Multi-Relational Embedding for Knowledge Graph Representation and Analysis

by

Hung Nghiep Tran

Dissertation

submitted to the Department of Informatics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

The Graduate University for Advanced Studies, SOKENDAI
September 2020

Committee

Advisor	Dr. Atsuhiro TAKASU Professor of National Institute of Informatics/SOKENDAI
Subadvisor	Dr. Akiko AIZAWA Professor of National Institute of Informatics/SOKENDAI
Subadvisor	Dr. Kenro AIHARA Professor of National Institute of Informatics/SOKENDAI
Examiner	Dr. Seiji YAMADA Professor of National Institute of Informatics/SOKENDAI
Examiner	Dr. Yusuke MIYAO Professor of The University of Tokyo

Abstract

Multi-relational data, such as knowledge graphs, bibliographic data, and information networks are prevalent in real-world datasets. Managing, exploring, and utilizing these large and complex datasets effectively are challenging. In recent years, multi-relational embedding methods have emerged as a new effective approach to model multi-relational data by representing both the entities and the relations as embedding vectors in semantic space. On knowledge graphs, multi-relational embedding methods aim to model the interactions between these embedding vectors to predict the relational link between entities. These knowledge graph embedding methods solve the important inherent task of link prediction for knowledge graph completion, but also provide the embedding representations that have various potential applications. The goal of this thesis is first to study multi-relational embedding on knowledge graphs to propose a new embedding model that explains and improves previous methods, then to study the applications of multi-relational embedding in representation and analysis of knowledge graphs.

For the first part of the thesis, we study the theoretical framework of knowledge graph embedding methods to explain and improve them. We review and analyze the popular class of semantic matching knowledge graph embedding methods, with a focus on the state-of-the-art trilinear-product-based models such as ComplEx. Based on our analysis, we identify two fundamental complementary aspects that a knowledge graph embedding model needs to address, that is, computational efficiency and model expressiveness. Previous trilinear-product-based models use specially designed interaction mechanisms to manually provide a trade-off between the two aspects. However, their interaction mechanisms are specially designed and fixed, potentially causing them to be suboptimal or difficult to extend. In this thesis, 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. The model combines advanced tensor representation formats and modern

deep learning techniques to achieve state-of-the-art performance on the link prediction task. The theoretical framework of the MEI model is then used as a general mechanism of knowledge graph embedding to analyze, explain, and generalize previous models. We also draw the connections to word embeddings and language modeling to provide some new insights and generalizations.

For the second part of the thesis, we study how to apply multi-relational embedding in representation and analysis of knowledge graphs. Unlike word embedding, the semantic structures such as similarity and analogy structures in knowledge graph embedding space are not well-studied, and thus not usually utilized for data representation and analysis. To demonstrate the application of multi-relational embedding, we formalize a framework for data representation and analysis by semantic queries on the multi-relational embedding space. We build a knowledge graph from scholarly data and show how various tasks on the original datasets can be approximated by appropriate semantic queries, which are multi-linear algebraic operations on the multi-relational embedding spaces. We also theoretically study the entity analogy reasoning task in multi-relational embedding space, which can be formulated as an open-relational query by examples task, doing relational query on unseen relations. Using the above mathematical connections between knowledge graph embeddings and word embeddings, we analyze the semantic structures in the knowledge graph embedding space and propose potential solution to the above entity analogy reasoning task. The goal of this endeavor is to explore potential applications of recent advancements in multi-relational embedding to data representation and analysis, especially to improve its effectiveness on scholarly data.

Acknowledgements

First, I would like to thank my advisor Professor Atsuhiro Takasu for his research guidance that have helped me navigate the long and challenging Ph.D. road, and for his constant supports that have given me the opportunity to stay focused and push my research forward. I also would like to thank all professors in my defense committee for their kind and valuable feedback that has helped me improve my research in many aspects, including Professor Akiko Aizawa and Professor Kenro Aihara who are also my subadvisors, and Professor Seiji Yamada and Professor Yusuke Miyao.

Second, I would like to thank people who have supported me during my Ph.D. study. I want to thank Takenaka-san, our lab secretary and also a friend, for her supports and accompanies in years. I want to thank NII and SOKENDAI staffs, NII Graduate Office, NII Library, and all other people who work behind the scene to keep the system run smoothly. I also want to thank Japan Government and MEXT Scholarship for financial and official supports so that I could come to study in Japan. My Ph.D. study would not have been filled with as good experiences without you.

Third, I would like to thank my friends, lab mates, and colleagues in NII, SOKENDAI for the time and experiences we had together. Especially I thank those who have helped me in the last stage of my study program, especially Van, my lab mate and also a friend, for her accompanies and supports in improving my presentation. I also want to thank the Vietnamese community in NII that have given me valuable advices on various aspects of life and research. I also want to thank my past teachers and mentors in Vietnam and other countries. Also thank everyone else for leaving me 10000 hours all to myself to focus on research and acquire the expertise.

I want to thank the books that I have read and learned most important things, the internet that has given me access to most resource I need, the musics that have kept me sane and joyful, and God that has kept me calm in some of the most depressing time.

Finally, I am grateful for my family, Mom and Dad, Sisters and Brothers, and others, together with my hope in science, for giving me reasons to try my best in all these years, belief, duty, honor, and courage.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Multi-relational data store information about the entities and multiple relationships between them, for example, knowledge graphs, bibliographic networks, and information networks. Large multi-relational data are prevalent in real-world datasets, such as knowledge graphs, including WordNet [58] representing English lexical knowledge, and Freebase [7] and Wikidata [88] representing general knowledge; and bibliographic datasets, including MAG[1] [70] and CORE[2] [45], which include millions of papers, authors, venues, and the relationships between them.

The knowledge graph is a popular and standard data representation format for multi-relational data. The main part of a knowledge graph is 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 [77]. This can also be formalized as a labeled directed multigraph where each triple (h, t, r) represents a directed edge from node h to node t with label r. Given this general format, it is straightforward to represent other multi-relational data in the knowledge graph standard, such as *(AuthorA, Paper1, write)* and *(Paper1, Paper2, cite)* triples in scholarly data. Knowledge graphs have become one of the cornerstones of modern semantic web technology. They have been used by large companies such as Google to provide semantic meanings into many traditional applications, such as semantic search engines, semantic browsing, and question answering [69]. Recently, knowledge graphs have also found applications in recommender systems, where they are used to integrate multiple sources of data and incorporate external knowledge [11] [96]. However, managing, exploring, and utilizing these large and complex datasets effectively are a challenging task.

In recent years, multi-relational embedding methods have emerged as a new effective

[1] Microsoft Academic Graph: `https://academic.microsoft.com/`
[2] Open access publications: `https://core.ac.uk/`

approach to model multi-relational data by representing both the entities and the relations as embedding vectors in semantic space. On knowledge graphs, multi-relational embedding methods aim to model the interactions between these embedding vectors to compute a score that predicts the existence of each triple. These knowledge graph embedding methods solve the important inherent task of link prediction for knowledge graph completion, but also provide the embeddings as a useful representation of the whole knowledge graph that may enable various potential applications of knowledge graphs in artificial intelligence tasks [78].

The goal of this thesis is first to study multi-relational embedding on knowledge graphs to propose a new embedding model that explains and improves previous methods, then to study the applications of multi-relational embedding, especially the knowledge graph embedding space, in representation and analysis of knowledge graphs.

1.1 Research problems and approaches

In this thesis, we address two main problems. The first problem is about the theoretical framework of knowledge graph embedding methods to explain, generalize, and improve them. The second problem is about studying the semantic structures in knowledge graph embedding space and its applications in representation and analysis of knowledge graphs.

1.1.1 Generalizing and improving knowledge graph embedding methods

Most previous works treat embedding as a whole and model the interaction between the whole embeddings. For example, the bilinear model RESCAL [60] and the recent model TuckER [3] can model very general interactions between every entry of the embeddings, but they cannot scale to large embedding size. One approach to this problem is to design special interaction mechanisms to restrict the interactions between only a few entries, for example, DistMult [95] and recent state-of-the-art models HolE [61], ComplEx [80], and SimplE [41] [47]. However, these interaction mechanisms are specifically designed and fixed, which may pose questions about optimality or extensibility on a specific knowledge graph.

In this thesis, 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 [82] 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 [14] in tensor calculus. The result is a multi-partition embedding interaction (MEI) model with block term format that combines advanced tensor representation formats and modern deep learning techniques to provide 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.

The framework of the MEI model is then used as a general mechanism to analyze, explain, and generalize previous knowledge graph embedding models. We also draw some connections to word embedding methods and provide some insights and new explanations to some intriguing phenomena in word embedding and network embedding.

1.1.2 Exploring and utilizing multi-relational embedding space

In the case of word embedding methods such as word2vec, embedding vectors are known to contain rich semantic information that enables them to be used in many semantic applications [57]. However, knowledge graph embedding vectors are usually only used for the inherent task of knowledge graph completion, but not for semantic applications. One of the reasons is that the semantic structures in knowledge graph embedding is not well-understood because of the vast diversity of the interaction mechanisms in knowledge graph embedding methods.

In this thesis, we first try to formalize a general framework for multi-relational data exploration and analysis using semantic queries on knowledge graph embedding space. The main component in this framework is the conversion templates from data exploration and analysis tasks on the original data to *semantic queries*, which are the multi-linear algebraic operations between the embedding vectors, that exploits the semantic structures of the embedding space to solve queries such as similarity query and relational query. We then build a scholarly knowledge graph and demonstrate how some important representation and analysis tasks on the original data can be efficiently approximated by semantic queries.

We also review the entity analogy reasoning on multi-relational embedding space task, which can be seen as an *open-relational query* by examples task. Towards solving this task, we study the semantic structures in the knowledge graph embedding space based on the connections between knowledge graph embedding methods and language modeling, in

particular, between CP_h [47] and word2vec skipgram [56]. We propose a generalized linear structure that extends the simple semantic direction structure in word2vec embedding space, namely $king - man = queen - woman$, to multi-relational embedding space. We then outline a potential solution to the above task.

1.2 Research contributions

The main contributions of this thesis are described as follows.

Theoretical aspects

- 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 and empirically show that MEI is efficient and can achieve state-of-the-art results using the popular and standard link prediction benchmarks.

- We theoretically analyze the framework of MEI to explain its intuitions and meanings. In addition, we are the first to formally study the parameter efficiency problem and derive a simple optimal trade-off criterion for MEI. We apply the theoretical framework of MEI to provide new intuitive explanations for the specially designed interaction mechanisms in several previous knowledge graph embedding models and show the advantages of MEI over previous models.

- We also draw the connections to word embeddings and language modeling to provide new insights and generalizations for both knowledge graph embedding and word embedding. We propose to view knowledge graph embedding from a language modeling perspective and vice versa, view language modeling from a knowledge graph embedding perspective. These connections enable us to provide new explanations to some intriguing phenomena in word embedding and network embedding.

Practical aspects

- We formalize a general framework for multi-relational data exploration and analysis using semantic queries on knowledge graph embedding space. The main component

in this framework is the conversion templates from data exploration and analysis tasks on the original data to *semantic queries*, which are the multi-linear algebraic operations between the embedding vectors on the embedding space.

- We build a knowledge graph from scholarly data and demonstrate how some important representation and analysis tasks on the original data can be solved by semantic queries using the formalized framework.

- We also review the entity analogy reasoning on multi-relational embedding space task, which can be seen as an *open-relational query* by examples task. Towards solving this task, we study the semantic structures in the knowledge graph embedding space, propose a general semantic analogy structure in multi-relational embedding space, then outline a potential solution to the above task.

1.3 Thesis organization

The remaining chapters in this thesis are organized as follows:

- In Chapter 2, we introduce the background knowledge of our studies, including the notations, general concepts and definitions that we use in this thesis. We first present the multi-relational data as a general data format and review related data formats in real-world datasets. We then summarize embedding methods and review tensor representation formats.

- In Chapter 3, we review the literature that is most relevant to our research. We first survey various embedding methods including text embedding, network embedding, and knowledge graph embedding for multi-relational data. In particular, we focus on the class of trilinear-product-based models, which includes the state-of-the-art models that are important objects of our study. We then review popular multi-relational datasets including knowledge graphs and scholarly datasets. We also review the applications of knowledge graph embedding methods in multi-relational data representation and reasoning.

- In Chapter 4, we present the main theoretical contributions of the thesis, namely the multi-partition embedding interaction model. This proposed model and the accompanying concepts provide a new perspective and general mechanism for knowledge graph embedding. We start with introducing the new concepts and perspectives of multi-partition embeddings and local partition interactions. We then propose the

multi-partition embedding interaction (MEI) model to generalize previous models and systematically address their drawbacks. We also show how MEI provides concrete advantages over previous embedding models, generalizes and explains their mechanisms, as well as explains some surprising phenomena in word embeddings.

- In Chapter 5, we evaluate the proposed knowledge graph embedding model using popular benchmarks. We first discuss the learning problem of the model. We then evaluate the performance of the model on the link prediction task, which is the standard benchmark of knowledge graph embedding method and can be seen as a simple data query task with a single predicate.

- In Chapter 6, we study and present various practical applications of knowledge graph embeddings, towards efficient multi-relational data analysis using semantic queries on knowledge graph embedding space. We first formalize a framework for data visualization, browsing, and querying applications using semantic queries, which are multi-linear algebraic operations on the embedding space and demonstrate some tasks on scholarly data. We also review the entity analogy reasoning in multi-relational embedding space task, study the semantic structures in the knowledge graph embedding space, and outline potential solution to the above task.

- Finally, in Chapter 7, we summarize the main results of in this thesis, discuss some important insights and lessons learned from our research, and list some open questions for future work.

Chapter 2

Background

In this chapter, we introduce the background knowledge of our studies, including the notations, general concepts and definitions that we use in this thesis. We first present the multi-relational data as a general data format and review related data formats in real-world datasets. We then summarize embedding methods and review tensor representation formats.

2.1 Notation

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 $\boldsymbol{M}$, and tensors by bold upper case sans serif such as T unless otherwise specified. A triple is denoted as a tuple of three scalars such as (h, t, r) corresponding to the indices of the head entity, tail entity, and relation. We also denote the collection of triples in a knowledge graph as $\mathcal{D}$, the set of all entities as $\mathcal{E}$, and the set of all relations as $\mathcal{R}$.

2.2 Multi-Relational Data

In this thesis, multi-relational data refer to the data about the entities and relationships between them, where there could be multiple relationships between two entities. Multi-relational data are a general data format that can represent various types of data. Therefore, multi-relational data are prevalent in real-world datasets, such as knowledge graphs, text corpora, and information networks. In the following, we review some popular data formats and show that they are special cases of multi-relational data.

2.2.1 Knowledge graph

In the context of semantic web, multi-relational data are knowledge bases or knowledge graphs in the Resource Description Framework (RDF) format [20]. Knowledge graph has recently become a standard multi-relational data format for representing knowledge about the relationships between entities [69].

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 $\mathcal{D}$ [79].

In addition, each entity and relation could have attributes and accompanying text. The attributes can also be represented as triple, where the attribute names become relation types and the attribute values become entities. For example, *I am 30 years old* can be represented as *(I, age, 30)*. Therefore, knowledge graph is a general format and can represent many types of data.

2.2.2 Textual data

Text corpora are a very popular data format that represents sequences of words. This format is fundamentally different from graph format and cannot be directly matched one-to-one to multi-relational data. However, if we allow information loss, we can capture some crucial information in textual data by defining an appropriate graph structure over the sequences of words. The general idea is that, each unique word corresponds to an entity and the absolute or relative positions of words in the text corpora can be encoded as relationships between them. For example, we can define the relation *in-context* to capture information about the co-appearance of words in a context. More extensively, we can define relations *same-sentence*, *appear-before*, and *appear-after* to capture more information about relative positions between words.

Theoretically speaking, when the number of defined relation types approaches infinity, we can capture all information in the text corpora and represent them as multi-relational data. Although this scenario is theoretical and unrealistic, defining an single implicit relation *in-context* [78] has been the primary tool that make neural word embedding methods work and achieve great success such as in word2vec [57]. Thus, it is possible and beneficial to model textual data as multi-relational data.

2.2.3 Network data

Network data refer to information networks such as co-author network, paper citation network, and drug-protein interaction network. These networks are usually non-labeled simple graphs with no loops and no parallel edges, which make them a direct special case of multi-relation data. When multiple related networks are merged together, there could be induced edge labels, loops, and parallel edges between nodes.

2.3 Multi-relational embedding methods

Embedding methods aim to learn the representations of data in low dimensional vector space. In this thesis, we consider the embedding methods on multi-relational data, specifically knowledge graph. We will then show that the embedding methods on other data formats can be reproduced as special cases of knowledge graph embedding methods.

2.3.1 Knowledge graph embedding

Link prediction in knowledge graphs

Real-world knowledge graphs are usually incomplete. Knowledge graph completion is the task that aims to predict new triples given the existing triples in the knowledge graph. The task of link prediction in knowledge graphs can refer to predicting all missing relational links between head and tail entities, that is, predicting all possible r given $(h, t, *)$. However, in practice, it usually refers to predicting all possible tail entities t given $(h, *, r)$ and all possible head entities h given $(*, t, r)$.

Knowledge graph embedding models

Knowledge graph embedding models usually take a triple of the form (h, t, r) as input and predict the existence of that triple. They represent entities and relations as embedding vectors in low dimensional spaces, then model the interactions between these embedding vectors to compute matching scores $\mathcal{S}(h, t, r)$. They usually model the existence of each triple by a Bernoulli distribution and use the standard logistic function $\sigma(\cdot)$ to compute its existence probability as:

$$P(1|h, t, r) = \sigma(\mathcal{S}(h, t, r)). \tag{2.1}$$

Note that the triple (h, t, r) is ordered, that is (h, t, r) is different from (t, h, r). Therefore, the score $\mathcal{S}(h, t, r)$ can be different from $\mathcal{S}(t, h, r)$, and thus their existence probabilities can be different.

Knowledge graph embedding models have the following general three-component architecture [77].

1. *Embedding lookup:* linear mapping from the input sparse high-dimensional discrete *one-hot vectors* to the dense low-dimensional continuous *embedding vectors*. A one-hot vector is a sparse discrete vector representing a discrete input, e.g., the first entity can be represented as $[1, 0, \ldots, 0]^\top$. A triple can then be represented as a tuple of three one-hot vectors corresponding to h, t, and r, respectively. An embedding vector is a dense low-dimensional continuous vector, which enables efficient distributed representations [32] [33].

2. *Interaction mechanism:* modeling the interactions between embedding vectors to compute the matching score of a triple. This is the key component of a knowledge graph embedding model and differentiates between different models.

3. *Link prediction:* using the matching score to predict the existence of each triple. Usually a higher score means that the triple is more likely to be existent. In training, the prediction results are compared with the true data to optimize the embedding vectors and the interaction mechanisms.

Fully expressiveness

Knowledge graph embedding is a very active research topic and thus there are many competing models. Traditionally, a desired property of a model is *fully expressiveness*, that is, there exists a model configuration that can represent any ground truth dataset. Here we formally define the fully expressiveness property.

Definition 2.1. ***(Knowledge graph)*** A knowledge graph $\mathcal{D}$ is a set of true triples.

Definition 2.2. ***(Data tensor)*** A data tensor is a representation of a knowledge graph $\mathcal{D}$ by a third-order binary tensor $\mathbf{G} \in \{0, 1\}^{|\mathcal{E}|\times|\mathcal{E}|\times|\mathcal{R}|}$, where $g_{htr} = 1$ if and only if (h, t, r) exists in $\mathcal{D}$ and 0 otherwise.

Definition 2.3. ***(Ground truth)*** A ground truth $\mathcal{G}$ over an entity set $\mathcal{E}$ and a relation set $\mathcal{R}$ is a full assignment of truth values to all triples that can be composed from these entities and relations.

Definition 2.4. ***(Positive ground truth, negative ground truth, non-positive ground truth, non-negative ground truth)*** A positive ground truth is a ground truth with all triples evaluated as positive. A negative ground truth is a ground truth with all triples evaluated as negative. A non-positive ground truth is a ground truth with at least one triple evaluated as negative. A non-negative ground truth is a ground truth with at least one triple evaluated as positive.

Definition 2.5. ***(Fully expressiveness)*** A model is fully expressive if and only if for any ground truth, there exists an assignment of values to the model's parameters such that the scores of all true triples are either strictly larger or strictly smaller than the scores of all false triples. When a model is shown to be fully expressive with respect to a ground truth, we say that the model accurately represents that ground truth.

Fully expressiveness is an important property of knowledge graph embedding models because if a model is not fully expressive, it cannot model some particular patterns in a knowledge graph, and thus, is inherently flawed. However, as we will discuss in Sect. 4.3.2, in practice, we care more about the model's ability to efficiently capture complex patterns in a knowledge graph, instead of the parameter upper bound for fully expressive.

2.3.2 Text embedding

Text embedding models, particularly word embedding models, are usually language models that take a context C, such as a sequence of context-words $c_1, \ldots, c_m$, as input and predict the target-word w [4], that is, predict the probability:

$$P(w|c_1, \ldots, c_m). \tag{2.2}$$

Context-words are usually defined by using a sliding window through the sequence of words. Target-words are usually either the next word or the center word in the sliding window. Some efficient models such as word2vec skipgram predict multiple context-words given the target-word instead [56].

More generally, text embedding models may take as input sequences of words, that is, documents. Each input word could be represented by a one-hot vector $[1, 0, \ldots, 0]^\top$, thus input are sequences of one-hot vectors. The output usually depends on specific tasks, such as predicting the next word in language modeling [39], predicting target-word in word2vec [57], and predicting the next sentence in skip thought [44].

Note that for the above text embedding models, they can be seen as special cases of knowledge graph embedding with a single implicit relation for each type of model. For

example, triples become pairs of words in word2vec, or pairs of sentences in skip thought, or tuple of document and word in doc2vec. This connection is mathematically analyzed in more details in Section 4.5.1.

2.3.3 Network embedding

Network embedding was originally developed based on text embedding, with DeepWalk [63] using random walks on the network to sample sequences of nodes, which are then used in the same manner as sequences of words in word2vec to learn the node embeddings.

Several later methods improved this approach by either changing the sequence sampling strategy, or restrict the sampling neighborhood. Some other methods use deep models like convolutional neural networks on the graph. They all share the same problem with text embedding, i.e., they do not model explicit relations, thus they cannot capture different semantic meanings of nodes in different relational contexts.

These models take as input a network, usually represented by an adjacency matrix or adjacency list denoting edges between pairs of nodes. Note that when data is a heterogeneous graph, meaning there are many edge types, it is usually represented by a collection of adjacency matrix or adjacency list, each one for each edge type instead of using edge labels as in knowledge graph. The output is usually predicting neighbor nodes. Note that there is no edge embedding, because there is only one edge type implicitly. Even for heterogeneous graph, they do not fully utilize different edge types and do not learn different edge embeddings.

2.4 Tensor

In this section, we review the mathematical background about tensor. Tensor is a generalization of vector and matrix to arbitrary orders [46]. A vector is a special case of first-order tensor with one lower index. A matrix is a special case of second-order tensor with one lower and one upper indices. A tensor can be viewed as a multi-dimensional array, where each entry is specified by multi-indices. For example, $t_{i,j,k}$ is the scalar entry at "row" i, "column" j, and "tube" k of the third-order tensor $\mathbf{T}$. In this thesis, we concern the finite-dimensional tensor.

2.4.1 Tensor product

The tensor product of two tensors is a new tensor whose entries are the products of every pair of entries from the two input tensors. Because of multiplying every pair of entries, the tensor product of two finite-dimensional tensors has dimension equal to the product of the dimensions of the two input tensors. We denote the tensor product as $\times$.

2.4.2 Tensor contraction

Tensor contraction is a generalization of trace in the case of matrix to the case of tensor. Tensor contraction refers to the process of summing over a pair of same indices, thus reduces the order of a tensor by 2. Tensor contraction can be applied to any tensor or tensor product, with the duplicate indices be summed over in the Einstein notation.

2.4.3 The n-mode tensor product with a matrix

The n-mode tensor product of a tensor with a matrix is a new tensor with the same order but different dimension to the input tensor. The mode n entries of the resulting tensor are computed by dot product of each mode n tube of the input tensor and each column of the input matrix. We denote the n-mode tensor product as $\times_n$.

