THE UNIVERSITY
of EDINBURGH

Tensors and Tensor Decompositions for Combining External Information with Knowledge Graph Embeddings

Esma Balkır

Doctor of Philosophy

Institute for Language, Cognition and Computation
School of Informatics
University of Edinburgh

2021

Abstract

The task of knowledge graph (KG) completion, where one is given an incomplete KG as a list of facts, and is asked to give high scores to correct but unseen triples, has been a well-studied problem in the NLP community. A simple but surprisingly robust approach for solving this task emerged as learning low dimensional embeddings for entities and relations by approximating the underlying KG directly through a scoring function.

Knowledge graphs have a natural representation as a binary three way array, also known as a 3rd order *tensor*, and certain classes of scoring functions can be characterized as finding a low-rank decomposition of this tensor. This dissertation extends this characterization, and investigates the suitability of tensors for modelling both knowledge graphs and related data, for learning low-rank representations of entities and relations that incorporate information from heterogeneous sources, and for reasoning with paths and rules using the learned representations.

Specifically, we present two joint tensor decomposition models for integrating external information in the process of learning KG embeddings. Our first model is a joint tensor-tensor decomposition model that learns representations based on both KG facts and type information on entities and relations. Our second model is a joint tensor-matrix decomposition for integrating cooccurrence information between entities and words from an entity linked corpus into knowledge graph embeddings, in order to learn better representations for the entities that are rarely seen in the knowledge graph.

We also investigate tensors as tools for enabling multi-step reasoning using learned embedding representations. To this end, we extend theoretical results for semiring weighted logic programs to tensors of semirings. Our results are broadly applicable to any area that uses dynamic programming algorithms for calculating tensor values. Such applications include incorporating embeddings of paths and rules for knowledge graph completion, and syntactic parsing with latent variable grammars.

Lay Summary

Our devices are increasingly able to interact with us on our own terms. We ask questions to our phones by speaking to them, and expect them to find the answer and tell it to us in the language we speak. Likewise when we write a question in web search, we expect the search engine to retrieve links that match what we *mean*. Having access to a large collection of general facts about the world is necessary for these applications to perform satisfactorily. This has led to the development of large repositories of world knowledge both in the public domain and in virtually all tech giants that offer these applications. These collections are called *knowledge graphs*. They model world facts in an interconnected way, as relations linking the entities. For example, *University of Edinburgh* and *Scotland* may be entities in a knowledge graph, and we'd expect the relation *Located-in* to connect the two.

Due to their scope and the sheer amount of information they need to capture, knowledge graphs invariably have missing links. The task of guessing which links might be missing just by looking at existing facts in a knowledge graph is called *knowledge graph completion* or *link prediction*. An effective way of achieving this is to learn representations of entities and relations as points in a high-dimensional space. Just as a point in three-dimensions can be specified by giving its distance from a chosen point as three quantities corresponding to *height*, *length* and *width*, a point in n dimensional space can be specified with n values. A common name for these representations is *embeddings*. Embeddings learned from knowledge graphs are used both for guessing which links are missing, but also as representations for other machine learning tasks that operate on the components of knowledge graphs.

Although knowledge graphs are an effective way for organizing knowledge, other types of information are usually also available for the entities and relations they contain. This thesis explores techniques for integrating additional information into embedding representations for knowledge graphs. The additional information considered includes types, such as *University of Edinburgh* being an organization and *Scotland* being a country, and collections of text that mention the entities, such as Wikipedia. We present novel models for learning embedding representations jointly from knowledge graphs and text or type data, and show that these embeddings perform better at predicting missing links compared to those learned from knowledge graphs only.

Finally, we consider rules that might apply to relations, such as: "If an entity A is

Located-in some location *B*, and location *B* is in turn *Located-in* another place *C*, then entity *A* is also *Located-in C*." If we already have the facts *(University of Edinburgh, Located-in, Scotland)* and *(Scotland, Located-in, United Kingdom)*, this rule allows us to deduce that *(University of Edinburgh, Located-in, United Kingdom)*. To be able to use these kinds of rules with embeddings, we develop theoretical foundations for integrating the learned knowledge graph embeddings in the process of deducing new links from existing ones by applying the given rules. The theoretical results presented are also applicable for automatic processing of human language beyond the immediate task of knowledge graph completion, and we discuss their applications to methods for uncovering the grammatical structure of sentences.

Acknowledgements

First and foremost, I would like to thank my supervisor Shay Cohen for his guidance and his support throughout this PhD. He was always there when I needed help, technical or otherwise. I learned much from our meetings, and this thesis would not have been possible without his scientific insights, patience and persistence. I am grateful to my second supervisor Ivan Titov for his careful eye and his constructive comments. His feedback throughout my PhD was essential in steering my research in the right direction. I was lucky to have Adam Lopez, Timothy Hospedales and Frank Keller as members of my thesis committee at different milestones, and would like to thank them for their valuable feedback. I would like to thank Stephen Clark and Mark Steedman for being my thesis examiners. I believe that their comments and corrections made this a stronger thesis.

I owe much to Masha Naslidnyk, Dave Palfrey and Arpit Mittal for the material that appears in Chapter 3. They have all been incredibly supportive throughout my internship at Amazon as colleagues, mentors and co-authors. I would especially like to thank Masha for her involvement in my project, for her attentiveness, and for foreseeing and tackling any problems head on before they could slow us down. I would like to thank Dave for his continuous presence and invaluable contributions to the discussion, for his encouragement, and for his enthusiasm about my ideas. I would like to thank Arpit for overseeing this work and for giving us his full support. Some of the work presented in Chapter 3 would not be possible if he did not go out of his way to make sure that the research code I created during my internship was open-sourced and accessible. I would also like to thank my fellow interns Fernando, Molly, Daniele, Ngoc, James, Sophie, Frederic and Alex.

I am grateful to have had the chance to work with Daniel Gildea on the material presented in Chapter 5. This project stemmed from his and Shay's ideas, and his input was essential in clarifying, correcting and distilling the theory as it developed.

I would like to thank everyone in Shay's research group, especially Marco, Nikos, Chunchuan, Joana, Jianming and Shashi, for helping me mature as a researcher through presentations, discussions and feedback sessions. I would also like to thank everyone who attended the Knowledge Graphs reading group, in particular Michael, Sabine, Javad and Sander, for expanding my knowledge and understanding of the topic. Finally, I'd like to thank everyone in EdinburghNLP. It was a privilege to be a part of

such a strong, diverse group of NLP researchers, and the weekly seminar was an priceless resource for a PhD student.

Spending one year preceding this degree working with Mehrnoosh Sadrzadeh, I gained insights and experience that have been invaluable for this PhD. I could not have started this program if it wasn't for her support. I would not have even considered this path if Prakash Panangaden hadn't mentored me during my undergraduate and introduced me to computer science research. I am very grateful to both of them for the support and encouragement I received.

Going into a PhD, I knew that it would be difficult. I was not, however, prepared for the particular ways in which it would get difficult, and I couldn't have made it through without the help and support of many people. I made many good friends in ILCC. Whenever I needed a cup of coffee and a respite from work, I could always find a few friends in the common areas and join them. I am thankful to everyone who has been a part of this group, especially Sameer, Ida, Nick, Seraphina, Sorcha, Federico, Naomi, Lucia, Elisabeth, Clara, Yevgen, Craig and Elif. I was lucky to share an office with Sabine. We commiserated, held each other accountable on our day-to-day tasks and on gym attendance. I am thankful to her and Joe for being there through some of the most difficult parts of my PhD. I am thankful to Toms for deciding to befriend me on my first weeks in the program, for the too-long coffee breaks and all the heated discussions we had. I would've had much less fun in Edinburgh if it wasn't for him and Kima.

The music group Baharat Collective made it possible for me to take a break from my work, relax, connect with people and make music just for the joy of it. Thank you Sarah, Chris, Ahmed, Masoud, Morteza, Konstantina, Maria, Izabela, Resul, Georgos, Vera, Jack, Paulo and everyone else who has been a part of Baharat Collective. Stella and Aris have opened their home for me during the last months of my PhD, and became my Edinburgh family. I am immensely grateful for their friendship. I am thankful to my friends Gülhiz and Selim for their continuous support, to Ulya, Ilgar and Efsun for welcoming me home to Istanbul whenever I went back and to Ipek, Farah and Zach for being there in Canada when I was writing up.

I would like to thank my parents Aslı and Babür for all their support during this PhD, and all that they have done for me before. I could not have got here without you. I am grateful for my grandmother Türkan for always being there for me, and for being a great listener. I am also thankful to my brother Esat and the rest of my family for

always being supportive and interested in what I do.

Finally, to my husband Martti, I am grateful to have had you by my side, sharing this experience. We finished two PhDs, and we did it together. Thank you for never wavering in your support. You are my rock.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Esma Balkır*)

Dedem Hüsamettin Ziler'e

Table of Contents

Chapter 1

Introduction

Knowledge graphs (KGs) are repositories of explicit world knowledge represented in the form of a graph where the entities are vertices, and facts are labelled edges connecting the entities. Although there is no consensus on the precise definition of a knowledge graph, we will use the convention from the Natural Language Processing (NLP) community that a KG is a collection of triples in the form (*entity-1*, `relation`, *entity-2*) such as (*University of Edinburgh*, `LocatedIn`, *Scotland*).[1] The overarching goal of KGs is to represent knowledge in a way that is unambiguous, interconnected, extendable and usable both by humans and algorithms on a wide variety of tasks. Examples of knowledge graphs include YAGO (Mahdisoltani et al., 2013), DBpedia (Auer et al., 2007), NELL (Carlson et al., 2010), Freebase (Bollacker et al., 2008) and Google Knowledge Vault (Dong et al., 2014). KGs are commonly used as resources in knowledge based applications in NLP and beyond, such as relation extraction, question answering and web search (see Wang et al. (2017) for a review).

Large knowledge graphs may be curated by automatically extracting knowledge from various sources (e.g. DBpedia, YAGO), crowdsourcing (e.g. Freebase) or collaborative manual contributions (e.g. Wikidata). Due to their scale and curation methods, KGs often contain errors and ommissions such as missing facts or duplicate entities. Hence it is also common to apply machine learning methods for cleaning and refining the KGs using their existing data. Such internal tasks include *knowledge graph completion*, *entity resolution*, and *link based clustering*. This thesis focuses on knowledge graph completion, where the goal is to identify the correct facts missing from the knowledge

[1] An in depth discussion of the issue of defining knowledge graphs can be found in Ehrlinger and Wöß (2016).

graph.

The terms *Knowledge Graph* and *Knowledge Base* have been used interchangeably in the literature, often employed to describe the same structures and resources. The term Knowledge Graph highlights that the knowledge in these resources is organized as a *multi-graph*, which is a graph with typed edges. When viewing a knowledge base as a multi-graph, each entity is viewed as a node, and each relation as a type of an edge. This type of representation has also been described as *relational data*, and learning performed on it as *statistical relational learning* (Nickel et al., 2016a).

It is possible to view a KG as a sparse, incomplete three way binary array (i.e. a *tensor*; see Section 2.1), with two of the modes corresponding to entities and one mode corresponding to relations. An intuitive way to think about this representation is to view each relation as defining a directed graph between the entities. Then, there is an adjacency matrix corresponding to each relation, and stacking these adjacency matrices gives the three way tensor representing the multi-graph.

