}$ that consist of the main groups of intermediate variables $\boldsymbol{\Xi}_{r_k,e_i}, \boldsymbol{\Xi}_{r_k,e_j}, \boldsymbol{\Xi}_{e_j,e_i}$ of N triplets, respectively, the diversity of the u-th dimension $\boldsymbol{\rho}[u]$ can be calculated by

$$\boldsymbol{\rho}[u] = \frac{1}{3p} \sum_{v=1}^{p} \Big(\frac{Std(\mathcal{S}^{(N)}_{r_k,e_i}[:,v,u])}{|\mu(\mathcal{S}^{(N)}_{r_k,e_i}[:,v,u])|} + \frac{Std(\mathcal{S}^{(N)}_{r_k,e_j}[:,v,u])}{|\mu(\mathcal{S}^{(N)}_{r_k,e_j}[:,v,u])|} + \frac{Std(\mathcal{S}^{(N)}_{e_j,e_i}[:,v,u])}{|\mu(\mathcal{S}^{(N)}_{e_j,e_i}[:,v,u])|}\Big), \quad (17)$$

where $Std(\cdot)$, $\mu(\cdot)$, and $|\cdot|$ denote the standard deviation, mean, and absolute value operators, respectively. Note that Equation (17) measures the normalized variance of the intermediate variables across different triplets.

Pruning strategy: With the dimension importance and diversity evaluations in SPAC, we further rank the results

Algorithm 1: Adaptive pruning strategy in SPAC

Input: Sparse core tensor $\tilde{\mathcal{G}}_{SP}$, embedding dimension d such that $d = p_0 c + q_0$ with intialization p_0, q_0, combination weighting α, minimum core tensor shape c_{min}, buffer size N, update period T, max training epoch h
Output: Pruned core tensor $\tilde{\mathcal{G}}'_{SP}$
$\tilde{\mathcal{G}}'_{SP} \leftarrow \tilde{\mathcal{G}}_{SP}$, $c \leftarrow c_0$, $q \leftarrow q_0$, $u \leftarrow 0$, $\boldsymbol{\eta}, \boldsymbol{\rho}, \boldsymbol{\gamma} \leftarrow \mathbf{0}$, $\mathcal{S}^{(N)}_{r_k,e_i}, \mathcal{S}^{(N)}_{r_k,e_j}, \mathcal{S}^{(N)}_{e_j,e_i} \in \mathbb{R}^{N \times p \times c} \leftarrow \mathbf{0}$
for $epoch = 1, 2, \cdots, h$ **do**
`do_training()`
if $(epoch \% T == T - 1) \wedge (c > c_{min})$ **then**
$\mathcal{T}^{(N)}_{tmp} \leftarrow$ `data_sampling`(N)
$\mathcal{S}^{(N)}_{r_k,e_i}, \mathcal{S}^{(N)}_{r_k,e_j}, \mathcal{S}^{(N)}_{e_j,e_i} \leftarrow$ `calculate_intermediate_output`$(\mathcal{T}^{(N)}_{tmp}, \tilde{\mathcal{G}}'_{SP})$
$\boldsymbol{\eta} \leftarrow$ `calculate_importance`$(\tilde{\mathcal{G}}'_{SP})$
$\boldsymbol{\rho} \leftarrow$ `calculate_diversity`$(\mathcal{S}^{(N)}_{r_k,e_i}, \mathcal{S}^{(N)}_{r_k,e_j}, \mathcal{S}^{(N)}_{e_j,e_i})$
$\boldsymbol{\gamma} \leftarrow \alpha\boldsymbol{\eta} + (1-\alpha)\boldsymbol{\rho}$
$u \leftarrow$ `argmin`$(\boldsymbol{\gamma})$
$\tilde{\mathcal{G}}'_{SP} \leftarrow$ `prune_core_tensor`$(\tilde{\mathcal{G}}'_{SP}, u)$
`embedding_repositioning`(u)
$c \leftarrow c - 1$, $q \leftarrow q + p$
end if
end for
return pruned core tensor $\tilde{\mathcal{G}}'_{SP}$

of the combined dimension importance and diversity in the intermediate output

$$\boldsymbol{\gamma} = \alpha\boldsymbol{\eta} + (1-\alpha)\boldsymbol{\rho}, \tag{18}$$

where α is the combination weighting. Then, the dimension with the lowest $\boldsymbol{\gamma}[u]$ in a certain period will be pruned until the number of dimensions in the main core tensor is equal to a predefined value c_{min}. The overall process of adaptive pruning is shown in Algorithm 1.