2.4.4 The n-mode tensor product with a vector

The n-mode tensor product of a tensor with a vector is similar to the n-mode tensor product with a matrix; however, the mode n with dimensionality 1 in the resulting tensor is contracted, and thus reduces the order of the resulting tensor by 1. We denote the n-mode tensor product with a vector as $\bar{\times}_n$.

2.5 Tensor representation formats

There are different ways to write the tensor as product of other tensors. These are called tensor representation formats. We will review some most important and relevant formats.

2.5.1 CP format

CP format is the common name for CANDECOMP, PARAFAC, or the *tensor rank format*. This is the most popular and simplest format, which was independently rediscovered many times under different names [46]. The tensor rank format represent a rank R tensor as a

sum of R rank 1 tensors, in an analogy to the matrix rank format. A rank 1 n^{th}-order tensor is a tensor such that it can be written as the tensor product of n 1-order tensors.

Entry-wise, we can write the rank R third-order tensor U in tensor rank format as:

$$u_{ijk} = \langle \boldsymbol{a}_i, \boldsymbol{b}_j, \boldsymbol{c}_k \rangle, \tag{2.3}$$

$$= \sum_{r=1}^{R} a_{ir} b_{jr} c_{kr}, \tag{2.4}$$

where

- $\langle \cdot, \cdot, \cdot \rangle$ denotes the trilinear product, which is an extension of dot product to more than two vectors,
- $\boldsymbol{a}_i, \boldsymbol{b}_j, \boldsymbol{c}_k \in \mathbb{R}^R$ are the factor vectors corresponding to the entry u_{ijk},
- a_{ir}, b_{jr}, and c_{kr} are the scalar elements of the factor matrices.

The rank of a tensor as presented here is the smallest number R that the tensor rank format using R rank-1 tensors can exactly represent it. Tensors have different definitions of rank and the concept is still not well understood, in opposite to the case of matrix rank. In general, finding the rank of a tensor is an NP-hard problem [31].

2.5.2 Tucker format

The *Tucker format* [82] is a more general and flexible format than the CP format. The Tucker format represents an n^{th}-order tensor by the n-mode tensor product of a *core tensor* with *factor matrices* on each mode.

For example, the Tucker representation format of a third-order data tensor $\mathsf{U} \in \mathbb{R}^{I \times J \times K}$ [46] is:

$$\mathsf{U} = \mathsf{W} \times_1 \boldsymbol{A} \times_2 \boldsymbol{B} \times_3 \boldsymbol{C}, \tag{2.5}$$

where

- $\mathsf{W} \in \mathbb{R}^{X \times Y \times Z}$ is the *third-order core tensor*,
- $\boldsymbol{A} \in \mathbb{R}^{I \times X}$, $\boldsymbol{B} \in \mathbb{R}^{J \times Y}$, and $\boldsymbol{C} \in \mathbb{R}^{K \times Z}$ are the *factor matrices*,
- $\times_n$ is the *n-mode tensor product with a matrix.*

More intuitively, each scalar element u_{ijk} of the tensor U can be written in the n-mode tensor product with vectors format [46] and in the summation format [82] as:

$$u_{ijk} = \mathsf{W} \;\bar{\times}_1\; \boldsymbol{a}_i \;\bar{\times}_2\; \boldsymbol{b}_j \;\bar{\times}_3\; \boldsymbol{c}_k, \tag{2.6}$$

$$= \sum_{x=1}^{X} \sum_{y=1}^{Y} \sum_{z=1}^{Z} w_{xyz} a_{ix} b_{jy} c_{kz}, \tag{2.7}$$

where

- $\boldsymbol{a}_i \in \mathbb{R}^X$, $\boldsymbol{b}_j \in \mathbb{R}^Y$, and $\boldsymbol{c}_k \in \mathbb{R}^Z$ are the factor *column vectors* corresponding to the rows of the factor matrices $\boldsymbol{A}, \boldsymbol{B}$, and $\boldsymbol{C}$, respectively,
- $\bar{\times}_n$ is the *n-mode tensor product with a column vector*, which makes the final result u_{ijk} a scalar,
- w_{xyz} is a scalar element of the core tensor W,
- a_{ix}, b_{jy}, and c_{kz} are the scalar elements of the factor matrices.

The factor matrices are usually constrained to have orthonormal column vectors, which make Tucker decomposition a form of higher order SVD [82] [46]. These orthonormal columns of the factor matrices can be thought of as the principle components that capture the most variance in data in each mode. The core tensor W consists of the weights of the interactions between these principle components and usually used as the compressed version of the input tensor. The Tucker format is the basic building block in tensor calculus that appears in many other formats [28], such as the block term format [14].

2.5.3 Block term format

Block term format [13] [14] [15] is a recent tensor representation format that generalizes the CP format to be more flexible as the Tucker format but still keeps the uniqueness property of CP format, instead of the lack of uniqueness in Tucker format.

Block term format represents each n^{th}-order tensor as the sum of T n^{th}-order tensors, where each of them is represented by Tucker format. For example, the block term format of a third-order data tensor $\mathsf{U} \in \mathbb{R}^{I \times J \times K}$, entry-wise, is:

$$u_{ijk} = \sum_{t=1}^{T} \left(\mathsf{W}_t \bar{\times}_1 \boldsymbol{a}_{i,t} \bar{\times}_2 \boldsymbol{b}_{j,t} \bar{\times}_3 \boldsymbol{c}_{k,t} \right), \tag{2.8}$$

These tensor representation formats have wide applications in many domains. The ideas of CP and Tucker formats was introduced in the work of Hitchcock in 1927 [34] but have only become popular in psychometrics in 1960s after the work of Tucker [82], whom the Tucker format was named after. Since then, these tensor formats have been being used increasingly in other domains such as signal processing [68], computer visions [85], and recommender systems [74].

These tensor representation formats are usually used for data analysis and finding latent factors that explain the data tensor. Traditional methods to solve the tensor decomposition under these tensor representation formats are usually alternating least squared error method for unbounded continuous values input tensors. There are other methods for representing and analyzing higher-order tensor data. However, in this thesis, we will show that these tensor representation formats are interesting and useful both theoretically and computationally to explain and compute knowledge graph embeddings.

Chapter 3

Related Work

In this chapter, we review the literature that is most relevant to our research. We first survey various embedding methods including text embedding, graph embedding, and knowledge graph embedding for multi-relational data. In particular, we focus on the class of trilinear-product-based models, which includes the state-of-the-art models that are important objects of our study. We then review popular multi-relational datasets including knowledge graphs and scholarly datasets. We also review the applications of knowledge graph embedding methods in multi-relational data representation and reasoning.

3.1 Multi-relational embedding methods

Knowledge graph is a universal data format that can represent both textual and graph data, thus knowledge graph embedding methods can be seen as a generalization of other embedding methods.

3.1.1 Knowledge graph embedding methods

Knowledge graph embedding is an active research topic with many different methods [89]. Based on the interaction mechanisms used in the second component of the general architecture, most popular knowledge graph embedding models can be categorized into three main categories [79]:

1. *Semantic matching models* are based on similarity measures between the head and tail embedding vectors such as bilinear map or trilinear product.

2. *Neural-network-based models* are based on using neural networks as universal approximators to compute the matching score for each triple.

3. *Translation-based models* are based on the geometric view of relation embeddings as translation vectors between the head and tail entity embeddings.

Semantic matching models

RESCAL [60] 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

$$\mathcal{S}(h,t,r) = \boldsymbol{h}^\top \boldsymbol{M}_r \boldsymbol{t}, \tag{3.1}$$

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 [3] is a recent model extending RESCAL by using the Tucker format [82]. 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 [95] 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

$$\mathcal{S}(h,t,r) = \langle \boldsymbol{h}, \boldsymbol{t}, \boldsymbol{r} \rangle = \sum_i h_i t_i r_i, \tag{3.2}$$

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 [61] uses a circular correlation between the head and tail embedding vectors; ComplEx [80] 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 [41] [47] represents each entity as two role-based embedding vectors and augments an inverse relation embedding vector. In our previous work [77], 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 [97]. How-

ever, 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 [34], Tucker format [82], and block term format [14]. 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 [46]. We analyze the connections to the related tensor decomposition methods in Section 4.2.3.

The trilinear-product-based models In this thesis, we focus on this class of trilinear-product-based models, including DistMult, CP, SimplE, and ComplEx. To facilitate further analysis, we review them in details. These models compute their scores by using the trilinear product between head, tail, and relation embeddings, with relation embedding playing the role of matching weights on the dimensions of head and tail embeddings:

$$\begin{aligned} \mathcal{S}(h,t,r) &= \langle \boldsymbol{h}, \boldsymbol{t}, \boldsymbol{r} \rangle \\ &= \boldsymbol{h}^\top diag(\boldsymbol{r}) \boldsymbol{t} \\ &= \sum_{d=1}^{D} \left(\boldsymbol{h} \odot \boldsymbol{t} \odot \boldsymbol{r} \right)_d \\ &= \sum_{d=1}^{D} h_d t_d r_d \end{aligned} \tag{3.3}$$

where

- $\boldsymbol{h}, \boldsymbol{t}, \boldsymbol{r}$ are embedding vectors of h, t, and r, respectively,
- $diag(\boldsymbol{r})$ is the diagonal matrix of $\boldsymbol{r}$,
- $\odot$ denotes the element-wise Hadamard product,
- D is the embedding size and d is the dimension for which h_d, t_d, and r_d are the scalar entries.

DistMult [95] embeds each entity and relation as a single real-valued vector. DistMult is the simplest model in this category. Its score function is symmetric, with the same

scores for triples (h, t, r) and (t, h, r). Therefore, it cannot model asymmetric data for which only one direction is valid, e.g., asymmetric triples such as *(Paper1, Paper2, cite)*. Its score function is:

$$\mathcal{S}(h, t, r) = \langle \boldsymbol{h}, \boldsymbol{t}, \boldsymbol{r} \rangle, \tag{3.4}$$

where $\boldsymbol{h}, \boldsymbol{t}, \boldsymbol{r} \in \mathbb{R}^D$.

CP [34] is similar to DistMult but embeds entities as head and as tail differently. Each entity e has two embedding vectors $\boldsymbol{e}$ and $\boldsymbol{e}^{(2)}$ depending on its role in a triple as head or as tail, respectively. Using different role-based embedding vectors leads to an asymmetric score function, enabling CP to also model asymmetric data. However, experiments have shown that CP's performance is very poor on unseen test data [47]. Its score function is:

$$\mathcal{S}(h, t, r) = \langle \boldsymbol{h}, \boldsymbol{t}^{(2)}, \boldsymbol{r} \rangle, \tag{3.5}$$

where $\boldsymbol{h}, \boldsymbol{t}^{(2)}, \boldsymbol{r} \in \mathbb{R}^D$.

A recent extension of CP was independently discovered under the name SimplE [41] and CP_h [47]. Its heuristic augments the training data by making an inverse triple $(t, h, r^{(a)})$ for each existing triple (h, t, r), where $r^{(a)}$ is the augmented relation corresponding to r. With this heuristic, CP_h significantly improves CP, achieving results competitive with ComplEx. Its score function is:

$$\begin{aligned} \mathcal{S}(h, t, r) = \quad & \langle \boldsymbol{h}, \boldsymbol{t}^{(2)}, \boldsymbol{r} \rangle \\ & \text{and } \langle \boldsymbol{t}, \boldsymbol{h}^{(2)}, \boldsymbol{r}^{(a)} \rangle, \end{aligned} \tag{3.6}$$

where $\boldsymbol{h}, \boldsymbol{h}^{(2)}, \boldsymbol{t}, \boldsymbol{t}^{(2)}, \boldsymbol{r}, \boldsymbol{r}^{(a)} \in \mathbb{R}^D$.

ComplEx [80] is an extension of DistMult that uses complex-valued embedding vectors that contain complex numbers. Each complex number c with two components, real a and imaginary b, can be denoted as $c = a + bi$. The complex conjugate $\overline{c}$ of c is $\overline{c} = a - bi$. The complex conjugate vector $\overline{\boldsymbol{t}}$ of $\boldsymbol{t}$ is form from the complex conjugate of the individual entries. Complex algebra requires using the complex conjugate vector of tail embedding in the inner product and trilinear product [1]. Thus, these products can be antisymmetric, which enables ComplEx to model asymmetric data [80] [81]. Its score function is:

$$\mathcal{S}(h, t, r) = \text{Re}(\langle \boldsymbol{h}, \overline{\boldsymbol{t}}, \boldsymbol{r} \rangle), \tag{3.7}$$

where $\boldsymbol{h}, \boldsymbol{t}, \boldsymbol{r} \in \mathbb{C}^D$ and $\text{Re}(c)$ means taking the real component of the complex number c.

As an extension of ComplEx, we proposed the more advanced Quaternion-based embedding model from the perspective of the weighted sum of trilinear products in our previous work [77], which were later confirmed in a concurrent work [97]. Several works have shown the benefit of using complex, quaternion, or other hyper-complex numbers in the hidden layers of deep neural networks [26] [59] [62]. To the best of our knowledge, we are the first to motivate and use quaternion numbers for the embedding vectors of knowledge graph embedding.

Quaternion numbers are extension of complex numbers to four components [40] [24]. Each quaternion number q, with one real component a and three imaginary components b, c, d, could be written as $q = a + b\boldsymbol{i} + c\boldsymbol{j} + d\boldsymbol{k}$ where $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$ are fundamental quaternion units, similar to the imaginary number i in complex algebra. As for complex conjugates, we also have a quaternion conjugate $\overline{q} = a - b\boldsymbol{i} - c\boldsymbol{j} - d\boldsymbol{k}$. An intuitive view of quaternion algebra is that each quaternion number represents a 4-dimensional vector (or 3-dimensional vector when the real component $a = 0$) and quaternion multiplication is rotation and scaling of this vector in 4- (or 3-)dimensional space. Compared to complex algebra, each complex number represents a 2-dimensional vector and complex multiplication is rotation and scaling of this vector in 2-dimensional plane [1]. The multiplication of the quaternion p with the quaternion q is a pure rotation of p by q when q has unit module, that is, $q = a + b\boldsymbol{i} + c\boldsymbol{j} + d\boldsymbol{k}, \sqrt{a^2 + b^2 + c^2 + d^2} = 1$. Restricting the relation embedding to unit quaternions for pure rotation is an effective regularization for the quaternion-based embedding model in practice.

Quaternion multiplication is noncommutative, thus there are multiple ways to multiply three quaternion numbers in the trilinear product. Here, we choose to write the score function of the model as:

$$\mathcal{S}(h, t, r) = \text{Re}(\langle \boldsymbol{h}, \bar{\boldsymbol{t}}, \boldsymbol{r} \rangle), \tag{3.8}$$

where $\boldsymbol{h}, \boldsymbol{t}, \boldsymbol{r} \in \mathbb{H}^D$.

By expanding this formula using quaternion algebra [40] and mapping the four components of a quaternion number to four embeddings in the multi-embedding interaction model [77], respectively, we can write the score function in the notation of the multi-

embedding interaction model as:

$$\begin{aligned}
\mathcal{S}(h,t,r) &= \text{Re}(\langle \boldsymbol{h}, \bar{\boldsymbol{t}}, \boldsymbol{r} \rangle) \\
&= \langle \boldsymbol{h}^{(1)}, \boldsymbol{t}^{(1)}, \boldsymbol{r}^{(1)} \rangle + \langle \boldsymbol{h}^{(2)}, \boldsymbol{t}^{(2)}, \boldsymbol{r}^{(1)} \rangle \\
&\quad + \langle \boldsymbol{h}^{(3)}, \boldsymbol{t}^{(3)}, \boldsymbol{r}^{(1)} \rangle + \langle \boldsymbol{h}^{(4)}, \boldsymbol{t}^{(4)}, \boldsymbol{r}^{(1)} \rangle \\
&\quad + \langle \boldsymbol{h}^{(1)}, \boldsymbol{t}^{(2)}, \boldsymbol{r}^{(2)} \rangle - \langle \boldsymbol{h}^{(2)}, \boldsymbol{t}^{(1)}, \boldsymbol{r}^{(2)} \rangle \\
&\quad + \langle \boldsymbol{h}^{(3)}, \boldsymbol{t}^{(4)}, \boldsymbol{r}^{(2)} \rangle - \langle \boldsymbol{h}^{(4)}, \boldsymbol{t}^{(3)}, \boldsymbol{r}^{(2)} \rangle \\
&\quad + \langle \boldsymbol{h}^{(1)}, \boldsymbol{t}^{(3)}, \boldsymbol{r}^{(3)} \rangle - \langle \boldsymbol{h}^{(2)}, \boldsymbol{t}^{(4)}, \boldsymbol{r}^{(3)} \rangle \\
&\quad - \langle \boldsymbol{h}^{(3)}, \boldsymbol{t}^{(1)}, \boldsymbol{r}^{(3)} \rangle + \langle \boldsymbol{h}^{(4)}, \boldsymbol{t}^{(2)}, \boldsymbol{r}^{(3)} \rangle \\
&\quad + \langle \boldsymbol{h}^{(1)}, \boldsymbol{t}^{(4)}, \boldsymbol{r}^{(4)} \rangle + \langle \boldsymbol{h}^{(2)}, \boldsymbol{t}^{(3)}, \boldsymbol{r}^{(4)} \rangle \\
&\quad - \langle \boldsymbol{h}^{(3)}, \boldsymbol{t}^{(2)}, \boldsymbol{r}^{(4)} \rangle - \langle \boldsymbol{h}^{(4)}, \boldsymbol{t}^{(1)}, \boldsymbol{r}^{(4)} \rangle,
\end{aligned} \tag{3.9}$$

where $\boldsymbol{h}, \boldsymbol{t}, \boldsymbol{r} \in \mathbb{H}^D$.

Neural-network-based models

These models use a nonlinear neural network to compute the matching score for a triple:

$$\mathcal{S}(h,t,r) = NN(\boldsymbol{h}, \boldsymbol{t}, \boldsymbol{r}), \tag{3.10}$$

where

- $\boldsymbol{h}, \boldsymbol{t}, \boldsymbol{r}$ are the embedding vectors of h, t, and r, respectively,
- NN is the neural network used to compute the score.

One of the simplest neural-network-based model is ER-MLP [17], which concatenates the input embedding vectors and uses a multi-layer perceptron neural network to compute the matching score. NTN [71] is a neural network extension of RESCAL that uses nonlinear neural network on top of the bilinear layer. Recent models such as ConvE [16] use convolution networks instead of fully-connected networks.

These models aim to learn a neural network, to automatically model the interaction. These models are usually complicated because they use neural networks as a black-box universal approximator to compute the matching score, which usually make them computational expensive and difficult to understand. Recent models using convolutional neural networks such as ConvE [16] can achieve good results by sharing the convolution weights.

However, they are restricted by the input format to the neural network [16], and the operations are generally less expressive than direct interactions between the entries of the embedding vectors [61]. We will empirically compare with them.

Translation-based models

These models translate the head entity embedding by summing with the relation embedding vector, then measuring the distance between the translated images of head entity and the tail entity embedding, usually by L^1 or L^2 distance:

$$\begin{aligned} \mathcal{S}(h,t,r) = & -||\boldsymbol{h}+\boldsymbol{r}-\boldsymbol{t}||_p \\ = & -\left(\sum_d^D |h_d + r_d - t_d|^p\right)^{1/p}, \end{aligned} \tag{3.11}$$

where

- $\boldsymbol{h}, \boldsymbol{t}, \boldsymbol{r}$ are embedding vectors of h, t, and r, respectively,
- p is 1 or 2 for L^1 or L^2 distance, respectively,
- D is the embedding size and d is the dimension for which h_d, t_d, and r_d are the scalar entries.

TransE [8] was the first model of this type, with score function basically the same as the above equation. There have been many extensions such as TransR [53], TransH [92], and TransA [93]. Most extensions are done by linear transformation of the entities into a relation-specific space before translation [53].

The main advantages of these models are their simple and intuitive mechanism with the relation embeddings as the translation vectors [8]. However, it has been shown that they have limitations in expressiveness because of over-strong assumptions about translation using relation embedding [41]. Therefore, they are unable to model some forms of data such as symmetric relation [91]. The recent model TorusE [18] 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.

3.1.2 Text embedding methods

Text embedding methods were developed independently of knowledge graph embedding. In this thesis, we restrict to the shallow text embedding models such as word2vec [57]

[56] and fastText [6] [38], that have achieved great success due to large training data and lots of engineering efforts from the community. They are a crucial ingredient in modern natural language processing and text mining systems.

The early models were usually based on language modeling of the form predicting the next word given context-words [4]. Since then, word embedding models have become more effective and computationally efficient. The most popular word embedding models in recent years are word2vec variants such as skipgram [57] [56]. The word2vec skipgram model define a context of m context-words, then predicts the context-words $c_i, i = 1, \ldots, m$ given the target-word w, that is:

$$\begin{aligned} &P(c_i|w), \\ &i = 1, \ldots, m. \end{aligned} \tag{3.12}$$

Computing these multinoulli distributions requires the expensive softmax function. In practice, word2vec skipgram avoids this by approximating them with negative sampling and solve for the Bernoulli distributions:

$$\begin{aligned} &P(1|c_i, w), \\ &i = 1, \ldots, m, \end{aligned} \tag{3.13}$$

which can be computed efficiently by using the standard logistic function σ:

$$\begin{aligned} &P(1|c_i, w) = \sigma(\boldsymbol{u}_{c_i}^\top \boldsymbol{v}_w), \\ &i = 1, \ldots, m, \end{aligned} \tag{3.14}$$

where $\boldsymbol{u}_{c_i}$ is the context-embedding vector of context-word c_i and $\boldsymbol{v}_w$ is the word-embedding vector of target-word w.

The resulted word embedding vectors, $\boldsymbol{w}$, have been shown to capture the syntactic and semantic information in the original text data [57], which enable them to be used as the pretrained feature vectors for various tasks such as name entity recognition, text classification, and question answering. Especially, it has been observed that the embedding space has the linear semantic structure, notably with the example *king* - *man* $\approx$ *queen* - *woman* [57] [52].

Later text embedding models are also log-bilinear model as word2vec but can integrate subword information such as fastText [6] [38]. For document embedding, one of the most popular model is doc2vec [48], which extends word2vec to additionally learn the document embedding by predicting words in the document. An extension of doc2vec is session2vec

[54], which also learn the embeddings for sessions that contain multiple documents. There are also several works on computing document embedding by averaging bag-of-word-embeddings such as [37] that use a deep multi-layer perceptron neural network on top of the bag-of-word-embeddings for document classification.

There are recent works on deep learning for text embeddings using long-short term memory (LSTM) [35] [64], convolutional neural network (Temporal CNN) [22], and Transformer [86] to compute contextual word embeddings at a higher computational cost. However, these models are outside the scope of this thesis.

3.1.3 Network embedding methods

There are many approaches for computing representations of a network. In this thesis, we restrict to the class of shallow network embedding models that are based on random walks and the skipgram model of word2vec.

The most popular models of this type is DeepWalk [63], which was directly developed based on word2vec. They use random walks on the network to sample sequences of nodes and use them in a similar manner to sequences of words in word2vec to learn node embeddings. A closely related model to DeepWalk is Node2Vec [25], where they introduced a strategy to balance between depth-first-search and breadth-first-search for the random walk. Another related model is LINE [75] where they limit the neighborhood to at most two-step away from each node.

There are recent works on deep learning for graph neural networks such as message passing using graph convolutional networks (GCN) [43], message passing using aggregation–readout functions (GraphSAGE, GIN) [29] [94], attention-based message passing (GAT) [87] that try to learn contextual representations of nodes and networks at a higher computational cost. However, these models are outside the scope of this thesis.

3.2 Datasets

3.2.1 Knowledge graphs

Knowledge graphs have become more popular in recent years and many knowledge graphs have been built to store many different type of data. Specialized knowledge graphs about a domain, such as Wordnet [58] about lexical information. Large real-world knowledge graphs, such as Freebase [7] and Wikidata [88].

3.2.2 Information networks

There are many networks of interests that have driven the researches and developments in network data mining over the years. For example, social networks [51] and especially biomedical networks, such as drug-protein interaction networks [98].