A promising approach for tackling learning tasks on KGs is to find a low rank and approximate decomposition of the incomplete tensor. This is essentially a hidden variable view of the problem, where one hopes that the observed high dimensional multi-relational data can be explained by global correlations of much smaller dimensions. This also results in dense vector representations for entities and relations, commonly referred to as *embeddings*, which can be used for downstream tasks.

Although embeddings are obtained primarily from the relational data contained in the KG, a variety of relevant data complementary to the pure (*entity*, `relation`, *entity*) triples is also available due to the nature and scope of contemporary KGs. This includes type data on entities and relations, text data linked to entities, and logical rules that can be used to derive unknown facts from known ones.

From a practical perspective, this thesis pursues the goal of integrating these types of external data into knowledge graph embeddings using tools from tensor algebra. Our primarily objective is to improve model performance on knowledge graph completion. In doing so, our methods fuse information from several different sources into entity and relation embeddings, resulting in rich representations that can easily be used for other knowledge-based tasks. From the scientific perspective, the work we present investigates the suitability of the tensor view of KGs and the tensor decomposition characterization of its embeddings when the data we aim to capture extends beyond

the KG triples. The research thesis we set out to prove is as follows:

Thesis Statement. *Tensor representations provide a unified framework for modelling knowledge graphs and different types of related data. Tensor methods can be applied both for learning effective representations of entities and relations through knowledge fusion, and for efficiently reasoning over the learned representations.*

1.1 Knowledge Graph Embeddings in Wider Context

It is worth asking the question: given that knowledge graphs are already in a form that is unambiguous and machine readable, why it is desirable to convert the factual knowledge contained in them into embeddings at all? The standard answer is that the two representations have complementary strengths. KG triples are well suited for logical methods, whereas embeddings capture notions such as similarity between the entities or the relations, and allow incremental changes to their representations. These properties of embeddings are important because they enable the representations to be adjusted according to some well-defined performance metric on a particular task. This view is useful both for contextualizing the research effort dedicated to KG embeddings within the history and current practice of AI research, and as a unifying thread for different approaches we take throughout this thesis.

Declarative & procedural knowledge. Among different disciplines concerned with characterizing knowledge, a common distinction is made between *procedural* and *declarative* knowledge.[2] Procedural knowledge is characterized by skillful behavior; it is knowing how to *do* something regardless of the ability to articulate it, such as riding a bike or recognising a face in a crowd. This type of knowledge is implicit and task specific. Common machine learning systems aim to capture this type of information through training. Declarative knowledge, on the other hand, is any type of knowledge that can be explicitly stated, such as "The Queen of England is also the Queen of Canada" or "I have written this sentence". Facts in knowledge graphs are

[2]The distinction between declarative and procedural knowledge was brought to prominence in philosophical discourse by Ryle (2009) as the difference between *knowing-that* and *knowing-how*. A related distinction exists between declarative and procedural memory in Cognitive Psychology, and a famous case study (Scoville and Milner, 1957) has shown that a patient with brain lesions who could not form new declarative memories could nevertheless acquire new procedural knowledge, suggesting that the distinction is somewhat natural.

examples of declarative knowledge. Unlike procedural knowledge, declarative knowledge is task-agnostic, and can be used in a variety of different ways. The two types of knowledge also suggest two different ways of learning. Procedural knowledge often implies learning by doing, whereas declarative knowledge enables agents to acquire knowledge generated or collected by others.

Knowledge representations in the history AI. In early AI research as recounted by Nilsson (2009), declarative knowledge was considered to be knowledge explicitly encoded in sentences of a formal language, which took truth values according to the state of the world. Procedural knowledge on the other hand corresponded to the programs that achieved a specific goal. Hence the distinction between the two was intimately tied to their representations in AI systems. Researchers focusing on pattern recognition tasks attempted to capture procedural knowledge as weights on low level features, or as neural nets that were sometimes implemented directly as hardware. Others viewed the goal of AI as the "mechanization of thought processes"[3], and worked on tasks that were believed to require complex reasoning, such as playing chess, proving theorems in geometry, or answering questions by deducing the answer from a set of given facts.

In the 80s, the focus of AI shifted to knowledge-based systems that performed reasoning over large collections of domain specific facts using rules provided by domain experts. Knowledge-based systems were designed around the ability to use symbolic, declarative knowledge as input. It was essential for their design to decouple the declarative knowledge embodied by facts and rules, and the procedural knowledge embodied by their abstract reasoning capabilities. This approach stands in sharp contrast to current *end-to-end* machine learning systems that aim to obtain all the knowledge relevant for a task by trying to do the task over and over again, and incrementally changing the knowledge representations to bring the behaviour closer and closer to competence. In other words, the knowledge representations that these systems can natively generate and use are procedural. The type of tasks they have been very successful at reflects this. These are the tasks which are so natural for humans that explicitly providing a program to achieve them have proven to be impossible.

Translating between declarative and procedural knowledge. Because end-to-end systems rely strongly on procedural knowledge and learning by doing, how to enable them to use the relevant declarative knowledge that is already available is an area of

[3]This was the name of one of the first symposiums on AI, held in 1958.

active research. KG embeddings are one way to bridge this divide. A common approach for translating from declarative representations to procedural ones is to try to capture the semantic content of the symbols used to represent declarative knowledge through a procedural task. In this spirit, knowledge graph embeddings aim to capture the procedural knowledge of *correctly predicting the missing entity* in a KG triple, and in this way convert the declarative knowledge represented by the triples in the KG into embedding representations.

The types of additional information we aim to integrate into KG embeddings in this thesis also consist of different forms of declarative knowledge. For integrating type information and text data we follow a similar strategy as above, and try to capture the relevant knowledge by modifying the task. Specifically, we introduce auxiliary tasks and data augmentation techiques to force embeddings to capture information from both the KG and the other data source. For using logical rules, we flip the roles of embeddings and the knowledge source, and consider the difficult problem of performing deduction using embedding representations. To this end, we explore theoretical foundations for incorporating distributed, procedural knowledge embodied by KG embeddings into deduction methods that operate on symbolic representations of declarative knowledge.

1.2 Overview and Contributions

Chapter 2: Background. In this chapter we first introduce the technical background necessary for work presented in later chapters. This includes fundamental concepts in tensor algebra, and two standard forms for decomposing tensors: *CP* and *Tucker* decompositions. We also provide a precise description of the task of knowledge graph completion, and present the standard datasets for the task.

The rest of the chapter consists of a literature review of previous work on KG completion. We first present some important models suggested for the task in three groups: *translational models*, *bilinear models* and *neural network models*. We then provide an extensive review of KG completion models that use additional information besides the set of triples in the KG, such as types, text, paths and rules. Finally, we present different choices that have been employed for the training of these models, underlining that these often have a larger impact on the success of a model than the architecture.

Chapter 3: Incorporating Type Information in Knowledge Graph Embeddings. Here we investigate how to modify the training of bilinear models so that implicit or explicit type information for entities and relations can be used to improve model performance.

We start by giving a precise characterization of the resource we aim to utilize in terms of a *type-ontology*. We present a framework to use this information both as an auxiliary task in the form of joint tensor-tensor decomposition, and for biasing the generation of negative examples. Then we outline a procedure for generating an approximate type-ontology from training triples in the absence of an explicit one.

For our experiments, we first re-implement three bilinear models, and analyze effects of different training settings and hyperparameters on final performance. We then generate type-ontologies from standard datasets using the suggested procedure, and show that our framework results in consistent improvements across different datasets and bilinear models even when using an approximate type-ontology. We draw from our analysis of baselines to compare the effect of our framework when training with different hyperparameter settings, and conclude that our framework is especially beneficial with hyperparameters that result in the smallest memory footprint, which reflects training conditions on very large datasets. We support this claim by showing that our framework indeed increases performance when run on a dataset with 1.9 million entities.

This chapter is based on findings published in:

- E. Balkır, M. Naslidnyk, D. Palfrey and A. Mittal. Using pairwise occurrence information to improve knowledge graph completion on large-scale datasets. *in the Proceedings of EMNLP-IJCNLP*. 2019.
- E. Balkır, M. Naslidnyk, D. Palfrey and A. Mittal. Improving knowledge graph embeddings with inferred entity types. *Relational Representation Learning workshop at NeurIPS*. 2018.

Chapter 4: Learning Entity Embeddings from Knowledge Graph and Corpus. In this chapter we present a method for learning embeddings jointly from a KG and a text corpus with links to KG entities, with the goal of improving performance on KG completion on entities rarely seen in training data. Our model combines the word embedding method GloVe (Pennington et al., 2014) with the bilinear KG embedding model DistMult (Yang et al., 2015) under the joint matrix-tensor factorization frame-

work of Acar et al. (2013). We present two versions of our method, one that ties the embeddings obtained from the KG and from the corpus completely, and one that penalizes their divergence.

We modify standard datasets to capture different causes for an entity to have few or no facts in training data. We perform experiments on standard and modified datasets to assess model performance on rare and unseen entities. We compare our models with baseline DistMult, and also with two models that learn embeddings of words and entities seperately before learning a mapping between the two. Our findings show that our joint model with tied embeddings has superior performance on rare or unseen entities compared to baselines on some of the datasets.

This chapter contains material which was written jointly with my supervisor Shay Cohen.

Chapter 5: Tensors over Semirings for Weighted Logic Programs. This chapter lays the theoretical foundations for using tensors as values in Weighted Logic Programs (WLPs). WLPs facilitate representing and reasoning about dynamic programming algorithms by abstracting the program structure from its value calculations. Previous work by Goodman (1999) has shown that the same WLP specification can be used with any set of values belonging to an algebraic structure called a *semiring*, and that one can obtain dynamic programming algorithms to calculate different values of interest just by changing the semiring.

We extend this work to WLPs with tensors over semiring values as weights, in order to allow the same program representations to be applied to latent variable models. We motivate our work with two applications: parsing with latent-variable context free grammars, and integrating representations for paths and rules to KG-embedding models. Specifically, we provide precise conditions on rules and their tensor weights for them to fulfill the relevant semiring axioms, and present the tensor formulations for *inside* and *outside* calculations.

This chapter is based on the findings published in:

- E. Balkır, D. Gildea and S. Cohen. Tensors over Semirings for Latent-Variable Weighted Logic Programs. *in the Proceedings of International Conference on Parsing Technologies (IWPT)*. 2020.

Chapter 6: Conclusion. In this chapter we summarize the presented findings in relation to our research thesis, discuss some of their limitations, and suggest some directions for future work.

Chapter 2

Background

2.1 Tensor Preliminaries and Notation

Tensors are generalizations of matrices, where each entry can be indexed by n different indices instead of just two. An n-th order tensor is an n-way array, with n independent indexes that specify each entry. A first order tensor is a vector, and a second order tensor is a matrix. Here we will be concerned mostly with tensors of order three, although the terminology and the methods can often be extended to higher order tensors.

Many of the essential concepts from matrix theory can be extended to the tensor case – such as singular value decompositions – but the properties of these extensions are surprisingly different than from the matrix case. These properties has been exploited in various subfields of machine learning, notably in spectral learning for moment based estimation of latent variables (Anandkumar et al., 2014). It also comes with new challenges: decomposing tensors, or even determining the rank of a tensor is in general NP-hard (Kolda and Bader, 2009). In practice, the work relying on tensor decompositions for methods with provable guarantees either assumes that the tensor in question belongs to a subclass where these hardness results do not apply, or provides methods to transform the tensor to one in such a subclass.

