ECAI 2020
G.D. Giacomo et al. (Eds.)

doi:10.3233/FAIA200173

Multi-Partition Embedding Interaction with Block Term Format for Knowledge Graph Completion

Hung Nghiep Tran[1] and **Atsuhiro Takasu**[1,2]

Abstract. Knowledge graph completion is an important task that aims to predict the missing relational link between entities. Knowledge graph embedding methods perform this task by representing entities and relations as embedding vectors and modeling their interactions to compute the matching score of each triple. Previous work has usually treated each embedding as a whole and has modeled the interactions between these whole embeddings, potentially making the model excessively expensive or requiring specially designed interaction mechanisms. In this work, we propose the multi-partition embedding interaction (MEI) model with block term format to systematically address this problem. MEI divides each embedding into a multi-partition vector to efficiently restrict the interactions. Each local interaction is modeled with the Tucker tensor format and the full interaction is modeled with the block term tensor format, enabling MEI to control the trade-off between expressiveness and computational cost, learn the interaction mechanisms from data automatically, and achieve state-of-the-art performance on the link prediction task. In addition, we theoretically study the parameter efficiency problem and derive a simple empirically verified criterion for optimal parameter trade-off. We also apply the framework of MEI to provide a new generalized explanation for several specially designed interaction mechanisms in previous models.

1 Introduction

Knowledge graphs are a popular data format for representing knowledge about entities and their relationships as a collection of triples, with each triple (h, t, r) denoting the fact that relation r exists between head entity h and tail entity t. Large real-world knowledge graphs, such as Freebase [3] and Wikidata [27] have found important applications in many artificial intelligence tasks, such as question answering, semantic search, and recommender systems, but they are usually incomplete. Knowledge graph completion, or link prediction, is a task that aims to predict new triples based on existing triples. Knowledge graph embedding methods perform this task by representing entities and relations as embeddings and modeling their interactions to compute a score that predicts the existence of each triple. These models also provide the embeddings as a useful representation of the whole knowledge graph that may enable new applications of knowledge graphs in artificial intelligence tasks [24].

In a knowledge graph embedding model, the matching score is computed based on the *interaction* between the entries of embeddings. The *interaction mechanism* is the function that computes the score from the embedding entries. The *interaction pattern* specifies which entries interact with each other and how; thus, it can define the interaction mechanism in a simple manner. For example, in DistMult [29], the interaction pattern is the diagonal matching matrix between head and tail embedding vectors, as detailed in Section 2.

Most previous works treat embedding as a whole and model the interaction between the whole embeddings. For example, the bilinear model RESCAL [19] and the recent model TuckER [1] can model very general interactions between every entry of the embeddings, but they cannot scale to large embedding size. One popular approach to this problem is to design special interaction mechanisms to restrict the interactions between only a few entries, for example, DistMult [29] and recent state-of-the-art models HolE [18], ComplEx [25], and SimplE [11, 14]. However, these interaction mechanisms are specifically designed and fixed, which may pose questions about optimality or extensibility on a specific knowledge graph.

In this work, we approach the problem from a different angle. We explicitly model the internal structure of the embedding by dividing it into multiple partitions, enabling us to restrict the interactions in a triple to only entries in the corresponding embedding partitions of head, tail, and relation. The local interaction in each partition is modeled with the classic Tucker format [26] to learn the most general linear interaction mechanisms, and the score of the full model is the sum score of all local interactions, which can be viewed as the block term format [5] in tensor calculus. The result is a multi-partition embedding interaction (MEI) model with block term format that provides a systematic framework to control the trade-off between expressiveness and computational cost through the partition size, to learn the interaction mechanisms from data automatically through the local Tucker core tensors, and to achieve state-of-the-art performance on the link prediction task using popular benchmarks.

In general, our contributions include the following.

- We introduce a new approach to knowledge graph embedding, the multi-partition embedding interaction, which models the internal structure of the embeddings and systematically controls the trade-off between expressiveness and computational cost.
- In this approach, we propose the standard multi-partition embedding interaction (MEI) model with block term format, which learns the interaction mechanism from data automatically through the Tucker core tensors.
- We theoretically analyze the framework of MEI and apply it to provide intuitive explanations for the specially designed interaction mechanisms in several previous models. In addition, we are the first to formally study the parameter efficiency problem and derive a simple optimal trade-off criterion for MEI.
- We empirically show that MEI is efficient and can achieve state-of-the-art results on link prediction using popular benchmarks.

[1] The Graduate University for Advanced Studies, SOKENDAI, Tokyo, Japan. Email: nghiepth@nii.ac.jp
[2] National Institute of Informatics, Tokyo, Japan. Email: takasu@nii.ac.jp

2 Related Work

In this section, we introduce the notations and review the related knowledge graph embedding models.

2.1 Background

In general, we denote scalars by normal lower case such as a, vectors by bold lower case such as $\boldsymbol{a}$, matrices by bold upper case serif such as $\boldsymbol{A}$, and tensors by bold upper case sans serif such as $\mathbf{A}$.

A knowledge graph is a collection of triples $\mathcal{D}$, with each triple denoted as a tuple (h, t, r), such as *(UserA, Movie1, Like)*, where h and t are head and tail entities in the entity set $\mathcal{E}$ and r belongs to the relation set $\mathcal{R}$. A knowledge graph can be modeled as a labeled-directed multigraph, where the nodes are entities and each edge corresponds to a triple, with the relation being the edge label. A knowledge graph can also be represented by a third-order binary *data tensor* $\mathbf{G} \in \{0, 1\}^{|\mathcal{E}| \times |\mathcal{E}| \times |\mathcal{R}|}$, where each entry $g_{htr} = 1 \Leftrightarrow (h, t, r)$ exists in $\mathcal{D}$.

Knowledge graph embedding models usually take a triple (h, t, r) as input and then represent it as embeddings and model their interactions to compute a matching score $\mathcal{S}(h, t, r)$ that predicts the existence of that triple.

2.2 Knowledge Graph Embedding Methods

Knowledge graph embedding is an active research topic with many different methods. Based on the interaction mechanisms, they can be roughly divided into three main categories: (1) *semantic matching models* are based on similarity measures between the head and tail embedding vectors, (2) *neural-network-based models* are based on neural networks as universal approximators to compute the matching score, and (3) *translation-based models* are based on the geometric view of relation embeddings as translation vectors [23, 28].