3.2.3 Scholarly data

Scholarly data is a large and important type of multi-relational data. They contain two types of data, bibliographic knowledge graphs and scientific knowledge graphs.

There exist several large scale bibliographic datasets thanks to the recent trend in open science and open access publishing. The popular ones are listed in Table 3.1. Out of them, the MAG dataset is the most complete and official [70]. MAG was used in some online challenges like KDD challenge 2016[1], WSDM challenge 2016[2].

Table 3.1: Some popular bibliographic datasets.

Dataset	Source	Scope	Notes
DBLP	https://dblp.uni-trier.de/	Computer Science	The first open bibliographic dataset. Lacking some important information: paper abstract, fulltext, citation, etc.
MAG	Online at https://academic.microsoft.com/	Multidisciplinary	Having full relational information, but lacking text content like paper abstract, fulltext. One of the most complete and official dataset.
MAS	http://academic.research.microsoft.com/	Computer Science	Old version of MAG. Having paper abstract. Crawled in 2011-2012.
ArnetMiner	https://aminer.org/open-academic-graph/	Multidisciplinary	Similar to MAG, less complete.
CORE	https://core.ac.uk/services#dataset	Multidisciplinary	Collection of open-access papers, no closed-access papers. Having fulltext content of about 10% of its papers.
arXiv	https://arxiv.org/	Some quantitative fields: Computer Science, Physics, Maths, etc.	Pre-print. Having fulltext content.

Bibliographic knowledge graphs

Bibliographic knowledge graphs contain information about the bibliographic data or metadata of the papers, such as authorship, publishing venue, citations. They can be constructed directly using the network structure of bibliographic data. Bibliographic data

[1] https://kddcup2016.azurewebsites.net/Data/
[2] http://www.wsdm-conference.org/2016/wsdm-cup.html

have an inherent multi-relational structure. For example, the Microsoft Academic Graph dataset [70] has scholarly entities such as *paper*, *author*; the relation types such as *paper-cites-paper*, *author-writes-paper*; and entity attributes such as *paper-title*, *author-name*.

There have been several attempts to build bibliographic knowledge graphs in recent years. The approaches include either adopting the bibliographic network structure directly [90], or derive the knowledge graph from some similarity measures [83] [66], or constructing the knowledge graph from survey paper [21]. Figure 3.1 show an overview of a bibliographic knowledge graph that is directly constructed from the bibliographic network structure.

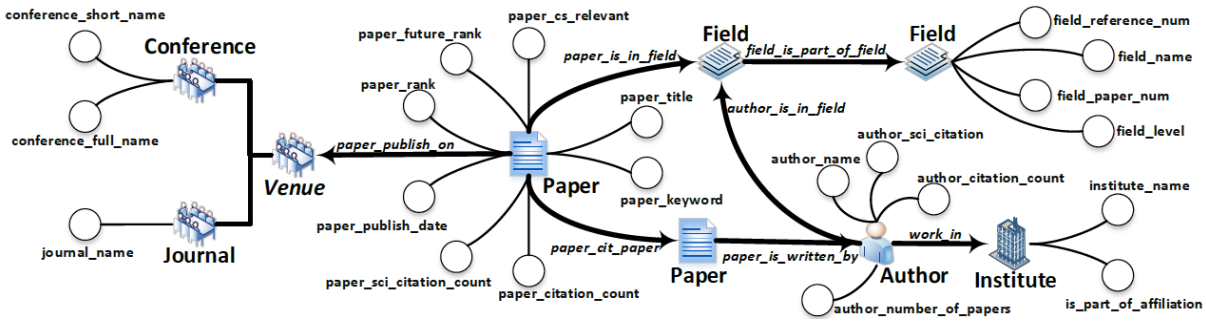

Figure 3.1: Example of a bibliographic knowledge graph (source [90]).

Scientific knowledge graphs

Scientific knowledge graphs contain information about scientific concepts and knowledge. Traditionally, they can be manually constructed by experts and scientists in a scientific domain. Recently, scientific knowledge graphs have been automatically constructed from the content of scientific papers using named entity recognition and integrated with other sources of data. For example, chemical knowledge graphs such as Chematica[3] include organic chemical compounds as entities and chemical reactions as relations, respectively. The triples in such knowledge graphs represent the existing reactions. The tasks of link prediction or relational querying would be useful for discovering new reaction routes, optimizing existing reactions, or avoiding hazardous by-products.

There are several scientific knowledge graphs, including biomedical knowledge graphs such as drug–protein interaction networks, protein–protein interaction networks, drug–drug side effect networks [98].

[3] `https://neo4j.com/news/the-chemical-knowledge-graph/`

3.3 Applications of embedding methods

Large real-world knowledge graphs, such as Freebase [7] and Wikidata [88] have found important applications in many artificial intelligence tasks, such as question answering, semantic search, and recommender systems. Google acquired the Freebase and make it their own knowledge graphs [69].

Knowledge graph embedding methods perform link prediction for knowledge graph completion [79]. These models also provide the embeddings as a useful representation of the whole knowledge graph that can be used for data visualization and analysis. They may also boost applications of knowledge graphs in artificial intelligence tasks such as explicit reasoning on knowledge graph embedding space [30] and implicit reasoning on knowledge graph embedding space with query by examples [78].

In this thesis, we concern the applications of knowledge graph embedding methods and the knowledge graph embedding space for scholarly data analysis and reasoning. Especially on the bibliographic knowledge graphs of COVID-19 papers and the scientific knowledge graphs of COVID-19 medical terms, drugs, and diseases.

Chapter 4

Multi-Partition Embedding Interaction: A General Mechanism for Knowledge Graph Embedding

In this chapter, we present the main theoretical contributions of the thesis, namely the multi-partition embedding interaction model. This proposed model and the accompanying concepts provide a new perspective and general mechanism for knowledge graph embedding. We start with introducing the new concepts and perspectives of multi-partition embeddings and local partition interactions. We then propose the multi-partition embedding interaction (MEI) model to generalize previous models and systematically address their drawbacks. We also show how MEI provides concrete advantages over previous embedding models, generalizes and explains their mechanisms, as well as explains some surprising phenomena in word embeddings.

4.1 Motivation

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 [7] and Wikidata [88] 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 em-

beddings 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 [78].

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 [95], the interaction pattern is the diagonal matching matrix between head and tail embedding vectors, as detailed in Section 3.1.1.

Most previous works treat embedding as a whole and model the interaction between the whole embeddings. For example, the bilinear model RESCAL [60] and the recent model TuckER [3] 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 [95] and recent state-of-the-art models HolE [61], ComplEx [80], SimplE [41] [47], and Quaternion embedding model [77] [97]. 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 [82] 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 [14] in tensor calculus. The result is a multi-partition embedding interaction (MEI) model with block term format that combines advanced tensor representation formats and modern deep learning techniques to provide 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 [79].

4.2 The multi-partition embedding interaction model

In this section, we first identify two fundamental concepts in our model, namely *multi-partition embedding* and *modeling the interaction patterns*. We then propose the multi-partition embedding interaction model and discuss its details.

4.2.1 Fundamental principles and concepts

Based on our analysis of previous work, we identify two fundamental aspects that a knowledge graph embedding model needs to address as follow.

1. *Restricting the interaction of embedding entries* for sparsity and high computational efficiency.

2. *Designing special interaction patterns between embedding entries* for high expressiveness.

Most previous models focus on either one of the aspects and neglect the other one. In RESCAL and TuckER, they model the full interactions between the embedding vectors. Thus, they can maximize the second aspect but have low computational efficiency. In trilinear-product-based models, they use specially designed interaction mechanisms to impose special architectural bias on the interaction patterns. However, their interaction patterns are specially designed and fixed, limiting their expressiveness, potentially causing them to be suboptimal or difficult to extend.

In the MEI model, we address both aspects and construct MEI with following two corresponding fundamental 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 [82], 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 [14] emerges from the sum score of all local interactions.

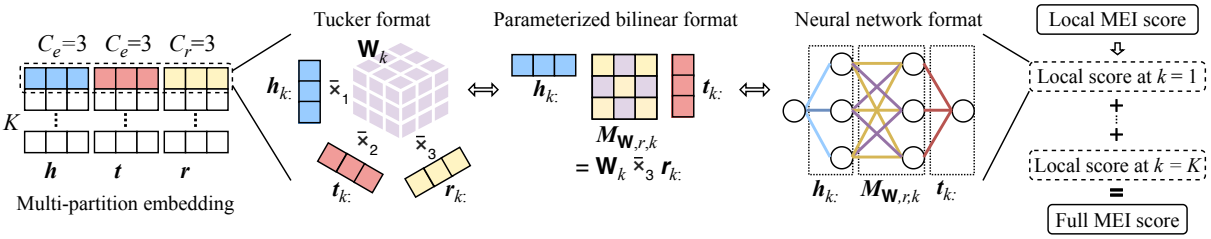

Figure 4.1: MEI architecture: multi-partition embedding vectors that interact only between the corresponding partitions. This figure illustrates a MEI model with block term format in three different views for the local-partition interaction: Tucker format, parameterized bilinear format, and neural network format.

Note that the concept of multi-partition embedding interaction is highly general and intuitive, as discussed in Section 4.3.1. For simplicity, we specifically adopt the Tucker and block term tensor formats to realize a simple yet general standard MEI model.

4.2.2 Model definition

The MEI model architecture is illustrated in Fig. 4.1. 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,

$$\mathcal{S}(h, t, r; \boldsymbol{\theta}) = \sum_{k=1}^{K} \left(\mathbf{W}_k \bar{\times}_1 \boldsymbol{h}_{k:} \bar{\times}_2 \boldsymbol{t}_{k:} \bar{\times}_3 \boldsymbol{r}_{k:} \right), \tag{4.1}$$

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* [46], 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

$$\mathcal{S}(h, t, r; \boldsymbol{\theta}) = \sum_{k=1}^{K} \left(\sum_{x=1}^{C_e} \sum_{y=1}^{C_e} \sum_{z=1}^{C_r} w_{xyz,k} h_{kx} t_{ky} r_{kz} \right), \tag{4.2}$$

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.

4.2.3 Model details

Let us analyze the details of MEI, with a focus on the local interactions in MEI, called local MEI, which are the building blocks of the full MEI model.

4.2.3.1 Tucker format and block term format

We choose to model the local interaction at each partition by the *Tucker format* [82] of third-order tensor

$$\mathcal{S}_k(h, t, r; \boldsymbol{\theta}) = \mathbf{W}_k \bar{\times}_1 \boldsymbol{h}_{k:} \bar{\times}_2 \boldsymbol{t}_{k:} \bar{\times}_3 \boldsymbol{r}_{k:} \tag{4.3}$$

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* [14]. 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.

We use the Tucker format and block term format to construct a simple and efficient model that realizes our ideas of multi-partition embedding interaction. In addition, they provide an elegant mathematical framework that facilitates the theoretical analysis of our model. This MEI model with block term format is an advanced development of our earlier preliminary work on analyzing knowledge graph embedding methods from a multi-embedding interaction perspective [77].

4.2.3.2 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. 4.3 as a *parameterized bilinear model*, by rewriting

the tensor products as

$$\begin{aligned}
\mathcal{S}_k(h,t,r;\boldsymbol{\theta}) &= \mathbf{W}_k \bar{\times}_1 \boldsymbol{h}_{k:} \bar{\times}_2 \boldsymbol{t}_{k:} \bar{\times}_3 \boldsymbol{r}_{k:} \\
&= (\mathbf{W}_k \bar{\times}_3 \boldsymbol{r}_{k:}) \bar{\times}_1 \boldsymbol{h}_{k:} \bar{\times}_2 \boldsymbol{t}_{k:} && (4.4) \\
&= \boldsymbol{h}_{k:}^\top (\mathbf{W}_k \bar{\times}_3 \boldsymbol{r}_{k:}) \boldsymbol{t}_{k:} && (4.5) \\
&= \boldsymbol{h}_{k:}^\top \boldsymbol{M}_{\mathbf{W},r,k} \boldsymbol{t}_{k:}, && (4.6)
\end{aligned}$$

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.

4.2.3.3 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 [72] and batch normalization [36] to reduce overfitting and improve the convergence rate. Specifically, Eq. 4.6 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. This type of network architectures is usually known as hypernetworks in the neural network literature [27]. 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.

4.2.4 Model constraints and variants

The MEI model architecture and its fundamental concepts are general. Here we consider some most important constraints and model variants.

4.2.4.1 Core tensor: non-shared core vs. shared core

As we discussed in Section 4.2.3, the core tensors help the model learn the interaction patterns between the embedding vectors. In general, each local MEI model in different partitions have different core tensors and can learn different interaction patterns. However, if we assume that each dataset has some specific interaction patterns, it makes sense to reuse the core tensors across different partitions. We consider the the most simple case, where all local MEI models share a single core tensor:

$$\mathbf{W} = \mathbf{W}_1 = \cdots = \mathbf{W}_K. \tag{4.7}$$

This *shared core* constraint has several benefits. First, it reduces the number of learnable parameters in the core tensors. A MEI model with K partitions of size C will have KC^3 parameters in its core tensors. On the other hand, sharing core tensor will reduce the number of parameters in the core tensor to only C^3, that is, independently from the number of partitions. Second, it can act as a regularization constraint to avoid overfitting and improve generalization. We will empirically compare its performance in the experiments.

4.2.4.2 Matching matrix: max rank and the orthogonality constraint

The relational matching matrix $\boldsymbol{M}_{\mathbf{W},r,k}$ linearly transforms the head embedding partition $\boldsymbol{h}_{k:}$ before matching it to the corresponding tail embedding partition $\boldsymbol{t}_{k:}$. A general linear transformation can be of many different types, such as rotation, reflection, translation, scale, or shear. When a matrix is not full rank, in other words, when it is singular or degenerate, the transformation result will lose information. In the MEI model, this may become a more critical issue when the matching matrix is parameterized by a small relation embedding partition.

To counter this issue, we can use a max rank constraint to influence the matching matrix to have as large rank as possible. A particularly interesting constraint that have this effect is the soft orthogonality constraint, expressed by the following loss.

$$\mathcal{L}_{\text{ortho}} = \sum_{k=1}^{K} ||\boldsymbol{M}_{\mathbf{W},r,k}^{\top} \boldsymbol{M}_{\mathbf{W},r,k} - \boldsymbol{I}||_2^2, \tag{4.8}$$

where $\boldsymbol{I} \in \mathbb{R}^{C_e \times C_e}$ is the identity matrix.

This constraint is equivalent to enforcing the linear transformation to be a rotation with a potential reflection when the determinant equal minus 1. The soft orthogonality

constraint has several benefits, such as being simple and computationally feasible, being easily integrated into the loss function as an additional loss term, and the orthogonal transformation preserving the norm of the embedding and the distance in the embedding space thus making the matching scores more stable. This constraint influences the matching matrix to be full rank and can be important to avoid overfitting, especially when the relation embedding is small.

4.3 Theoretical analysis

In this section, we discuss the theoretical foundations, intuitions, and insights of MEI, then we study the optimal parameter efficiency.

4.3.1 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.

4.3.1.1 Sparse modeling

The full MEI model can be seen as a special form of *sparse parameterized bilinear models*, which are the bilinear models whose matching matrices are sparse block-diagonal matrices parameterized by the core tensors and the relation embeddings. 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

$$\boldsymbol{M}_{\mathbf{W},r}^{(s)} = \begin{bmatrix} \boldsymbol{M}_{\mathbf{W},r,1} & 0 & \cdots & 0 \\ 0 & \boldsymbol{M}_{\mathbf{W},r,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{M}_{\mathbf{W},r,K} \end{bmatrix}. \tag{4.9}$$

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

$$\mathcal{S}(h,t,r;\boldsymbol{\theta}) = \boldsymbol{h}^\top \boldsymbol{M}_{\mathbf{W},r}^{(s)} \boldsymbol{t}, \tag{4.10}$$

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

$$\mathcal{S}(h, t, r; \boldsymbol{\theta}) = \mathbf{W}^{(s)} \bar{\times}_1 \boldsymbol{h} \bar{\times}_2 \boldsymbol{t} \bar{\times}_3 \boldsymbol{r}. \tag{4.11}$$

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.

4.3.1.2 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. 4.1, 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.

4.3.1.3 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-dimensional entry of head embedding then matches it with the corresponding meta-dimensional entry of tail embedding. This framework can serve as a novel general design pattern of knowledge graph embedding methods, as we show in Section 4.5 how it can explain the previous specially designed models.

4.3.2 Computational analysis

4.3.2.1 Complexity

For simplicity, we consider the same embedding size $D = D_e = D_r$ and same partition size $C = C_e = C_r$ for both entity and relation, such that $D = KC$. 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. Therefore, the core tensor sizes can be considered as growing linearly with K in the case of non-shared-core MEI, and as a constant in the case of shared-core MEI.

4.3.2.2 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 4.3.1, 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 4.1. ***(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 4.2. ***(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 4.1. ***(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 the 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 4.2 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[1]. 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 4.1 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 5.2.2, where we show that models with partition sizes that are closer to the predicted optimal values usually achieve better results.

There are some limitations in using the proposed criterion. First, we assume the same partition size for both entity and relation, that is, $C = C_e = C_r$; and the same partition size for all partitions, that is, $C = C_1 = \cdots = C_K$. These restrictions are only for theoretical analysis purpose. In practice, the MEI model can use different partition sizes

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

flexibly, which can be tuned by cross-validation similar to other hyperparameters. Second, the predicted optimal partition size can be seen as a pessimistic prediction when assuming no special patterns in the data, that is, the data tensor entries are distributed uniformly at random. When there exist special patterns in the data, the amount of information to be encoded is reduced, and thus the optimal partition sizes may be smaller. Therefore, the proposed criterion only provides a general guideline 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. In practice, we observed that when the dataset is very large, sparse, and unevenly distributed, it may be preferable to restrict the partition size C to a small value such as from 10 to 100 and use multiple partitions $K \geq 3$, to keep the model sparse and try to maximize the empirical benefit of the ensemble boosting effect on multiple small local MEI models for practical purpose.

Comparison to other model selection indices In statistical learning, model selection is the problem of selecting the best model in a set of models given the training dataset by estimating the true test performance [49]. Popular methods for estimating the true test performance include k-fold cross validation and using model selection indices, such as Mallow's C_p, Akaike information criterion (AIC), Bayesian information criterion (BIC), and modified R^2 [9].

Cross validation is usually the most general and practical approach to estimate the true test performance. This is done by randomly dividing the training data into k separate subsets, then keeping each subset for evaluation and training on the rest $k-1$ subsets. However, cross validation is expensive because each model needs to be trained and evaluated k times corresponding to k folds. The model selection indices aim to estimate the true test performance by adjusting the training set performance according to the model size to account for the overfitting issue. However, they need to train and obtain the training set performance for each model, which is still expensive when the number of models is large.

The proposed optimal parameter efficiency criterion can be seen as a model selection index specifically developed for the MEI model. It offers two main advantages. First, it accounts for the specific architecture of the MEI model instead of general linear models as in the above model selection indices. Second, it directly estimates the capacity of the model on the dataset, which avoids the excessive cost of training and obtaining training set performance for every model configuration, especially with the very large number of possible MEI model configurations. However, there are some limitations in using the proposed optimal parameter efficiency criterion as a model selection index. First, it only

concerns the relationship between the partition size and the number of partitions; thus, other hyperparameters are still needed to be tuned using cross-validation similarly to other embedding models. Second, it only address the problem of parameter efficiency, which aims to best fit the data given a fixed amount of parameters. This does not address the overfitting issue and thus cannot be used to estimate the test performance directly. Therefore, in practice, the partition size may still need to be fine-tuned using cross-validation.

Similarly to our use of the degrees of freedom, there is the related Vapnik–Chervonenkis dimension (VC dimension) that aims to directly estimate the capacity of a model [84]. Traditional usage of VC dimension in predicting test performance is usually restricted to the case that the VC dimension is much smaller than the training set size. In addition, VC dimension is defined in a specific way that can be difficult to apply to the MEI model. We measure the capacity of the model using the degrees of freedom in the MEI model, which is more natural and intuitive for the MEI architecture.

4.4 Revisiting knowledge graph embedding models

In this section, we revisit knowledge graph embedding methods to provide a new generalization and explanation for previous model, reproduce them, and show the advantages of MEI.

4.4.1 Connections to specially designed interaction mechanisms

There exist a few generalizations of previous embedding models that include DistMult, ComplEx, and SimplE; such as [41] explaining them using a bilinear model, [3] using a vanilla Tucker model, and [77] 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.

4.4.1.1 Multi-partition embedding interaction patterns of trilinear-product-based models

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. 4.9 and Eq. 4.10. 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 [34] 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 [77].

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 the same partition k. These properties express two basic ideas of previous models, *restricting the interaction of entries* for sparsity and efficiency, and *designing special interaction patterns* for expressiveness. Note that they use the same interaction pattern for all partitions and all datasets.

The MEI model addresses both ideas and systematically improves them. First, the sparsity and efficiency is achieved by restricting the interaction to corresponding local partitions, which can be adapted for each dataset by setting the partition sizes. Second, the interaction patterns are modeled by the core tensors, which can be different for each dataset and each partition, and automatically learned from data.

4.4.1.2 Core tensors for reproducing trilinear-product-based models

Here we derive the core tensor to reproduce the above specially designed interaction mechanisms. We use a single shared core tensor for all partitions and let the partition size $C_e = C_r = 1$ in the case of DistMult, $C_e = C_r = 2$ in the case of ComplEx, CP, and SimplE (CP_h). These models can be reproduced from MEI exactly by setting the core tensor following the *core tensor* column in Table 4.1. They can also be seen as the weighted sum of the trilinear products of the column embedding vectors by setting the weights following the *weighted terms* column in Table 4.1 [77].

Table 4.1: Core tensors for reproducing specially designed interaction mechanisms.

Core tensor	**Weighted terms**	**DistMult**	**ComplEx**	**ComplEx equiv. 1**	**ComplEx equiv. 2**	**ComplEx equiv. 3**	**CP**	**CP equiv.**	**CP$_h$**	**CP$_h$ equiv.**
w_{111}	$\langle \boldsymbol{h}_{:1}, \boldsymbol{t}_{:1}, \boldsymbol{r}_{:1} \rangle$	1	1	1	0	0	0	0	0	0
w_{112}	$\langle \boldsymbol{h}_{:1}, \boldsymbol{t}_{:1}, \boldsymbol{r}_{:2} \rangle$	–	0	0	1	1	0	0	0	0
w_{121}	$\langle \boldsymbol{h}_{:1}, \boldsymbol{t}_{:2}, \boldsymbol{r}_{:1} \rangle$	–	0	0	-1	1	1	0	1	0
w_{122}	$\langle \boldsymbol{h}_{:1}, \boldsymbol{t}_{:2}, \boldsymbol{r}_{:2} \rangle$	–	1	-1	0	0	0	0	0	1
w_{211}	$\langle \boldsymbol{h}_{:2}, \boldsymbol{t}_{:1}, \boldsymbol{r}_{:1} \rangle$	–	0	0	1	-1	0	1	0	1
w_{212}	$\langle \boldsymbol{h}_{:2}, \boldsymbol{t}_{:1}, \boldsymbol{r}_{:2} \rangle$	–	-1	1	0	0	0	0	1	0
w_{221}	$\langle \boldsymbol{h}_{:2}, \boldsymbol{t}_{:2}, \boldsymbol{r}_{:1} \rangle$	–	1	1	0	0	0	0	0	0
w_{222}	$\langle \boldsymbol{h}_{:2}, \boldsymbol{t}_{:2}, \boldsymbol{r}_{:2} \rangle$	–	0	0	1	1	0	0	0	0

For DistMult, we can see the equivalence directly. For ComplEx, we need to expand its score function following complex algebra [1]:

$$\begin{aligned} \mathcal{S}(h,t,r) = &\quad \mathrm{Re}(\langle \boldsymbol{h}, \bar{\boldsymbol{t}}, \boldsymbol{r} \rangle) \\ = &\quad \langle \mathrm{Re}(\boldsymbol{h}), \mathrm{Re}(\boldsymbol{t}), \mathrm{Re}(\boldsymbol{r}) \rangle + \langle \mathrm{Re}(\boldsymbol{h}), \mathrm{Im}(\boldsymbol{t}), \mathrm{Im}(\boldsymbol{r}) \rangle \\ &- \langle \mathrm{Im}(\boldsymbol{h}), \mathrm{Re}(\boldsymbol{t}), \mathrm{Im}(\boldsymbol{r}) \rangle + \langle \mathrm{Im}(\boldsymbol{h}), \mathrm{Im}(\boldsymbol{t}), \mathrm{Re}(\boldsymbol{r}) \rangle, \end{aligned} \tag{4.12}$$

where

- $\boldsymbol{h}, \boldsymbol{t}, \boldsymbol{r} \in \mathbb{C}^D$,
- $\mathrm{Re}(\boldsymbol{c})$ and $\mathrm{Im}(\boldsymbol{c})$ mean taking the real and imaginary components of the complex vector $\boldsymbol{c}$, respectively.