Throughout this thesis, we will use regular lowercase letters a, b, c etc. to denote scalars, and boldface lowercase letters $\mathbf{v}, \mathbf{w}$ etc. to denote vectors and boldface uppercase letters $\mathbf{M}, \mathbf{T}$ etc. to denote matrices and tensors. v_i will denote the ith entry of the vector $\mathbf{v}$, $M_{i,j}$ the ijth entry of the matrix $\mathbf{M}$ and $T_{i,j,k}$ the ijkth entry of the rank-3 tensor $\mathbf{T}$. We will use the shorthand $[n]$ for the set $\{k|k = 1, \ldots, n\}$. We will use $\mathbf{I}$ to

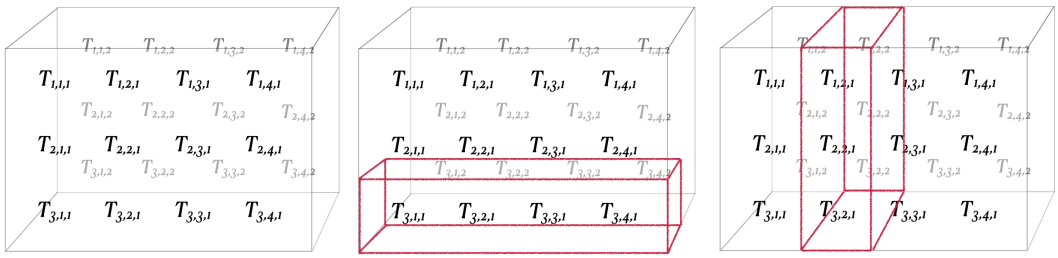


Figure 2.1: (a) An example tensor $\mathbf{T} \in \mathbb{R}^{3\times4\times2}$, (b) an example fibre $T_{3,:,1}$ and (c) an example slice $T_{:,2,:}$.

denote the identity matrix, where $I_{i,j} = 1$ if $i = j$ and 0 otherwise.

Formally, an n-th order tensor $\mathbf{A} \in \mathbb{R}^{d_1\times\ldots\times d_n}$ is an element of tensor the product of n Euclidean spaces $\bigotimes_{i=1}^{n} \mathbb{R}^{d_i}$, possibly of different dimensions d_i. Each of these n Euclidean spaces will be referred as a *mode* of the tensor. Tensor *fibres* are columns of the tensor along a particular mode, where all indexes but one are fixed. *Slices* are matrix cuts of the tensor along two chosen modes. Analogous to a tensor fibre, a tensor slice is defined by fixing all but two indices of the tensor. We will denote tensor fibers and slices by using ":" in the corresponding mode. For example $T_{:,j,k}$ will denote a tensor fiber on the first mode and $T_{:,j,:}$ will denote a tensor slice on the first and third modes. Figure 2.1 shows a 3rd order tensor $\mathbf{T}$, and a fibre and a slice on $\mathbf{T}$.

Just as matrices can be thought of a linear maps, higher order tensors can be thought of as multilinear maps: a tensor $\mathbf{A} \in \mathbb{R}^{d_1\times d_2\times d_3}$ represents a bilinear map $A : \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} \to \mathbb{R}^{d_3}$. One could also view the same tensor as a trilinear map $A : \mathbb{R}^{d_1} \times \mathbb{R}^{d_2} \times \mathbb{R}^{d_3} \to \mathbb{R}$, or a linear map: $A : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2\times d_3}$.

In general an n-way tensor $\mathbf{A} \in \mathbb{R}^{d_1\times\ldots\times d_n}$ as a multilinear operator takes as arguments a set of n matrices $\{\mathbf{V}^i \in \mathbb{R}^{d_i\times m_i}\}$, and returns an nth order tensor $\mathbf{A}(\mathbf{V}^1,\ldots,\mathbf{V}^n) \in \mathbb{R}^{m_1\times\ldots\times m_n}$, where each entry is defined as following:

$$\left(\mathbf{A}(\mathbf{V}^1,\ldots,\mathbf{V}^n)\right)_{i_1,i_2,\ldots,i_n} := \sum_{j_1,\ldots,j_n\in[d_1],\ldots,[d_n]} A_{j_1,j_2,\ldots j_n} V^1_{j_1,i_1} \times V^2_{j_2,i_2} \times \ldots \times V^n_{j_n,i_n} \quad (2.1)$$

This operation is referred to as tensor contraction.

A special case for tensor contraction is where all $\mathbf{V}^i$ are vectors. For example, consider the contraction of a 3rd order tensor $\mathbf{T} \in \mathbb{R}^{d_1\times d_2\times d_3}$ with some or all of the vectors

$\mathbf{u} \in \mathbb{R}^{d_1}, \mathbf{v} \in \mathbb{R}^{d_2}, \mathbf{w} \in \mathbb{R}^{d_3}$:

$$\mathbf{T}(\mathbf{u}, \mathbf{v}, \mathbf{w}) := \sum_{j \in [d_1], k \in [d_2], l \in [d_3]} u_j v_k w_l T_{j,k,l} \in \mathbb{R} \tag{2.2}$$

$$\mathbf{T}(\mathbf{I}^{d_1}, \mathbf{v}, \mathbf{w}) := \sum_{k \in [d_2], l \in [d_3]} v_k w_l T_{:,k,l} \in \mathbb{R}^{d_1} \tag{2.3}$$

$$\mathbf{T}(\mathbf{I}^{d_1}, \mathbf{I}^{d_2}, \mathbf{w}) := \sum_{l \in [d_3]} w_l T_{:,:,l} \in \mathbb{R}^{d_1, d_2} \tag{2.4}$$

where $\mathbf{I}^{d_i}$ is the identity matrix in $\mathbb{R}^{d_i \times d_i}$. The use of the identity matrix as one of the arguments to be contracted allow the contraction operation to capture trilinear, bilinear and linear maps that can be represented by the same tensor $\mathbf{T}$.

In the more general case where $\mathbf{U} \in \mathbb{R}^{d_1 \times d_4}$, $\mathbf{V} \in \mathbb{R}^{d_2 \times d_5}$ and $\mathbf{W} \in \mathbb{R}^{d_3 \times d_6}$, the analogous values $\mathbf{T}(\mathbf{U}, \mathbf{V}, \mathbf{W}) \in \mathbb{R}^{d_4 \times d_5 \times d_6}$, $\mathbf{T}(\mathbf{I}^{d_1}, \mathbf{V}, \mathbf{W}) \in \mathbb{R}^{d_1 \times d_5 \times d_6}$ and $\mathbf{T}(\mathbf{I}^{d_1}, \mathbf{I}^{d_2}, \mathbf{W}) \in \mathbb{R}^{d_1 \times d_2 \times d_6}$ are:

$$\mathbf{T}(\mathbf{U}, \mathbf{V}, \mathbf{W})_{p,q,r} := \sum_{j \in [d_1], k \in [d_2], l \in [d_3]} U_{j,p} V_{k,q} W_{l,r} T_{j,k,l} \tag{2.5}$$

$$\mathbf{T}(\mathbf{I}^{d_1}, \mathbf{V}, \mathbf{W})_{:,q,r} := \sum_{k \in [d_2], l \in [d_3]} V_{k,q} W_{l,r} T_{:,k,l} \tag{2.6}$$

$$\mathbf{T}(\mathbf{I}^{d_1}, \mathbf{I}^{d_2}, \mathbf{W})_{:,:,r} := \sum_{l \in [d_3]} W_{l,r} T_{:,:,l} \tag{2.7}$$

2.2 Tensor Decompositions

For data that naturally comes in the form of tensors, an important question is how to approximate it as a reconstruction from smaller matrices or tensors. For real valued matrices, *Singular Value Decomposition (SVD)* provides a principled way to obtain such a representation.

Given a matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$, SVD finds orthonormal matrices $\mathbf{U} \in \mathbb{R}^{m \times m}$ and $\mathbf{V} \in \mathbb{R}^{n \times n}$ and a diagonal matrix $\Sigma \in \mathbb{R}^{m \times n}$ such that $\mathbf{U}\Sigma\mathbf{V}^\top = \mathbf{M}$, where $V^\top$ denotes the transpose of V. The diagonal elements $\sigma_{i,i}$ of Σ corresponds to the singular values of $\mathbf{M}$, and the number of non-zero singular values correspond to the *rank* of $\mathbf{M}$. The columns of $\mathbf{U}$ and $\mathbf{V}$ correspond to the right and the left singular vector respectively. Note that one can equivalently view the operation defined by $\mathbf{U}\Sigma\mathbf{V}^*$ as a sum-of-outer-products representation. Let k be the smaller of m and n. Then,

$$\mathbf{M} = \sum_{i=0}^{k} \sigma_{i,i} U_{:,i} \left(V_{:,i}\right)^\top. \tag{2.8}$$

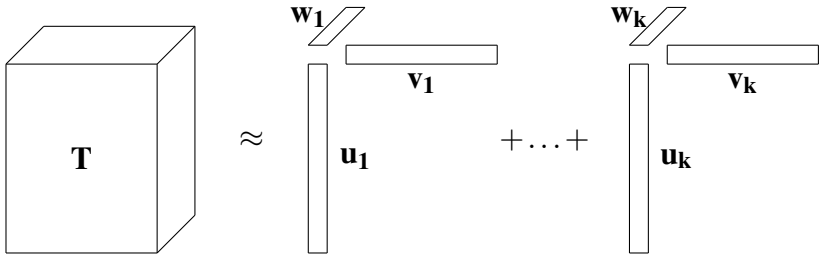


Figure 2.2: CP-decomposition of a 3rd order tensor $\mathbf{T}$ into k components.

The sum of outer products view makes it clear that discarding all rows and columns with zero singular values from Σ, and the corresponding columns of singular vectors from $\mathbf{U}$ and $\mathbf{V}$ wouldn't affect the reconstructed matrix. This gives a *low rank decomposition* of $\mathbf{M}$. Discarding all but k largest singular values and the corresponding singular vectors likewise provides the best rank-k approximation of $\mathbf{M}$.

With tensors of order larger than two, it is in general not possible to have a factorization so that the factor matrices are both minimal and have orthogonal columns. The two standard tensor decompositions, *CP-decomposition* and *Tucker Decomposition* essentially preserve one of these properties while sacrificing the other. See Kolda and Bader (2009) for a comprehensive review of other decompositions proposed in the literature and their applications.

2.2.1 CP-Decomposition

CP-decomposition is based on the idea of the *rank* of tensor. Recall that a matrix $\mathbf{M}$ is rank-1 if it can be written as the outer product of two vectors: $\mathbf{M} = \mathbf{v}\mathbf{w}^\top$. Likewise, we can define a nth order tensor $\mathbf{A}$ to be rank-1 if it can be written in the form: $\mathbf{A} = \mathbf{v}^1 \circ \mathbf{v}^2 \circ \ldots \circ \mathbf{v}^n$ where the generalized outer product is defined as follows:

$$\left(\mathbf{v}^1 \circ \mathbf{v}^2 \circ \ldots \circ \mathbf{v}^n\right)_{i_1,\ldots,i_n} := v^1_{i_1} \cdot v^2_{i_2} \cdot \ldots \cdot v^n_{i_n}. \tag{2.9}$$

A tensor of order n is rank-1 if it can be written as the outer product of n vectors. The *CP-rank* (or simply *rank*) of a tensor is the smallest number of rank-1 tensors it could be written as the sum of:

$$\mathbf{A} = \sum_{j=1}^{k} \mathbf{v}^{1,j} \circ \mathbf{v}^{2,j} \circ \ldots \circ \mathbf{v}^{n,j}. \tag{2.10}$$

The decomposition into such rank one components is known as *Canonical Polyadic*

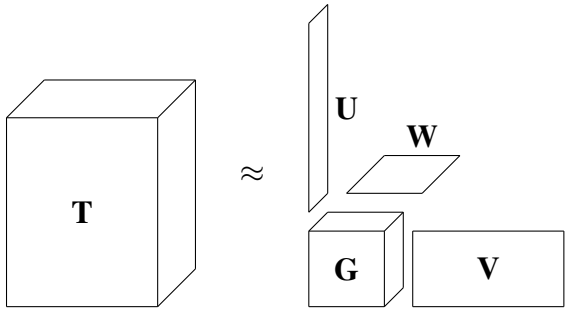


Figure 2.3: Tucker decomposition of a 3rd order tensor $\mathbf{T}$ into a smaller core tensor $\mathbf{G}$, and factor matrices $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{W}$.