Experiments

In this section, we conduct comprehensive experiments to evaluate the performance of SPAC on three benchmark datasets, i.e., FB15k-237 (Toutanova and Chen 2015), WN18RR (Dettmers et al. 2018), and YAGO3-10 (Mahdisoltani, Biega, and Suchanek 2013).

A comparison of prediction performance on different datasets is shown in Table 1. It is observed that SPAC outperforms state-of-the-art TD models, such as MEIM, NePTuNe, ComplEx-DURA and other simple models, and achieves the best performance on these three datasets. More specifically, for the FB15k-237 dataset, SPAC achieves 0.280 in Hits@1, which is 0.4% higher than ComplEx-DURA, 0.6% higher than MEIM, and 0.8% higher than NePTuNe. For the WN18RR dataset, SPAC achieves 0.591 in Hits@10, which is 1.8% higher than ComplEx-DURA, 1.4% higher than MEIM, and 3.4% higher than NePTuNe. For the large dataset YAGO3-10, SPAC still slightly outperforms MEIM and significantly outperforms other models like DistMult and ComplEx-N3.

Note that the last four rows of Table 1 show the perfromances of KGC models that use either language models or additional information except the training triples. For example, KG-BERT uses a pretrained language model BERT and treats triples as token sequences when calculating the scoring function. The results also demonstrate that the SPAC model without any additional information can achieve competitive performance compared to these models and even beat all of them on some metrics. This implies the effectiveness of the SPAC model. In the following section, we will discuss the effectiveness and mechanism of SPAC and provide a complexity analysis compared to other models.

Complexity Analysis

After the pruning stage, the core tensor size of SPAC becomes $(1-\tau)d + \tau^3 \frac{d^3}{p^2}$. The core tensor size of SPAC includes a linear term $(1-\tau)d$, representing the auxiliary core dimensions, and a scaling factor τ^3 for the higher-order term d. In most cases, $\tau^3 \frac{d^3}{p^2} \gg (1-\tau)d$, which makes the linear term almost negligible. Therefore, the total parameter size of SPAC is approximately $|\mathcal{E}|d + |\mathcal{R}|d + \tau^3 \frac{d^3}{p^2} + 2d^2$. In Table 2, a comparison of the parameter size and computational speed of different TD models and neural network models across three datasets is summarized. It demonstrates that SPAC is more efficient than most of the baseline models in terms of computational cost and parameter size. For example, for the case with $d = 300$ and $\tau = 0.1$ running on the FB15k-237 dataset, SPAC is 2x faster than Tucker ($d = 200$) while using only half of the parameters. The result also shows that SPAC with parameter size being 1/4 and 1/14 of ComDensE and InteractE is approximately 12x and 9x faster than ComDensE and InteractE, respectively, which further proves the efficiency of SPAC. Note that all the experiments are conducted on a single RTX 4080 laptop GPU.

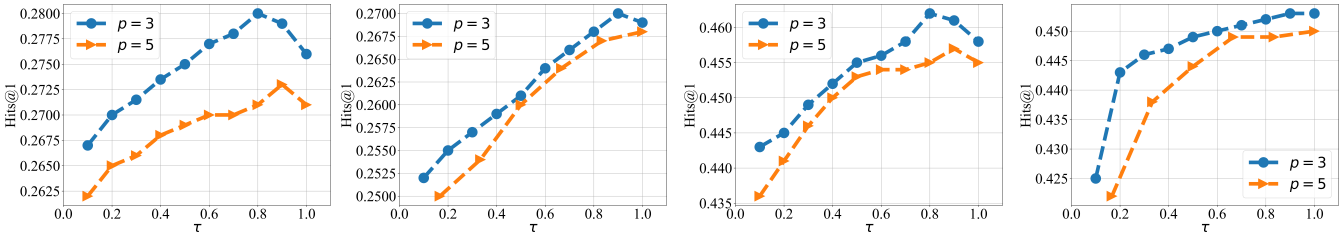
Effectiveness of sparse partitioning

To evaluate the effectiveness of the sparse partitioning mechanism in SPAC, we conduct more experiments on FB15k-237 and WN18RR with different embedding dimensions d, number of main groups p, and dense rates τ. The results are shown in Figure 2. Note that the dense rate τ is defined as the ratio of the number of dimensions in the main groups to the total number of dimensions in the core tensor. It is calculated by $\tau = \frac{pc}{d}$ for a given $d = pc + q$.

