Towards Efficient and Expressive Knowledge Graph Embedding for Link Prediction

Hung-Nghiep TRAN† and Atsuhiro TAKASU†

† National Institute of Informatics
Hitotsubashi, Chiyoda-ku, Tokyo 101–8430, Japan
E-mail: †{nghiepth,takasu}@nii.ac.jp

Abstract In this paper, we overview the emerging field of knowledge graph embedding and present our recent results in efficient and expressive knowledge graph embedding methods for link prediction. Knowledge graph embedding methods aim to perform link prediction by learning the embeddings of entities and relations. 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. We proposed 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 model the interactions at the internal-structure level of the embeddings, enabling MEI to control the trade-off between computational cost and expressiveness, learn the interaction mechanisms from data automatically, and achieve state-of-the-art performance on the link prediction task. Our framework of MEI also provides a new generalized explanation for several specially designed interaction mechanisms in previous models.

Key words machine learning, neural networks, block term tensor format, multi-relational embedding, knowledge graph embedding, link prediction.

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 [2] 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 [19].

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 [24], 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 [14] 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 [24] and recent state-of-the-art models HolE [15], ComplEx [21], and SimplE [10]. However, these interaction mechanisms are specifically designed and fixed, which may pose questions about optimality or extensibility on a specific knowledge graph.

As a different approach to this problem, we motivated [18] and proposed [20] the multi-partition embedding interaction model. Here we present a comprehensive overview of this approach and include some new experimental results. In this approach, 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 [22] 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 [4] 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.

In general, the main points of this paper include the following.

- We present the multi-partition embedding interaction approach to knowledge graph embedding, which models the internal structure of the embeddings to trade-off between efficiency and expressiveness.
- In this approach, we present 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 several previous models.
- We empirically show that MEI can achieve state-of-the-art results on link prediction using popular benchmarks.

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 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 [18].

a) Semantic Matching Models

RESCAL [14] 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 [22]. 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 [24] 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 $\boldsymbol{M}_r = \mathrm{diag}(\boldsymbol{r})$. 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 [15] uses a circular correlation between the head and tail embedding vectors; ComplEx [21] 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 [10] represents each entity as two role-based embedding vectors and augments an inverse relation embedding vector. In our previous work [18], 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 [25]. 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 [8], Tucker format [22], and block term format [4]. 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 [11]. We analyze the connections to the related tensor decomposition methods in Section 3 2.

b) 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 [5] can achieve good results by sharing the convolution weights. However, they are restricted by the input format to the neural network [5], and the operations are generally less expressive than direct interactions between the entries of the embedding vectors [15]. We will empirically compare with them.

c) Translation-based Models

The main advantages of these models are their simple and intuitive mechanism with the relation embeddings as the translation vectors [3]. However, it has been shown that they have limitations in expressiveness [10]. The recent model TorusE [6] 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 [22], which is the most general linear model that computes the weighted sum of all entry product combinations in the interacting partitions. The block term format [4] 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* [11], 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.

a) Tucker Format and Block Term Format

We choose to model the local interaction at each partition by the *Tucker format* [22] of third-order tensor

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

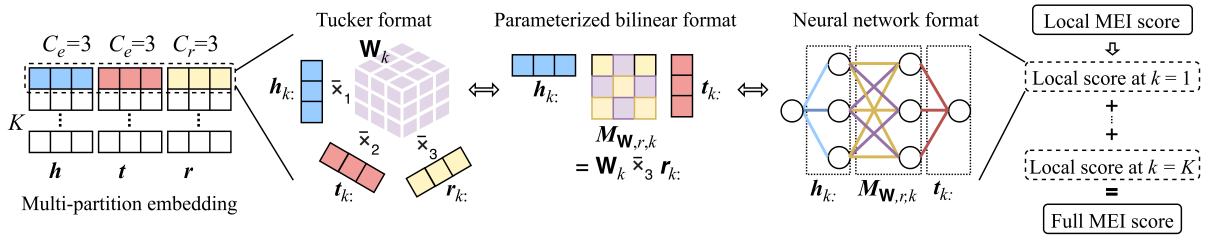

Figure 1 MEI architecture: multi-partition embedding vectors that interact only between the corresponding partitions.