Decomposition or *CANDECOMP/PARAFAC*, or *CP-Decomposition* for short.[1] An essential property of CP-decomposition is that unlike the matrix case where without orthonormality restrictions on left and right singular vectors, there are infinitely many such decompositions, CP-decomposition is unique in general, up to scaling and permutation. Tensor rank might also exceed the dimensions of the tensor. Again unlike the matrix case, obtaining CP-decomposition for general tensors is NP-hard, and so is determining the rank of a tensor (Hillar and Lim, 2013; Sidiropoulos et al., 2017).

Another difference between matrices and higher order tensors is that, the best rank-k approximation for matrices is given by leading k factors of its SVD, but this is not the case for tensors. There are cases where the best rank-k approximation does not even exist, i.e. any given rank-k approximation can be improved upon to approximate the tensor slightly better. These are referred to as *CP-degeneracies* (Comon et al., 2009). Despite this, the CP-decomposition is often used to find a low-rank and approximate representation of a tensor akin to the truncated SVD. This is achieved by finding k factors which provide a good enough approximation to the tensor $\mathbf{T}$, where k is smaller than the true CP-rank. An illustration of an approximate CP-decomposition is given in Figure 2.2.

2.2.2 Tucker Decomposition

Tucker decomposition is a different method of decomposing a tensor than the CP-decomposition. The main idea is to decompose a tensor of order n into a core tensor $\mathbf{G}$

[1] The two different names of the decomposition, even though they were independently discovered in different scientific communities, conveniently happen to share the same initials.

of order n, and n factor matrices. For a 3rd order tensor $\mathbf{T}$ this would be of the form:

$$\mathbf{T} = \sum_{p,q,r} G_{p,q,r} \left(U_{:,p} \circ V_{:,q} \circ W_{:,r} \right), \tag{2.11}$$

where $\mathbf{T} \in \mathbb{R}^{I \times J \times K}$, $\mathbf{G} \in \mathbb{R}^{P,Q,R}$, $\mathbf{U} \in \mathbb{R}^{I,P}$, $\mathbf{V} \in \mathbb{R}^{J,Q}$ and $\mathbf{W} \in \mathbb{R}^{K,R}$, and p,q,r ranges over P,Q,R respectively. The illustration of this decomposition is given in Figure 2.3.

Note that a trivial Tucker decomposition would be to assign $U,V,W = I$ and $\mathbf{G} = \mathbf{T}$. Hence, a desired Tucker decomposition seeks $\mathbf{G}$ such that it is either a smaller dimension than the full tensor $\mathbf{T}$, or that it is sparse.

Unlike the CP-decomposition, Tucker decomposition is non-unique. This allows, for a given Tucker decomposition with a core tensor $\mathbf{G}$ and factor matrices $\mathbf{U},\mathbf{V},\mathbf{W}$, to find an invertible matrix $\mathbf{M}$ such that $\mathbf{UM}$ would be orthonormal, and then absorb the inverse $\mathbf{M}^{-1}$ in the core tensor $\mathbf{G}$ so that $(\mathbf{G}(\mathbf{M}^{-1}), \mathbf{UM}, \mathbf{V}, \mathbf{W})$ is still a Tucker decomposition of the full tensor $\mathbf{T}$. Continuing in this fashion, it is possible to find a Tucker decomposition with orthonormal factorization matrices.

It is possible to think of CP-decomposition as a special type of Tucker decomposition, where the core tensor $\mathbf{G}$ is potentially larger than the full tensor $\mathbf{T}$, but has non-zero values only on the diagonal:

$$G_{i,j,k} = \begin{cases} \lambda_i & \text{if } i = j = k \\ 0 & \text{otherwise}, \end{cases} \tag{2.12}$$

where λ_i is the scalar value in Equation 2.10. This means intuitively that only the the interactions of the matching columns of the factor matrices contribute to the values in the full tensor.

It is also possible to view Tucker as a special type of CP-decomposition if we allow the CP-decomposition to be non-minimal, and allow the factor matrices of the CP-decomposition to have repeated rows. For a Tucker decomposition with core tensor $\mathbf{G} \in \mathbb{R}^{n \times m \times o}$, and factor matrices $\mathbf{U} \in \mathbb{R}^{l \times n}$, $\mathbf{V} \in \mathbb{R}^{k \times m}$ and $\mathbf{W} \in \mathbb{R}^{j \times o}$, we can construct the factor matrices $\mathbf{A} \in \mathbb{R}^{l \times (n \cdot m \cdot o)}$, $\mathbf{B} \in \mathbb{R}^{k \times (n \cdot m \cdot o)}$ and $\mathbf{C} \in \mathbb{R}^{l \times (n \cdot m \cdot o)}$ of the corresponding CP-decomposition by assigning $\mathbf{A},\mathbf{B}$ and $\mathbf{C}$ to be the repeating of the columns of $\mathbf{U},\mathbf{V}$ and $\mathbf{W}$ respectively. Then, each potential interaction between the columns of $\mathbf{U},\mathbf{V}$ and $\mathbf{W}$ can be expressed as a CP type outer product, the corresponding entry in the core tensor $\mathbf{G}$ becoming the scalar weight λ_i in the CP-decomposition.

2.2.3 Joint Decompositions

Sometimes data naturally occurs in the form of multiple matrices or tensors from different sources, where one or more modes from different data tensors correspond to the same underlying objects. For example, Chapter 3 of this thesis explores the case where one tensor represents the knowledge graph, and the other represents the type relationships within the triples. Chapter 4 similarly considers the knowledge graph tensor coupled with a word-word co-occurrence matrix. In such cases, it is desirable to find a joint decomposition of the data tensors so that the objects that are shared amongst different tensors obtain the same latent representation.

For data matrices that are coupled in one mode, Singh and Gordon (2008) present *Collective Matrix Factorization* (CMF) to jointly decompose the two datasets in the context of relational learning. Specifically for two matrices $\mathbf{X} \in \mathbb{R}^{d_1,d_2}$ and $\mathbf{Y} \in \mathbb{R}^{d_1,d_3}$ that are coupled in their first mode, CMF finds factor matrices $\mathbf{U} \in \mathbb{R}^{d_1,d_4}$, $\mathbf{V} \in \mathbb{R}^{d_2,d_4}$ and $\mathbf{W} \in \mathbb{R}^{d_3,d4}$ such that the reconstruction error for both are minimized:

$$\min_{\mathbf{U},\mathbf{V},\mathbf{W}} \|\mathbf{X} - \mathbf{U}\mathbf{V}^\top\|_2^2 + \|\mathbf{Y} - \mathbf{U}\mathbf{W}^\top\|_2^2 \tag{2.13}$$

Acar et al. (2013) present *Coupled Matrix and Tensor Factorization* (CMTF) that extend CMF to tensors of order larger than 2. Given a tensor $\mathbf{T}$ and a matrix $\mathbf{M}$ that are coupled in the first mode, the goal of CMTF is to find lower dimensional matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\mathbf{V}$ such that $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ provide an approximate CP-decomposition for $\mathbf{T}$, and the product of $\mathbf{A}$ and $\mathbf{V}^\top$ is an approximation of $\mathbf{M}$. This can be achieved by minimizing the reconstruction loss:

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{A}} \|\mathbf{T} - \mathbf{A} \circ \mathbf{B} \circ \mathbf{C}\|_2^2 + \|\mathbf{M} - \mathbf{A}\mathbf{V}^\top\|_2^2, \tag{2.14}$$

Although the original CMTF framework uses the Euclidean norm, it is possible to replace this with other norms. It is also straightforward to extend the formulation to tensors of higher orders, or to more than two tensors.

2.3 Knowledge Graph Completion

Formally, a knowledge graph (KG) is a tuple $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{F})$ where $\mathcal{E}$ is the set of entities such as *Tom Cruise*, *Hawaii* or *1984 (book)*, $\mathcal{R}$ is the set of relations such as `LocatedIn` or `StarredIn`, and $\mathcal{F} \subset (\mathcal{E} \times \mathcal{R} \times \mathcal{E})$ is the set of facts. These are triples

that express the relations that hold between the entities. The task of *knowledge graph (KG) completion* (also referred to as *link prediction*) operates from the assumption that many of the large knowledge graphs are incomplete, especially with regards to triples. KG completion methods aim to learn effective ways to assign scores to possible triples so that the facts that should be added to $\mathcal{F}$ receive higher scores than the other candidates.

2.3.1 Evaluation Protocol

Due to the absence of triples in KGs that are known to be false, the evaluation of KG completion models is framed as a ranking task (Bordes et al., 2013). The model is evaluated on previously unseen triples from the KG as follows: for each test triple, either the head or the tail entity is replaced with every entity in the knowledge graph, and for each of these triples a score is obtained by the model being tested. Then these triples are sorted from the highest to lowest scored. Bordes et al. (2013) initially report the *mean rank (MR)* of the correct triples, and the proportion of the entities ranked in the first 10, which they refer to as *hits@10*. Later works report *mean reciprocal rank* either in addition to, or instead of MR because MR could be disproportionately affected by a few low ranked triples, whereas taking the inverse of the rank before averaging eliminates this issue. Later works also report *hits@1* and *hits@3*.

When generating the list of triples to be ranked and compared against a correct triple, some of triples that are in the KG could be among those that are generated and labelled as incorrect. To avoid penalising the model for ranking these correct triples above the test triple in question, Bordes et al. (2013) remove from the list any triples that appear either in training, validation or the test set. They call this version of their protocol *filtered*, and the protocol without this step *raw*. Later works often only report results on the filtered setting.

2.3.2 Datasets

Two datasets, WN18 and FB15K were introduced in Bordes et al. (2013) together with the evaluation protocol outlined above in Section 2.3.1:

- **WN18** is a subset of *WordNet* (Miller, 1995), a hand constructed word ontology consisting of *synsets* which correspond to word senses, and lexical relationships between synsets such as *hyponymy* (is-a relationships) and *meronymy* (part-of

relationships). This dataset contains 40,943 entities, 18 relations and 151,442 triples in total.

- **FB15K** is a subset of *Freebase* (Bollacker et al., 2008), a very large, crowd sourced knowledge graph that contains general facts about the world.[2] FB15K is chosen from entities and relationships that also exist in Wikilinks database and have at least 100 mentions in Freebase. It contains 14,951 entities, 1,345 relations and 592,213 triples in total.