In Figure 2, it is observed that for both datasets, the setting $p = 5$ outperforms $p = 3$, and the setting $d = 300$ outperforms $d = 180$. This is because the increase of main groups p results in the decrease of interactions between dimensions, and the increase of embedding dimension d leads to the increase of model capacity. On the other hand, $\tau = 0.8$ or 0.9 may outperform $\tau = 1.0$ in some cases, which indicates that appropriately incorporating sparse interactions dominated by auxiliary cores into dense interactions can effectively enhance model performance. This improvement is due to the ability of the sparse interactions from the auxiliary cores to capture general, coarse-grained semantic information, while the dense interactions from the main cores capture fine-grained semantic details.

Model	Extra info	FB15k-237				WN18RR				YAGO3-10			
		Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10	MRR
TransE (Bordes et al. 2013)	✗	0.199	0.369	0.471	0.290	0.422	–	0.512	0.465	0.406	–	0.674	0.501
DistMult (Yang et al. 2014)	✗	0.155	0.336	0.419	0.241	0.390	0.452	0.490	0.430	0.240	–	0.540	0.340
CompGCN (Vashishth et al. 2019)	✗	0.264	0.390	0.535	0.355	0.443	0.494	0.546	0.494	–	–	–	–
R-GCN (Schlichtkrull et al. 2018)	✗	0.151	–	0.417	0.249	–	–	–	–	–	–	–	–
ConvE (Dettmers et al. 2018)	✗	0.237	0.356	0.501	0.325	0.400	0.440	0.520	0.430	0.399	–	0.658	0.488
CP (Lacroix, Usunier, and Obozinski 2018)	✗	0.244	–	0.507	0.332	0.416	–	0.485	0.438	0.495	–	0.696	0.567
CP-N3 (Friedland and Lim 2018)	✗	0.261	–	0.542	0.355	0.432	–	0.541	0.469	0.504	–	0.703	0.575
TuckER (Balažević, Allen, and Hospedales 2019)	✗	0.266	0.394	0.544	0.358	0.443	0.482	0.526	0.470	–	–	–	–
ComplEx-DURA (Zhang, Cai, and Wang 2020)	✗	0.276	–	0.560	0.371	0.449	–	0.573	0.491	0.511	–	0.713	0.584
RotatE (Nickel et al. 2011)	✗	0.241	0.375	0.533	0.338	0.417	0.492	0.552	0.462	0.402	0.550	0.670	0.495
ComDensE (Kim and Baek 2022)	✗	0.265	–	0.536	0.356	0.440	–	0.538	0.473	–	–	–	–
InteractE (Vashishth et al. 2020)	✗	0.263	0.535	0.354	0.430	0.528	0.463	0.462	0.687	0.541			
MEI (Tran and Takasu 2020)	✗	0.271	0.402	0.552	0.365	0.444	0.496	0.551	0.496	0.505	0.622	0.709	0.578
MEIM (Nghiep Tran and Takasu 2022)	✗	0.274	0.406	0.557	0.369	0.458	0.518	0.577	0.499	0.514	0.625	0.716	0.585
NePTuNe (Sonkar, Katiyar, and Baraniuk 2022)	✗	0.272	0.404	0.547	0.366	0.455	0.507	0.557	0.491	–	–	–	–
SAttLE (Baghershahi, Hosseini, and Moradi 2023)	✗	0.268	0.396	0.545	0.360	0.454	0.508	0.558	0.491	–	–	–	–
SPAC	✗	**0.280**	**0.411**	**0.562**	**0.372**	**0.462**	**0.521**	**0.591**	**0.501**	**0.517**	**0.627**	**0.720**	**0.586**
KG-BERT (Yao, Mao, and Luo 2019)	✓	–	–	0.420	–	–	–	0.524	–	–	–	–	–
kNN-KGE (Wang et al. 2023)	✓	0.280	0.404	0.550	0.370	0.525	0.604	0.683	0.579	–	–	–	–
KG-R3 (Pahuja et al. 2023)	✓	0.315	0.413	0.539	0.390	0.439	0.481	0.537	0.472	–	–	–	–
Relphormer (Bi et al. 2024)	✓	0.314	–	0.481	0.371	0.448	–	0.591	0.495	–	–	–	–

Table 1: **A comparison of prediction performance on different datasets.** The best result is in **bold**, and the second best result is underlined (**The last 4 rows are not considered in the ranking**). Note that *Extra info* indicates that the model uses either language models or additional information except the training data.