Semantic Matching Models RESCAL [19] is a general model that uses a bilinear map to model the interactions between the whole head and tail entity embedding vectors, with the relation embedding being used as the matching matrix, such that

$$\mathcal{S}(h, t, r) = \boldsymbol{h}^\top \boldsymbol{M}_r \boldsymbol{t}, \tag{1}$$

where $\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^D$ are the embedding vectors of h and t, respectively, and $\boldsymbol{M}_r \in \mathbb{R}^{D \times D}$ is the relation embedding matrix of r, with D being the embedding size. However, the matrix $\boldsymbol{M}_r$ grows quadratically with embedding size, making the model expensive and prone to overfitting. TuckER [1] is a recent model extending RESCAL by using the Tucker format [26]. However, it also models the interactions between the whole head, tail, and relation embedding vectors, making the core tensor in the Tucker format grow cubically with the embedding size, and also quickly becomes expensive.

One approach to reducing computational cost is to design special interaction mechanisms that restrict the interactions between a few entries of the embeddings. For example, DistMult [29] is a simplification of RESCAL in which the relation embedding is a diagonal matrix, equivalently a vector $\boldsymbol{r} \in \mathbb{R}^D$, such that $\boldsymbol{M}_r = \text{diag}(\boldsymbol{r})$. Its score function can also be written as a trilinear product

$$\mathcal{S}(h, t, r) = \langle \boldsymbol{h}, \boldsymbol{t}, \boldsymbol{r} \rangle = \textstyle\sum_i h_i t_i r_i, \tag{2}$$

which is an extension of the dot product to three vectors.

DistMult is fast but restrictive and can only model symmetric relations. Most recent models focus on designing interaction mechanisms that aim to be richer than DistMult while achieving a low computational cost. For example, HolE [18] uses a circular correlation between the head and tail embedding vectors; ComplEx [25] uses complex-valued embedding vectors, $\boldsymbol{h}, \boldsymbol{t}, \boldsymbol{r} \in \mathbb{C}^D$, and a special complex-valued vector trilinear product; and SimplE [11, 14] represents each entity as two role-based embedding vectors and augments an inverse relation embedding vector. In our previous work [23], we analyzed knowledge graph embedding methods from the perspective of a weighted sum of trilinear products to propose a more advanced Quaternion-based interaction mechanism and showed its promising results, which were later confirmed in a concurrent work [30]. However, these interaction mechanisms are specially designed and fixed, potentially causing them to be suboptimal or difficult to extend.

In this work, we propose a multi-partition embedding interaction framework to automatically learn the interaction mechanism and systematically control the trade-off between expressiveness and computational cost.

Semantic matching models are related to tensor decomposition methods where the embedding model can employ a standard tensor representation format in tensor calculus to represent the data tensor, such as the CP tensor rank format [9], Tucker format [26], and block term format [5]. However, when applied to knowledge graph embedding, there are some differences, such as changing from continuous tensor to binary tensor, relaxation of constraints for data analysis, and different solvers [13]. We analyze the connections to the related tensor decomposition methods in Section 3.2.

Neural-Network-based Models These models aim to learn a neural network, to automatically model the interaction. Recent models using convolutional neural networks such as ConvE [6] can achieve good results by sharing the convolution weights. However, they are restricted by the input format to the neural network [6], and the operations are generally less expressive than direct interactions between the entries of the embedding vectors [18]. We will empirically compare with them.

Translation-based Models The main advantages of these models are their simple and intuitive mechanism with the relation embeddings as the translation vectors [4]. However, it has been shown that they have limitations in expressiveness [11]. The recent model TorusE [8] improves the translation-based models by embedding in the compact torus space instead of real-valued vector space and achieves good results. We will also empirically compare with them.

3 Multi-Partition Embedding Interaction with Block Term Format

In this section, we motivate, formulate, and analyze the MEI model, illustrated in Fig. 1. We construct MEI with two main concepts:

1. *Multi-Partition Embedding Interaction:* Each embedding vector $\boldsymbol{v} \in \mathbb{R}^D$ is divided into K partitions, and the interactions in each triple are restricted to only entries in the corresponding partitions $\boldsymbol{v}_{k:}$. For simplicity, we assume all partitions have the same size C, then $\boldsymbol{v}$ can be denoted conveniently as a matrix $\boldsymbol{V} \in \mathbb{R}^{K \times C}$, where $D = KC$, each row vector $\boldsymbol{v}_{k:}$ is called a partition, and each column vector $\boldsymbol{v}_{:c}$ is called a component.
2. *Modeling the Interaction with Block Term Format:* The local interaction is modeled with the Tucker format [26], which is the most

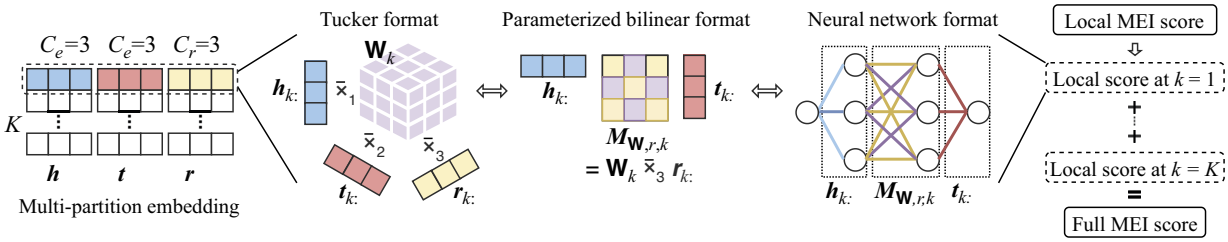

Figure 1. MEI architecture: multi-partition embedding vectors that interact only between the corresponding partitions. This figure illustrates a MEI model with block term format in three different views for the local-partition interaction: Tucker format, parameterized bilinear format, and neural network format.

general linear model that computes the weighted sum of all entry product combinations in the interacting partitions. The block term format [5] emerges from the sum score of all local interactions.

Note that the concept of multi-partition embedding interaction is highly general and intuitive, as discussed in Section 3.2.2. In this paper, we specifically adopt the Tucker and block term tensor formats to realize a simple yet general standard MEI model.