It has been noted by Toutanova et al. (2015) and Dettmers et al. (2018) that WN18 and FB15K suffer from test leakage where for most test instances (e_1, r, e_2), the training set has the inverse relation (e_2, r', e_1). This reduces the task largely to memorizing the relevant pairs in the training data. Dettmers et al. (2018) show that a simple rule based model that first learns whether r_1 is the reverse of r_2 by assessing if for triples (e_1, r_1, e_2), there is the reverse triple (e_2, r_2, e_1) in training data with high frequency and vice versa, and then during test time ranks triples based only on this information, beats all the state of the art models on FB15K and WN18 datasets at the time. The following datasets are constructed to avoid this kind of data leakage:

- **FB15K-237** (Toutanova et al., 2015) is a subset of Freebase that is derived from FB15K by first choosing the most frequent 401 relations, and then filtering out reverse and near-duplicate relations, resulting in 237 relations.
- **WN18RR** (Dettmers et al., 2018) is a subset of WordNet, derived from WN18 dataset by removing the reverse relations from the dataset. It contains 93,003 triples, 40,943 entities and 11 relations, although most of the relations are hyponymy/hypernymy relations.
- **YAGO3-10** (Dettmers et al., 2018) is a dataset constructed from YAGO (Mahdisoltani et al., 2013), which is a KG automatically generated from WordNet, Wikipedia and Wikidata. YAGO3-10 includes only the entities in YAGO that have more than 10 relations. It has a total of 123,182 entities and 37 relations.

[2]Freebase has been officially shut down in 2016 after most of its data was migrated to Wikidata (Pellissier Tanon et al., 2016).

2.4 Knowledge Graph Embeddings

In the last few years, numerous models have been suggested for the task of KG completion. Most of these works focus on learning low dimensional embeddings for entities and relations by optimizing a scoring function that scores correct triples above incorrect ones. These models can be roughly categorized into three different groups based on their scoring functions: *translational models*, *bilinear models* and *neural network models*. We review some of the prominent models from each of these categories below.

2.4.1 Translational Models

The basic idea of translational models is to represent the relation as an operation in the vector space that translates the embedding of the head entity to the embedding of the tail entity.

TransE (Bordes et al., 2013). This method is one of the earliest models for link prediction, and the simplest translational model that all other models in this subsection has built upon. It aims to learn embeddings of entities and relations in the same vector space $\mathbb{R}^k$, where for a correct (h,r,t) pair, $\mathbf{h}+\mathbf{r}$ is close to $\mathbf{t}$. This is achieved by a distance function $d(\mathbf{h}+\mathbf{r},\mathbf{t})$ where d is either the L_1 or the L_2 distance.

TransE is simple to implement and easy to scale, so there has been a number of methods that build on the basic idea to improve on it one way or another. Among them are TransH (Wang et al., 2014b), TransR (Lin et al., 2015b), TransD (Ji et al., 2015), TranSparse (Ji et al., 2016) and STransE (Nguyen et al., 2016). See (Nguyen, 2020) for a comprehensive review. We describe some of these models below.

TransH (Wang et al., 2014b). This model is motivated by the shortcoming of TransE of modelling reflexive (e.g. `neighborOf`), one-to-many (e.g. `leadActorIn`), many-to-one (e.g. `bornIn`) and many-to-many (e.g. `descendantOf`) relations. The authors observe that if the embeddings satisfy the constraint imposed by TransE perfectly for all correct triples, then for a reflexive relation r, embedding for r needs to be 0, and the embeddings for the head entity h and tail entity t need to be equal to each other for all correct triples (h,r,t). Likewise for a one-to-many relation r, if the triples $\{(h,r,t^i)\}_{i=1\ldots n}$ are all correct, then TransE forces that $\mathbf{t^i}=\mathbf{t^j}$ for all $i,j\in[n]$.

TransH solves this issue by projecting the head and the tail entity to a relation specific

hyperplane before applying the translational scoring function of TransE. It learns two embeddings $\mathbf{w_r}$ and $\mathbf{d_r}$ for each relation by optimizing the scoring function:

$$s(h,r,t) = \|(\mathbf{h} - \mathbf{w_r}^\top \mathbf{h}\mathbf{w_r}) + \mathbf{d}_r - (\mathbf{t} - \mathbf{w_r}^\top \mathbf{t}\mathbf{w_r})\|_2^2 \tag{2.15}$$

For $\mathbf{v} - \mathbf{w_r}^\top \mathbf{v}\mathbf{w_r}$ to be the projection of $\mathbf{v}$ onto the hyperplane defined by $\mathbf{w_r}$ it needs to be that $\|\mathbf{w_r}\| = 1$. A further constraint that needs to be imposed is that $\mathbf{d_r}$ corresponds to a translation within the hyperplane defined by $\mathbf{w_r}$. TransH integrates these latter constraints as a soft constraint in its optimization procedure, and enforces the former by projecting $\mathbf{w_r}$ onto the unit ball after every update.

TransR (Lin et al., 2015b). This model further improves TransH by learning a full projection matrix $\mathbf{M_r}$ for each relation r by optimizing the scoring function

$$s(h,r,t) = \|\mathbf{h}\mathbf{M_r} + \mathbf{r} - \mathbf{t}\mathbf{M_r}\|_2^2 \tag{2.16}$$

The geometric intuition behind TransR is to learn an entity space and separate relation spaces for each relation. Learning the matrices $\mathbf{M_r}$ correspond to learning mappings from the entity space to the space of relation r. Even though TransR improves model performance on the KG completion task, it increases the model parameters significantly compared to TransE and TransH.

KG2E (He et al., 2015) This work argues that representing entities and relations as points in embedding space ignores the inherent uncertainty of these objects, and proposes a method that represents entities and relations as Gaussian distributions instead. In this framework each entity and relation is represented by a mean vector μ and a covariance matrix Σ that together define the associated distribution.

Let $\mathcal{H} \sim \mathcal{N}(\mu_h, \Sigma_h)$ be the Gaussian embedding corresponding to the head entity, and let $\mathcal{R}$ and $\mathcal{T}$ be the analogously defined embeddings of the relation and the tail entity respectively. KG2E extends the translational requirement to Gaussian distributions as follows: given a triple (h,r,t) the model first calculates a combined entity distribution $\mathcal{H} - \mathcal{T} = \mathcal{P}_e \sim \mathcal{N}(\mu_h - \mu_t, \Sigma_h + \Sigma_t)$. The scoring function is then defined as the KL-divergence between the relation embedding $\mathcal{R}$ and the combined entity distribution $\mathcal{P}_e$. The authors report that using an asymmetric measure such as KL-divergence provides improvement over symmetric measures such as expected-likelihood.

ManifoldE. (Xiao et al., 2016) This is another extension of TransE that addresses the limitations in expressiveness of point based embeddings. The model represents entities as points, but relations are represented as manifolds. The geometric intuition for ManifoldE is that for a given head entity h and a tail entity t, the set of correct relations r_i for the pair would lie in a manifold within the embedding space. The authors consider two families of manifolds, spheres and hyperplanes. They further enrich these with a number of kernels to enlarge the space of possible shapes for manifold representations. They report that the method is particularly good at *hits@1*, which corresponds to predicting the correct entity among all possible ones.

RotatE (Sun et al., 2019). This work models relations as rotations in the complex vector space that go from the embedding of the head entity h to the embedding of the tail entity t. More specifically, let $\mathbf{h}, \mathbf{r}, \mathbf{t} \in \mathbb{C}^d$ be d-dimensional vector embeddings for the entities and relation in the complex space, and $\|r\| = 1$. The scoring function for RotatE is defined as:

$$s(h,r,t) = \|\mathbf{h} \circ \mathbf{r} - \mathbf{t}\|. \tag{2.17}$$

Despite the simplicity of the idea, the authors show that RotatE is fully expressive and performs well on benchmarks. Furthermore, they introduce a *self-adversarial* sampling procedure for generating negative triples during training. This procedure depends on the current parameters of the model, and gives more weight to the incorrect triples that the model scores highly.

2.4.2 Bilinear Models

Bilinear models are the class that has the strongest links to tensor decompositions. Below, we survey a few prominent models in this category.

RESCAL (Nickel, 2013). A notable example of a tensor decomposition algorithm that is designed for knowledge bases is RESCAL. It is a special case of Tucker decomposition of a tensor. Recall that Tucker decomposition for a tensor $\mathbf{T}$ finds three factor matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$, and a low dimensional core tensor $\mathbf{G}$ such that

$$\mathbf{T} = \sum_{l,m,n} G_{l,m,n} \left(A_{:,l} \circ B_{:,m} \circ C_{:,n} \right), \tag{2.18}$$

where $\circ$ is the generalized outer product operator. RESCAL is a Tucker decomposition with the additional constraints: $\mathbf{C} = \mathbf{I}$ and $\mathbf{A} = \mathbf{B}$:

$$\mathbf{T} = \sum_{l,m,n} G_{l,m,n} \left(A_{:,l} \circ A_{:,m} \circ I_{:,n} \right). \tag{2.19}$$

The justification of the first constraint is that in RESCAL, one views the nth slice of the core tensor as the latent representation of the relation, and the factor matrix $\mathbf{A}$ defines the embeddings of the entities. The constraint $\mathbf{A} = \mathbf{B}$ ensures that an entity has the same embedding representation regardless of whether it appears as the subject or the object in the triple. The authors argue that this results in more effective information propagation during the decomposition, and leads to better experimental results.

The original algorithm for RESCAL is based on the Alternating Least Squares (ALS) algorithm, adapted specifically for the structure in RESCAL. Later in Nickel et al. (2016b) the authors report that by training the factorization with Stochastic Gradient Descent together with AdaGrad (Duchi et al., 2001) obtains significant improvements on the performance of the model.

In both cases, the RESCAL factorization is cast as an optimization problem of finding the entity embedding matrix $\mathbf{A}$ and the relation tensor $\mathbf{R}$ that minimizes the following loss function:

$$\|\mathbf{X} - \mathbf{A} \circ \mathbf{R} \circ \mathbf{A}\|^2 + \lambda_{\mathbf{A}} \|\mathbf{A}\|^2 + \lambda_{\mathbf{R}} \|\mathbf{R}\|^2 \tag{2.20}$$

Here the first term measures how well the approximated decomposition fits the adjacency tensor $\mathbf{X}$, $\lambda_{\mathbf{A}}$ and $\lambda_{\mathbf{R}}$ are regularization terms.

Semantic Matching Energy Model (SME) (Bordes et al., 2014) this model first combines embeddings $\mathbf{h}, \mathbf{r}$ for the head entity h and the relation r, and the embeddings $\mathbf{r}, \mathbf{t}$ of the tail entity h and r separately. Then as the second step the combined representation of (h,r) and (r,t) are matched via an energy function. While the model leaves open the possibility of complex matching functions between the representations of the two pairs, the experiments only consider the simple dot-product between the two embeddings.

SME has both linear and bilinear versions, which dictates how (h,r) and (r,t) are combined. The equations for SME-linear are as follows:

$$\mathbf{h_r} = \mathbf{W}_1\mathbf{h} + \mathbf{W}_2\mathbf{r} + \mathbf{b}_1 \tag{2.21}$$

$$\mathbf{t_r} = \mathbf{W}_3\mathbf{t} + \mathbf{W}_4\mathbf{r} + \mathbf{b}_2 \tag{2.22}$$

$$s(h,r,t) = \mathbf{h_r}^\top \mathbf{t_r} \tag{2.23}$$

Where the matrices $\mathbf{W_1}, \mathbf{W_2}, \mathbf{W_3}, \mathbf{W_4}$ and the vectors $\mathbf{b}_1, \mathbf{b}_2$ are the model-wide parameters to be learned.