Mapping $\mathrm{Re}(\boldsymbol{h})$ to $\boldsymbol{h}_{:1}$, $\mathrm{Im}(\boldsymbol{h})$ to $\boldsymbol{h}_{:2}$, $\mathrm{Re}(\boldsymbol{t})$ to $\boldsymbol{t}_{:1}$, $\mathrm{Im}(\boldsymbol{t})$ to $\boldsymbol{t}_{:2}$, $\mathrm{Re}(\boldsymbol{r})$ to $\boldsymbol{r}_{:1}$, and $\mathrm{Im}(\boldsymbol{r})$ to $\boldsymbol{r}_{:2}$, we can rewrite the score function of ComplEx as:

$$\begin{aligned} \mathcal{S}(h,t,r) = &\quad \mathrm{Re}(\langle \boldsymbol{h}, \bar{\boldsymbol{t}}, \boldsymbol{r} \rangle) \\ = &\quad \langle \boldsymbol{h}_{:1}, \boldsymbol{t}_{:1}, \boldsymbol{r}_{:1} \rangle + \langle \boldsymbol{h}_{:1}, \boldsymbol{t}_{:2}, \boldsymbol{r}_{:2} \rangle \\ &- \langle \boldsymbol{h}_{:2}, \boldsymbol{t}_{:1}, \boldsymbol{r}_{:2} \rangle + \langle \boldsymbol{h}_{:2}, \boldsymbol{t}_{:2}, \boldsymbol{r}_{:1} \rangle, \end{aligned} \tag{4.13}$$

which is equivalent to the weighted sum using the weight vectors in Table 4.1. Note that by the symmetry between h and t, we can obtain the equivalent weight vector *ComplEx equiv. 1*. By symmetry between the embedding components, we can obtain the equivalent weight vectors *ComplEx equiv. 2* and *ComplEx equiv. 3*.

For CP, note that the two role-based embedding vectors for each entity can be mapped to two-embedding vectors in our model and the relation embedding vector can be mapped to $\boldsymbol{r}_{:1}$. For CP$_h$, further note that its data augmentation is equivalent to adding the score of the original triple and the inverse triple when optimizing the likelihood using stochastic

gradient descent (SGD):

$$\begin{aligned}\mathcal{S}(h,t,r) = \quad & \langle \boldsymbol{h}, \boldsymbol{t}^{(2)}, \boldsymbol{r} \rangle \\ & + \langle \boldsymbol{t}, \boldsymbol{h}^{(2)}, \boldsymbol{r}^{(a)} \rangle.\end{aligned} \tag{4.14}$$

We can then map $\boldsymbol{r}^{(a)}$ to $\boldsymbol{r}_{:1}$ and rewrite the score function of CP_h as:

$$\begin{aligned}\mathcal{S}(h,t,r) &= \langle \boldsymbol{h}_{:1}, \boldsymbol{t}_{:2}, \boldsymbol{r}_{:1} \rangle + \langle \boldsymbol{t}_{:1}, \boldsymbol{h}_{:2}, \boldsymbol{r}_{:2} \rangle \\ &= \langle \boldsymbol{h}_{:1}, \boldsymbol{t}_{:2}, \boldsymbol{r}_{:1} \rangle + \langle \boldsymbol{h}_{:2}, \boldsymbol{t}_{:1}, \boldsymbol{r}_{:2} \rangle,\end{aligned} \tag{4.15}$$

which is equivalent to the weighted sum using the weight vectors given in Table 4.1. By symmetry between h and t, we can obtain the equivalent weight vector *CP equiv.* and *CP_h equiv.*

Similarly, the score function of the quaternion-based embedding model [77] can also be reproduced by the core tensor of MEI. Note that when representing these models in the MEI framework, it is natural to multiply the head embedding with the relation-based transformation matrix first, then multiply the result with all the tail embeddings. This operation order is more efficient if the number of tail entities is large compared to the number of relations.

4.4.2 Connections to other knowledge graph embedding models

4.4.2.1 Tensor representation formats in knowledge graph embedding

Note that the Tucker format is the basic building block of tensor calculus, similar to the role of multi-layer perceptron in neural networks. The Tucker format can be used in different ways and appear in different models, similarly to multi-layer perceptron appears in convolutional neural networks. Multi-layer perceptron is general but expensive, whereas convolutional neural networks are more efficient and work better for specific use cases. Therefore, for a new embedding model that includes the Tucker format, it is important to notice how the Tucker format is used and what advantages it provides.

Recently, the Tucker format has been used in knowledge graph embedding models independently to our work. However, they apply the Tucker format directly on the embedding vector as a whole [3], and thus this vanilla Tucker model suffers from the scalability problem when the embedding size increases. We use the Tucker format as a building block for modeling the local interactions in our model, which implements our ideas of multi-partition embedding interaction and essentially aims to solve the scalability problem. The

application of vanilla Tucker format is a special case of MEI when $K = 1$. When $K > 1$, the MEI model is sparse and can be more efficient.

In addition, the multi-partition embedding interaction framework is general approach towards knwledge graph embedding, which is not tied to the Tucker or block term formats. The MEI framework can be used to explain previous embedding models, as discussed in Section 4.3.1 and extended to other model variants, as discussed in Section 4.2.4.

4.4.2.2 Semantic matching models

The standard semantic matching models such as RESCAL [60] compute a bilinear map between the head and tail embedding vectors. The connection to semantic matching models is apparent when we notice that the matching matrices in RESCAL can be parameterize arbitrarily by the core tensor and the relation embedding vectors. Considering only one local MEI model by setting the number of partition to $K = 1$ and one-hot vectors as the relation embedding vector $\boldsymbol{r}$, each matching matrix $\boldsymbol{M}_r$ is a third mode slice of the core tensor W. Hence, for any RESCAL model, there exists a local MEI model that can reproduce it exactly. When $K > 1$, MEI is a sparse block-diagonal bilinear model where each block is parameterized by a shared core tensor making MEI more parameter efficient than RESCAL.

One important advantage of MEI compared to RESCAL is that the matching matrices are parameterized by the core tensors and the relations embedding vectors. Apparently, MEI has fewer parameters and is less expensive when the number of relations is large. More importantly, the core tensor parameters are shared between all relations, which facilitates learning embeddings of rare relations. In addition, the core tensors have intuitive meaning as they define the interaction patterns in the model.

4.4.2.3 Translation-based models

The connection to translation-based models such as TransE [8] is apparent when we notice that the translation transformation in n-dimensional space can be represented as a linear transformation in $(n+1)$-dimensional space using a specific format of the transformation matrix. Local MEI is a parameterized bilinear model that can specify any format of the transformation matrix M_r. Hence, for any TransE model with embedding size n, there exists a local MEI model with embedding size $n + 1$ that can reproduce it exactly.

When $K > 1$, MEI can model the sum of multiple local translation interactions, which is not different from the single partition translation in TransE . However, when we consider the translation-based model extensions such as TransR, which uses linear transformation

before translation, MEI can provide benefits similarly to that in the case of RESCAL. Generally speaking, MEI suggests that any extension of translation-based models should be done on multi-partition embedding vector instead of the full vector. Therefore, the approach of MEI can be applied to all previous translation-based model extensions.

4.4.3 Advantages of MEI over previous knowledge graph embedding models

We have discussed the general advantages of MEI over previous knowledge graph embedding models, including its generality and flexibility, its theoretical framework, and its applications in explaining and extending previous models. In addition, here we discuss in more details the concrete advantages of MEI over specific models. In particular, we concern two cases, first the vanilla Tucker model ; and second a variant of the ComplEx model called RotatE [73].

Regarding the vanilla Tucker model, it is equivalent to MEI when $K = 1$. The MEI model has two major advantages, parameter efficiency and practical flexibility. First, the vanilla Tucker model only has one embedding partition and one core tensor that model the interactions on this partition. The core tensor grows cubically to the embedding size and the model becomes inefficient when the embedding size is larger than the optimal partition size. Second, in practice, depending on the applications we may want to obtain highly informative embedding vectors. In the vanilla Tucker model, increasing the size of the embedding vectors comes at a large cost due to the size of the core tensor. Moreover, we may assume that a large part of the information captured by the model is stored in the core tensor instead of the embedding vectors. In this case, MEI can provide a practical trade-off between the embedding vectors and the core tensor, and thus may be more suitable for such applications.

Regarding RotatE, it is a variant of the ComplEx model where each relation partition is normalized to unit norm, that is, $C_e = C_r = 2$ and each relation partition satisfies $||\boldsymbol{r}_{k,:}||_2 = 1, \forall k \in [1, K]$. Recall in Section 4.4.1, we show that ComplEx has the rotation pattern on each block of the matching matrix. However, the block is not normalized to orthogonal so the transformation is not exactly rotation but a rotation with scaling. RotatE makes the transformation exactly rotation to make it easier to optimize and avoid overfitting. Here we state a simple result about the parameter efficiency advantage of MEI, and provide an example about the expressiveness advantage of MEI.

Corollary 4.1.1. ***(More expressive than trilinear-product-based models)*** *In real-*

world datasets with more than 4 entities and relations, given an arbitrary number of parameters, there exists a MEI model that is more expressive than the previous trilinear-product-based models including DistMult, CP, SimplE, and ComplEx.

This result follows directly from the Theorem 4.1. When $|\mathcal{E}| + |\mathcal{R}| > 4$, we have optimal $C > 2$. Because the listed previous trilinear-product-based model has $C = 1$ or $C = 2$, they are suboptimal compared to MEI. Therefore, for an arbitrary fixed number of parameters, MEI is more expressive than previous trilinear-product-based models.

To give a concrete example of how MEI is more expressive than previous model, we will show the limit of the RotatE model, and to some extent, the ComplEx model. We first notice that rotation in 2-dimensional space is non-commutative. Therefore, the models that use a rotation in 2-dimensional space such as RotatE cannot differentiate between different orders of the relations, as shown in [10].

Example 4.1. ***(Limitation of RotatE)*** Considering two relations *'s son* and *'s wife*. There are two different order combinations, namely *A's son's wife* and *A's wife's son*. It is clearly that different orders of the relations can result in different meanings. However, the RotatE model cannot differentiate the different orders of the relations.

MEI provides a general approach to increase the expressiveness of the model enabling it to solve this example. In particular, MEI with $C \geq 2$ can solve this problem, because rotation in 3-dimensional space and above is non-commutative, thus, it can differentiate the different orders of the relations.

4.5 Revisiting word embedding models

We revisit word embedding and language modeling to provide a broader connection that potentially benefits the research and development in both knowledge graph embedding and word embedding domains [78].

4.5.1 Connections between CP_h and word2vec skipgram

Word2vec skipgram is well-studied both theoretically and empirically [57] [52]. Its word embedding vectors contain rich information and have application in various tasks. CP_h is a recent state-of-the-art knowledge graph embedding models. We will show the theoretical connections between them to better understand and use both these models and their embedding vectors.

Let us look at Eq. 3.14 of the word2vec skipgram model and consider only one context-word c for simplicity. We can write the probability in proportional format as:

$$P(1|c,w) \propto \exp\left(\boldsymbol{u}_c^\top \boldsymbol{v}_w\right), \tag{4.16}$$

where $\boldsymbol{u}_c$ is the context-embedding vector of context-word c and $\boldsymbol{v}_w$ is the word-embedding vector of target-word w.

We now note that the pair of context-word c and target-word w is ordered because word2vec skipgram use different role-based embedding vectors for each word when it is a context-word or a target-word. Also note that in word2vec skipgram, the target-word is the central word in the sliding window, e.g., w_i is the target-word and $w_{i-k}, \ldots, w_{i-1}$, $w_{i+1}, \ldots, w_{i+k}$ are context-words. Therefore, the roles in each pair of words are symmetric. Each word can be either a target-word or a context-word of each other when the sliding window slides through the text. When maximizing the likelihood by stochastic gradient descent, we can write the approximate probability of unordered pair of words as:

$$P(1|c,w;w,c) \propto \exp\left(\boldsymbol{u}_c^\top \boldsymbol{v}_w + \boldsymbol{u}_w^\top \boldsymbol{v}_c\right), \tag{4.17}$$

where

- $\boldsymbol{u}_c$ and $\boldsymbol{v}_c$ are the context-embedding and word-embedding vectors of c, respectively,
- $\boldsymbol{u}_w$ and $\boldsymbol{v}_w$ are the context-embedding and word-embedding vectors of w, respectively,

The dot product in the right hand side of Eq. 4.17 can be expanded as:

$$P(1|c,w;w,c) \propto \exp\left(\sum_{d=1}^{D} u_{cd} v_{wd} + \sum_{d=1}^{D} u_{wd} v_{cd}\right), \tag{4.18}$$

where u_{cd}, v_{cd}, u_{wd}, and v_{wd} are the scalar entries of context-embedding and word-embedding vectors.

We now return to Eq. 3.6 of CP_h score function. We can also write the knowledge graph embedding probability in Eq. 2.1 in proportional format and expand the trilinear

product according to Eq. 3.3 as:

$$P(1|h,t,r) \propto \exp\left(\mathcal{S}_{CP_h}(h,t,r)\right) \tag{4.19}$$

$$\propto \exp\left(\langle \boldsymbol{h}, \boldsymbol{t}^{(2)}, \boldsymbol{r}\rangle + \langle \boldsymbol{t}, \boldsymbol{h}^{(2)}, \boldsymbol{r}^{(a)}\rangle\right) \tag{4.20}$$

$$\propto \exp\left(\sum_{d=1}^{D} h_d t_d^{(2)} r_d + \sum_{d=1}^{D} t_d h_d^{(2)} r_d^{(a)}\right), \tag{4.21}$$

where

- $\boldsymbol{h}, \boldsymbol{h}^{(2)}, \boldsymbol{t}, \boldsymbol{t}^{(2)}, \boldsymbol{r}, \boldsymbol{r}^{(a)}$ are knowledge graph embedding vectors $\in \mathbb{R}^D$,
- $h_d, h_d^{(2)}, t_d, t_d^{(2)}, r_d, r_d^{(a)}$ are the scalar entries.

Comparing Eq. 4.18 of word2vec skipgram and Eq. 4.21 of CP_h, we can see they have essentially the same format. Except that in CP_h, each embedding dimension is weighted differently for each relation r, by the corresponding entry of the relation embedding vectors. Note that the embedding vectors in word2vec skipgram are learned by aligning each target-word to different context-words and vice versa. This mechanism is essentially the same for CP_h by aligning each tail entity to different head entities in different triples and vice versa, with regards to the different dimension weightings by different relations. Given these equivalence, we will state the following duality.

Word2vec skipgram from CP_h perspective Word2vec skipgram is a special case of CP_h where there is only one relation and the the relation embedding is the identity vector. In word2vec skipgram, there is a single implicit relation that capture the information about the context defined by the sliding window. The technique of using a sliding window to define the context words and the target words can be seen as a simple method to construct a knowledge graph with a single relation from the textual data. Because there is only one relation, its true embedding can be implicitly absorbed into the word embeddings, resulting in the identity relation embedding vector as we see. This observation will be used to explain some recent phenomena in word embedding space below.

CP_h from word2vec skipgram perspective CP_h is a generalization of word2vec skipgram to model multi-relational data. Note that the basic foundation of CP_h is the CP tensor rank format, but the reverse relation heuristics is not theoretically justified and still an open question [47]. From the connection to word2vec skipgram, we can see that the reverse relations stem from the equivalent role of the head and tail entities in the

knowledge graph, similarly to the equivalent role of the context-word and the target-word in the corpora. The relation embedding vectors have the role of the weight vectors that scale the dimensions of the embedding space between two entity embeddings.

4.5.2 Connections between knowledge graph embedding and language modeling

More generally, we extend the above connections to between knowledge graph embedding and language modeling.

Knowledge graph embedding as a language modeling task In this perspective, knowledge graph embedding is similar to language modeling that learns to predict the target-entity given the context-entity and the additional relational contexts, with the target-entity being either the head entity or the tail entity in each triple. In regard to the MEI framework in Section 4.3.1, the model learns to transform the context-entity using a relation-based transformation, then match it with the target-entity by the dot product.

Language modeling as a knowledge graph embedding task In this perspective, we will construct the knowledge graph for the corpora by defining the relations that capture the useful information including semantic and syntactic cues, such as co-occurrence of words in sliding windows, relative position between words, or absolute position of words in a sentence. After that, we can use knowledge graph embedding methods to model the constructed knowledge graph and solve the language modeling task.

4.5.3 Explaining some intriguing phenomena in embedding space

4.5.3.1 The global calibration matrix of bag-of-word embeddings

Word embeddings are well-developed and well-studied, however, there are still some unknown phenomena in the word embedding space. In a recent paper [2], the authors show that in the word embedding space obtained by word2vec skipgram and related models such as Glove, there exists a single global matrix, namely the "calibration" matrix, that maps the average of the context-embedding of context words to the word-embedding of the target word. Formally speaking:

$$\boldsymbol{v}_w \approx \boldsymbol{A} \sum_{c \in \text{context of } w} \boldsymbol{u}_c, \tag{4.22}$$

There are two important and surprising properties [2]:

- First, the average of $\boldsymbol{u}_c$ do not approximate or proportionate to $\boldsymbol{u}_w$, despite their seemingly relationship in the word2vec skipgram Equation 3.14.
- Second, the "calibration" matrix $\boldsymbol{A}$ is the same for all words and context words in a given dataset.

The authors justified the matrix $\boldsymbol{A}$ by proposing a random walk model. Here we provide a new explanation using the connections between knowledge graph embedding and word embedding in Section 4.5.1. Following the above connections, there is an implicit relation that capture the information about co-occurrence of words in the sliding windows. The embedding of this relation is absorbed into the word embeddings.

We note that the Equation 4.22 can be written as:

$$\boldsymbol{v}_w \approx \sum_{c \in \text{context of } w} \boldsymbol{A}\boldsymbol{u}_c, \tag{4.23}$$

where $\boldsymbol{A}$ can be seen as a new relation-based transformation matrix that match the context words to the target words.

About the first property, the average of $\boldsymbol{u}_c$ cannot be used directly to approximate $\boldsymbol{v}_w$ because the relation defined in the post process of computing context-embedding is different from and the relation defined in sampling, weighting, and training of word2vec skipgram and Glove models. The matrix $\boldsymbol{A}$ accounts for the difference between the two relations. About the second property, the relation is fixed on a dataset, thus the matrix $\boldsymbol{A}$ is global for a given dataset. This also suggests that if the relation is defined differently, such as extending the sliding window context or including the position information, the matrix $\boldsymbol{A}$ will be different.

This observation solidifies our ideas that language modeling could be viewed as a knowledge graph embedding task, and the ideas about extensions to other type of relation in textual data.

4.5.3.2 The Hadamard product for edge features in Node2Vec

In the context of network analysis, there are tasks such as edge labeling that require the edge features, which are the features of a pair of nodes. However, network embedding methods such as DeepWalk and Node2vec only learn the node features. A surprisingly effective method to compute the edge features from the node features is to compute the

element-wise Hadamard product between them. However, this technique is not theoretically justified and this is a long-standing question in network embedding [25].

Here we provide an explanation for this technique using the proposed connections in Section 4.5.1. Note that DeepWalk and Node2Vec are graph embedding extensions of word2vec skipgram. The models sample random walks of node sequences and use word2vec skipgram to compute the node embeddings. Following the above connections, there is an implicit relation that capture the information about co-occurrence of nodes on the random walks. The embedding of this relation is absorbed into the node embeddings. The technique using Hadamard product can be seen as approximately reconstructing the implicit relation embedding vector from the node embedding vectors. Formally speaking:

$$\text{Edge features } f(h,t) = \boldsymbol{h} \odot \boldsymbol{t}. \tag{4.24}$$

For simplicity, we first assume the case that each entity has only one embedding vector, similarly to the DistMult embedding model. The edge features between two nodes h and t are:

$$\text{Edge features } f(h,t) = \boldsymbol{h} \odot \boldsymbol{t} \approx Z \oslash \boldsymbol{r}, \tag{4.25}$$

where $\oslash$ denotes element-wise division, such that each entry i of the edge features is

$$f_i = (\boldsymbol{h} \odot \boldsymbol{t})_i = h_i t_i \approx Z / r_i, \tag{4.26}$$

r is the implicit relation, and Z is a global scalar value for a given dataset. The extracted edge features $f(h,t)$ contain information about the implicit relation r between the two nodes h and t, and thus can be used for the tasks on the pair of nodes.

Now we return to the actual case of Node2Vec and DeepWalk as graph embedding extensions of word2vec skipgram, where each entity has two role-based embedding vectors. The edge features that capture the information of the implicit relation between h and t are:

$$\text{Edge features } f(h,t) = \text{concat}\left(\boldsymbol{h} \odot \boldsymbol{t}^{(2)}, \boldsymbol{h}^{(2)} \odot \boldsymbol{t}\right) \approx \text{concat}\left(Z \oslash \boldsymbol{r}, Z \oslash \boldsymbol{r}^{(a)}\right), \tag{4.27}$$

where $\boldsymbol{r}$ and $\boldsymbol{r}^{(a)}$ are the relation embedding vectors of the implicit relation in the dataset.

The Node2Vec edge feature extraction function in Eq. 4.24 is similar to the simple case of single embedding vector in Eq. 4.25. It can be seen as an approximation of Eq. 4.27 because it does not account for the two role-based embedding vectors. To facilitate

further analysis, we assume an arbitrary context node c that appears in the contexts of both h and t. The Node2Vec edge features between h and t can be written as the element-wise extraction from their embedding vectors through their connections to the context-embedding vectors of the context node c. Formally speaking:

$$\begin{aligned} \text{Edge features } f(h,t) &= \boldsymbol{h} \odot \boldsymbol{t} && (4.28)\\ &\approx \left(Z \oslash \left(\boldsymbol{r} \odot \boldsymbol{c}^{(2)}\right)\right) \odot \left(Z \oslash \left(\boldsymbol{c}^{(2)} \odot \boldsymbol{r}^{(a)}\right)\right) && (4.29)\\ &\approx Z^2 \oslash \left(\boldsymbol{r} \odot \boldsymbol{c}^{(2)} \odot \boldsymbol{c}^{(2)} \odot \boldsymbol{r}^{(a)}\right). && (4.30) \end{aligned}$$

Because c is arbitrary, the extracted edge features capture the information about the element-wise product of the relation embedding vectors $\boldsymbol{r} \odot \boldsymbol{r}^{(a)}$ and thus contain information about the implicit relation r between two nodes h and t.

4.6 Summary

4.6.1 Contribution and impact discussions

In general, the main contributions and impacts of our work are as follows.

- We analyze and identify two fundamental complementary aspects in knowledge graph embedding, namely computational efficiency and model expressiveness. We then address both aspects by introducing 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. About the impact, although our work is not the first one to try to trade-off between the computational efficiency and model expressiveness in knowledge graph embedding, most previous works address this problem in a manual or heuristic way. To the best of our knowledge, we are the first to systematically address this problem by proposing the new multi-partition embedding interaction approach, that generalizes previous methods. Therefore, our work potentially present a conclusive hindsight to some recent researches in knowledge graph embedding.

- To realize our approach, we propose the standard multi-partition embedding interaction (MEI) model with block term format, to control the trade-off between computational efficiency and model expressiveness through the partition size, and to learn the interaction mechanisms from data automatically through the local

Tucker core tensors. We empirically show that MEI is efficient and effective, as it can achieve state-of-the-art results using the popular and standard link prediction benchmarks. About the impact, the MEI model presents a combination of advanced tensor representation formats and modern deep learning techniques for knowledge graph embeddings, that can provide advantages over previous models. This approach of combination may potentially be a promising direction for future research in knowledge graph embeddings.