3.1 The Model

In each triple (h, t, r), the entities and relations embedding vectors $\boldsymbol{h}, \boldsymbol{t} \in \mathbb{R}^{D_e}$, and $\boldsymbol{r} \in \mathbb{R}^{D_r}$ are divided into multiple partitions conveniently denoted as the multi-partition embedding matrices $\boldsymbol{H}, \boldsymbol{T} \in \mathbb{R}^{K \times C_e}$, and $\boldsymbol{R} \in \mathbb{R}^{K \times C_r}$, respectively. Note that the embedding sizes of entity and relation are not necessarily the same.

Formally, the score function of MEI is defined as the sum score of K local interactions, with each local interaction being modeled by the Tucker format,

$$\mathcal{S}(h, t, r; \boldsymbol{\theta}) = \sum_{k=1}^{K} \left(\mathbf{W}_k \bar{\times}_1 \boldsymbol{h}_{k:} \bar{\times}_2 \boldsymbol{t}_{k:} \bar{\times}_3 \boldsymbol{r}_{k:} \right), \tag{3}$$

where $\boldsymbol{\theta}$ denotes all parameters in the model; $\mathbf{W}_k \in \mathbb{R}^{C_e \times C_e \times C_r}$ is the global core tensor at partition k; $\boldsymbol{h}_{k:}$, $\boldsymbol{t}_{k:}$, and $\boldsymbol{r}_{k:}$ are the corresponding partitions k; and $\bar{\times}_n$ denotes the *n-mode tensor product with a vector* [13], which contracts the modes of the resulting tensor to make the final result a scalar. The tensor product can be expanded as the following weighted sum

$$\mathcal{S}(h, t, r; \boldsymbol{\theta}) = \sum_{k=1}^{K} \left(\sum_{x=1}^{C_e} \sum_{y=1}^{C_e} \sum_{z=1}^{C_r} w_{xyz,k} h_{kx} t_{ky} r_{kz} \right), \tag{4}$$

where $w_{xyz,k}$ is a scalar element of the core tensor $\mathbf{W}_k$ and h_{kx}, t_{ky}, and r_{kz} denote the entries in the local partitions k.

3.2 Theoretical Analysis

Let us discuss the theoretical foundations of MEI, draw connections to previous models, and study the optimal parameter efficiency.

3.2.1 Local Interaction Modeling

We first focus on analyzing the local interactions in MEI, called local MEI, which are the building blocks of the full MEI model.

Tucker Format and Block Term Format We choose to model the local interaction at each partition by the *Tucker format* [26] of third-order tensor

$$\mathcal{S}_k(h, t, r; \boldsymbol{\theta}) = \mathbf{W}_k \bar{\times}_1 \boldsymbol{h}_{k:} \bar{\times}_2 \boldsymbol{t}_{k:} \bar{\times}_3 \boldsymbol{r}_{k:} \tag{5}$$

because the Tucker format provides the most general linear interaction mechanism between the embedding vectors, and its core tensor totally defines the interaction mechanism. With local interactions in Tucker format, the full MEI model computed by summing the scores of all local MEI models is in *block term format* [5]. Both Tucker format and block term format are standard representation formats in tensor calculus. When applied in knowledge graph embedding, there are some important modifications, such as the data tensor contains binary instead of continuous values, which change the data distribution assumptions, guarantees, constraints, and the solvers. In our work, we express the model as a neural network and use deep learning techniques to learn its parameters as detailed below.

Recently, the Tucker format was independently used in knowledge graph embedding for modeling the interactions on the embedding vector as a whole [1], while we only use the Tucker format for modeling the local interactions in our model. Thus, their model corresponds to a vanilla Tucker model, which is the special case of MEI when $K = 1$. Note that this vanilla Tucker model suffers from the scalability problem when the embedding size increases, whereas MEI essentially solves this problem. Moreover, MEI provides a general framework to reason about knowledge graph embedding methods, as discussed in Section 3.2.2.

Parameterized Bilinear Format To better understand how the core tensor defines the interaction mechanism in local MEI, we can view the local interaction in Eq. 5 as a *parameterized bilinear model*, by rewriting the tensor products as

$$\begin{aligned} \mathcal{S}_k(h, t, r; \boldsymbol{\theta}) &= \mathbf{W}_k \bar{\times}_1 \boldsymbol{h}_{k:} \bar{\times}_2 \boldsymbol{t}_{k:} \bar{\times}_3 \boldsymbol{r}_{k:} \\ &= (\mathbf{W}_k \bar{\times}_3 \boldsymbol{r}_{k:}) \bar{\times}_1 \boldsymbol{h}_{k:} \bar{\times}_2 \boldsymbol{t}_{k:} \qquad (6) \\ &= \boldsymbol{h}_{k:}^\top (\mathbf{W}_k \bar{\times}_3 \boldsymbol{r}_{k:}) \boldsymbol{t}_{k:} \qquad (7) \\ &= \boldsymbol{h}_{k:}^\top \boldsymbol{M}_{\mathbf{W},r,k} \boldsymbol{t}_{k:}, \qquad (8) \end{aligned}$$

where $\boldsymbol{M}_{\mathbf{W},r,k} \in \mathbb{R}^{C_e \times C_e}$ denotes the matching matrix of the bilinear model. Note that $\boldsymbol{M}_{\mathbf{W},r,k}$ defines the interaction patterns of the bilinear map between $\boldsymbol{h}_{k:}$ and $\boldsymbol{t}_{k:}$, but itself is defined by $\mathbf{W}_k \bar{\times}_3 \boldsymbol{r}_{k:}$. Specifically, each element $m_{\mathbf{W},r,k_{xy}}$ of the matching matrix $\boldsymbol{M}_{\mathbf{W},r,k}$ is a weighted sum of the entries in $\boldsymbol{r}_{k:}$, weighted by the mode-3 tube vector $\boldsymbol{w}_{xy:,k}$ of $\mathbf{W}_k$. Therefore, the core tensor $\mathbf{W}_k$ defines the *interaction patterns* or the *interaction mechanisms* at partition k. Compared with the standard bilinear model RESCAL, local MEI is

more flexible and efficient because its matching matrices are generated from the relation embedding vectors. Moreover, the global core tensors enable information sharing between all entities and relations, which is particularly useful when the data are sparse.