(a) FB15k-237, $d = 300$. (b) FB15k-237, $d = 180$. (c) WN18RR, $d = 300$. (d) WN18RR, $d = 180$.

Figure 2: The performance of SPAC with different embedding dimensions d, number of main groups p, and dense rate τ.

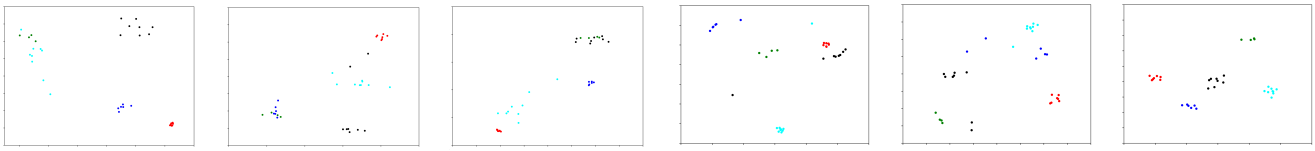
It is expected that the auxiliary cores can assist the main cores to some extent in capturing semantics. To better illustrate this mechanism, the embedding vectors of the main groups with and without auxiliary cores on the FB15k-237 dataset are visualized by using t-SNE, and compared to the embeddings of ComplEx, MEIM, and Tucker. The t-SNE visualization is shown in Figure 3. Compared to ComplEx, MEIM, and Tucker in Figures 3a, 3b, and 3c, which fail to discriminate different classes of entities well, SPAC achieves a far more reasonable clustering performance, which indicates its strong representation ability and generalization ability. In Figure 3d, without the auxiliary cores, the clustering of the main groups is generally reasonable, but there are a few outliers. In Figure 3e, the auxiliary groups itself can only capture general semantic information, resulting in poor clustering performance. In Figure 3f, with the presence of the auxiliary cores, the embeddings of the main groups achieve better clustering without obvious outliers. The result demonstrates that the auxiliary groups can assist the main groups in effective fine-grained semantic extraction.

Ablation study

The adaptive pruning strategy can adjust the core tensor automatically, which significantly reduces the number of parameters and only remains the important ones. On the other hand, the intermediate variable gating mechanism effectively controls the flow of information, enhances the interactions between sub-core tensors, and incorporates nonlinearity into the tensor decomposition.

To investigate the effectiveness of gating mechanism and pruning strategy in SPAC, we conduct an ablation study on the FB15k-237 dataset. The result is shown in Figure 4, where the symbols *S, G, P* denote the sparse partitioning, gating mechanism, and adaptive pruning strategy, respectively, and *M* denotes the model without auxiliary cores and auxiliary groups. The result shows that removing the gating mechanism or the dynamic pruning process from the complete SPAC model leads to a performance degradation. More specifically, the performance degradation caused by removing the dynamic pruning process increases with the increase of τ. This indicates that an effective core tensor pruning strategy can minimize the parameter count while preserving the original model's capability to the greatest extent. Moreover, this performance gain is more pronounced for sparser core tensors. This is because the pruning strategy uses diversity as one of the criteria, which tends to remain the main core components that can extract differentiated semantics.

On the other hand, discarding the auxiliary cores results



(a) Embeddings of **ComplEx**. (b) Embeddings of **MEIM**. (c) Embeddings of **Tucker**. (d) **Without** auxiliary cores. (e) **Auxiliary groups only**. (f) **With** auxiliary cores.

Figure 3: **The t-SNE visualization** of the embedding vectors on FB15k-237, where $d = 300$, $p = 3$, and $\tau = 0.8$. Note that each point represents an entity, and the different colors represent different classes of entities.

Model	Total parameter count	Time per batch (ms)
CP	30M, 82M, 123M	**5.6**, **5.7**, **5.9**
ComplEx	60M, 164M, 246M	*6.1*, 6.1, *6.4*
SimplE	30M, 82M, 123M	6.1, *6.4*, *6.4*
Tucker	11M, 16M, –	12.83, 12.8, –
MEI	*5.5M*, *13.3M*, 62.5M	9.9, 9.8, 17.6
MEIM	7.5M, 15.3M, 66.5M	10.3, 10.3, 18.1
InteractE	18M, 60M, –	75.4, 78.1, –
ComDensE	66M, 33M, –	54.2, 56.8, –
SPAC ($\tau = 0.5$)	5.06M, 12.86M, *62.63M*	6.8, 6.8, 9.1
SPAC ($\tau = 0.1$)	**4.68M**, **12.48M**, **62.00M**	6.1, 6.1, 6.2

Table 2: Parameter efficiency and training cost. The three numbers in each column correspond to FB15k-237, WN18RR, and YAGO3-10, respectively. The **bold**, underlined, and the *italic* styles indicate the best, the second best and the third best results, respectively.