- We theoretically analyze the framework of MEI to explain its intuitions and meanings. In addition, we are the first to formally study the parameter efficiency problem and derive a simple optimal trade-off criterion for the model size of MEI. We apply the theoretical framework of MEI to provide intuitive explanations for the specially designed interaction mechanisms in several previous knowledge graph embedding models. About the impact, MEI is not just a model, but an approach and theoretical framework to knowledge graph embedding. There are many potential variants and extensions that can improve the model, of which some variants have been discussed above. The theoretical framework of MEI may serve to assist in analyzing previous models and may be readily applied to improve them.

- We also draw the connections from knowledge graph embedding to word embeddings and language modeling to provide some new insights and generalizations. About the impact, these connections may potentially benefit the research and development in both domains of representation learning.

In comparison to previous work, the proposed model systematically control the trade-off between computational efficiency and model expressiveness instead of manually designing interaction mechanisms in previous models. The proposed model is highly efficient compared to non-sparse models RESCAL [60] and TuckER [3], and highly expressive compared to previous state-of-the-art trilinear-product-based models ComplEx [80], CP_h [47], and SimplE [41].

In addition, our analysis also provides another framework to analyze knowledge graph embedding models, specifically generalizing trilinear-product-based models, and provide intuitive explanation for their mechanisms and suggestions for their extensions.

4.6.2 Scopes and future work

The proposed MEI framework provides a general approach towards knowledge graph embedding, which can be used to explain some previous embedding models and extend

them to new effective variants, in particular, the recent state-of-the-art trilinear-product-based models. A limitation of MEI is that it does not generalize some other knowledge graph embedding models, such as complicated neural-network-based models. However, complicated models are usually outperformed by MEI and other simpler models on the link prediction for knowledge graph completion task. Further investigation to find more effective but simple embedding models is an important direction for future work.

Another main limitation of the proposed method as well as most previous knowledge graph embedding methods is that they are *transductive learning* methods, that is, they can only learn the embeddings for the entities and relations that are existing in training. Therefore, one big future direction is to extend the proposed method to *inductive learning* approach, where it can compose embedding for new unseen entities and relations.

Chapter 5

Multi-Partition Embedding Interaction: Learning and Evaluation

In this chapter, we evaluate the proposed knowledge graph embedding model using popular benchmarks. We first discuss the learning problem of the model. We then evaluate the performance of the model on the link prediction task, which is the standard benchmark of knowledge graph embedding method and can be seen as a simple data query task with a single predicate.

5.1 Learning problem

We have so far examined in details the theoretical aspects of the MEI model, including its operations, properties, and meanings. In this section, we discuss the problem of learning the parameters in MEI, which includes the embedding vectors and the core tensors.

5.1.1 Learning the interaction patterns

One of the main advantages of MEI compared to previous embedding models is that it can learn the interaction patterns in the matching matrices automatically from data. The core tensors $\mathbf{W}_k$ play an important role in MEI because they define the interaction patterns. In the MEI model, we learn the core tensors together with the embedding vectors.

We treat this learning problem as a neural-network optimization problem by viewing each local MEI model as a dynamic neural network as discussed in Section 4.2.3. This treatment enables us to combine advanced tensor representation formats and modern deep learning optimization and regularization techniques. Specifically, the model is divided into

layers enabling us to use modern weight initialization approaches such as Xavier initialization [23], layer-wise dropout [72] and batch normalization [36] to improve convergence rate and generalization performance. Negative sampling technique [56] [8] [16] enables tractable and effective optimization of the loss function defined over the output layer of the neural network. The learning of the model is in an end-to-end fashion using mini-batch stochastic gradient descent with modern adaptive learning rate algorithms such as Adam [42].

5.1.2 Loss function

The loss function is an important part of a model and usually crucial to its performance. Here we discuss the main loss functions of knowledge graph embedding that we will use in our experiments, including the *binary cross-entropy* and the *full softmax cross-entropy* loss functions.

5.1.2.1 Binary cross-entropy loss

Traditionally, 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 [56], 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) [8]. The set of existent triples is called the true data $\mathcal{D}$, and the set of nonexistent triples is called the negative sampled data $\mathcal{D}'$.

Note that convergence rate and generalization performance depends on the quality of negative samples, which in turn depends on the *noise distribution* $\mathbf{Q}$ used to sample invalid triples. Usually, the *uniform distribution* is used for sampling [8]. Some researches use *uni-gram frequency to the power of* $\frac{3}{4}$ to scale down frequent words such as in word2vec [56]. Some other researches reduce false negative samples by modifying the distribution based on the cardinality of the relation [92].

To construct the binary cross-entropy loss function, we first define a Bernoulli distribution over each entry of the binary data tensor $\mathbf{G}$ to model the existence probability of each triple as

$$\hat{p}_{htr} = g_{htr}, \tag{5.1}$$

where g_{htr} is the entry of the observed knowledge graph data tensor.

The predicted probability of the model is computed by using the standard logistic function on the matching scores as

$$p_{htr} = \sigma(\mathcal{S}(h,t,r;\boldsymbol{\theta})) = \frac{\mathrm{e}^{\mathcal{S}(h,t,r;\boldsymbol{\theta})}}{e^{\mathcal{S}(h,t,r;\boldsymbol{\theta})}+1}. \tag{5.2}$$

We can then learn both the embeddings and the core tensor from data by minimizing the binary cross-entropy loss:

$$\begin{aligned}\mathcal{L}(\mathcal{D},\mathcal{D}';\boldsymbol{\theta}) = \; - \sum_{(h,t,r)\in\mathcal{D}\cup\mathcal{D}'} \big(&\hat{p}_{htr}\log p_{htr} \\ &+(1-\hat{p}_{htr})\log(1-p_{htr})\big),\end{aligned} \tag{5.3}$$

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

More concisely, by defining the class label $\mathrm{Y}_{(h,t,r)} = 2\hat{p}_{(h,t,r)} - 1$, that is, labels of positive triples are 1 and negative triples are -1, this loss can be rewritten in the softplus format as:

$$\mathcal{L}(\mathcal{D},\mathcal{D}';\boldsymbol{\theta}) = \sum_{(h,t,r)\in\mathcal{D}\cup\mathcal{D}'} \log(1+\mathrm{e}^{-\mathrm{Y}_{(h,t,r)}\mathcal{S}(h,t,r;\boldsymbol{\theta})}) \tag{5.4}$$

with $\mathcal{D}$ is true data, $\mathcal{D}'$ is negative sampled data.

5.1.2.2 Full softmax cross-entropy loss

Some recent works have found that the full softmax cross-entropy loss can greatly improve the performance of previous models over the binary cross-entropy loss [47] [12]. In this case, the learning problem in knowledge graph embedding methods can be modeled as the multi-class classification of existent triples among nonexistent triples. Specifically, each existent triple (h,t,r) is classified among all negative-tail triples $\{(h,t',r)|t'\in\mathcal{E}\setminus t\}$ and among all negative-head triples $\{(h',t,r)|h'\in\mathcal{E}\setminus h\}$, which are negative sampled triples constructed by replacing the h or t entities with *all* other entities in the knowledge graph to obtain the locally related nonexistent triples [47] [12]. 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 full softmax cross-entropy loss function, we first define a categorical distribution over each triple (h,t,r) and its negative-tail and negative-head sampled triples to model the existence probability of each triple. There are two approaches to defining this distribution. The first one is called *1-vs-all*, that considers each existent triple (h,t,r)

separately, defines its probability as 1 and all of its negative samples' probabilities as 0.

$$\begin{aligned} \hat{p}_{htr} &= 1, \\ \hat{p}_{ht'r} &= 0, \\ \hat{p}_{h'tr} &= 0. \end{aligned} \tag{5.5}$$

where $(h, t, r) \in \mathcal{D}$, $t' \in \mathcal{E} \setminus t$, and $h' \in \mathcal{E} \setminus h$.

The second approach is called *k-vs-all*, that considers all existent triples that share the entity and relation pair together, that is, in the tail direction with all $\{(h, \dot{t}, r) | \dot{t} \in \mathcal{E}\}$ together, and in the head direction with all $\{(\dot{h}, t, r) | \dot{h} \in \mathcal{E}\}$ together. Note that the probabilities of each triple have different values for the tail direction and the head direction due to different normalization factors.

$$\begin{aligned} \hat{p}_{h\dot{t}r} &= \frac{g_{h\dot{t}r}}{\sum_{\dot{t} \in \mathcal{E}} g_{h\dot{t}r}}, \\ \hat{p}_{\dot{h}tr} &= \frac{g_{\dot{h}tr}}{\sum_{\dot{h} \in \mathcal{E}} g_{\dot{h}tr}}, \end{aligned} \tag{5.6}$$

where g_{htr} is the entry of the observed knowledge graph data tensor, $g_{htr} = 1 \iff (h, t, r) \in \mathcal{D}$ and $g_{htr} = 0$ otherwise.

The predicted probability of the model is computed by using the softmax function on the matching scores. We also compute different probabilities in the tail and in the head directions.

$$\begin{aligned} p_{h\dot{t}r} &= \frac{\mathrm{e}^{\mathcal{S}(h,\dot{t},r;\boldsymbol{\theta})}}{\sum_{\dot{t} \in \mathcal{E}} e^{\mathcal{S}(h,\dot{t},r;\boldsymbol{\theta})}}, \\ p_{\dot{h}tr} &= \frac{\mathrm{e}^{\mathcal{S}(\dot{h},t,r;\boldsymbol{\theta})}}{\sum_{\dot{h} \in \mathcal{E}} e^{\mathcal{S}(\dot{h},t,r;\boldsymbol{\theta})}}, \end{aligned} \tag{5.7}$$

We can then learn both the embeddings and the core tensor from data by minimizing the full softmax cross-entropy loss at both the tail and head directions:

$$\begin{aligned} \mathcal{L}(\mathcal{D}, \mathcal{E}; \boldsymbol{\theta}) = -\sum_{(h,t,r) \in \mathcal{D}} \Big(& \sum_{\dot{t} \in \mathcal{E}} \hat{p}_{h\dot{t}r} \log p_{h\dot{t}r} \\ & + \sum_{\dot{h} \in \mathcal{E}} \hat{p}_{\dot{h}tr} \log p_{\dot{h}tr} \Big). \end{aligned} \tag{5.8}$$

5.1.3 Optimization

The loss function can be optimized by stochastic gradient descent (SGD) on mini-batch data. The implementation is detailed in Section 5.2.2. Instead of using traditional meth-

ods for solving tensor decompositions, we solve it using modern deep learning optimization techniques, which is important because the learning problem is an approximation problem, which is prone to overfitting and requires very good regularization. The blend of traditional mathematical models and modern deep learning optimization techniques is useful in two ways. First, the mathematical models provide solid background and intuition for the embedding models. Second, modern deep learning techniques helps to solve the embedding models to a satisfied level of success, that is unobtainable using traditional methods otherwise.

In particular, we use negative sampling to obtain training data and approximate the true data distribution in training. The loss function is modified for discrete data instead of continuous data in traditional tensor decompositions. The model use *dropout* and *batch normalization* for regularization and improve the dynamics of model training. Training is done by mini-batch stochastic gradient descent using adaptive learning rate optimizer with momentum.

5.2 Link prediction for knowledge graph completion experiments

In this section, we present the experiments and analyses to evaluate the performance and efficiency of MEI on the link prediction task. We describe the experimental settings for the link prediction task on four popular benchmarks. We then present and analyze the results of MEI and other previous state-of-the-art models.

5.2.1 Experimental Settings

Tasks

Link prediction task is the standard and most useful task for knowledge graph embedding methods [8]. 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 [8]. The goal of the model now is to rank the true triple (h, t, r) before the corrupted triples based on the predicted score $\mathcal{S}$.

Datasets

We use four popular benchmark datasets for link prediction, as shown in Table 5.1. WN18 [8] and WN18RR [16] are subsets of WordNet [58], which contains lexical relationships

Table 5.1: Datasets statistics of the link prediction benchmarks.

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

between words. FB15K [8] and FB15K-237 [76] are subsets of Freebase [7], which contains general facts. WN18 and FB15K were popular datasets but have been shown to be suffering from data leakage [76] [16]. WN18RR and FB15K-237 are recent datasets that have the above problem fixed and have become more standard benchmark datasets for this task.

Evaluations

We evaluate and analyze MEI on the link prediction task [8]. 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) [80]. 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 [8]. 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. In addition, we also compare MEI with models that use new expensive settings and techniques, such as TorusE that uses larger embedding size; ComplEx at $K = 400$ that was retuned with N3 weight decay, reciprocal relation, and full softmax loss; and

RotatE that uses larger embedding size 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 [42] with AMSGrad variant [65]. We followed the 1-N scoring procedure in [16] 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 [3]. All hyperparameters in our experiments are tuned by random search [5], including batch size $\in \{32, 64, 128, 512, 1024\}$, learning rate $\in$ {1e-2, 5e-3, 3e-3, 1e-3, 5e-4}, exponential decay rate $\in \{0.99, 0.995, 0.9975, 1.0$ (no learning rate decay)$\}$, dropout rate $\in \{0.0$ (no dropout), $0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7\}$, and batch normalization on each layer or not. Preliminary results show that for all models, the best batch size is 128, the best batch normalization is on $\boldsymbol{h}$ and $\boldsymbol{h}^\top \boldsymbol{M}$. Trainings were early stopped by checking the filter MRR metric on the validation sets. Reported results are the median results regarding filtered MRR on the validation set selected from three independent runs with different random seeds.

Note that in the main experiments to compare with the baselines, we only use the binary cross-entropy loss function to be comparable with previous work. In the new experiments for small MEI model, we use the new full softmax cross-entropy loss function, which have been shown to greatly improve the link prediction performance of previous models.

The best hyperparameters for $\text{MEI}_{1\times200}$ are learning rate 5e-3, exponential decay rate 0.995, drop rates 0.1 on $\boldsymbol{M}$, 0.2 on $\boldsymbol{h}$, 0.2 on $\boldsymbol{h}^\top \boldsymbol{M}$, label smoothing 0.1 on WN18; learning rate 3e-3, exponential decay rate 0.99, drop rates 0.2 on $\boldsymbol{M}$, 0.2 on $\boldsymbol{h}$, 0.3 on $\boldsymbol{h}^\top \boldsymbol{M}$ on FB15K; learning rate 1e-2, drop rates 0.2 on $\boldsymbol{M}$, 0.2 on $\boldsymbol{h}$, 0.3 on $\boldsymbol{h}^\top \boldsymbol{M}$, label smoothing 0.1 on WN18RR; learning rate 5e-4, drop rates 0.4 on $\boldsymbol{M}$, 0.3 on $\boldsymbol{h}$, 0.5 on $\boldsymbol{h}^\top \boldsymbol{M}$, label smoothing 0.1 on FB15K-237.

The best hyperparameters for $\text{MEI}_{3\times100,\text{ shared core}}$ are learning rate 1e-3, drop rates 0.3 on $\boldsymbol{r}$, 0.3 on $\boldsymbol{M}$, 0.5 on $\boldsymbol{h}$, 0.5 on $\boldsymbol{h}^\top \boldsymbol{M}$ on WN18; learning rate 1e-3, exponential decay rate 0.995, drop rates 0.1 on $\boldsymbol{r}$, 0.1 on $\boldsymbol{M}$, 0.3 on $\boldsymbol{h}$, 0.3 on $\boldsymbol{h}^\top \boldsymbol{M}$ on FB15K; learning rate 1e-3, drop rates 0.6 on $\boldsymbol{h}$, 0.6 on $\boldsymbol{h}^\top \boldsymbol{M}$ on WN18RR; learning rate 3e-3, exponential decay rate 0.9975, drop rates 0.65 on $\boldsymbol{h}$, 0.6 on $\boldsymbol{h}^\top \boldsymbol{M}$ on FB15K-237.

The best hyperparameters for $\text{MEI}_{3\times100,\text{ non-shared core}}$ are learning rate 1e-3, drop rates

0.5 on $\boldsymbol{h}$, 0.5 on $\boldsymbol{h}^\top \boldsymbol{M}$ on WN18; learning rate 1e-3, exponential decay rate 0.995, drop rates 0.1 on $\boldsymbol{r}$, 0.1 on $\boldsymbol{M}$, 0.35 on $\boldsymbol{h}$, 0.35 on $\boldsymbol{h}^\top \boldsymbol{M}$ on FB15K; learning rate 1e-3, drop rates 0.65 on $\boldsymbol{h}$, 0.6 on $\boldsymbol{h}^\top \boldsymbol{M}$ on WN18RR; learning rate 3e-3, exponential decay rate 0.9975, drop rates 0.65 on $\boldsymbol{h}$, 0.6 on $\boldsymbol{h}^\top \boldsymbol{M}$ on FB15K-237.

Table 5.2: Link prediction results on WN18 and FB15K. † are reported in [61], ‡ are reported in [80], other results are reported in their papers. Best results are in bold, second-best results are underlined, best baseline results are in italic.

	WN18				**FB15K**			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
TransE [8] †	0.495	0.113	0.888	0.943	0.463	0.297	0.578	0.749
TransR [53] †	0.605	0.335	0.876	0.940	0.346	0.218	0.404	0.582
ER-MLP [17] †	0.712	0.626	0.775	0.863	0.288	0.173	0.317	0.501
R-GCN [67]	0.814	0.686	0.928	0.955	0.651	0.541	0.736	0.825
ConvE [16]	0.943	0.935	0.946	*0.956*	0.657	0.558	0.723	0.831
RESCAL [60] †	0.890	0.842	0.904	0.928	0.354	0.235	0.409	0.587
CP [34] ‡	0.075	0.049	0.080	0.125	0.326	0.219	0.376	0.532
DistMult [95] ‡	0.822	0.728	0.914	0.936	0.654	0.546	0.733	0.824
ComplEx [80]	0.941	0.936	0.945	0.947	0.692	0.599	0.759	0.840
SimplE [41]	0.942	0.939	0.944	0.947	0.727	0.660	*0.773*	0.838
TorusE [18]	*0.947*	*0.943*	*0.950*	0.954	0.733	*0.674*	0.771	0.832
ComplEx new tuning [50]	–	–	–	–	*0.790*	–	–	*0.872*
$\text{MEI}_{1\times200}$	**0.953**	**0.949**	**0.955**	**0.958**	0.795	0.741	0.833	0.892
$\text{MEI}_{3\times100,\text{ shared core}}$	0.950	0.946	0.952	0.957	0.806	0.754	0.843	0.893
$\text{MEI}_{3\times100,\text{ non-shared core}}$	0.950	0.946	0.953	0.956	**0.809**	**0.756**	**0.845**	**0.898**

5.2.2 Main Results

5.2.2.1 Link Prediction Performance Compared to Traditional Baselines

Most previous models reported results using traditional settings and training techniques, specifically the binary cross-entropy loss function. We first report the link prediction performance using the same traditional settings and training techniques for fair comparison write reported baselines.

Tables 5.2 and 5.3 show the main results of the MEI model on comparable model sizes and experimental settings with the baseline models. In general, MEI strongly outperforms the baselines, including translation-based models, neural-network-based models, and semantic matching models. 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.

Table 5.3: Link prediction results on WN18RR and FB15K-237. † are reported in [19], ‡ are reported in [16], other results are reported in their papers. Best results are in bold, second-best results are underlined, best baseline results are in italic.

	WN18RR				**FB15K-237**			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
TransE [8] †	0.182	0.027	0.295	0.444	0.257	0.174	0.284	0.420
R-GCN [67]	–	–	–	–	0.248	0.153	0.258	0.414
ConvE [16]	0.43	0.40	0.44	*0.52*	*0.325*	*0.237*	*0.356*	*0.501*
DistMult [95] ‡	0.43	0.39	0.44	0.49	0.241	0.155	0.263	0.419
ComplEx [80] ‡	0.44	0.41	0.46	0.51	0.247	0.158	0.275	0.428
TorusE [19]	*0.452*	*0.422*	*0.464*	0.512	0.305	0.217	0.335	0.484
RotatE w/o adv [73]	–	–	–	–	0.297	0.205	0.328	0.480
$\text{MEI}_{1\times200}$	**0.470**	**0.443**	**0.482**	**0.526**	0.358	<u>0.266</u>	0.394	<u>0.544</u>
$\text{MEI}_{3\times100,\text{ shared core}}$	<u>0.458</u>	<u>0.426</u>	<u>0.470</u>	<u>0.521</u>	<u>0.359</u>	<u>0.266</u>	<u>0.395</u>	<u>0.544</u>
$\text{MEI}_{3\times100,\text{ non-shared core}}$	0.457	0.422	<u>0.470</u>	<u>0.521</u>	**0.361**	**0.269**	**0.397**	**0.545**

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 4.3.1.

Compared with TorusE, which uses a simple and efficient interaction mechanism, enabling it to use very large embedding size $D = 10000$, MEI can still achieve better results across all datasets and metrics. These 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 ComplEx and RotatE that use new and expensive techniques. Moreover, note that MEI is highly general compared to previous models and potentially preferable for more sophisticated datasets and difficult benchmarks.

5.2.2.2 Link Prediction Performance of Small Models with Modern Training Techniques

In real-world applications, the knowledge graphs are usually very large compared to the benchmark datasets, with millions to billions of entities. To make training and inference feasible, practical embedding models need to use relatively small embedding sizes compared to the data sizes. Here we simulate such scenarios on the benchmark datasets by restricting the embedding size to a small value $D = 100$. The goal is to examine the state-of-the-art performance of relatively small embedding models on large data.

Small model sizes enable us to expand our hyperparameter search space and tune

Table 5.4: Link prediction results of small $\text{MEI}_{10\times10}$ models variants tuned with recent training techniques. ComplEx results were tuned by [47] and reported at their github page[1], representing previous state-of-the-art results of small models. RotatE results were reported in [73], provided as a reference for recent state-of-the-art results of large models. Best results of small models are in bold. Best results of large models are in bold and italicized if they are best overall results.

		Param. count	MRR	H@1	H@3	H@10
WN18	$\text{RotatE}_{500\times2}$	40.961M	0.949	0.944	0.952	0.959
	$\text{ComplEx}_{50\times2}$	4.098M	0.950	0.940	0.950	0.950
	$\text{MEI}_{10\times10,\text{ shared core}}$	4.099M	0.950	0.945	**0.953**	0.957
	$\text{MEI}_{10\times10,\text{ non-shared core}}$	4.108M	0.950	**0.946**	0.952	0.956
	$\text{MEI}_{10\times10,\text{ non-shared core, orthogonal}}$	4.108M	**0.951**	**0.946**	**0.953**	**0.960**
FB15K	$\text{RotatE}_{1000\times2}$	32.592M	0.797	0.746	***0.830***	***0.884***
	$\text{ComplEx}_{50\times2}$	1.630M	0.780	0.730	0.810	0.860
	$\text{MEI}_{10\times10,\text{ shared core}}$	1.631M	0.790	0.746	0.817	0.870
	$\text{MEI}_{10\times10,\text{ non-shared core}}$	1.640M	0.798	**0.757**	0.820	0.874
	$\text{MEI}_{10\times10,\text{ non-shared core, orthogonal}}$	1.640M	**0.800**	**0.757**	**0.823**	**0.878**
WN18RR	$\text{RotatE}_{500\times2}$	40.954M	0.476	0.428	0.492	***0.571***
	$\text{ComplEx}_{50\times2}$	4.097M	0.460	0.430	0.470	0.520
	$\text{MEI}_{10\times10,\text{ shared core}}$	4.098M	0.468	0.434	0.482	0.531
	$\text{MEI}_{10\times10,\text{ non-shared core}}$	4.107M	0.477	0.442	0.489	0.543
	$\text{MEI}_{10\times10,\text{ non-shared core, orthogonal}}$	4.107M	**0.481**	**0.446**	**0.494**	**0.550**
FB15K-237	$\text{RotatE}_{1000\times2}$	29.556M	0.338	0.241	0.375	***0.533***
	$\text{ComplEx}_{50\times2}$	1.502M	0.340	0.250	0.370	0.520
	$\text{MEI}_{10\times10,\text{ shared core}}$	1.503M	0.347	0.256	0.380	0.531
	$\text{MEI}_{10\times10,\text{ non-shared core}}$	1.512M	0.349	0.257	0.383	**0.533**
	$\text{MEI}_{10\times10,\text{ non-shared core, orthogonal}}$	1.512M	**0.350**	**0.258**	**0.385**	**0.533**

[1] `https://github.com/facebookresearch/kbc`

more extensively the loss function, optimizer, and regularization on feasible computational resource. The small model sizes also enable us to use more advanced recent training techniques, including the full softmax cross-entropy loss function, which have been found to greatly improve link prediction performance in some recent works [47] [12]. This loss function has not been traditionally used by most previous baseline work, including our reported results of MEI in Section 5.2.2.