Dynamic Neural Network Format For parameter learning, we express the Tucker format as a *neural network* to employ standard deep learning techniques such as dropout [20] and batch normalization [10] to reduce overfitting and improve the convergence rate. Specifically, Eq. 8 can be seen as a *linear neural network*, where $\boldsymbol{h}_{k:}$ is the input of the network, $\boldsymbol{M}_{\mathbf{W},r,k}$ is the weight of the hidden layer, $\boldsymbol{h}_{k:}^{\top}\boldsymbol{M}_{\mathbf{W},r,k}$ is the output of the hidden layer, $\boldsymbol{t}_{k:}$ is the weight of the output neuron, and $\mathcal{S}_k$ is the output of the network. Note that the weight of the hidden layer, $\boldsymbol{M}_{\mathbf{W},r,k}$, can be seen as the output of another neural network, where $\boldsymbol{r}_{k:}$ is the input and the core tensor $\mathbf{W}_k$ is the weight. Under this format, there are four layers to apply dropout and batch normalization: $\boldsymbol{r}_{k:}$, $\boldsymbol{M}_{\mathbf{W},r,k}$, $\boldsymbol{h}_{k:}$, and $\boldsymbol{h}_{k:}^{\top}\boldsymbol{M}_{\mathbf{W},r,k}$, which are tuned as hyperparameters.

3.2.2 Multi-Partition Embedding Interaction

There are several reasons why *Multi-Partition Interaction* is superior and preferable to *Local-Partition Interaction*. Here, we present some interpretations of the full MEI model to explain its properties.

Sparse Modeling The full MEI model can be seen as a special form of *sparse parameterized bilinear models*. The matching matrix of the full MEI model is constructed by the direct sum of the matching matrices of all local MEI models, and the result is a sparse parameterized block-diagonal matrix

$$\boldsymbol{M}_{\mathbf{W},r}^{(s)} = \begin{bmatrix} \boldsymbol{M}_{\mathbf{W},r,1} & 0 & \cdots & 0 \\ 0 & \boldsymbol{M}_{\mathbf{W},r,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{M}_{\mathbf{W},r,K} \end{bmatrix}. \tag{9}$$

The score function of the full MEI model can then be written as a bilinear model

$$\mathcal{S}(h,t,r;\boldsymbol{\theta}) = \boldsymbol{h}^{\top}\boldsymbol{M}_{\mathbf{W},r}^{(s)}\boldsymbol{t}, \tag{10}$$

where $\boldsymbol{h}$, $\boldsymbol{t}$, and $\boldsymbol{r}$ are the original embedding vectors before dividing into K partitions. Similarly, we can view MEI in the form of a special *sparse Tucker model*, where the sparse core tensor $\mathbf{W}^{(s)}$ of MEI is constructed by the direct sum of the K local core tensors $\mathbf{W}_1, \ldots \mathbf{W}_K$ and the score function is written as

$$\mathcal{S}(h,t,r;\boldsymbol{\theta}) = \mathbf{W}^{(s)} \bar{\times}_1 \boldsymbol{h} \bar{\times}_2 \boldsymbol{t} \bar{\times}_3 \boldsymbol{r}. \tag{11}$$

This view provides a concrete explanation for the interaction mechanism in the MEI model, as it can be seen as imposing a sparsity constraint on the core tensor, or equivalently the matching matrices, to make the model efficient.

Multiple Interactions and the Ensemble Boosting Effect An intuitive explanation of MEI is that it models *multiple relatively independent interactions* between the head and tail entities in a knowledge graph. These interactions correspond to the separate local partitions of the embedding vectors and together define the final matching score. Technically, MEI forms an ensemble of K local interactions by summing their scores, as seen in Eq. 3, similarly to ensemble averaging. However, we argue that MEI works as an *ensemble boosting* model in a similar manner to gradient boosting methods because the summing operation is done in training and all local MEI models are optimized together. This view intuitively explains the success of MEI when each local interaction is very simple, such as when the partition size is only 1 or 2. It also suggests the empirical benefit of the ensemble boosting effect in MEI with $K > 1$ over the vanilla Tucker.

Vector-of-Vectors Embedding and the Meta-Dimensional Transforming–Matching Framework An important insight of MEI is that the embedding can be seen as a *vector of vectors*, which means a meta-vector where each meta-dimension corresponding to a local partition contains a vector entry instead of a scalar entry. Compared to scalar entry, a vector entry contains more information and allows more expressive yet simple transformation on each entry. By using this notion of vector-of-vectors embedding, we can view MEI as a *transforming–matching framework*, where the model simply transforms each meta-dimension entry of head embedding then matches it with the corresponding meta-dimension entry of tail embedding. This framework can serve as a novel general design pattern of knowledge graph embedding methods, as we show in Section 3.2.3 how it can explain the previous specially designed models.

3.2.3 Connections to Previous Specially Designed Interaction Mechanisms

There exist a few generalizations of previous embedding models that include DistMult, ComplEx, and SimplE; such as [11] explaining them using a bilinear model, [1] using a vanilla Tucker model, and [23] using a weighted sum of trilinear products. However, these generalizations consider the embedding as a whole, here we present a new generalization that considers the embedding as a multi-partition vector to provide a more intuitive explanation of these models and their specially designed interaction mechanisms.

We first construct the multi-partition embedding vector for these models. DistMult is trivial with $C = 1$ and $D = K$. For ComplEx and SimplE, $C = 2$ and $D = 2K$. In ComplEx, each partition k consists of the real and imaginary components of the entry k in a ComplEx embedding vector. In SimplE, each partition k consists of the two entries k in the two role-based embedding vectors. With this correspondence, these previous models can be written in the sparse bilinear model form of MEI in Eq. 9 and Eq. 10. For DistMult, each matching block $\boldsymbol{M}_{\mathbf{W},r,k}$ is just a scalar entry of the relation embedding vector. More interestingly, for ComplEx, each matching block is a 2×2 matrix with the *rotation pattern*, parameterized by the relation embedding vector,

$$\boldsymbol{M}_{\mathbf{W},r,k} = \begin{bmatrix} Re(r_k) & -Im(r_k) \\ Im(r_k) & Re(r_k) \end{bmatrix}.$$

For SimplE, each matching block is a 2×2 matrix with the *reflection pattern*, parameterized by the relation embedding vector,

$$\boldsymbol{M}_{\mathbf{W},r,k} = \begin{bmatrix} 0 & r_k \\ r^{(a)}{}_k & 0 \end{bmatrix},$$

where $\boldsymbol{r}^{(a)}$ is the augmented inverse relation embedding vector. CP [9] is similar to SimplE, but missing $\boldsymbol{r}^{(a)}$, making the matching matrix lose the geometrical interpretation, which is probably the reason why CP does not generalize well to new data, as reported in [23].