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* [4]. 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.

b) 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.

c) 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 [16] and batch normalization [9] 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.

a) 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

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

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

as a bilinear model

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

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 $\mathbf{W}_1, \dots \mathbf{W}_K$ 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.

b) 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.

c) 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 [10] explaining them using a bilinear model, [1] using a vanilla Tucker model, and [18] 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 [8] 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 [18].

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 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 [12], which replaces the h or t entities in each existent triple (h,t,r) with other random entities to obtain

Table 1 Datasets statistics.

Dataset	$\|\mathcal{E}\|$	$\|\mathcal{R}\|$	Train	Valid	Test
WN18RR	40,943	11	86,835	3,034	3,134
FB15K-237	14,541	237	272,115	17,535	20,466

the locally related nonexistent triples (h', t, r) and (h, t', r) [3]. 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

a) Datasets

We use two popular benchmark datasets for link prediction, as shown in Table 1. WN18RR [5] is a subset of WordNet [13], which contains lexical relationships between words. FB15K-237 [17] are subsets of Freebase [2], which contains general facts.

b) Evaluations

We evaluate and analyze MEI on the link prediction task [3]. 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) [21]. 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 [3]. 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.

c) Baselines

We 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.

d) Implementations

We trained MEI using mini-batch stochastic gradient descent with Adam optimizer. We followed the 1-N scoring procedure in [5] 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]. All hyperparameters of $\text{MEI}_{3\times100}$ are tuned by random search, including batch size, learning rate, decay rate, batch normalization, and dropout rates. There are two variants of the $\text{MEI}_{3\times100}$ model, *shared core* using the same core tensor for all K partitions as an analogy to single interaction patterns in previous specially designed models and *non-shared core* using different core tensors for different partitions.

4 2 Results

a) Link Prediction Performance

Table 2 shows the main results. In general, MEI strongly outperforms the baselines. MEI and ConvE both aim to learn the interaction between the embedding vectors, however, 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 and ComplEx, 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 MEI can still outperform them on comparable settings.

b) Parameter Trade-off Analysis

There are two kinds of parameters in the MEI model, the embeddings and the core tensors, which can be traded off by changing the number of partitions K. There are several factors affecting this trade-off, such as the high expressiveness favoring the larger partition size C, the low computational cost favoring the smaller partition size C, the ensemble boosting effect favoring larger K and smaller C. We argue that due to these effects, MEI with $K > 1$ can have empirical advantages compared with MEI with $K = 1$. Table 2 shows that on FB15K-237, MEI with three partitions outperforms MEI with one partition. To further evaluate our claim, we analyze the performance of MEI models with approximately the same parameter counts but different core-tensor sizes on

Table 2 Link prediction results on WN18RR and FB15K-237. † are reported in [7], ⸸ are reported in [23], ‡ are reported in [5], 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 [3] †	0.182	0.027	0.295	0.444	0.257	0.174	0.284	0.420
ConvE [5]	0.43	0.40	0.44	0.52	0.325	0.237	0.356	0.501
RESCAL [14] ⸸	0.420	–	–	0.447	0.270	–	–	0.427
DistMult [24] ‡	0.43	0.39	0.44	0.49	0.241	0.155	0.263	0.419
ComplEx [21] ‡	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
$\text{MEI}_{1\times200}$	**0.470**	**0.443**	**0.482**	**0.526**	0.358	0.266	0.394	0.544
$\text{MEI}_{3\times100,\text{ shared core}}$	0.458	0.426	0.470	0.521	0.359	0.266	0.395	0.544
$\text{MEI}_{3\times100,\text{ non-shared core}}$	0.457	0.422	0.470	0.521	**0.361**	**0.269**	**0.397**	**0.545**

FB15K-237. To disambiguate the effects of larger core tensor, we made sure that the models with larger core tensors would have equal or smaller parameter counts. Table 3 shows that the models with larger core tensor consistently achieve better results with even fewer total parameters due to higher expressiveness. More 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.