The previous state-of-the-art results for small model ($D = 100$) were set by $\text{ComplEx}_{50\times2}$ model, tuned by [47]. They used recent training techniques, notably the new full softmax cross-entropy loss function, inverse relation, and L_3 weight decay. We will report their

results as the baselines. We also report the results of RotatE [73], provided as a reference for recent state-of-the-art results of large models. We use an improved model variant of MEI with small embedding size $\text{MEI}_{10\times10}$, which makes the model sizes of MEI roughly the same to the baseline $\text{ComplEx}_{50\times2}$ model.

Table 5.4 shows the link prediction performance of small $\text{MEI}_{10\times10}$ models, tuned with recent training techniques on the four standard benchmark datasets. We first notice that the results of the small MEI models improve significantly thanks to the recent training techniques and our new tuning, especially the results on WN18 and WN18RR, which outperforms larger MEI models trained with traditional techniques. The $\text{MEI}_{10\times10}$ models strongly outperform $\text{ComplEx}_{50\times2}$ models on equivalent settings and set new state-of-the-art performance record for small models with embedding size $D = 100$ on the four benchmark datasets.

Notably, the small $\text{MEI}_{10\times10}$ models can even outperform the recent state-of-the-art results of large RotatE models on most metrics, using only a fraction of the parameter counts. These results support our argument that MEI models with larger partition sizes are more expressive than previous embedding models using fixed smaller partition sizes. This is in line with a recent group-theoretic analysis [10] showing some examples of limitations of RotatE, which our MEI model provides a general framework to systematically address. In general, the results demonstrate one of the key advantages of the MEI model, that is, being both efficient and expressive.

5.2.2.3 Model Constraints and Variants

Tables 5.2, and 5.3 also show the results of different model constraints and variants, including shared core and non-shared core tensor, for $\text{MEI}_{3\times100}$ with traditional training techniques. On simple datasets, WN18 with only 18 relations and WN18RR with only 11 relations, the performance of non-shared core and shared core variants are mostly close to each other with some small differences. On more difficult datasets, FB15K and FB15K-237, the non-shared core variant usually has slightly better results. The reason is probably that on the simpler datasets, the model is more easily to get overfitting and the shared core constraint is beneficial. On more difficult datasets, the model may be more difficult to get overfitting; therefore, non-shared core has an advantage because the shared core constraint decreases the capacity of the model. These results suggest that shared core tensor is a simple yet effective regularization constraint for MEI, that reduces the model size and may even achieve better results.

Table 5.4 shows the results of different model constraints and variants for $\text{MEI}_{10\times10}$

with recent advanced training techniques. For shared core and non-shared core tensor variants, the differences in performance become more notable, with non-shared core variants usually achieve better results. However, the shared core tensor variant still performs comparably and slightly better on WN18.

For the orthogonality variants, the results on WN18 and WN18RR significantly improve, especially on the H@10 metric, probably because the orthogonality constraint is suitable for the WordNet knowledge graph. On FB15K and FB15K-237, the orthogonality variants decrease the link prediction results, probably because the orthogonality constraint is not suitable for the FreeBase knowledge graph.

5.2.2.4 Optimal Parameter Efficiency

Empirical results agree with the predictions of Theorem 4.1 on the optimal parameter efficiency. On WN18 and WN18RR, $\text{MEI}_{1\times200}$ is closer to the optimal partition size than $\text{MEI}_{3\times100}$ and the latter has more parameters. On FB15K and FB15K-237, the relative model sizes are reversed due to different numbers of entities and relations. $\text{MEI}_{1\times200}$ has two times more parameters than $\text{MEI}_{3\times100}$ but the latter is closer to the optimal partition size.

On WN18 and WN18RR, $\text{MEI}_{1\times200}$ consistently outperforms $\text{MEI}_{3\times100}$ using fewer parameters as predicted. On FB15K, $\text{MEI}_{3\times100}$ consistently outperforms $\text{MEI}_{1\times200}$ as predicted. On FB15K-237, $\text{MEI}_{3\times100}$ outperforms $\text{MEI}_{1\times200}$ most of the time, although not by a large margin, 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 $K = 1$.

5.2.3 Analyses

5.2.3.1 Parameter Scale Comparison

Table 5.5 compares the performance of MEI with that of ConvE [16], 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.

Table 5.5: Parameter scaling on FB15K-237.

Model	Param. count	Emb. size	MRR	H@ 1	H@ 3	H@ 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 5.6: Parameter trade-off analysis on FB15K-237.

Emb. size	Param. count	$\mathbf{W}$ 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

5.2.3.2 Parameter Trade-off Analysis

There are two kinds of parameters in the MEI model, the embeddings and the core tensors. Theorem 4.1 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 5.6 shows that the models with larger core tensor consistently achieve better results with even fewer total parameters, once again agreeing with Theorem 4.1. Interestingly, MEI with $K = 3$ achieves competitive results compared with MEI with $K = 1$, which suggest that the ensemble boosting effect provides additional advantages for MEI with $K > 1$, as we argued.

5.2.3.3 The Effects of Hyperparameters

About the the effects of hyperparameters, we observed that the full softmax cross-entropy loss function significantly improves the link prediction performance compared to the traditionally used binary cross-entropy loss function, confirming observations in recent works

[47] [12]. We also note that standard deep learning techniques such as dropout [72] and batch normalization [36] played an important role in both the convergence rate and the final results of our model. The inverse relation heuristics [47] helped to much improve convergence rate in training but did not affect final results much. The *k-vs-all* negative sampling procedure [16] that we adopted also helped to speed up each epoch training and improved final results. Interestingly, small mini-batch was important in training using binary cross-entropy loss as it helped the model converge much faster and tended to achieve higher results than with large batch size in general. Small mini-batch also achieved better results when training small models using the full softmax cross-entropy loss. However, large mini-batch achieved better results when training large models using the full softmax cross-entropy loss.

5.3 Summary

In this chapter, we present the learning problem and empirical evaluation of the MEI model. We first discussed how to learn the embeddings and the core tensors together by treating the problem as neural network optimization and using modern deep learning optimization techniques. We then showed that MEI model and its variants can achieve state-of-the-art results on the link prediction task using popular benchmarks. Especially, MEI outperforms previous state-of-the-art models using less parameters, which demonstrates the key advantage of MEI, that is, being both efficient and expressive.

Further analyses showed that MEI provides good parameter efficiency in comparision to neural-network-based models. We also showed that the optimal parameter efficiency is a simple yet effective criterion to choose model size configurations for MEI. The extending variants of MEI are also shown to be effective and help the model to further improve performance.

Chapter 6

Multi-Relational Embedding: Applications

In this chapter, we study and present various practical applications of knowledge graph embeddings, towards efficient multi-relational data analysis using semantic queries on knowledge graph embedding space. We first formalize a framework for data visualization, browsing, and querying applications using semantic queries, which are multi-linear algebraic operations on the embedding space and demonstrate some tasks on scholarly data. We also review the entity analogy reasoning in multi-relational embedding space task, study the semantic structures in the knowledge graph embedding space, and outline potential solution to the above task.

6.1 Motivation

In recent years, digital libraries have moved towards open science and open access with several large scholarly datasets being constructed. Most popular datasets such as MAG[1] [70] and CORE[2] [45] include millions of papers, authors, venues, and other information. Their large size and heterogeneous contents make it very challenging to effectively manage, explore, and utilize these datasets.

These bibliographic datasets contain multi-relational information and can be efficiently represented in a knowledge graphs format. The main part of a knowledge graph is 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. This can also be formalized as a labeled directed multi-

[1] Microsoft Academic Graph: `https://academic.microsoft.com/`
[2] Open access publications: `https://core.ac.uk/`

graph where each triple (h, t, r) represents a directed edge from node h to node t with label r. Therefore, it is straightforward to build knowledge graphs for scholarly data by representing natural connections between scholarly entities with triples such as *(AuthorA, Paper1, write)* and *(Paper1, Paper2, cite)*. There have been several attempts at building and using knowledge graphs for scholarly data [90] [66] [83].

Another advantage of knowledge graph is that, it enables the application of knowledge graph embedding methods to model the dataset as a low dimensional semantic space and operations on this space. Knowledge graph embedding is an emerging research topic with various new and effective methods [89] [77]. These methods were originally developed to solve the link prediction for knowledge graph completion task, but they also provide the embedding representations of the data. The data representations may enable new, more efficient, and effective data representation and analysis applications.

In the case of word embedding methods such as word2vec, embedding vectors are known to contain rich semantic information that enables them to be used in many semantic applications [57]. However, knowledge graph embedding vectors are usually only used for the inherent task of knowledge graph completion, but not for semantic applications. One of the reasons is that the semantic structures in knowledge graph embedding is not well-understood because of the vast diversity of the interaction mechanisms in knowledge graph embedding methods. Therefore, the knowledge graph embedding space remains absent in the toolbox for data representation and analysis, although they have the potential to enable very effective and efficient application. In this chapter, we address these issues by providing a theoretical understanding of knowledge graph embedding space and proposing a general framework for their applications.

We first try to formalize a general framework for multi-relational data exploration and analysis using semantic queries on knowledge graph embedding space. The main component in this framework is the conversion templates from data exploration and analysis tasks on the original data to *semantic queries*, which are the multi-linear algebraic operations between the embedding vectors, that exploits the semantic structures of the embedding space to solve queries such as similarity query and relational query. For example, the framework can solve the related paper recommendation task by running the *semantic similarity query* between the paper entity embedding on a bibliographic knowledge graph embedding space. We then build a scholarly knowledge graph and demonstrate how some important representation and analysis tasks on the original data can be efficiently approximated by semantic queries.

We also review the entity analogy reasoning on multi-relational embedding space task,

which can be seen as an *open-relational query* by examples task. Towards solving this task, we study the semantic structures in the knowledge graph embedding space. Based on the discussed connections between knowledge graph embedding methods and language modeling, as discussed in Section 4.5.1, we propose a general semantic analogy structure that extend the simple semantic direction structure in word2vec embedding space, namely $king - man = queen - woman$, to multi-relational embedding space. We then outline a potential solution to the above task.

6.2 Semantic query on knowledge graph embedding space

In this section, we define semantic query and formalize the semantic query framework on knowledge graph embedding space.

6.2.1 Semantic structure

When an embedding model is trained on a dataset, semantic information in that dataset, such as similarity or analogy relationships, may be encoded by the resulting embedding space. A structure on the embedding space can be seen as an operation, relation, or metric on the embedding vectors. The structure is a semantic structure if it can be assigned with a semantic meaning, such as similarity or analogy.

Definition 6.1. ***(Semantic structure)*** A semantic structure on an embedding space is a mathematical structure (such as operation, relation, or metric) on the embedding vectors that was assigned with a semantic meaning.

The advantage of such semantic structures is that we can use the mathematical operations on the embedding space to represent and model some semantic relationships on the original dataset. For example, the *semantic similarity structure* on the entity embedding vectors represents the similarity relationship between the entities. In regard to the MEI model and most other embedding model, the matching between embedding vectors is simply by a dot product. Therefore, we can define the semantic similarity structure between two entity a and b as:

$$sim(a, b) = \boldsymbol{a}^\top \boldsymbol{b}. \tag{6.1}$$

6.2.2 Semantic queries

We define the semantic query and semantic query on a knowledge graph embedding space as following.

Definition 6.2. ***(Semantic query)*** A semantic query is a precise relational-type operation that explicitly uses the relational information to perform a data representation and analysis task.

Definition 6.3. ***(Semantic query on the knowledge graph embedding space)*** A semantic query on the knowledge graph embedding space is defined as the multi-linear algebraic operations on that knowledge graph embedding space to approximate a given data representation and analysis task.

For example, data visualization, similarity query, and question answering are some important tasks that can be solved by semantic queries. Semantic structures can be seen as more basic building blocks that can be used for specific operations in a semantic query. We will discuss them in details in Section 6.2.4.

6.2.3 The semantic query framework

To facilitate the application of multi-relational embedding in data representation and analysis tasks, we formalize the steps of semantic queries in a semantic query framework [78]. Figure 6.1 illustrates the overall architecture of the proposed framework using the symbols loosely based on Yourdon and Coad's data flow diagram convention. There are three main components, namely *data processing*, *task processing*, and *query processing*.

Component 1: Data processing include two steps, *constructing the knowledge graph from multi-relational data* and *learning the knowledge graph embeddings*. These can be done only once for a scholarly dataset as the resulted knowledge graph embeddings are reused for multiple queries.

- *Constructing the knowledge graph for multi-relational data:* we can build the knowledge graph by directly using entities and relations in the multi-relational dataset. For example, on a scholarly dataset such as Microsoft Academic Graph (MAG), the entities mainly include *authors*, *papers*, *venues*; the relations mainly include *author-write-paper*, *paper-cite-paper*, *paper-publish-in-venue*. Note that the knowledge graph can be extended by including other natural or augmented nodes and edges to extensively integrate information from the original data.

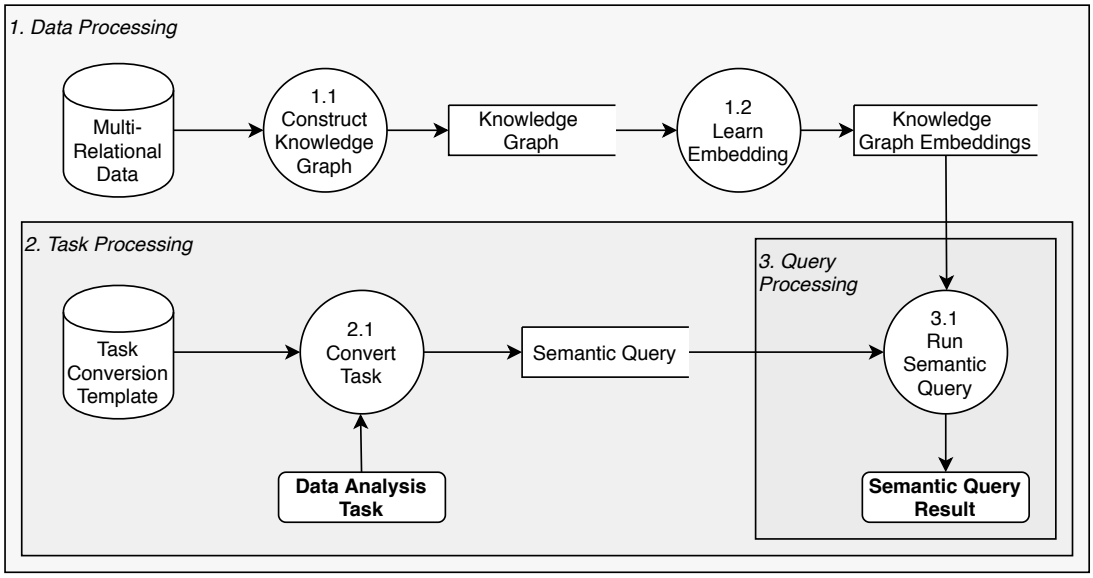

Figure 6.1: Architecture of the semantic query framework. Three main components include Data Processing, Task Processing, and Query Processing. Notations based on Yourdon and Coad's data flow diagram convention, with circle denoting process, cylinder denoting database, open rectangle denoting data store, rectangle denoting external input and output.

- *Learning the knowledge graph embeddings:* we learn the embedding vectors by knowledge graph completion. The procedure can be found in [47] and [77].

Component 2: Task processing converting data analysis tasks to algebraic operations on the embedding space. The *task conversion* can be done by following some templates for specific tasks. Some important tasks and their conversion templates are discussed in Section 6.2.4.

Component 3: Query processing executing semantic query on the embedding space and return results. Note that the algebraic operations on embedding vectors are linear and can be performed in parallel. Therefore, the *semantic query* is very efficient.

Note that the proposed semantic query framework aims to be general. It makes no assumption on the specific knowledge graph embedding models and the induced embedding spaces. Any embedding space that contains rich semantic information can be applied in this framework with an appropriate query processing method. We study some typical tasks and their corresponding semantic queries below.

6.2.4 Representation and Analysis Tasks

In the following, we present some specific semantic query tasks. Based on the analysis in Section 6.4.2, we are interested in three main tasks on the knowledge graph embedding space, (1) data visualization, (2) similarity query, and (3) relational query. For each task, we propose a conversion template for converting it to appropriate multi-linear algebraic operations on the embedding space to perform semantic query.

6.2.4.1 Task 1: Data visualization

Tasks Given a multi-relational dataset, visualize its entities and identify semantic clusters of the entities.

Examples This task can give an intuitive overview of a multi-relational dataset such as the bibliographic data or biomedical data. The semantic clusters of the entities give hints about the potential interesting information and can be used for further processing.

Solution We can solve this task by first learning the multi-relational embedding of the dataset, for example using a knowledge graph embedding method such as MEI. After that, we can use a dimension reduction method such as PCA or t-SNE to reduce the dimension of the entity embeddings to 2 or 3-dimensional space. Assuming the embedding space captures some semantic information in the dataset, it is expected that semantically related entities will be close to each other in the embedding space and the visualization. Moreover, following the analysis of the semantic structures in knowledge graph embedding space, entities with different types will form large semantic clusters corresponding to the entity types, which means the embedding space provide some concrete semantic information about entity types. However, this may be an advantage in some applications and a disadvantage in some other applications. In the latter case, we need to use more advanced semantic queries, that is, multi-linear algebraic operations, to overcome the separation of embedding by entity types.

6.2.4.2 Task 2: Similarity query

Tasks Given an entity e, find other entities that are similar to e.

Examples In a bibliographic dataset, find papers that are related to a given paper based on its references and citations. In a COVID-19 biomedical network, find drugs that are similar to a given drug based on its chemical interactions.

Solution We first note that this task only make sense when querying in the same entity type. For example, a paper is not similar to some authors directly, but a paper's author can be similar to some authors. Thus, when querying, we need to restrict the type of the candidate entities.

We can solve this task by finding the entities with the top largest similarity to the given entity e as measure by the semantic similarity structure in Eq. 6.1. The most similar entity to e is:

$$\text{Result} = \arg\max_{e_i \in \mathcal{E} \setminus e} sim(\boldsymbol{e}_i, \boldsymbol{e}) \tag{6.2}$$

$$= \arg\max_{e_i \in \mathcal{E} \setminus e} \boldsymbol{e}_i^\top \boldsymbol{e}. \tag{6.3}$$

6.2.4.3 Task 3: Relational query

Tasks Given an entity e and relation r, find the entities that are related to e through r.

Examples In a bibliographic dataset, find papers that are potentially cited by an author. In this case, the entity e is the *author*, the relation r is *citing*. In a COVID-19 biomedical network, find all drugs that can potentially act as inhibitor of a specific enzyme. In this case, the entity e is the *enzyme*, the relation r is *being inhibited*.

Solution To solve this task, we notice that when the entity e and the relation r exist in the training data, even if there is no triple that contains e and r, we can still treat it as a link prediction task on the embedding space. Thus, when querying, we need to restrict the type of the candidate entities.

We can solve this task by transforming the entity embedding $\boldsymbol{e}$ using the relation-based transformation $T_r(\cdot)$, then finding most similar entities as measure by the semantic similarity structure in Eq. 6.1. The most related entity to e through r is:

$$\text{Result} = \arg\max_{e_i \in \mathcal{E}} sim(\boldsymbol{e}_i, T_r(\boldsymbol{e})) \tag{6.4}$$

$$= \arg\max_{e_i \in \mathcal{E}} \boldsymbol{e}_i^\top T_r(\boldsymbol{e}), \tag{6.5}$$

where $T_r(\cdot)$ is the relation-based transformation depending on the specific embedding space.

6.3 Experiments

In this section, we present how we implemented the semantic query framework and conducted the experiments for the three discussed tasks, (1) data visualization, (2) similarity query, and (3) relational query.

6.3.1 Experiment settings

Here we will describe the procedure to implement the semantic query framework, including the source datasets, their multi-relational data format, and the embedding learning.

Data For experiments, we use a popular bibliographic dataset to construct a scholarly knowledge graph.

Bibliographic dataset We use a subset of the popular MAG dataset[3] [70], which is one of the largest and most complete bibliographic datasets. We constructed a curated subset of influential computer science papers published in top conferences between 1990 and 2010. The top conference list are based on the 2018 CORE ranking[4] A* conferences.

We removed conferences with less than 300 publications because they are likely incomplete data. We also remove papers with less than 20 citations recorded in MAG. The final bibliographic dataset, namely MAG20C, includes papers from 20 top conferences, sorted alphabetically, including AAAI, AAMAS, ACL, CHI, COLT, DCC, EC, FOCS, ICCV, ICDE, ICDM, ICML, ICSE, IJCAI, NIPS, SIGGRAPH, SIGIR, SIGMOD, UAI, and WWW. Data statistics of MAG and the curated subset are shown in Table 6.1.

Table 6.1: Data statistics of MAG and the curated bibliographic dataset.

Dataset	Paper	Author	Affiliation	Conference	Journal	Domain	Year
MAG	123,056,983	114,698,044	19,843	1,283	23,404	53,834	1800–2017
MAG20C	5,047	8,680	692	20	0	1,923	1990–2010

Knowledge graph construction We represent the bibliographic dataset in a knowledge graph format by defining the entities, the relations, and constructing the triples. We use five intrinsic *entity types* including *Paper*, *Author*, *Affiliation*, *Venue*, and *Domain*. We also use five intrinsic *relation types* between the entities including *author_in_affiliation*, *author_write_paper*, *paper_in_domain*, *paper_cite_paper*, and *paper_in_venue*.

[3] Microsoft Academic Graph: `https://academic.microsoft.com/`
[4] `http://portal.core.edu.au/conf-ranks/`

We then split the triple dataset uniformly at random into the training, valid, and test sets. We made sure that no triple is duplicate in these splits. The five relation types are intrinsic, so that no relation is directly implied by another relation. Hence, the data are not redundant, that is, no triple can be easily implied by other triples. This property is similar to how the recent standard benchmark datasets WN18RR and FB15K-237 for link prediction are constructed, which constitutes a difficult benchmark. We also made sure that all existing entities and relations appear in the training set so that their embeddings can be learned in training and used in evaluation. The statistics of the resulted knowledge graph, namely KG20C, are shown in Table 6.2.

Table 6.2: Data statistics of the KG20C knowledge graph.

Dataset	$\|\mathcal{E}\|$	$\|\mathcal{R}\|$	Training	Validation	Test
KG20C	16,362	5	48,213	3,670	3,724

Learning the knowledge graph embedding We learn the embedding space using two knowledge graph embedding methods, our MEI model and the CP_h model as a multi-relational embedding baseline. We also learn the embedding space using word2vec skip-gram as a single-relational embedding baseline.

Similarly to the experiments in Chapter 5, we trained the models using mini-batch stochastic gradient descent with Adam optimizer [42]. We use the full softmax cross-entropy loss and try both *1-vs-all* and *k-vs-all* negative sampling [16] [47] [12]. For simplicity, after preliminary experiments, we fix the batch size at 128, learning rate at 1e-3, batch normalization on $\boldsymbol{h}$ and $\boldsymbol{h}^\top \boldsymbol{M}$. We also fix the embedding size of MEI to 10×10, and CP_h and word2vec skipgram to 50×2 so that they have similar model sizes and embedding sizes. These embedding sizes are relatively small compared to the size of the dataset, however, they enable faster training and evaluating and thus more extensive hyperparameter tuning. The small embedding sizes also make our analyses and demonstrations more tractable. Note that we do not use a larger partition size for MEI because larger partitions will result in larger number of parameters in the core tensors and thus smaller embedding size to keep the model size unchanged. In our experiments, we want to keep both the model sizes and embedding sizes of MEI, CP_h, and word2vec skipgram at similar values.