The interaction mechanisms of these models are totally characterized by the simple and fixed patterns in their matching blocks $\boldsymbol{M}_{\mathbf{W},r,k}$, which also specify the interaction restriction between the entries. In MEI, the interaction restriction can be varied by setting the partition size, and more importantly, the interaction patterns can be automatically learned from data.

3.2.4 Computational Analysis

Complexity For simplicity, we consider the same embedding size $D = KC$ for both entity and relation. The parameters in a MEI model include the embedding vectors of all entities, all relations, and the core tensors. On a knowledge graph with $|\mathcal{E}|$ entities and $|\mathcal{R}|$ relations, the number of parameters in MEI is $O(|\mathcal{E}|D + |\mathcal{R}|D + KC^3) = O(|\mathcal{E}|D + |\mathcal{R}|D + D^3/K^2)$. In this paper's experiments, we restrict them to the simplified case of one single shared-core tensor for all K partitions, so the number of parameters in this case is $O(|\mathcal{E}|D + |\mathcal{R}|D + C^3) = O(|\mathcal{E}|D + |\mathcal{R}|D + D^3/K^3)$.

We note a few interesting observations. First, the core tensor size of the vanilla Tucker (when $K = 1$) is much larger than the sparse core of MEI, up to K^2 times in non-shared-core MEI and K^3 times in shared-core MEI. These factors can become crucial in practice; for example, with $D = 1000$ and $K = 10, C = 100$, the vanilla Tucker core has 1 billion parameters, making it infeasible on most GPUs, while shared-core MEI has only 1 million parameters in the core tensor. Second, the partition size C can be set independently from the embedding size D; thus, the core tensor sizes can be considered as growing linearly with K in the former case of non-shared-core MEI, and as constant in the latter case of shared-core MEI.

Parameter Efficiency By using Tucker format for local interactions, MEI with block term format is *fully expressive*. However, in practice, we usually do not care about the parameter *upper bound* for fully expressiveness of the model. The more interesting property of the model is its ability to efficiently capture complex patterns in the knowledge graph. In this regard, we define the criteria to measure the expressiveness and parameter efficiency of the model. To the best of our knowledge, we are the first to formally study the parameter efficiency in knowledge graph embedding.

From the interpretation of MEI as a transforming–matching framework in Section 3.2.2, where the model first transforms each head embedding partition then simply matches it with the corresponding tail embedding partition, we see that the ability to capture complex patterns depends totally on the transformation system.

Definition 1. ***(Expressiveness)*** The expressiveness of the MEI model is measured by the degrees of freedom of the model provided by its transformation system.

For example, a linear transformation in a 3-dimensional space has 9 degrees of freedom: 3 for translation, 3 for rotation, and 3 for scaling. For a MEI model with two partitions of size $C = 3$, the sum score of two local interactions has $9 + 9 = 18$ degrees of freedom.

As mentioned earlier, the vanilla Tucker model can become excessively expensive when the embedding size is large, in which case, it is necessary to use a MEI model with a smaller partition size. To compare fairly across models, we define the *parameter efficiency*.

Definition 2. ***(Parameter efficiency)*** The parameter efficiency of a model is measured by the ratio of its expressiveness and the number of parameters.

The size of a MEI model depends on the number of partitions and the partition size. Changing any of them affects the parameter count of the model, its expressiveness, and its parameter efficiency. The effect is rather complicated; when the partition size is small, the expressiveness and model size depend mainly on the number of entities and relations; however, when the partition size becomes large enough, the effects of the core tensor outweigh that of the embeddings. Interestingly, we show that the optimal partition size can be determined on any dataset with mild assumptions as stated in the following theorem.

Theorem 1. ***(Optimal parameter efficiency)*** *Given any MEI model that represents an arbitrary knowledge graph over $|\mathcal{E}|$ entities and $|\mathcal{R}|$ relations, it is optimal in terms of maximizing the parameter efficiency P if and only if the partition size*

$$C = \min(\lfloor\sqrt{|\mathcal{E}| + |\mathcal{R}|}\rceil_P, D),$$

where $\lfloor\cdot\rceil_P$ denotes a special rounding function that selects the floor or ceiling values depending on where P evaluates to a larger value.

Proof. Consider an arbitrary knowledge graph over $|\mathcal{E}|$ entities and $|\mathcal{R}|$ relations, where $|\mathcal{E}|, |\mathcal{R}| \in \mathbb{Z}^+$ fixed for this knowledge graph, and an arbitrary MEI model representing the given knowledge graph with partition size C, number of partitions K, and embedding size $D = KC$, where $C, K, D \in \mathbb{Z}^+$. The total parameter count is

$$T = |\mathcal{E}|D + |\mathcal{R}|D + KC^3 = |\mathcal{E}|D + |\mathcal{R}|D + DC^2.$$

There are $|\mathcal{R}|$ distinct matching matrices corresponding to the number of relations, each of which include K local interactions, so the total expressiveness of the model is

$$E = |\mathcal{R}|KC^2 = |\mathcal{R}|DC.$$

The parameter efficiency of the model as defined in Definition 2 is $P = \frac{E}{T}$. For simplicity, consider its inverse,

$$P^{-1} = \frac{T}{E} = \frac{|\mathcal{E}|+|\mathcal{R}|}{|\mathcal{R}|C} + \frac{C}{|\mathcal{R}|}$$

and assume its continuous extension by interpolation[3]. Noting that P^{-1} only depends on C, we can take its first derivative w.r.t. C as

$$\frac{\mathrm{d}}{\mathrm{d}C}[P^{-1}] = -\frac{|\mathcal{E}|+|\mathcal{R}|}{|\mathcal{R}|C^2} + \frac{1}{|\mathcal{R}|},$$

which evaluates to 0 when $C = \sqrt{|\mathcal{E}| + |\mathcal{R}|}$. The second derivative of P^{-1} w.r.t. C is

$$\frac{\mathrm{d}^2}{\mathrm{d}C^2}[P^{-1}] = 2\frac{|\mathcal{E}|+|\mathcal{R}|}{|\mathcal{R}|C^3},$$

which is positive everywhere.
($\Leftarrow$) By the derivative tests, $C = \sqrt{|\mathcal{E}| + |\mathcal{R}|}$ is the global maximum of the unimodal parameter efficiency function P; thus, the optimal partition sizes must be its floor or ceiling values, which are selected depending on P evaluations, that is, $C = \lfloor\sqrt{|\mathcal{E}| + |\mathcal{R}|}\rceil_P$. When the embedding size $D < \lfloor\sqrt{|\mathcal{E}| + |\mathcal{R}|}\rceil_P$, we use the largest possible partition size; thus, the optimal $C = \min(\lfloor\sqrt{|\mathcal{E}| + |\mathcal{R}|}\rceil_P, D)$, as required.
($\Rightarrow$) By Fermat's theorem on stationary points, all local maxima occur at critical points. $C = \sqrt{|\mathcal{E}| + |\mathcal{R}|}$ is the only feasible critical point; thus, $C = \min(\lfloor\sqrt{|\mathcal{E}| + |\mathcal{R}|}\rceil_P, D)$ must be the only possible optimal partition sizes, as required. □