SME-linear models the interaction of the triple as the sum of pairwise interactions. SME-bilinear however models three-way interaction all at once by letting the model-wide parameters $\mathbf{W}_1$ and $\mathbf{W}_2$ to be rank-3 tensors:

$$\mathbf{h_r} = \mathbf{W}_1\left(\mathbf{r}, \mathbf{h}, \mathbf{I}\right) + \mathbf{b}_1 \tag{2.24}$$

$$\mathbf{t_r} = \mathbf{W}_2\left(\mathbf{r}, \mathbf{t}, \mathbf{I}\right) + \mathbf{b}_2 \tag{2.25}$$

$$s(h,r,t) = \mathbf{h_r}^\top \mathbf{t_r} \tag{2.26}$$

Where $\mathbf{W}(\mathbf{a}, \mathbf{b}, \mathbf{I})$ is tensor contraction as defined in Equation 2.4.

DistMult (Yang et al., 2015). This model simplifies RESCAL to avoid overfitting by constraining relation matrices to be diagonal. It can equivalently be thought of as a CP-decomposition of the adjacency tensor where like RESCAL, the first and the third component matrices are constrained to be equal to each other.

The model optimizes a bilinear scoring function:

$$s(h,r,t) = \mathbf{h}^\top \mathrm{Diag}(\mathbf{r})\,\mathbf{t}. \tag{2.27}$$

where the $\mathbf{h}, \mathbf{t}$ are the vector embeddings for the head and tail entities and $\mathrm{Diag}(\mathbf{r})$ is a diagonal matrix embedding for the relation. The original implementation optimizes a margin based ranking loss with AdaGrad. However, later implementations improve performance significantly by using either binary cross entropy loss or negative log likelihood (NLL) loss. We describe these loss functions in detail in Section 2.6.2.

HolE (Nickel et al., 2016b). This work presents a model named *Holographic Embeddings (HolE)* which uses circular correlations to create compositional representations.

The motivation behind this model is to find a good middle ground between the expressiveness of a fully bilinear model such as RESCAL where all possible interactions between the different dimensions of the embeddings are modelled explicitly, and simplicity of distance models such as TransE. The scoring function for HolE is defined as:

$$s(h,r,t) = \sigma\left(\mathbf{r}^\top(\mathbf{h}\star\mathbf{t})\right), \tag{2.28}$$

where $\star : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ is the circular correlation operator defined as:

$$[\mathbf{a}\star\mathbf{b}]_k = \sum_{i=0}^{d-1} a_i b_{(k+i)\bmod(d)}. \tag{2.29}$$

The intuition behind holographic embeddings is that it is a compressed version of the tensor product, where each component corresponds to the sum of a section of all pairwise interactions. Circular correlation can be computed efficiently via fast Fourier transform (FFT):

$$\mathbf{a}\star\mathbf{b} = \mathcal{F}^{-1}\left(\overline{\mathcal{F}(\mathbf{a})}\cdot\mathcal{F}(\mathbf{b})\right), \tag{2.30}$$

where $\mathcal{F}$ is the FFT and $\mathcal{F}^{-1}$ its inverse, $\overline{\mathbf{x}}$ is the complex conjugate of $\mathbf{x}$, and $\cdot$ denotes the entrywise Hadamard product. It has later been shown by Trouillon and Nickel (2017) that HolE is equivalent to ComplEx.

ComplEx (Trouillon et al., 2016). This model is closely related to DistMult, however it embeds the entities and relations in the complex vector space $\mathbb{C}^n$ rather than $\mathbb{R}$. The objective is to be able to account for anti-symmetric relations (e.g. *parentOf*) by exploiting the fact that the Hermitian dot product, which is the generalization of the regular dot product to the complex field, is anti-linear in the imaginary part of its inputs. The authors observe that the simple tensor product in the complex case is thus enough to model anti-symmetric relationships that DistMult struggles to capture.

The scoring function for the model is:

$$s(h,r,t) = \mathrm{Re}\left(\mathbf{h}^\top \mathrm{Diag}(\mathbf{r})\,\bar{\mathbf{t}}\right), \tag{2.31}$$

where $\mathrm{Re}(\mathbf{x})$ denotes the real part of $\mathbf{x}\in\mathbb{C}^n$, and like the scoring function of DistMult, $\mathrm{Diag}(\mathbf{r})$ is a diagonal matrix with $\mathbf{r}$ on its diagonal.

Lacroix et al. (2018) introduce a novel regularization term based on tensor nuclear p-norm to ComplEx, and together with a data agumentation technique of adding reverse triples to the training set, beat the state-of-the-art at the time on several datasets.

ANALOGY (Liu et al., 2017) This model is motivated by the goal to find the appropriate restrictions on relation matrices to capture analogical inference. The authors argue that the class of *normal matrices* $\mathcal{M} = \{M|MM^* = M^*M\}$, where M^* is the conjugate transpose of M, is the appropriate class for this, since it subsumes several classes which were shown to be important for modeling relations such as symmetric and anti-symmetric matrices, rotation matrices and circulant matrices. ANALOGY is a bilinear model akin to RESCAL, where the relation matrices are constrained to be normal. The authors then reformulate the constraint as learning a block diagonal matrix, hence allowing the model to be optimized via SGD.

SimplE (Kazemi and Poole, 2018) Earlier works such as Schlichtkrull et al. (2018) implement CP-decomposition as a baseline model where the same entity has different embeddings in head and tail positions, and show that not tying the weights for head and tail entities reduces the performance of the model greatly. SimplE uses data augmentation to overcome this limitation of CP-decomposition by enriching the training set with reverse relations. For each relation r, a reverse relation r^{-1} is added to the set of relations, and for each triple (h,r,t), the corresponding reverse triple (t,r^{-1},h) is added to the training set. Each entity e hence has two corresponding embeddings: a head embedding h_e and a tail embedding t_e. The score for a triple (e,r,f) is defined as the average of the CP-reconstruction for (e,r,f) and $(f,r^{-1}e)$:

$$s(e,r,f) = \frac{1}{2}\left(\mathbf{h_e}^\top \text{Diag}(\mathbf{r})\mathbf{t_f} + \mathbf{h_f}^\top \text{Diag}(\mathbf{r}^{-1})\mathbf{t_e}\right). \tag{2.32}$$

The data augmentation technique of adding the reverse triples to the training set has also been suggested by Lacroix et al. (2018), and subsequently used in a number of models, including TuckER.

TuckER (Balazevic et al., 2019). This is a KG embedding model that is based on Tucker decomposition of the binary adjacency tensor:

$$\mathbf{T} = \sum_{i,j,k} \left(Z_{i,j,k}\left(E_{:,i} \circ R_{:,j} \circ E_{:,k}\right)\right). \tag{2.33}$$

The model learns vector embeddings of entities and relations via the factor matrices $\mathbf{E}$ and $\mathbf{R}$. In addition to the vector embeddings, it learns a 3rd order core tensor $\mathbf{Z}$, which consists of shared parameters for the entity and relation representations. Scoring a triple (h,r,t) is achieved by contracting the embeddings $\mathbf{h},\mathbf{r},\mathbf{t}$ with the core tensor $\mathbf{Z}$:

$$s(h,r,t) = \mathbf{Z}(\mathbf{h},\mathbf{r},\mathbf{t}). \tag{2.34}$$

The authors show that DistMult, ComplEx and SimplE are special cases of TuckER.

2.4.3 Neural Models

In this section we review some of the influential models suggested for KG completion that have hidden layers and nonlinearities in their formulations.

Neural Tensor Network (NTN) (Socher et al., 2013). This work modifies a standard feed-forward neural architecture for relational data by adding a *bilinear* tensor layer. In NTN, each entity has its corresponding vector embedding $\mathbf{e} \in \mathbb{R}^d$, and each relation r has parameters associated with a network: a linear transformation defined by a matrix $\mathbf{V_r} \in \mathbb{R}^{k \times 2d}$, a bilinear transformation defined by a 3rd order tensor $\mathbf{W_r} \in \mathbb{R}^{d \times d \times k}$, a bias vector $\mathbf{b_r} \in \mathbb{R}^k$ and another vector $\mathbf{u_r} \in \mathbb{R}^k$ defining the parameters of the final layer. The scoring function for a triple is:

$$s(h,r,t) = \mathbf{u_r}^\top \tanh\left(\mathbf{W_r}(\mathbf{h},\mathbf{r},\mathbf{I}) + \mathbf{V_r}\begin{pmatrix}\mathbf{h}\\ \mathbf{t}\end{pmatrix} + \mathbf{b_r}\right) \tag{2.35}$$

Notice that with $\mathbf{V_r}$, the two entities can only interact indirectly through the nonlinearity. The addition of the bilinear form $\mathbf{W_r}(\mathbf{h},\mathbf{r},\mathbf{I})$ allows NTN to also model three-way interactions directly.

This work is also the first one to use pre-trained word vectors to initialize the entity embeddings. For entities with multi-word labels, the authors take the average of the corresponding word embeddings. They argue that this allows them to share statistical information between related entities, e.g. *tiger* and *Bengal tiger*.

ConvE (Dettmers et al., 2018) This model could be understood as an encoder-decoder framework, where the encoder part uses a 2D convolutional layer to combine representations for the head entity h and the relation r. It achieves this by first reshaping the vectors to form 2D matrices, then applying a convolutional operation on their concatenation. The output of the convolution is then re-flattened into a vector and passed through a linear layer. For decoding, the inner product of the output of the encoder and the embedding of the tail entity gives the score for the triple. Formally, the scoring function is defined as follows:

$$s(h,r,t) = f(\mathrm{vec}(f([g(\mathbf{h});g(\mathbf{t})] * \omega))\mathbf{W})\mathbf{t}, \tag{2.36}$$

Where f is the pointwise nonlinearity, g is the function that reshapes the embeddings from 1D vectors to 2D matrices, and $*$ is the convolution operator with the filter ω.

A number of works (Nguyen et al., 2018; Balažević et al., 2019; Vashishth et al., 2020) propose convolutional architectures that improve on that of ConvE.

R-GCN (Schlichtkrull et al., 2018). Relational Graph Convolutional Neural Network extends Graph Convolutional Networks (GCNs) (Kipf and Welling, 2017) for graphs with labelled edges.

The idea behind GCNs is to create a multi-layer neural network where each layer reflects the structure of the underlying graph, with convolutional filters spanning the direct neighbors of each node. At each layer, the network calculates the representation of a given node by combining the parameters of the node and its neighbors at the previous layer. Hence, at the n-th layer, each node representation contains information about nodes n removed from it.

R-GCN adds relation specific linear transformations to the basic GCN architecture. To avoid adding too many parameters via these transformations, the model constrains the corresponding matrices to either be block-diagonal, or low rank. For KG completion, R-GCN is used as the encoder part of an encoder-decoder framework, with DistMult as the decoder.

Other models that use Graph Convolutional Networks include Shang et al. (2019), Nathani et al. (2019) and Vashishth et al. (2020).

2.5 Using Auxiliary Information for Knowledge Graph Completion

Although the task of KG completion has been formulated as finding missing triples based only on the existing knowledge in the KG, the success of embedding methods motivated researchers to combine KG completion models with other modes of data that are often available for KGs. In this section we review these works grouped by the types of data they use. We focus on embedding models, although we also present some earlier works that have had significant impact on approaches that integrate information beyond triples from the KG into the embeddings.

2.5.1 Models that Use Information from Text

There has been a number of models proposed in the literature to integrate textual knowledge in knowledge graph embeddings. Early models for KG embeddings used word embeddings as initialization (Socher et al., 2013), although the effectiveness of this strategy has been questioned (Yang et al., 2015). Later models used mentions in a corpus, mentions where two of the entities appear in the same sentence, and entity descriptions as text sources.