5 Conclusion and Future Work

In this paper, we presented an overview of MEI, the multi-partition embedding interaction model with block term format [20], as an approach towards efficient and expressive knowledge graph embedding. In addition, 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 provide 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.

Acknowledgments 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).

Table 3 Parameter trade-off analysis on FB15K-237.

Emb. size	Param. count	**W** size	MRR	H@ 1	H@ 3	H@ 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

References

[1] I. Balažević, C. Allen, and T. M. Hospedales. TuckER: Tensor Factorization for Knowledge Graph Completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, pages 5185–5194, 2019.

[2] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250, 2008.

[3] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating Embeddings for Modeling Multi-Relational Data. In *Advances in Neural Information Processing Systems*, pages 2787–2795, 2013.

[4] L. 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.

[5] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel. Convolutional 2D Knowledge Graph Embeddings. In

Proceedings of the 32nd AAAI Conference on Artificial Intelligence, pages 1811–1818, 2018.

[6] T. Ebisu and R. Ichise. TorusE: Knowledge Graph Embedding on a Lie Group. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 1819–1826, 2018.

[7] T. Ebisu and R. Ichise. Generalized Translation-based Embedding of Knowledge Graph. *IEEE Transactions on Knowledge and Data Engineering*, 32(5):941–951, 2019.

[8] F. 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, 1927.

[9] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 448–456, 2015.

[10] S. M. Kazemi and D. Poole. SimplE Embedding for Link Prediction in Knowledge Graphs. In *Proceedings of the 32nd Conference on Neural Information Processing Systems*, pages 4289–4300, 2018.

[11] T. G. Kolda and B. W. Bader. Tensor Decompositions and Applications. *SIAM Review*, 51(3):455–500, 2009.

[12] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. In *Workshop Proceedings of the 2013 International Conference on Learning Representations*, page 12, 2013.

[13] Miller, George A. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[14] M. Nickel, V. Tresp, and H.-P. Kriegel. A Three-Way Model for Collective Learning on Multi-Relational Data. In *Proceedings of the 28th International Conference on Machine Learning*, pages 809–816, 2011.

[15] M. Nickel, L. Rosasco, and T. Poggio. Holographic Embeddings of Knowledge Graphs. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 1955–1961, 2016.

[16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[17] K. Toutanova and D. Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and Their Compositionality*, pages 57–66, 2015.

[18] H. N. Tran and A. 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*, page 7, 2019.

[19] H. N. Tran and A. 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*, pages 154–162, 2019.

[20] H. N. Tran and A. Takasu. Multi-Partition Embedding Interaction with Block Term Format for Knowledge Graph Completion. In *Proceedings of the European Conference on Artificial Intelligence*, pages 833–840, 2020.

[21] T. Trouillon, J. Welbl, S. Riedel, Eric Gaussier, and Guillaume Bouchard. Complex Embeddings for Simple Link Prediction. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 2071–2080, 2016.

[22] L. R. Tucker. Some Mathematical Notes on Three-Mode Factor Analysis. *Psychometrika*, 31(3):279–311, 1966.

[23] Y. Wang, D. Ruffinelli, R. Gemulla, S. Broscheit, and C. Meilicke. On Evaluating Embedding Models for Knowledge Base Completion. *arXiv:1810.07180 [cs, stat]*, Oct. 2018.

[24] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *Proceedings of the International Conference on Learning Representations*, page 12, 2015.

[25] S. Zhang, Y. Tay, L. Yao, and Q. Liu. Quaternion Knowledge Graph Embedding. In *Advances in Neural Information Processing Systems*, pages 2735–2745, 2019.