Theorem 1 predicts that on WN18 and WN18RR with $\approx 40,000$ entities and relations, the optimal partition size would be ≈ 200. On FB15K and FB15K-237 with $\approx 15,000$ entities and relations, the optimal partition size would be ≈ 122. When C increases, P increases and is maximized at the optimal partition sizes and then starts decreasing. Thus, when the computational budget is high enough for a large embedding size $D = KC$, it is more parameter efficient to keep the partition size C close to the optimal value and increase the number of partitions K. These predictions are empirically verified in Section 4.2. Note that this criterion only provides a general guideline

[3] Not to be confused with analytic continuation of analytic functions.

for choosing model size, but there are other detailed factors that can affect the model performance in practice, such as data sparsity, data distribution, and the ensemble boosting effect. When the dataset is very large, sparse, and unevenly distributed, it may be preferable to restrict C and try to maximize the empirical benefit of the ensemble boosting effect with a large number K of small local MEI models.

3.3 Learning

The learning problem in knowledge graph embedding methods can be modeled as the binary classification of every triple as existence and nonexistence. Because the number of nonexistent triples w.r.t. a knowledge graph is usually very large, we only sample a subset of them by the negative sampling technique [16], which replaces the h or t entities in each existent triple (h, t, r) with other random entities to obtain the locally related nonexistent triples (h', t, r) and (h, t', r) [4]. The set of existent triples is called the true data $\mathcal{D}$, and the set of nonexistent triples is called the negative sampled data $\mathcal{D}'$.

To construct the loss function, we define a Bernoulli distribution over each entry of the binary data tensor $\mathbf{G}$ to model the existence probability of each triple as $\hat{p}_{htr} = g_{htr}$. The predicted probability of the model is computed by using the standard logistic function on the matching score as $p_{htr} = \sigma(\mathcal{S}(h, t, r; \boldsymbol{\theta}))$. We can then learn both the embeddings and the core tensor from data by minimizing the cross-entropy loss:

$$\mathcal{L}(\mathcal{D}, \mathcal{D}'; \boldsymbol{\theta}) = -\sum_{(h,t,r)\in\mathcal{D}\cup\mathcal{D}'} \big(\hat{p}_{htr} \log p_{htr} + (1 - \hat{p}_{htr}) \log(1 - p_{htr})\big), \tag{12}$$

where $\hat{p} = 1$ in $\mathcal{D}$ and 0 in $\mathcal{D}'$.

4 Experiments

4.1 Experimental Settings

Datasets We use four popular benchmark datasets for link prediction, as shown in Table 1. WN18 [4] and WN18RR [6] are subsets of WordNet [17], which contains lexical relationships between words. FB15K [4] and FB15K-237 [22] are subsets of Freebase [3], which contains general facts. WN18 and FB15K are more popular, whereas WN18RR and FB15K-237 are recently built and more competitive.

Table 1. Datasets statistics.

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	Train	Valid	Test
WN18	40,943	18	141,442	5,000	5,000
FB15K	14,951	1,345	483,142	50,000	59,071
WN18RR	40,943	11	86,835	3,034	3,134
FB15K-237	14,541	237	272,115	17,535	20,466

Evaluations We evaluate and analyze MEI on the link prediction task [4]. In this task, for each true triple (h, t, r) in the test set, we replace h and t by every other entity to generate corrupted triples (h', t, r) and (h, t', r), respectively. The goal of the model is to rank the true triple (h, t, r) before the corrupted triples based on the score $\mathcal{S}$. We compute popular evaluation metrics including MRR (mean reciprocal rank, which is robust to outlier rankings) and $H@k$ for $k \in \{1, 3, 10\}$ (Hits at k, which is how many true triples are correctly ranked in the top k) [25]. The higher MRR and $H@k$ are, the better the model performs. To avoid false-negative error, i.e., some corrupted triples are actually existent, we follow the protocols used in other works for filtered metrics [4]. In this protocol, all existent triples in the training, validation, and test sets are removed from the corrupted triples set before computing the rank of the true triple.

Baselines To evaluate the prediction on the optimal parameter efficiency, we compare $\text{MEI}_{1\times200}$ (vanilla Tucker model) and $\text{MEI}_{3\times100}$. The aim is to show that the model with optimal parameter efficiency can achieve better results with even fewer parameters. We also evaluate MEI against several strong baselines including classic models such as TransE, RESCAL, DistMult, and recent state-of-the-art models such as ComplEx, SimplE, and ConvE. We also compare MEI with TorusE that uses larger embedding size, ComplEx at $K = 400$ that was retuned with reciprocal relation and full softmax loss, and RotatE without the adversarial sampling technique as this technique is not subjected to a specific model.

Implementations We trained MEI using mini-batch stochastic gradient descent with Adam optimizer [12]. We followed the 1-N scoring procedure in [6] for negative sampling of (h, t, r), where negative samples are reused multiple times for computation efficiency and the number of negative samples is different for each triple. The results of $\text{MEI}_{1\times200}$ are reproduced from the vanilla Tucker model in [1]; note that the relation embedding size $D_r = 30$ on WN18 and WN18RR only. All hyperparameters of $\text{MEI}_{3\times100}$ are tuned by random search [2], including batch size, learning rate, decay rate, batch normalization, and dropout rates, which we will publish together with the code. Note that in these experiments, we restrict them to the simplified case of one single shared-core tensor for all K partitions, as an analogy to single interaction patterns in previous specially designed models.