All other hyperparameters in both models are tuned by random search [5], including L^3 regularization $\in$ {1, 3e-1,1e-1, 3e-2,1e-2, 3e-3,1e-3, 3e-4,1e-4, 0.0 (no weight decay)} and dropout rate $\in \{0.0 \text{ (no dropout)}, 0.05, 0.1, 0.2, 0.3, 0.4\}$. The best hyperparameters for

word2vec are full softmax cross-entropy loss with *k-vs-all* negative sampling, L^3 weight decay with strength 3e-1; for CP_h are full softmax cross-entropy loss with *k-vs-all* negative sampling, L^3 weight decay with strength 1e-1; and for MEI are full softmax cross-entropy loss with *k-vs-all* negative sampling, non-shared core tensor, drop rates 0.4 on $\boldsymbol{h}$, 0.4 on $\boldsymbol{h}^\top \boldsymbol{M}$, and no weight decay.

Trainings were early stopped by checking the filter MRR metric on the validation sets and the embedding spaces at the best epoch are used. Reported results are the median results regarding filtered MRR on the validation set selected from three independent runs with different random seeds.

6.3.2 Experimental results and discussion

6.3.2.1 Task 1: Data visualization

In this task, we will visualize the semantic cluster of the entity types on each embedding space learned by word2vec, CP_h, and MEI. This task will provide an overview of the embedding space, its basic semantic structures, and some comparisons between the embedding methods. In the embedding space, the embedding vectors may form clusters. A natural question is what these clusters stand for, or in other words, whether there is any semantic information represented by these clusters. The following results try to answer this question is some extent.

Semantic clusters of entity types To visualize the embedding space, we need to reduce its dimension, for example by using principal component analysis (PCA). However, PCA is a linear method and cannot visualize complex structures. Therefore, we also use UMAP [55], which is a non-linear dimension reduction and visualization method based on k-NN. Figures 6.2 and 6.3 show the 2-dimensional projections using PCA and UMAP of the entity embeddings obtained by random vectors (for simple baseline reference), word2vec, CP_h, and MEI, respectively.

Between the PCA visualizations, we see that word2vec can capture the entity information to some extents. Word2vec does this by using the co-occurrence information of the entities, not the relations between them. CP_h can more strongly separate the entities of different types because it can infer the entity types using the relations. Surprisingly, MEI seems not able to separate the entity types in the PCA visualization. To further investigate this problem, we look at the UMAP visualizations.

We observe that UMAP can more clearly visualize the structure of the embedding spaces. The general results are similar to what we saw in PCA visualization, word2vec

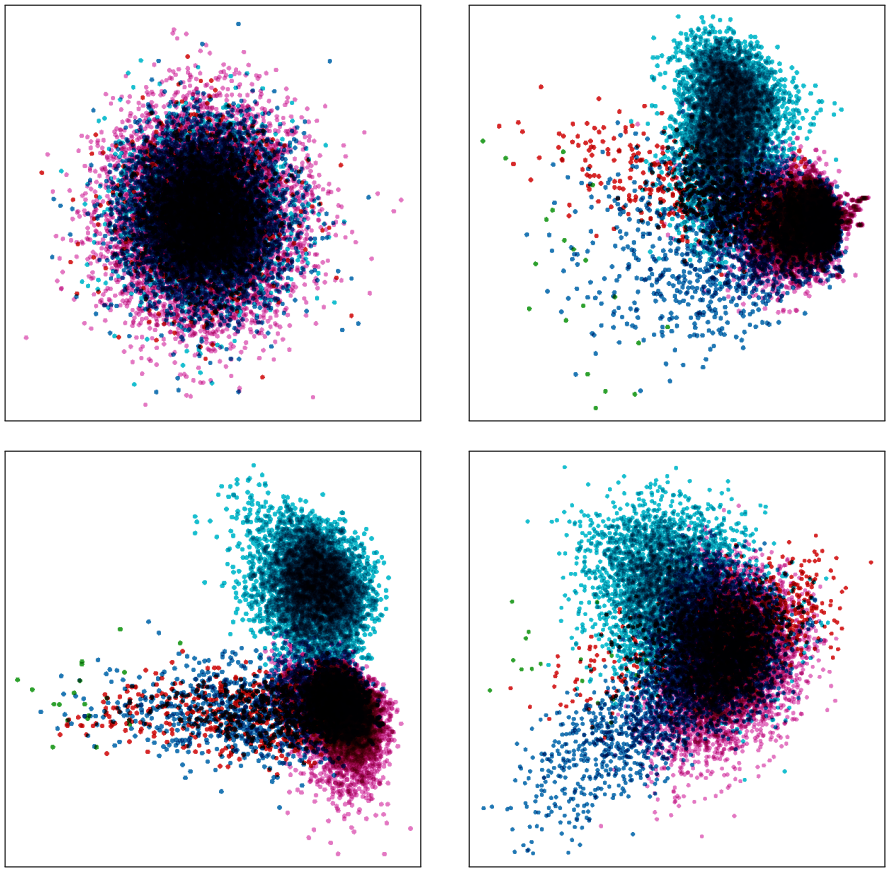

Figure 6.2: PCA visualization of embedding spaces obtained by *random vectors* (top–left), *word2vec* (top–right), CP_h (bottom–left), and MEI (bottom–right), with authors, papers, domains, affiliations, and conferences.

can separate the main entity types including paper and author, but fails to separate other entity types. CP_h can more strongly separate the entity types. Especially, MEI can clearly separate the entity types almost perfectly.

In *Task 3 relational query* we will quantitatively show that the ability to capture and encode the entity type information is an important advantage of multi-relational embedding methods.

6.3.2.2 Task 2: Similarity query

In this task, we will present the case study of similarity query between conferences on the embedding spaces.

Case study: conference similarity query We consider the 20 top conferences in the KG20C dataset. We will measure the similarity between each of them in the embed-

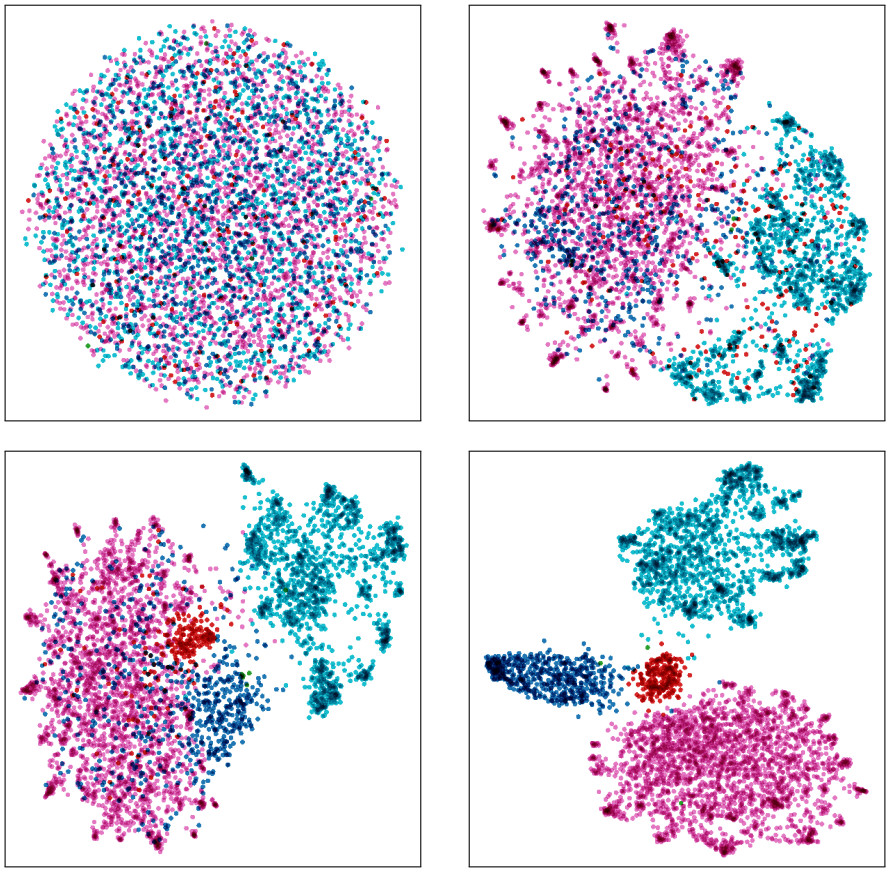

Figure 6.3: UMAP visualization of embedding spaces obtained by *random vectors* (top–left), *word2vec* (top–right), CP_h (bottom–left), and MEI (bottom–right), with authors, papers, domains, affiliations, and conferences.

ding spaces obtained by word2vec, CP_h, and MEI, by the semantic similarity structure defined in Eq. 6.1. We can expect that semantically similar conferences will be close to each other and vice versa. Figures 6.4, 6.5, and 6.6 visualize the similarity matrices between every pair of conferences on the embedding spaces obtained by word2vec, CP_h, and MEI, respectively. The similarities in each row are normalized to L_1 unit norm.

We first observe that the similarities on the word2vec embedding space are mostly lower than those on the CP_h and MEI embedding spaces, as shown in generally darker color squares. This is expected because multi-relational embedding can more easily recognize that the conferences are of the same type and similar to each other. The similarities results on CP_h and MEI are mostly similar and manually inspection shows that their results agree to intuition.

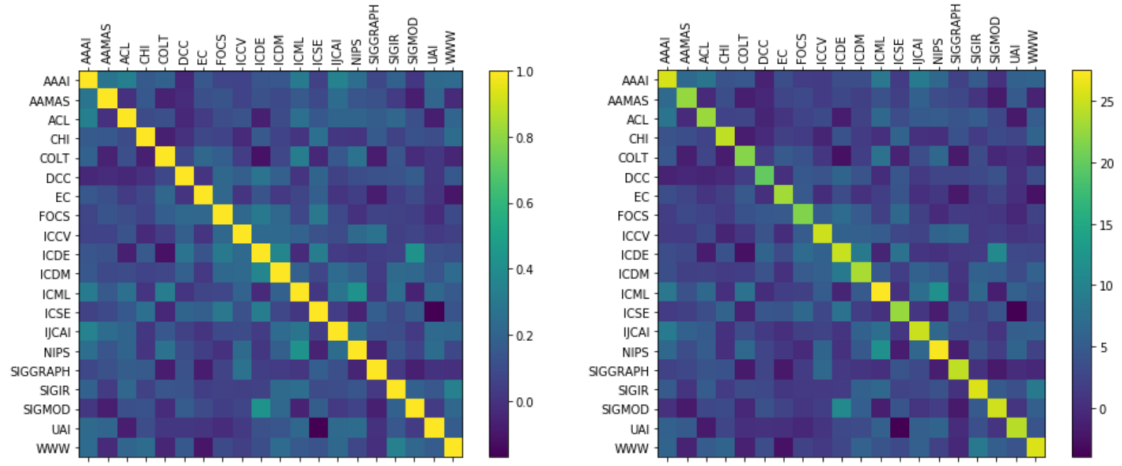

Figure 6.4: Similarity matrix of conferences based on word2vec embeddings, computed using cosine (left) and dot product (right).

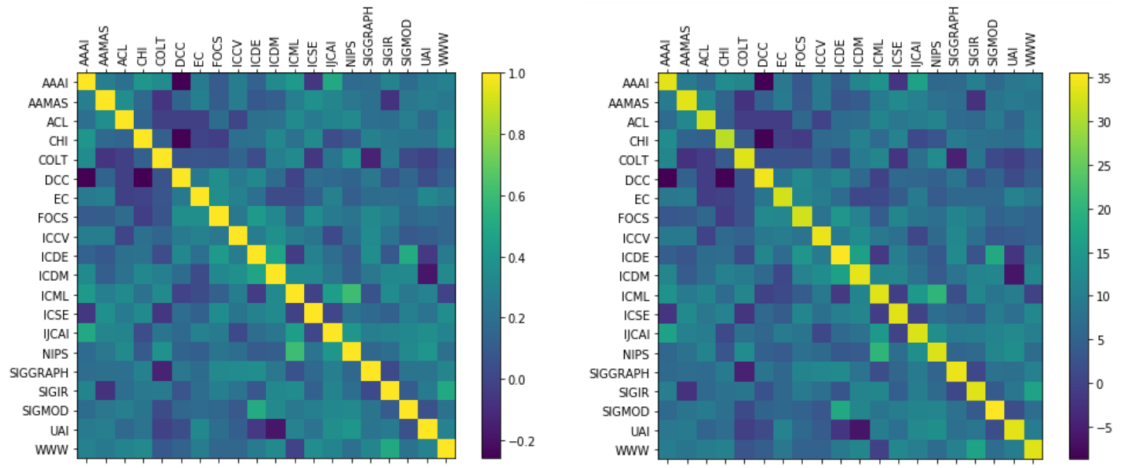

Figure 6.5: Similarity matrix of conferences based on CP_h embeddings, computed using cosine (left) and dot product (right).

6.3.2.3 Task 3: Relational query

This is one of the most important tasks in semantic query. We will quantitatively evaluate the relational query performance on the embedding spaces of random guess, word2vec, CP_h, and MEI in details.

Standard link prediction benchmark Link prediction is a relational query task given a relation and the head or tail entity to predict the corresponding tail or head entities. We first evaluate the using standard metrics on the link prediction task. Table 6.3 shows the standard link prediction results for random guess, word2vec, CP_h, and MEI, respectively.

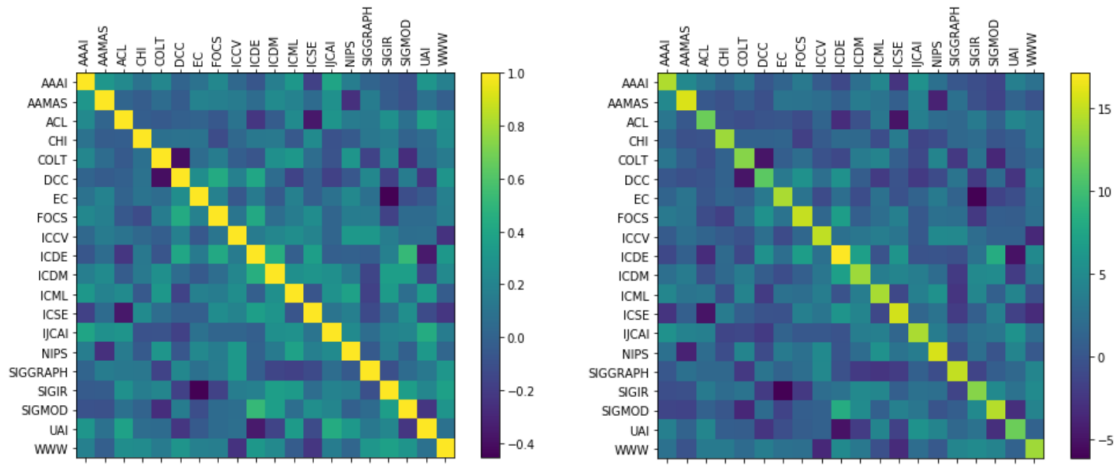

Figure 6.6: Similarity matrix of conferences based on MEI embeddings, computed using cosine (left) and dot product (right).

We first observe that random guess completely fails on this task. This demonstrates that the task and the benchmark dataset are very difficult. The second important observation is multi-relational embedding methods significantly outperform single-relational embedding methods such as word2vec. In addition, we see that MEI strongly outperforms the CP_h method because MEI is more expressive and parameter efficient.

The standard link prediction task is equivalent to evaluating relational query using all the queries on all relations in the test set triples and averaging the results, with no restriction on predicted entity types. In real world, we usually know the types of each entity and which types are compatible with each query. For example, the answer to the query *(author, ?, write)* should be of type *paper*. Therefore, we also provide evaluation results filtered by the predicted entity types for each relation in KG20C in Table 6.4.

Interestingly, we see that filtering by the entity types significantly improves the results of word2vec, whereas the results of CP_h and MEI are only improved very slightly. This can be explained by the fact that word2vec cannot capture the information about entity types well, thus explicitly providing this information is crucial for the word2vec model. On multi-relational embedding methods, the entity type information is well captured and encoded, thus the additional filter does not much affect the results. Note that although the result of word2vec improves with the explicit entity type filter, it is still lower than the results of multi-relational embedding methods.

Detailed analysis Here we will evaluate the detailed performance of the MEI model on each relational query. We express 10 relational queries for 5 relations in KG20C in human language form. For example, the triple *(author, paper, write)* is corresponding

Table 6.3: Link prediction results on KG20C.

Models	**MRR**	**Hit@1**	**Hit@3**	**Hit@10**
Random	0.001	< 5e-4	< 5e-4	< 5e-4
Word2vec	0.068	0.011	0.070	0.177
CP_h	0.215	0.148	0.234	0.348
MEI	**0.230**	**0.157**	**0.258**	**0.368**

Table 6.4: Link prediction results on KG20C filtered by entity types.

Models	**MRR**	**Hit@1**	**Hit@3**	**Hit@10**
Random	0.011	0.003	0.008	0.025
Word2vec	0.203	0.137	0.221	0.330
CP_h	0.216	0.148	0.235	0.350
MEI	**0.231**	**0.157**	**0.259**	**0.369**

to two queries *(author, ?, write)* and *(?, paper, write)*, which can be expressed in natural language as *What papers may this author write?* and *Who may write this paper?*, respectively. The results are shown in Table 6.5 and demonstrate a good performance on situations simulating real-world contexts.

Table 6.5: Detailed relational query results with MEI on KG20C.

Queries	**MRR**	**Hit@1**	**Hit@3**	**Hit@10**
Who may work at this organization?	0.299	0.221	0.342	0.440
Where may this author work at?	0.626	0.562	0.669	0.731
Who may write this paper?	0.247	0.164	0.283	0.405
What papers may this author write?	0.273	0.182	0.324	0.430
Which papers may cite this paper?	0.116	0.033	0.120	0.290
Which papers may this paper cite?	0.193	0.097	0.225	0.404
Which papers may belong to this domain?	0.052	0.025	0.049	0.100
Which may be the domains of this paper?	0.189	0.114	0.206	0.333
Which papers may publish in this conference?	0.148	0.084	0.168	0.257
Which conferences may this paper publish in?	0.693	0.542	0.810	0.976

6.4 Beyond word analogy: entity analogy reasoning in multi-relational embedding space

In this section, we review the entity analogy reasoning task as an extension of the word analogy reasoning task on multi-relational embedding space. We first define the task, give examples, and provide an new equivalent definition for it in multi-relational embedding

space as the open-relational query by examples task. We then study the semantic analogy structures in multi-relational embedding space. Finally we propose a potential solution based on the studied semantic structures.

6.4.1 Entity analogy reasoning as the open-relational query by examples task

Word analogy is a popular concept introduced by the word2vec paper [57]. Its typical example has the form *A is to B as C is to ?*, for example, *king is to man as queen is to woman*. Extending this task directly to multi-relational data, we define the entity analogy reasoning task as follows.

Definition 6.4. ***(Entity analogy reasoning task)*** Given an entity e and some entity pairs $X = \{(a_i, b_i)|i = 1 \dots n\}$, where there is an analogical relationship such that a_i is to b_i as a_j is to b_j, $i = 1 \dots n, j = 1 \dots n$. Find the entity e' such that a_i is to b_i as e is to e', $i = 1 \dots n$.

In multi-relational data format, there are explicit relationship information between entities, we can state the analogical relationship as an explicit relation. This enables us to treat the task as an open-relational query task, where the open relations can be specified by some example pairs $(a_i, b_i), i = 1 \dots n$ at the query time. Specifically we define the equivalent Open-relational query by examples task as follows.

Definition 6.5. ***(Open-relational query by examples task)*** Given an entity e and some examples of the form $X = \{(a_i, b_i)|i = 1 \dots n\}$, where there is a new unseen relation r between the entities in each pair (a_i, b_i). Find the entity e' such that e' is the result of relational query $(e, ?, r)$.

This task enables the possibility to query open and arbitrary relationship on a multi-relational dataset, which can be very useful because a multi-relational dataset cannot store all possible relationships between the entities. For example, in a bibliographic dataset, the knowledge graph format may only include direct bibliographic relations and not the co-author relation. We can query co-authorship after obtaining the fixed pretrained embedding space by solving this task. We can also query other types of relationship, such as who is the editors, the reviewers, or the supervisor of an author.

Below, we will show that the formulation of entity analogy reasoning task as open-relational query by examples task is essential in studying the semantic analogy structure in the multi-relational embedding space.

6.4.2 Semantic analogy structures in the multi-relational embedding space

The semantic analogy structure in the word embedding space is formulated as the *semantic directions* by vector difference between word embedding vectors. Specifically, the analogy *A is to B as C is to D* is represented by the structure

$$\boldsymbol{v}_A - \boldsymbol{v}_B = \boldsymbol{v}_C - \boldsymbol{v}_D. \tag{6.6}$$

This structure was made popular by the word2vec model, where they showed the intriguing relationship between the word embedding vectors such as $king - man = queen - woman$. This simple semantic direction structure does not necessarily hold in multi-relational embedding space such as the knowledge graph embedding space. In this section, we will explain why and propose a new general semantic analogy structure for multi-relational embedding space.

First, we explain why the simple semantic direction structure does not necessarily hold. In regard to the MEI transforming and matching framework, the relationship between two entities in a triple is modeled by a relation-based linear transformation. Because linear transformation are composable, the relationship between any two entities in the knowledge graph can also be modeled by a linear transformation. Considering a linear transformation T that models the gender relationship, so that $T(king) = Z \cdot queen$ and $T(man) = Z \cdot woman$, where Z is a scalar value accounting for the different scale in the matching step. Note that linear transformations preserve linear combinations, so we can write

$$\begin{aligned} T(king - man) &= T(king) - T(man) && \text{(by linearity of T)} && (6.7)\\ &= Z \cdot queen - Z \cdot woman && \text{(by definition of T)} && (6.8)\\ &= Z \cdot (king - man) && \text{(by claiming the simple linear structure).} && (6.9) \end{aligned}$$

Therefore, the simple semantic direction structure as in the word embedding space only holds when the linear transformation T is the identity transformation up to a scale by Z, such that $T(king - man) = Z \cdot (king - man)$. Such an identity relation embedding is present in single-relational embedding methods such as word2vec and was discussed in Section 4.5.1. In general multi-relational embedding space, there are multiple different linear transformations so the simple semantic direction structure does not hold.

6.4.2.1 General semantic analogy structure

Now, we define the *general semantic analogy structure* in multi-relational embedding space. As we have discussed above, the entity analogy reasoning task is equivalent to the open-relational query by examples task, where the analogical relationship is denoted as an unseen relation r. Intuitively, the general semantic analogy structure can be defined based on the embedding of this relation r, which can be computed based on a general linear map between a and b. Denote the general semantic analogy structure between two entities a and b as $sem(a, b)$, we can then write $sem(king, man) = sem(queen, woman)$. The function $sem(\cdot, \cdot)$ can be seen as an edge feature extractor or a relation embedding reconstructor for r.

The detailed definition of the function $sem(\cdot, \cdot)$ depends on the specific multi-relational embedding methods used to obtain the embedding space. In the general case of the MEI model,

$$sem(a, b) = \text{concat}_{k=1}^{K} \left(Z \oslash \left(\mathbf{W}_k \bar{\times}_1 \boldsymbol{a}_{k:} \bar{\times}_2 \boldsymbol{b}_{k:} \right) \right), \tag{6.10}$$

where $\oslash$ denotes element-wise division, Z is a global scalar value for a given dataset, and $\mathbf{W}_k$ are the global core tensors given a dataset. In the specific case, for example CP_h,

$$sem(a, b) = \text{concat} \left(Z \oslash \left(\boldsymbol{a}_{1:} \odot \boldsymbol{b}_{2:} \right), Z \oslash \left(\boldsymbol{a}_{2:} \odot \boldsymbol{b}_{1:} \right) \right), \tag{6.11}$$

where $\oslash$ denotes element-wise division and Z is a global scalar value for a given dataset.

6.4.3 Towards a solution for entity analogy reasoning on multi-relational embedding space

In this section, we outline a potential solution of semantic query, that is, multi-linear algebraic operations on multi-relational embedding space, to approximate the entity analogy reasoning task.