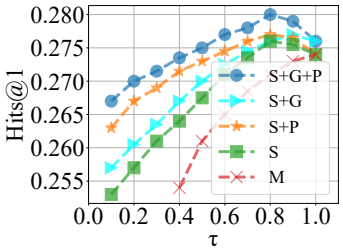
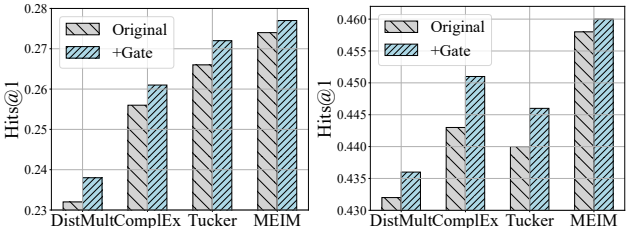


Figure 4: Ablation study on FB15k-237.

in a significant performance loss, which leads to a sharp decline in semantic space representation ability when the dimensionality is reduced. The presence of the auxiliary cores allows the model to achieve a substantial performance gain with only a small increase in core tensor parameters.

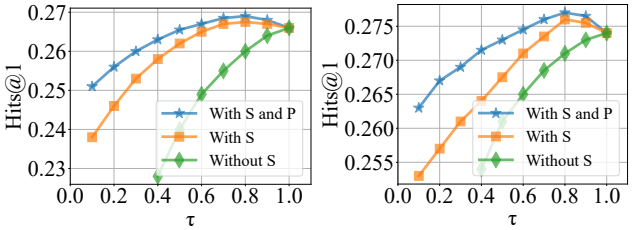
Extension to other TD models

The gating mechanism used in SPAC can be effectively extended to other TD models without altering the original model structure. For example, we apply the gating mechanism to DistMult, ComplEx, Tucker, and MEIM. The results in Figure 5 indicate that the gating mechanism contributes to a performance improvement to these models with few additional parameters. In addition, we also apply SPAC model's pruning strategy to the Tucker and MEIM models, as shown in Figures 6. The results show that the pruning strategy can



(a) FB15k-237. (b) WN18RR.

Figure 5: Gating mechanism extended to other models.



(a) Tucker. (b) MEIM.

Figure 6: Pruning strategy extended to Tucker and MEIM on FB15k-237.

effectively reduce the number of parameters while maintaining the original model's performance.

Conclusion

In this paper, we proposed a novel tensor decomposition model for KGC tasks, which was called SPAC. This model employed sparse partitioning to group embeddings into main and auxiliary groups, capturing semantic information at different granularities and enhancing the extraction of semantics. Moreover, SPAC optimized inter-group interactions by introducing gating intermediate variables and dynamically and adaptively pruning the core tensor, which minimized the number of parameters and maintained the model's expressive power. Comprehensive experimental results demonstrated that our model achieved state-of-the-art performance on three benchmark datasets, FB15k-237, WN18RR and YAGO3-10, and outperformed existing TD models with fewer parameters. Ablation studies were conducted to demonstrate the effectiveness of gating mechanism and adaptive pruning strategy.

Acknowledgments

This work was supported in part by the National Key Research and Development Program of China(No.2021YFB2900600), and in part by the National Natural Science Foundation of China under Grant 62371045 and Grant 6240011751.

References

Baghershahi, P.; Hosseini, R.; and Moradi, H. 2023. Self-attention presents low-dimensional knowledge graph embeddings for link prediction. *Knowledge-Based Systems*, 260: 110124.

Balažević, I.; Allen, C.; and Hospedales, T. M. 2019. Tucker: Tensor factorization for knowledge graph completion. *arXiv preprint arXiv:1901.09590*.