4.2 Main Results

Link Prediction Performance Tables 2 and 3 show the main results. In general, MEI strongly outperforms the baselines. MEI and ConvE both aim to learn the interaction between the embedding vectors, and interestingly, the multi-partition embedding interaction used in MEI can achieve better results than the convolutional neural networks used in ConvE. MEI also outperforms the general bilinear model RESCAL and other recent state-of-the-art bilinear models DistMult, ComplEx, and SimplE, which is explained by the fact that they are special cases of MEI with specific interaction patterns, as shown in Section 3.2. Compared with TorusE, the results show that an expressive interaction mechanism can help a smaller model outperform a much larger model. There are some recent techniques that help to improve the performance of old models, but we show that MEI can still outperform retuned ComplEx and RotatE reported with comparable settings. Moreover, note that MEI is highly general and potentially preferable for sophisticated datasets.

Optimal Parameter Efficiency Empirical results agree very well with the predictions of Theorem 1 about the optimal parameter efficiency. On WN18 and WN18RR, $\text{MEI}_{1\times200}$ consistently outperforms $\text{MEI}_{3\times100}$ using fewer parameters. On FB15K and FB15K-237, the model sizes are reversed due to different numbers of entities and relations, with $\text{MEI}_{1\times200}$ having two times more parameters than $\text{MEI}_{3\times100}$. On FB15K, as predicted, $\text{MEI}_{3\times100}$ consistently outperforms $\text{MEI}_{1\times200}$. On FB15K-237, $\text{MEI}_{3\times100}$ outperforms $\text{MEI}_{1\times200}$ most of the time, although not by a large margin,

Table 2. Link prediction results on WN18 and FB15K. † are reported in [18], ‡ are reported in [25], other results are reported in their papers. Best results are in bold, second-best results are underlined.

	WN18				FB15K			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
TransE [4] †	0.495	0.113	0.888	0.943	0.463	0.297	0.578	0.749
ConvE [6]	0.943	0.935	0.946	0.956	0.657	0.558	0.723	0.831
RESCAL [19] †	0.890	0.842	0.904	0.928	0.354	0.235	0.409	0.587
DistMult [29] ‡	0.822	0.728	0.914	0.936	0.654	0.546	0.733	0.824
ComplEx [25]	0.941	0.936	0.945	0.947	0.692	0.599	0.759	0.840
SimplE [11]	0.942	0.939	0.944	0.947	0.727	0.660	0.773	0.838
TorusE [8]	0.947	0.943	0.950	0.954	0.733	0.674	0.771	0.832
ComplEx new tuning [15]	–	–	–	–	0.790	–	–	0.872
$\text{MEI}_{1\times200}$	**0.953**	**0.949**	**0.955**	**0.958**	0.795	0.741	0.833	0.892
$\text{MEI}_{3\times100}$	0.950	0.946	0.952	0.957	**0.806**	**0.754**	**0.843**	**0.893**

Table 3. Link prediction results on WN18RR and FB15K-237. † are reported in [7], ‡ are reported in [6], other results are reported in their papers. Best results are in bold, second-best results are underlined.

	WN18RR				FB15K-237			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
TransE [4] †	0.182	0.027	0.295	0.444	0.257	0.174	0.284	0.420
ConvE [6]	0.43	0.40	0.44	0.52	0.325	0.237	0.356	0.501
DistMult [29] ‡	0.43	0.39	0.44	0.49	0.241	0.155	0.263	0.419
ComplEx [25] ‡	0.44	0.41	0.46	0.51	0.247	0.158	0.275	0.428
TorusE [7]	0.452	0.422	0.464	0.512	0.305	0.217	0.335	0.484
RotatE w/o adv [21]	–	–	–	–	0.297	0.205	0.328	0.480
$\text{MEI}_{1\times200}$	**0.470**	**0.443**	**0.482**	**0.526**	0.358	**0.266**	0.394	**0.544**
$\text{MEI}_{3\times100}$	0.458	0.426	0.470	0.521	**0.359**	**0.266**	**0.395**	**0.544**

but uses only half the number of parameters. These results are particularly interesting because they suggest that when the embedding size D is large enough, MEI with $K > 1$ can both scale to larger embedding sizes and have better results than MEI with $K = 1$ partition.

4.3 Analyses

Parameter Scale Comparison Table 4 compares the performance of MEI with that of ConvE [6], which aims to learn interaction mechanisms by a neural network, at different parameter scales. The results show that MEI achieves better results than ConvE at the same parameter count. Moreover, the small MEI model at 0.95M parameters remarkably outperforms the other model at 1.89M parameters. These results suggest that MEI is an effective framework to utilize the parameters of the model and to learn the interaction mechanisms automatically for knowledge graph embedding.

Parameter Trade-off Analysis There are two kinds of parameters in the MEI model, the embeddings and the core tensors. Theorem 1 provides a guideline to trade-offs between them. For example, on FB15K-237, the parameter efficiency increases when the partition size increases up to $C \approx 122$. However, there are other factors affecting this trade-off, such as the ensemble boosting effect that favors larger K and smaller C. We argue that due to this effect, MEI with $K > 1$ has an empirical advantage compared with MEI with $K = 1$. To evaluate this claim, we analyze the performance of MEI models with approximately the same parameter counts but different core-tensor sizes on FB15K-237. To disambiguate the effects of larger core tensor, we made sure that the models with larger core tensors would have smaller parameter counts. Table 5 shows that the models with larger core tensor consistently achieve better results with even fewer total parameters, agreeing very well with Theorem 1. Interestingly, MEI with $K = 3$ achieves competitive results compared with MEI with $K = 1$, which suggest that the ensemble boosting effect benefits MEI with $K > 1$, as we argued.

Table 4. Parameter scaling on FB15K-237.

Model	Param. count	Emb. size	MRR	H@ 1	3	10
ConvE	1.89M	96	.32	.23	.35	.49
ConvE	0.95M	54	.30	.22	.33	.46
MEI	1.89M	3×40	.34	.25	.38	.53
MEI	0.95M	3×20	.33	.24	.36	.51

Table 5. Parameter trade-off analysis on FB15K-237.

Emb. size	Param. count	**W** size	MRR	H@ 1	3	10
12×11	1.95M	1K	0.335	0.247	0.367	0.514
6×21	1.87M	9K	0.339	0.249	0.371	0.518
3×40	1.84M	64K	**0.344**	0.253	**0.378**	**0.527**
1×82	1.76M	551K	**0.344**	**0.255**	**0.378**	0.522

5 Conclusion and Future Work

In this work, we proposed MEI, the multi-partition embedding interaction model with block term format, to systematically control the trade-off between expressiveness and computational cost, to learn the interaction mechanisms from data automatically, and to achieve state-of-the-art performance on the link prediction task. In addition, we theoretically studied the parameter efficiency problem and derived a simple criterion for optimal parameter trade-off. We discussed several interpretations and insights of MEI as a novel general design pattern for knowledge graph embedding, and we applied the framework of MEI to present a new generalized explanation for several specially designed interaction mechanisms in previous models.