Wang et al. (2014a) present the first model in the literature for learning word embeddings and knowledge graph embeddings jointly. Their model consists of three parts: the knowledge model, the text model and the alignment model. For knowledge model they use TransE, and for text model they use word2vec (Mikolov et al., 2013). They use two different methods for aligning text with the KG. The first method uses Wikipedia anchors to replace the mention of the word with its entity identifier from the KG. The second method uses entity names to extend the KG: for each triple (h,r,t), they add triples (w_h,r,w_t), (w_h,r,t) and (h,r,w_t) to the KG for each word w_h and w_t that corresponds to the head entity h and the tail entity t respectively. Zhong et al. (2015) present an improvement on this model by using *entity descriptions* for the alignment model rather than entity names or Wikipedia anchors. Their alignment model treats the description as the context for the entity, and encourages the model to embed the entity and the words that appear in its description close together.

Xie et al. (2016a) propose *DKRL*, a model that integrates entity descriptions in KG embeddings. Their model jointly optimizes two representations of each entity e: the structure-based representation e_S and the description-based representation e_D. They use the scoring function of TransE for both increasing the scores of correct triples compared to the incorrect ones, and to discourage e_S and e_D from diverging. They also experiment with two different encoding functions for obtaining e_D from the description text: a continuous bag of words (CBOW) model and a convolutional neural network (CNN) encoder. They report improved performance with the CNN encoder over both CBOW encoder and baseline TransE. Several extensions of the DKRL model have been proposed, such as integrating relation descriptions (He et al., 2019), and replacing the CNN encoder with an LSTM network (Wu et al., 2016). Other models (Xu et al., 2017; Zhou et al., 2019) suggest the use of gating mechanisms to combine the description based representation e_D with the structure based representation e_S for a unified embedding. Xiao et al. (2017) takes a different approach, applying non-

negative matrix factorization (NMF) on entity descriptions and using these to project entity embeddings to triple specific *topic* subspaces.

A competing paradigm for integrating textual knowledge into knowledge graph embeddings has roots in using KGs to provide *distant supervision* for relation extraction (Bunescu and Mooney, 2007; Mintz et al., 2009). The main idea for this is that if two entities occur together in a sentence, then it can be assumed that the sentence expresses a relation that holds between these two entities in the KG. Riedel et al. (2013) uses this intuition to extend the KG with triples that correspond to entities and surface patterns that were observed in text, and applies collective matrix factorization to learn embeddings for *entity pairs* and relations from a KG and text corpus jointly. While the early works using the occurrences of entity-pairs in text focus on the information extraction task, Toutanova and Chen (2015) adapt this approach to KG completion by modulating the effect of the triples that were extracted from text on the final representation. They demonstrate that augmenting the KG with textual mentions can improve the performance of KG embedding models, and provide a large textual co-occurrance dataset for FB15K and FB15K-237 which consists of dependency parsed sentences extracted from ClueWeb09 where two entities co-occur. Toutanova et al. (2015) add a CNN encoder to this approach in order to generate compositional representations of the textual mentions using their dependency structure. An et al. (2018) present a BiLSTM model with attention, which matches each triple with its potential mentions in text, and uses the encoding of these mentions to enrich the relation embedding. They also use entity descriptions to enrich the entity embeddings, hence combining two separate text sources in their final model. Han et al. (2018) propose a similar approach using a CNN architecture with attention to jointly learn entity and relation representations from KG and text without the use of dependency parses for the relation mentions.

Although many of the neural network architectures described above use pretrained embeddings when encoding the section of text that is to be integrated into the KG embeddings, some models differ in their approach as they aim to integrate word embedding information in the KG embeddings without the use of any non-linearities. Among those is TEKE (Wang and Li, 2016) which uses cooccurrences of entities and words in text to augment the KG embeddings. It constructs *context embeddings* of entities by taking the weighted average of embeddings of its context words in corpora, and learns a mapping matrix to combine the context embedding with the KG embedding. It also constructs context embeddings for *pairs* of entities by finding the overlapping context

words for the two entities and taking the weighted average of this set. The context embeddings for pairs are then used to learn a mapping matrix for the relation in the triple.

Veira et al. (2019) similarly presents two models, one for using entity descriptions and one for using corpus cooccurrences. The first is a model that uses entity descriptions to augment the entity embeddings. It learns a weighted sum of the word embeddings in the entity description, and combines this with the entity embedding learned from the KG after applying a linear transformation. The second model uses cooccurrences in corpora to fine-tune the word2vec embedding of the entity, while learning a mapping matrix to combine this with the entity embedding learned from the KG. Yao et al. (2019) proposes a method to utilize contextualized embeddings in KG completion. The proposed model fine-tunes pre-trained BERT (Devlin et al., 2019) for scoring a triple using the descriptions or names of the the entities. Hosseini et al. (2019) adopt a different approach, and utilize information from a text corpus for KG completion by first constructing an entailment graph using word co-occurrences with relation mentions, then using the resulting entailment graph to improve link prediction on a KG constructed from assertions in raw text. Although their method uses textual information, it is also a technique for integrating logical constraints in the form of implications between relations.

2.5.2 Models that Use Entity Type Information

Modern knowledge graphs often differ from other highly structured knowledge representations such as frames in that they don't impose strict type hierarchies to categorize entities. Nevertheless, entities that occur in KGs tend to fall intuitively in different classes. For example, the majority of the entities that occur in YAGO3-10 are either people, locations or organizations. Most relations likewise have implicit constraints on them as to which entity types can appear as the head or the tail. For example, the relation `playsFor` in YAGO3-10 appears with people as its head, and organizations as its tail entities.

Chang et al. (2014) integrate type information into RESCAL by only considering entries corresponding to triples where the types of the entities match the type constraints of the relation during tensor factorization, with the aim of making the technique more scalable to large scale datasets. Likewise, Krompaβ et al. (2015) suggest adding type constraints to RESCAL and TransE by considering only the candidate triples where

the head and the tail entity types match that of the relation during the training phase. Xie et al. (2016b) extend TransR to include type specific projection matrices, and use the type hierarchy to calculate the values for these projections. They also sample type-consistent negative samples with higher probability. Zhang et al. (2018) generalize this approach by suggesting two auxilliary losses: *entity-type cost* and *relation-type cost* which could be used to enrich any embedding model. The intuition behind their framework is that the suggested losses encourage the model to embed entities in areas of the embedding space that mirror the structure of the type hierarchy.

Garcia-Durán et al. (2016) present *TATEC*, a model that combines bigram and trigram interactions. They interpret the bigram terms as implicitly characterising type compatibility between the entities and the relation, and similarity between the head and the tail entity. The trigram interaction is modelled by RESCAL, and the bigram is modelled by three inner product terms, one between head entity h and the relation r, one between the tail entity t and r, and the last one between the two entities h and t. Jain et al. (2018a) present *JointDM* and *JointComplex*. These could be viewed as a simplification of TATEC where the relation representation for the trigram term is either DistMult or ComplEx, and the bigram term that captures the similarity between the head and the tail entity is discarded.

2.5.3 Models that Use Relation Paths

A prominent method for KG completion that does not rely on learning embeddings of entities and relations is the Path Ranking Algorithm (PRA) (Lao et al., 2011). PRA calculates the score for a candidate triple (h,r,t) by estimating the random walk probabilities of different paths between h and t and combining these values via a logistic regression model based on the relation r. Specifically, the model makes use of *path-constrained random walks* where, given an edge labelled path $P = r_1, ...r_n$ and a start entity h, a path constrained random walk $w_{h,P}$ gives a distribution over all entities e, which captures the probability of ending up in e if a random walk starts at h and follows the path P, choosing uniformly at random whenever there is more than one edge with the correct label at step t. The algorithm uses the values $w_{h,P}$ for a number of paths P as features for logistic regression, which learns to weight each feature proportional to how informative it is given the relation r.

Several works suggested different ways of integrating path information through random walk probabilities into KG embeddings, usually by providing an operation that

combines the embeddings of the relations in the path and maps them to the same vector space as the relation embeddings. Lin et al. (2015a) augment TransE to use path information by adding an auxiliary *path loss* to the main model. This loss term is summed over paths between the head and the tail entities, and considers both the PRA-style random-walk score of the given path between h and t and the informativeness of the path for the relation r. As the composition operation for obtaining path embeddings, they consider addition, multiplication and a RNN. Guu et al. (2015) investigate which operations for composing relation embeddings are suitable for which model, arguing that matrix multiplication is appropriate for RESCAL and DistMult, and vector addition is the correct choice for TransE. Neelakantan et al. (2015) present an RNN architecture that learns to compose relation representations, and show that this approach is effective in a zero-shot scenario. Toutanova et al. (2016) present a dynamic programming algorithm to efficiently incorporate all paths of a bounded length for RESCAL and DistMult. Other models such as that of Das et al. (2017) tackle the issue of the high computational cost of enumerating and scoring paths by instead learning how to navigate within the KG to arrive at the answer. Their model uses neural reinforcement learning (RL) to find predictive paths from the query entity to the answer entity. Lin et al. (2018) and Godin et al. (2019) present improvements on this model by addressing some of the technical challenges of the RL framework.

2.5.4 Models that Use Logical Rules

Both applying given rules to a knowledge base to deduce new facts, and learning logical rules from data has a long history in AI research. In the context of modern knowledge graphs, one approach that has been employed in large projects such as *NELL*(Carlson et al., 2010) is to mine logical rules from the given facts in the KG, and then to apply these rules for inferring new correct facts from the known ones. These methods mostly employ Inductive Logic Programming (ILP) to learn Horn clauses that best describe the data. Example systems include First Order Inductive Learner (FOIL) (Quinlan, 1990) and variants (Landwehr et al., 2007, 2010), AIME (Galárraga et al., 2013), and AIME+ (Galárraga et al., 2015).

If one wants to learn logical rules and apply these to reason over KGs that are incomplete or contain errors, it is essential that the rules and the deduction procedure allows for modelling uncertainty. Most rule mining systems intended for use in KGs provide probabilities with the mined rules that reflect how reliable the rule is. A number

of frameworks have been proposed to reason with probabilistic rules. These include probabilistic Datalog (Fuhr, 2000), Markov Logic Networks (Richardson and Domingos, 2006), ProbLog (De Raedt et al., 2007), Probabilistic Similarity Logic (Bröcheler et al., 2010) and ProPRR (Wang et al., 2015b).

The main advantage of rule-based systems such as those listed above is that they are interpretable. The rules they learn often offer high precision but low recall. The computational costs for both rule mining and probabilistic deduction have also been prohibitive for large scale KGs, although a considerable amount of research effort has been dedicated to improve the scalability of these approaches. Embedding methods on the other hand provide high coverage with moderate precision, are easily scalable, but lack interpretability. The complementary strengths of the two approaches have motivated a number of works that attempt to integrate them.

Earliest works combine logical and embedding approaches by using one to reduce the search space of the other. Wang et al. (2015a) suggests a method that uses Integer Linear Programming (ILP) to impose logical constraints on the candidate facts obtained via embedding methods. Wei et al. (2015) similarly generates candidate facts via embedding models first, then uses Markov Logic Networks to perform inferences on the much smaller set of facts.