Bi, Z.; Cheng, S.; Chen, J.; Liang, X.; Xiong, F.; and Zhang, N. 2024. Relphormer: Relational Graph Transformer for Knowledge Graph Representations. *Neurocomputing*, 566: 127044.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

Cha, H.-J.; Choi, S.-W.; Lee, E.-B.; and Lee, D.-M. 2023. Knowledge Retrieval Model Based on a Graph Database for Semantic Search in Equipment Purchase Order Specifications for Steel Plants. *Sustainability*, 15(7): 6319.

Dettmers, T.; Minervini, P.; Stenetorp, P.; and Riedel, S. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Friedland, S.; and Lim, L.-H. 2018. Nuclear norm of higher-order tensors. *Mathematics of Computation*, 87(311): 1255–1281.

Kazemi, S. M.; and Poole, D. 2018. Simple embedding for link prediction in knowledge graphs. *Advances in neural information processing systems*, 31.

Kim, M.; and Baek, S. 2022. ComDensE: Combined Dense Embedding of Relation-aware and Common Features for Knowledge Graph Completion. In *2022 26th International Conference on Pattern Recognition (ICPR)*, 1989–1995. IEEE.

Lacroix, T.; Usunier, N.; and Obozinski, G. 2018. Canonical tensor decomposition for knowledge base completion. In *International Conference on Machine Learning*, 2863–2872. PMLR.

Mahdisoltani, F.; Biega, J.; and Suchanek, F. M. 2013. Yago3: A knowledge base from multilingual wikipedias. In *CIDR*.

Nghiep Tran, H.; and Takasu, A. 2022. MEIM: Multi-partition embedding interaction beyond block term format for efficient and expressive link prediction. *arXiv e-prints*, arXiv–2209.

Nickel, M.; Tresp, V.; Kriegel, H.-P.; et al. 2011. A three-way model for collective learning on multi-relational data. In *Icml*, volume 11, 3104482–3104584.

Pahuja, V.; Wang, B.; Latapie, H.; Srinivasa, J.; and Su, Y. 2023. A retrieve-and-read framework for knowledge graph link prediction. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 1992–2002.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Van Den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling relational data with graph convolutional networks. In *The Semantic Web: 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*, 593–607. Springer.

Sonkar, S.; Katiyar, A.; and Baraniuk, R. 2022. NePTuNe: Neural Powered Tucker Networkfor Knowledge Graph Completion. In *Proceedings of the 10th International Joint Conference on Knowledge Graphs*, IJCKG '21, 177–180. New York, NY, USA: Association for Computing Machinery. ISBN 9781450395656.

Toutanova, K.; and Chen, D. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, 57–66.

Tran, H. N.; and Takasu, A. 2020. Multi-partition embedding interaction with block term format for knowledge graph completion. *arXiv preprint arXiv:2006.16365*.

Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, É.; and Bouchard, G. 2016. Complex embeddings for simple link prediction. In *International conference on machine learning*, 2071–2080. PMLR.

Vashishth, S.; Sanyal, S.; Nitin, V.; Agrawal, N.; and Talukdar, P. 2020. Interacte: Improving convolution-based knowledge graph embeddings by increasing feature interactions. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 3009–3016.

Vashishth, S.; Sanyal, S.; Nitin, V.; and Talukdar, P. 2019. Composition-based multi-relational graph convolutional networks. *arXiv preprint arXiv:1911.03082*.

Wang, P.; Xie, X.; Wang, X.; and Zhang, N. 2023. Reasoning through memorization: Nearest neighbor knowledge graph embeddings. In *CCF International Conference on Natural Language Processing and Chinese Computing*, 111–122. Springer.

Yang, B.; Yih, W.-t.; He, X.; Gao, J.; and Deng, L. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Yang, Y.; Huang, C.; Xia, L.; and Huang, C. 2023. Knowledge graph self-supervised rationalization for recommendation. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*, 3046–3056.

Yao, L.; Mao, C.; and Luo, Y. 2019. KG-BERT: BERT for knowledge graph completion. *arXiv preprint arXiv:1909.03193*.

Zhang, S.; Tay, Y.; Yao, L.; and Liu, Q. 2019a. Quaternion knowledge graph embeddings. *Advances in neural information processing systems*, 32.

Zhang, Z.; Cai, J.; and Wang, J. 2020. Duality-Induced Regularizer for Tensor Factorization Based Knowledge Graph

Completion. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 21604–21615. Curran Associates, Inc.

Zhang, Z.; Han, X.; Liu, Z.; Jiang, X.; Sun, M.; and Liu, Q. 2019b. ERNIE: Enhanced language representation with informative entities. *arXiv preprint arXiv:1905.07129*.