In future work, we plan to conduct more experiments with MEI, especially regarding the ensemble boosting effect and the meta-dimensional transforming–matching framework. Other interesting directions include more in-depth studies of the embedding internal structure and the nature of multi-partition embedding interaction, especially with applications in other domains such as natural language processing, computer vision, and recommender systems.

ACKNOWLEDGEMENTS

This work was supported by the Cross-ministerial Strategic Innovation Promotion Program (SIP) Second Phase, "Big-data and AI-enabled Cyberspace Technologies" by the New Energy and Industrial Technology Development Organization (NEDO).

REFERENCES

[1] Ivana Balažević, Carl Allen, and Timothy M. Hospedales, 'TuckER: Tensor Factorization for Knowledge Graph Completion', in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, (2019).

[2] James Bergstra and Yoshua Bengio, 'Random Search for Hyper-Parameter Optimization', *Journal of Machine Learning Research*, **13**, 281–305, (2012).

[3] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor, 'Freebase: A collaboratively created graph database for structuring human knowledge', in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pp. 1247–1250, (2008).

[4] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko, 'Translating embeddings for modeling multi-relational data', in *Proceedings of the 27th Conference on Neural Information Processing Systems*, (2013).

[5] Lieven De Lathauwer, 'Decompositions of a higher-order tensor in block terms—Part II: Definitions and uniqueness', *SIAM Journal on Matrix Analysis and Applications*, **30**(3), 1033–1066, (2008).

[6] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel, 'Convolutional 2D Knowledge Graph Embeddings', in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, (2018).

[7] T. Ebisu and R. Ichise, 'Generalized Translation-based Embedding of Knowledge Graph', *IEEE Transactions on Knowledge and Data Engineering*, 1–1, (2019).

[8] Takuma Ebisu and Ryutaro Ichise, 'TorusE: Knowledge Graph Embedding on a Lie Group', in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, (2018).

[9] Frank L. Hitchcock, 'The Expression of a Tensor or a Polyadic as a Sum of Products', *Journal of Mathematics and Physics*, **6**(1-4), 164–189, (April 1927).

[10] Sergey Ioffe and Christian Szegedy, 'Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift', in *Proceedings of the 32nd International Conference on Machine Learning*, pp. 448–456, (2015).

[11] Seyed Mehran Kazemi and David Poole, 'SimplE Embedding for Link Prediction in Knowledge Graphs', in *Proceedings of the 32nd Conference on Neural Information Processing Systems*, (February 2018).

[12] Diederik P. Kingma and Jimmy Ba, 'Adam: A Method for Stochastic Optimization', in *International Conference on Learning Representations*, (December 2014).

[13] Tamara G. Kolda and Brett W. Bader, 'Tensor Decompositions and Applications', *SIAM Review*, **51**(3), 455–500, (August 2009).

[14] Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski, 'Canonical Tensor Decomposition for Knowledge Base Completion', in *Proceedings of the 35th International Conference on Machine Learning*, (June 2018).

[15] Adam Lerer, Ledell Wu, Jiajun Shen, Timothee Lacroix, Luca Wehrstedt, Abhijit Bose, and Alex Peysakhovich, 'PyTorch-BigGraph: A Large-scale Graph Embedding System', in *Proceedings of the 2 Nd SysML Conference*, p. 12, (2019).

[16] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, 'Efficient Estimation of Word Representations in Vector Space', in *Workshop Proceedings of the 2013 International Conference on Learning Representations*, (January 2013).

[17] Miller, George A., 'WordNet: A lexical database for English', *Communications of the ACM*, 39–41, (1995).

[18] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio, 'Holographic Embeddings of Knowledge Graphs', in *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, (2016).

[19] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel, 'A Three-Way Model for Collective Learning on Multi-Relational Data', in *Proceedings of the 28th International Conference on Machine Learning*, pp. 809–816, (2011).

[20] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, 'Dropout: A simple way to prevent neural networks from overfitting', *The Journal of Machine Learning Research*, **15**(1), 1929–1958, (2014).

[21] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang, 'RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space', in *International Conference on Learning Representations*, (2019).

[22] Kristina Toutanova and Danqi 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*, pp. 57–66, Beijing, China, (July 2015). Association for Computational Linguistics.

[23] Hung Nghiep Tran and Atsuhiro Takasu, 'Analyzing Knowledge Graph Embedding Methods from a Multi-Embedding Interaction Perspective', in *Proceedings of the Data Science for Industry 4.0 Workshop at EDBT/ICDT*, (2019).

[24] Hung Nghiep Tran and Atsuhiro Takasu, 'Exploring Scholarly Data by Semantic Query on Knowledge Graph Embedding Space', in *Proceedings of the 23rd International Conference on Theory and Practice of Digital Libraries*, pp. 154–162, (2019).

[25] Theo Trouillon, Johannes Welbl, Sebastian Riedel, Eric Gaussier, and Guillaume Bouchard, 'Complex Embeddings for Simple Link Prediction', in *Proceedings of the 33rd International Conference on Machine Learning*, (2016).

[26] Ledyard R Tucker, 'Some mathematical notes on three-mode factor analysis', *Psychometrika*, **31**(3), 279–311, (September 1966).

[27] Denny Vrandečić and Markus Krötzsch, 'Wikidata: A free collaborative knowledgebase', *Communications of the ACM*, **57**(10), 78–85, (September 2014).

[28] Q. Wang, Z. Mao, B. Wang, and L. Guo, 'Knowledge Graph Embedding: A Survey of Approaches and Applications', *IEEE Transactions on Knowledge and Data Engineering*, **29**(12), (December 2017).

[29] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng, 'Embedding Entities and Relations for Learning and Inference in Knowledge Bases', in *International Conference on Learning Representations*, (2015).

[30] Shuai Zhang, Yi Tay, Lina Yao, and Qi Liu, 'Quaternion Knowledge Graph Embedding', in *Advances in Neural Information Processing Systems*, (2019).