Note that the analogy reasoning task cannot be solved as a relational query task directly, because the analogical relation is absent from the training data, and thus unavailable in the pretrained multi-relational embedding space. Therefore, to treat it as a relational query task, we need an efficient method to obtain the relation-based transformation function of the new relationship from the existing knowledge graph embedding space. In addition, the information about the analogical relation is usually very scarce, including only a few example. Therefore, it is difficult and inefficient to use the few data

points in the examples to fine-tune the existing embedding space.

To solve this problem, one potential approach is to use the general semantic analogy structure proposed in Section 6.4.2. The essence here is to compute the general semantic analogy structure between the entities in the example pairs and use them to construct the relation-based transformation function to use in relational query. Here we outline the solution on the embedding space obtained by the MEI model. For simplicity, we consider a local MEI model with a single partition to omit k and concat$(\cdot)$ in the notation.

We first compute the average generalized linear structure between the example entities:

$$sem_X = \frac{1}{n} \sum_{i=1...n} sem(a_i, b_i) \tag{6.12}$$

$$= \frac{1}{n} \sum_{i=1...n} Z \oslash (\mathbf{W} \bar{\times}_1 \boldsymbol{a}_i \bar{\times}_2 \boldsymbol{b}_i) \,. \tag{6.13}$$

We then use the computed generalized linear structure sem_X to reconstruct the relation-based transformation matrix of the new relationship:

$$\boldsymbol{M}_X = \mathbf{W} \bar{\times}_3 sem_X. \tag{6.14}$$

Finally we use $\boldsymbol{M}_X$ to construct the relation-based transformation function:

$$T_X(e) = \left(\boldsymbol{e}^\top \boldsymbol{M}_X\right)^\top \tag{6.15}$$

$$= \boldsymbol{M}_X^\top \boldsymbol{e} \tag{6.16}$$

$$= \left(\mathbf{W} \bar{\times}_3 sem_X\right)^\top \boldsymbol{e} \tag{6.17}$$

At this step, we have converted the open-relational query task to the known relational query task and thus can reuse the known solution for the remaining steps. The most related entity to e through the open relationship given by X is:

$$\text{Result} = \arg\max_{e_i \in \mathcal{E}} sim(\boldsymbol{e}_i, T_X(\boldsymbol{e})) \tag{6.18}$$

$$= \arg\max_{e_i \in \mathcal{E}} \boldsymbol{e}_i^\top T_X(\boldsymbol{e}), \tag{6.19}$$

where $T_X(\cdot)$ is the new open relation-based transformation computed above.

Note that after acquiring the generalized linear structure sem_X, we can use the few data points in the examples to continue fine-tuning it using the standard knowledge graph embedding training procedure.

6.5 Summary

6.5.1 Contribution and impact discussions

In general, the main contributions and impacts of our work are as follows.

- We formalize a general framework for multi-relational data exploration and analysis using semantic queries on knowledge graph embedding space. The main component in this framework is the conversion templates from data representation and analysis tasks on the original data to *semantic queries*, which are the multi-linear algebraic operations between the embedding vectors on the embedding space. About the impact, although the framework is conceptually simple, to the best of our knowledge, we are the first to formalize such a framework. Our work potentially facilitates the applications of multi-relational embedding in data representation and analysis.

- We build a knowledge graph from scholarly data and demonstrate how some important representation and analysis tasks on the original data can be solved by semantic queries using the formalized framework. About the impact, we empirically demonstrate how the proposed framework and concepts can be realized on a real world bibliographic dataset. Our implementation can serve as a reference and example for further development of similar applications in practice.

- We also review the entity analogy reasoning on multi-relational embedding space task, which can be seen as an *open-relational query* by examples task. Towards solving this task, we study the semantic structures in the knowledge graph embedding space, propose a general semantic analogy structure in multi-relational embedding space, then outline a potential solution to the above task. About the impact, we explore a new aspect in multi-relational embedding application by extending the analogy reasoning task from word embedding space to multi-relational embedding space. The outline potential solution and theoretical analysis may potentially influence future research.

To the best of our knowledge, in comparison to previous work, the proposed framework is the first formalization and demonstration of a general framework for multi-relational embedding applications in data representation and analysis. It was designed as a simple and extensible framework that can be adopted for applications on new data or new tasks.

In addition, we review the entity analogy reasoning task, express it as a new relational query task on the open relations defined by examples. The analysis led to a potential

analogy structure in multi-relational embedding space and a potential solution to the entity analogy reasoning task.

6.5.2 Scopes and future work

The proposed semantic query framework aims to be general, such that it can be extended to new embedding spaces and new analysis tasks. However, the main limitation of this framework is that, for each task and each embedding space, we need to use an appropriate multi-algebraic operation. For example, the operation for similarity query is different from the operation for relational query, and the operation on translation-based embedding space is different from the operation on semantic matching embedding space. Therefore, one important direction for future work that may have practical impact is to study the appropriate semantic queries for different tasks and different embedding spaces.

The entity analogy reasoning task and the general semantic analogical structure in multi-relational embedding space are intriguing problems. However, in this work we stop at theoretical analysis and outlining the potential solution. We plan to go into more detailed analysis and implementation in future work.

Chapter 7

Conclusion

Multi-relational data, such as knowledge graphs, bibliographic data, and information networks are prevalent in real-world datasets. Managing, exploring, and utilizing these large and complex datasets effectively are challenging. In recent years, multi-relational embedding methods have emerged as a new effective approach to model multi-relational data by representing both the entities and the relations as embedding vectors in semantic space. On knowledge graphs, multi-relational embedding methods aim to model the interactions between these embedding vectors to predict the relational link between entities. These knowledge graph embedding methods solve the important inherent task of link prediction for knowledge graph completion, but also provide the embedding representations that have various potential applications. The goal of this thesis is first to study multi-relational embedding on knowledge graphs to propose a new embedding model that explains and improves previous methods, then to study the applications of multi-relational embedding in representation and analysis of knowledge graphs.

For the first part of the thesis, we study the theoretical framework of knowledge graph embedding methods to explain and improve them. We review and analyze the popular class of semantic matching knowledge graph embedding methods, with a focus on the state-of-the-art trilinear-product-based models such as ComplEx. Based on our analysis, we identify two fundamental complementary aspects that a knowledge graph embedding model needs to address, that is, computational efficiency and model expressiveness. Previous trilinear-product-based models use specially designed interaction mechanisms to manually provide a trade-off between the two aspects. However, their interaction mechanisms are specially designed and fixed, potentially causing them to be suboptimal or difficult to extend. In this thesis, we propose the multi-partition embedding interaction (MEI) model with block term format to systematically address this problem. MEI di-

vides 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. The model combines advanced tensor representation formats and modern deep learning techniques to achieve state-of-the-art performance on the link prediction task. The theoretical framework of the MEI model is then used as a general mechanism of knowledge graph embedding to analyze, explain, and generalize previous models. We also draw the connections to word embeddings and language modeling to provide some new insights and generalizations.

For the second part of the thesis, we study how to apply multi-relational embedding in representation and analysis of knowledge graphs. Unlike word embedding, the semantic structures such as similarity and analogy structures in knowledge graph embedding space are not well-studied, and thus not usually utilized for data representation and analysis. To demonstrate the application of multi-relational embedding, we formalize a framework for data representation and analysis by semantic queries on the multi-relational embedding space. We build a knowledge graph from scholarly data and show how various tasks on the original datasets can be approximated by appropriate semantic queries, which are multi-linear algebraic operations on the multi-relational embedding spaces. We also theoretically study the entity analogy reasoning task in multi-relational embedding space, which can be formulated as an open-relational query by examples task, doing relational query on unseen relations. Using the above mathematical connections between knowledge graph embeddings and word embeddings, we analyze the semantic structures in the knowledge graph embedding space and propose potential solution to the above entity analogy reasoning task. The goal of this endeavor is to explore potential applications of recent advancements in multi-relational embedding to data representation and analysis, especially to improve its effectiveness on scholarly data.

7.1 Contribution and impact discussions

For the first part of the thesis, the main contributions and impacts of our work are as follows.

- We analyze and identify two fundamental complementary aspects in knowledge graph embedding, namely computational efficiency and model expressiveness. We then address both aspects by introducing 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. About the impact, although our work is not the first one to try to trade-off between the computational efficiency and model expressiveness in knowledge graph embedding, most previous works address this problem in a manual or heuristic way. To the best of our knowledge, we are the first to systematically address this problem by proposing the new multi-partition embedding interaction approach, that generalizes previous methods. Therefore, our work potentially present a conclusive hindsight to some recent researches in knowledge graph embedding.

- To realize our approach, we propose the standard multi-partition embedding interaction (MEI) model with block term format, to control the trade-off between computational efficiency and model expressiveness through the partition size, and to learn the interaction mechanisms from data automatically through the local Tucker core tensors. We empirically show that MEI is efficient and effective, as it can achieve state-of-the-art results using the popular and standard link prediction benchmarks. About the impact, the MEI model presents a combination of advanced tensor representation formats and modern deep learning techniques for knowledge graph embeddings, that can provide advantages over previous models. This approach of combination may potentially be a promising direction for future research in knowledge graph embeddings.

- We theoretically analyze the framework of MEI to explain its intuitions and meanings. In addition, we are the first to formally study the parameter efficiency problem and derive a simple optimal trade-off criterion for the model size of MEI. We apply the theoretical framework of MEI to provide intuitive explanations for the specially designed interaction mechanisms in several previous knowledge graph embedding models. About the impact, MEI is not just a model, but an approach and theoretical framework to knowledge graph embedding. There are many potential variants and extensions that can improve the model, of which some variants have been discussed above. The theoretical framework of MEI may serve to assist in analyzing previous models and may be readily applied to improve them.

- We also draw the connections from knowledge graph embedding to word embeddings and language modeling to provide some new insights and generalizations. About the impact, these connections may potentially benefit the research and development

in both domains of representation learning.

For the first part of the thesis, the main contributions and impacts of our work are as follows.

- We formalize a general framework for multi-relational data exploration and analysis using semantic queries on knowledge graph embedding space. The main component in this framework is the conversion templates from data representation and analysis tasks on the original data to *semantic queries*, which are the multi-linear algebraic operations between the embedding vectors on the embedding space. About the impact, although the framework is conceptually simple, to the best of our knowledge, we are the first to formalize such a framework. Our work potentially facilitates the applications of multi-relational embedding in data representation and analysis.

- We build a knowledge graph from scholarly data and demonstrate how some important representation and analysis tasks on the original data can be solved by semantic queries using the formalized framework. About the impact, we empirically demonstrate how the proposed framework and concepts can be realized on a real world bibliographic dataset. Our implementation can serve as a reference and example for further development of similar applications in practice.

- We also review the entity analogy reasoning on multi-relational embedding space task, which can be seen as an *open-relational query* by examples task. Towards solving this task, we study the semantic structures in the knowledge graph embedding space, propose a general semantic analogy structure in multi-relational embedding space, then outline a potential solution to the above task. About the impact, we explore a new aspect in multi-relational embedding application by extending the analogy reasoning task from word embedding space to multi-relational embedding space. The outline potential solution and theoretical analysis may potentially influence future research.

7.2 Scopes and future work

The proposed MEI framework provides a general approach towards knowledge graph embedding, which can be used to explain previous embedding models and extend to new effective variants. However, the main limitation of the proposed method as well as most previous knowledge graph embedding methods is that they are *transductive learning*

methods, that is, they can only learn the embeddings for the entities and relations that are existing in training. Therefore, one big future direction is to extend the proposed method to *inductive learning* approach, where it can compose embedding for new unseen entities and relations.

The proposed semantic query framework aims to be general, such that it can be extended to new embedding spaces and new analysis tasks. However, the main limitation of this framework is that, for each task and each embedding space, we need to use an appropriate multi-algebraic operation. For example, the operation for similarity query is different from the operation for relational query, and the operation on translation-based embedding space is different from the operation on semantic matching embedding space. Therefore, one important direction for future work that may have practical impact is to study the appropriate semantic queries for different tasks and different embedding spaces.

The entity analogy reasoning task and the general semantic analogical structure in multi-relational embedding space are intriguing problems. However, in this work we stop at theoretical analysis and outlining the potential solution. We plan to go into more detailed analysis and implementation in future work.

Bibliography

[1] L. V. Ahlfors. Complex Analysis: An Introduction to the Theory of Analytic Functions of One Complex Variable. *New York, London*, page 177, 1953.

[2] S. Arora, Y. Li, Y. Liang, T. Ma, and A. Risteski. Linear Algebraic Structure of Word Senses, with Applications to Polysemy. *Transactions of the Association for Computational Linguistics*, 6(0):483–495, 2018.

[3] 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.

[4] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3(Feb):1137–1155, 2003.

[5] J. Bergstra and Y. Bengio. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

[6] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. FastText: Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[7] 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.

[8] 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.

[9] K. P. Burnham and D. R. Anderson. Practical Use of the Information-Theoretic Approach. In *Model Selection and Inference*, pages 75–117. Springer, 1998.

[10] C. Cai. Group Representation Theory for Knowledge Graph Embedding. In *arXiv:1909.05100 [Cs, Math]*, 2019.

[11] W. Carrer-Neto, M. L. Hernández-Alcaraz, R. Valencia-García, and F. García-Sánchez. Social Knowledge-based Recommender System. Application to the Movies Domain. *Expert Systems with Applications*, 39(12):10990–11000, Sept. 2012. ISSN 0957-4174.

[12] Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. You CAN Teach an Old Dog New Tricks! On Training Knowledge Graph Embeddings. In *Proceedings of the International Conference on Learning Representations*, page 20, 2020.

[13] L. De Lathauwer. Decompositions of a Higher-Order Tensor in Block Terms—Part I: Lemmas for Partitioned Matrices. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1022–1032, 2008.

[14] 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.

[15] L. De Lathauwer and D. Nion. Decompositions of a Higher-Order Tensor in Block Terms—Part III: Alternating Least Squares Algorithms. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1067–1083, 2008.

[16] 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.

[17] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion. In *Proceedings of the 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 601–610, 2014.

[18] 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.

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

[20] L. Ehrlinger and W. Wöß. Towards a Definition of Knowledge Graphs. In *Proceedings of the Posters and Demos Track of the 12th International Conference on Semantic Systems*, pages 1–4, 2016.

[21] S. Fathalla, S. Vahdati, S. Auer, and C. Lange. Towards a Knowledge Graph Representing Research Findings by Semantifying Survey Articles. In *Proceedings of the 21st International Conference on Theory and Practice of Digital Libraries*, pages 315–327, 2017.

[22] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional Sequence to Sequence Learning. *arXiv:1705.03122 [cs]*, 2017.

[23] X. Glorot and Y. Bengio. Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *Proceedings of the 13rd International Conference on Artificial Intelligence and Statistics*, volume 9, pages 249–256, 2010.

[24] R. Goldman. Rethinking Quaternions. *Synthesis Lectures on Computer Graphics and Animation*, 4(1):1–157, Oct. 2010. ISSN 1933-8996.

[25] A. Grover and J. Leskovec. Node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 855–864, 2016.

[26] N. Guberman. On Complex Valued Convolutional Neural Networks. *arXiv:1602.09046 [cs.NE]*, Feb. 2016.

[27] D. Ha, A. M. Dai, and Q. V. Le. HyperNetworks. In *Proceedings of the International Conference on Learning Representations*, page 18, 2016.

[28] W. Hackbusch. *Tensor Spaces and Numerical Tensor Calculus*. Number 42 in Springer Series in Computational Mathematics. Springer Science & Business Media, 2012.

[29] W. Hamilton, Z. Ying, and J. Leskovec. Inductive Representation Learning on Large Graphs. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[30] W. Hamilton, P. Bajaj, M. Zitnik, D. Jurafsky, and J. Leskovec. Embedding Logical Queries on Knowledge Graphs. In *Advances in Neural Information Processing Systems*, pages 2026–2037, 2018.

[31] C. J. Hillar and L.-H. Lim. Most Tensor Problems Are NP-Hard. *Journal of the ACM*, 60(6):1–39, 2013.

[32] G. E. Hinton. Learning Distributed Representations of Concepts. In *Proceedings of the Annual Conference of The Cognitive Science Society*, pages 1–12, 1986.

[33] G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. Distributed Representations. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1*, pages 77–109. MIT Press, 1984.

[34] 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.

[35] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[36] 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.

[37] M. Iyyer, V. Manjunatha, J. Boyd-Graber, and H. Daumé III. Deep Unordered Composition Rivals Syntactic Methods for Text Classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1681–1691, 2015.

[38] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. FastText: Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 427–431, 2017.

[39] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu. Exploring the Limits of Language Modeling. *arXiv:1602.02410 [cs]*, 2016.

[40] I. L. Kantor and A. S. Solodovnikov. *Hypercomplex Numbers: An Elementary Introduction to Algebras*. Springer, 1989.

[41] 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.

[42] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations*, 15, 2014.

[43] T. N. Kipf and M. Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the International Conference on Learning Representations*, page 14, 2017.

[44] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. Skip-Thought Vectors. In *Advances in Neural Information Processing Systems*, pages 3294–3302, 2015.

[45] P. Knoth and Z. Zdrahal. CORE: Three Access Levels to Underpin Open Access. *D-Lib Magazine*, 18(11/12):1–13, 2012.

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

[47] T. Lacroix, N. Usunier, and G. Obozinski. Canonical Tensor Decomposition for Knowledge Base Completion. In *Proceedings of the 35th International Conference on Machine Learning*, pages 2863–2872, 2018.

[48] Q. V. Le and T. Mikolov. Distributed Representations of Sentences and Documents. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1188–1196, 2014.

[49] H. Leeb and B. M. Pötscher. Model Selection and Inference: Facts and Fiction. *Econometric Theory*, 21(1):21–59, 2005.

[50] A. Lerer, L. Wu, J. Shen, T. Lacroix, L. Wehrstedt, A. Bose, and A. Peysakhovich. PyTorch-BigGraph: A Large-scale Graph Embedding System. In *Proceedings of the 2nd SysML Conference*, page 12, 2019.

[51] J. Leskovec and A. Krevl. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data/index.html, June 2014.

[52] O. Levy, Y. Goldberg, and I. Ramat-Gan. Linguistic Regularities in Sparse and Explicit Word Representations. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 171–180, 2014.

[53] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 2181–2187, 2015.

[54] P. Liu, K. K. Wu, and H. Meng. A Model of Extended Paragraph Vector for Document Categorization and Trend Analysis. In *Proceedings of the International Joint Conference on Neural Networks*, pages 2400–2406, 2017.

[55] L. McInnes and J. Healy. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv:1802.03426 [cs, stat]*, Feb. 2018.

[56] 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.

[57] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed Representations of Words and Phrases and Their Compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

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

[59] T. Minemoto, T. Isokawa, H. Nishimura, and N. Matsui. Feed forward neural network with random quaternionic neurons. *Signal Processing*, C(136):59–68, 2017. ISSN 0165-1684.

[60] 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.

[61] 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.

[62] T. Parcollet, M. Ravanelli, M. Morchid, G. Linarès, C. Trabelsi, R. D. Mori, and Y. Bengio. Quaternion Recurrent Neural Networks. In *Proceedings of the International Conference on Learning Representations*, 2019.

[63] B. Perozzi, R. Al-Rfou, and S. Skiena. DeepWalk: Online Learning of Social Representations. In *Proceedings of the 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 701–710, 2014.

[64] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 17th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018.

[65] S. J. Reddi, S. Kale, and S. Kumar. On the Convergence of Adam and Beyond. In *Proceedings of the International Conference on Learning Representations*, page 23, 2018.

[66] A. Sadeghi, C. Lange, M.-E. Vidal, and S. Auer. Integration of Scholarly Communication Metadata Using Knowledge Graphs. In *Proceedings of the 21st International Conference on Theory and Practice of Digital Libraries*, pages 328–341, 2017.

[67] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling. Modeling Relational Data with Graph Convolutional Networks. In *The Semantic Web*, Lecture Notes in Computer Science, pages 593–607. Springer International Publishing, 2018.

[68] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos. Tensor Decomposition for Signal Processing and Machine Learning. *IEEE Transactions on Signal Processing*, 65(13):3551–3582, 2017.

[69] A. Singhal. Official Google Blog: Introducing the Knowledge Graph: Things, Not Strings. https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html, 2012.

[70] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B.-J. P. Hsu, and K. Wang. An Overview of Microsoft Academic Service (MAS) and Applications. pages 243–246, 2015.

[71] R. Socher, D. Chen, C. D. Manning, and A. Y. Ng. Reasoning With Neural Tensor Networks for Knowledge Base Completion. In *Proceedings of the 27th Conference on Neural Information Processing Systems*, pages 926–934, 2013.

[72] 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.

[73] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *Proceedings of the International Conference on Learning Representations*, page 18, 2019.

[74] P. Symeonidis. Matrix and Tensor Decomposition in Recommender Systems. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 429–430, 2016.

[75] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. LINE: Large-scale Information Network Embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077, 2015.

[76] 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.

[77] 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.

[78] 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.

[79] 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*, page 8, 2020.

[80] 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.

[81] T. Trouillon, C. R. Dance, É. Gaussier, J. Welbl, S. Riedel, and G. Bouchard. Knowledge Graph Completion via Complex Tensor Factorization. *The Journal of Machine Learning Research*, 18(1):4735–4772, 2017.

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

[83] S. Vahdati, G. Palma, R. J. Nath, C. Lange, S. Auer, and M.-E. Vidal. Unveiling Scholarly Communities over Knowledge Graphs. In *Proceedings of the 22nd International Conference on Theory and Practice of Digital Libraries*, pages 103–115, 2018.

[84] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Science & Business Media, 2013.

[85] M. A. O. Vasilescu and D. Terzopoulos. Multilinear Analysis of Image Ensembles: TensorFaces. In *Proceedings of the 7th European Conference on Computer Vision*, pages 447–460, 2002.

[86] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention Is All You Need. In *Proceedings of the 31st Conference on Neural Information Processing Systems*, June 2017.

[87] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph Attention Networks. In *International Conference on Learning Representations*, 2018.

[88] D. Vrandečić and M. Krötzsch. Wikidata: A Free Collaborative Knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[89] 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):2724–2743, 2017.

[90] R. Wang, Y. Yan, J. Wang, Y. Jia, Y. Zhang, W. Zhang, and X. Wang. AceKG: A Large-scale Knowledge Graph for Academic Data Mining. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1487–1490, 2018.

[91] Y. Wang, R. Gemulla, and H. Li. On Multi-Relational Link Prediction with Bilinear Models. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 4227–4234, 2018.

[92] Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge Graph Embedding by Translating on Hyperplanes. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pages 1112–1119, 2014.

[93] H. Xiao, M. Huang, Y. Hao, and X. Zhu. TransA: An Adaptive Approach for Knowledge Graph Embedding. *arXiv:1509.05490 [cs.CL]*, 2015.

[94] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How Powerful are Graph Neural Networks? In *Proceedings of the International Conference on Learning Representations*, 2019.

[95] 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.

[96] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma. Collaborative Knowledge Base Embedding for Recommender Systems. In *Proceedings of the 22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 353–362, 2016.

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

[98] M. Zitnik, R. Sosic, S. Maheshwari, and J. Leskovec. BioSNAP Datasets: Stanford Biomedical Network Dataset Collection. http://snap.stanford.edu/biodata/index.html, Aug. 2018.

Appendix A

Publications

International conference papers (in the top conference list):

- Hung Nghiep Tran and Atsuhiro Takasu, "Multi-Partition Embedding Interaction with Block Term Format for Knowledge Graph Completion," in Proceedings of the European Conference on Artificial Intelligence (ECAI 2020), June 2020, page 8. (Full paper, top conference list: No. 113)

International conference papers (other refereed conferences):

- Hung Nghiep Tran and Atsuhiro Takasu, "Analyzing Knowledge Graph Embedding Methods from a Multi-Embedding Interaction Perspective," in Proceedings of the International Workshop on Data Science for Industry 4.0 at EDBT/ICDT 2019 Joint Conference (DSI4 2019), March 2019. (Full paper)
- 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 (TPDL 2019), September 2019, pp. 154–162. (Short paper)