Guo et al. (2016) presents *KALE*, a model that jointly embeds triples and logical rules by combining TransE with a scoring function for grounded formulas based on t-norm fuzzy logic. Wang and Cohen (2016) present a method to learn embeddings of logical rules by applying matrix factorization on binary matrices obtained via proof graphs generated by ProPPR, and use these embeddings to guide proof search during test-time. Guo et al. (2018) consider a KG together with a set of Horn clauses with confidence intervals such as those extracted from rule mining systems. They iteratively predict *soft labels*, which are values between 0 and 1, for unseen triples based on the application of the given Horn clauses, then refine the knowledge graph embeddings by adding the triples with the soft labels to the training examples. Zhang et al. (2019a) similarly present a method that iteratively learns embeddings, logical rules from the embeddings, and then extends the KG by the learned rules. Other works have suggested using logical rules for imposing constraints on KG embeddings during training (Minervini et al., 2017; Ding et al., 2018; Wang et al., 2018a), or for regularization via adversarial examples (Minervini et al., 2017). Neuro-symbolic approaches that learn to reason using logic-like structures have also been applied to the KG completion task. These

include TensorLog (Cohen et al., 2020; Yang et al., 2017), Neural Theorem Provers (Rocktäschel and Riedel, 2017) and Lifted Relational Neural Networks (Sourek et al., 2018).

2.5.5 Models that use Other Auxiliary Data

Integrating auxiliary information into KG embeddings besides text, paths, logical rules and types have also been explored in KG completion literature. For instance, Xie et al. (2017) presents a model that intergrates *images* associated with the entities to KG embeddings. Wu and Wang (2018) explores combining KG information with numeric attributes. Other works (Pezeshkpour et al., 2018; Mousselly-Sergieh et al., 2018; Wang et al., 2019) combine image data with other types of auxiliary information such as text and numeric attributes for general multimodal KG embeddings. There has also been works on enriching domain specific knowledge graphs with relevant auxiliary information, such as geographic location (Qiu et al., 2019; Mai et al., 2020).

A related area of research that has recently gained more attention from the community is *temporal knowledge graph completion* (Garcia-Duran et al., 2018; Goel et al., 2020; Jain et al., 2020). Even though this setting could be seen as a special case of enriching the representations with numeric attibutes, it goes beyond integrating auxiliary information, and changes the task: in temporal KG completion, each triple is associated with a time interval, and the goal is either to correctly predict the missing entity given the relation, the other entity and the time interval, or to predict the time interval given the triple.

2.6 Training Choices

The majority of the recent work in knowledge graph embeddings has focused on model architecture. The parametrization, however, is only one component of a framework that learns embeddings for KG completion. In fact, often a new architecture is introduced together with one or more improvements to the training procedure, and it is not always easy to disentangle the effect of these different components on the reported performance.

Reported results in subsequent implementations of DistMult and ComplEx serve well as an illustration of the magnitude of the effect training choices and implementa-

tion details have on performance. For DistMult, the original implementation of Yang et al. (2015) obtains an MRR of 0.36 on FB15K. Later implementations report 0.555 (Toutanova and Chen, 2015), 0.654 (Trouillon et al., 2016), and 0.798 (Kadlec et al., 2017). Likewise, the performance of ComplEx jumps from 0.692 MRR on FB15K in the original implementation, to 0.86 (Lacroix et al., 2018).

In this section we review some of the training strategies that have been found to have a large effect on the success of the models. For an extensive analysis and comparison of these and other strategies, see Ruffinelli et al. (2020) and Ali et al. (2020).

2.6.1 Negative Training Triples

A standard difficulty with large knowledge graphs is that they contain only facts that are believed to be true. Hence it is possible to interpret the missing triples in several different ways. *Closed world assumption* posits that any triple that is not included in the KG is false. This is in contrast with the *open world assumption* where any missing triple is deemed unknown rather than incorrect.

Even though the open world assumption is justified by the size and the incompleteness of the KGs that are often the target of KG completion approaches, most methods require examples of incorrect triples for training. A compromise between the open and the closed world assumptions is the *locally closed world assumption* (Nickel et al., 2016a). In this approach, the KG is assumed to be locally complete in the sense that if a triple (h,r,t) is observed in data, it can be assumed that (h,r,t') and (h',r,t) are incorrect if they aren't in the training data already; however the model doesn't make any assumptions about a triple (h',r',t') if neither $(h',r',\cdot)$ nor $(\cdot,r',t')$ is observed in the KG. This approach motivates the negative sampling strategy first introduced in Bordes et al. (2013) where a *corrupted triple* is generated by replacing the head or the tail entity of a training triple with a randomly chosen entity. Efficient implementations of this strategy can accidentally generate correct triples and label them as false. To reduce the change of this happening, Wang et al. (2014b) introduce *Bernoulli negative sampling*. This strategy adjusts the probability of corrupting the head or the tail depending on whether the relation is a 1-to-1, 1-to-many, many-to-1 or many-to-many.

Although the standard strategy for obtaining corrupted triples is still uniform sampling, several works have focused on improving the scores of KG completion models by more sophisticated methods to generate negative triples. Among those are works that use

Generative Adversarial Networks (GANs) (Cai and Wang, 2018; Wang et al., 2018b), and those that pick more informative samples based on the current state of the model (Qin et al., 2019; Zhang et al., 2019b; Sun et al., 2019).

Instead of refining the choice of sampled negatives, 1-to-all setting was used by Lacroix et al. (2018) to score all possible entities at once for a query $(h, r, ?)$ or $(?, r, t)$, considering all but the candidate entity as incorrect choices. Dettmers et al. (2018) similarly suggested using K-to-all scoring where each batch consists of a fiber of the adjacency tensor corresponding to $(h, r, \cdot)$ or $(\cdot, r, t)$, and the triple is labelled correct if it is in the training set and false otherwise. Even though this was introduced as a way of speeding up the training, it proved to be effective at increasing the performance of the models as well. See Kotnis and Nastase (2017) for an in-depth analysis of different strategies for negative sampling.

2.6.2 Loss Functions

A number of different loss functions have been suggested for training KG embeddings. RESCAL was initially implemented with L2 loss between the original adjacency tensor and the one reconstructed from the learned parameters. L2 loss directly corresponds to standard tensor decomposition models, and allows RESCAL to be optimized via the Alternating Least Squares (ALS) algorithm. The scalability of ALS means that negative sampling strategies are not necessary, and made it possible for the initial implementation to obtain embeddings for large-scale knowledge graphs for the first time (Nickel et al., 2012). A weakness of this approach from the point of view of the KG completion task is that the optimized objective does not correlate very well with the ranking task. Intuitively, reconstructing the binary adjacency tensor is too strong a requirement if the goal is merely to find low rank decomposition of a "score" tensor where the entries corresponding to 1s in the adjacency tensor are larger than those corresponding to 0s.

This is the observation that motivated Bouchard et al. (2015) to argue that standard tensor rank is not the right notion for KG completion. The tensor decomposition models should instead aim to find a representation with low *sign-rank*.[3] Even though it is difficult to optimize for sign-rank directly, the authors argue that *binary cross entropy loss (BCE)* and *hinge loss* are approximations of the desired objective.

[3] As a motivating example, they show that even though the identity matrix has full rank in the regular sense, it has a sign-rank of 2 regardless of its dimensions.

Binary cross entropy loss treats the problem as a binary classification problem. When labels $y_d = \{1, -1\}$ are provided for original and corrupted triples respectively, BCE loss for a triple d is calculated as follows:

$$\mathcal{L}_{BCE}(d) = \log(1 + \exp(-y_d \cdot s(d))). \tag{2.37}$$

BCE is used in the original implementations of ComplEx (Trouillon et al., 2016) and HolE (Nickel et al., 2016b).

The first models that directly optimize for the ranking task, including the initial implementations of TransE (Bordes et al., 2013) and DistMult (Yang et al., 2015), use a pairwise margin loss between the training triple, and a corrupted version of the triple where the head or the tail entity has been replaced by another entity.

Let $D = \{d_i | d_i = (h,r,t)_i\}$ be the set of correct triples in the training set and for $d_i = (h,r,t)$, let $D'_i = \{d'_i | d'_i = (h,r,t')$ or $d'_i = (h',r,t)\}$ be the set of corrupted triples for d_i. Let $s(d)$ denote the score of the model for the triple d. The margin based ranking criterion is defined as follows:

$$\mathcal{L}_{\text{margin}} = \sum_{d_i \in D} \sum_{d'_j \in D'_i} [\gamma + s(d_i) - s(d'_j)]_+ \tag{2.38}$$

Where $\gamma > 0$ is the margin hyperparameter and $[x]_+ = x$ if $x > 0$, and 0 otherwise. In words, the loss function is 0 if the correct pair is scored at least γ smaller than the corrupted pair, and increases linearly as the scores deviates from this case.

A downside of this loss compared to others is that the ratio of correct to corrupted triples the model sees during training is 1 by design: if the number of corrupted triples per one training triple is n, the model will see n copies of the training triple, each paired with one corrupted triple. This means that increasing the number of corrupted triples per training triple is equivalent to training with more epochs with the ratio set to 1. This is a disadvantage as this ratio seems to be an important hyperparameter in all other losses.

Toutanova and Chen (2015) introduce the use of *Negative Log Likelihood (NLL) of softmax* as a loss function that scores one correct triple against a number of corrupted triples at once. NLL of softmax loss treats the problem similar to a multi-class classification problem, where for each triple in the training set, the model needs to correctly choose the correct triple over all the corrupted triples. Let d_i be a triple from the training set, and set D'_i denote the set of corrupted triples obtained by corrupting the head

or the tail of d_i. NLL of softmax is defined as follows:

$$\mathcal{L}_{NLL}(d_i) = -\log\left(\frac{\exp(s(d_i))}{\exp(s(d_i)) + \sum_{d'_j \in D'_i} \exp(s(d'_j))}\right). \tag{2.39}$$

This loss is used in a number of works that re-implement baselines to achieve results competitive with the state-of-the-art at the time they were published (Kadlec et al., 2017; Lacroix et al., 2018).

2.6.3 Regularization

The choice of regularizer often depends on the choice of loss function and the architecture. The L2 regularizer is common for models that directly learn embeddings, especially if the loss function is binary cross entropy (Trouillon et al., 2016; Nickel et al., 2016b). However, Lacroix et al. (2018) argue that an L3 norm based on the tensor nuclear p-norm is more appropriate for the task. When margin-based ranking loss is used, it is common to constrain the L_2 norm of the entity embeddings to 1 in order to avoid the case where the model increases the norms of the embeddings to trivially minimize the loss (Bordes et al., 2013; Yang et al., 2015). This done by projecting the embeddings on to the unit sphere after every gradient step. A number of models also employ dropout in hidden layers to prevent overfitting (Dettmers et al., 2018; Balazevic et al., 2019).

Chapter 3

Incorporating Entity Types in Knowledge Graph Embeddings

3.1 Introduction

A common type of additional information that can be obtained about entities and relations in a knowledge graph is the types of the entities, and the type restrictions on the relations. For example, one would expect a common type of entity in a knowledge graph to be *person*, and that a relation `isCitizenOf` would only have entities of this type as its head. We refer to such information on entities and relations of a KG as a *type-ontology*, and explore how to utilize such information for improving KG embedding models.

The main contribution presented in this chapter is a training framework that improves the performance of a number of standard bilinear models for KG completion. We name our framework *JoBi*, short for ***Jo**int **Bi**ased Training*. Our approach can be seen as a joint tensor-tensor decomposition of the adjacency tensor of the knowledge graph and a type tensor constructed from a type-ontology. Our framework can utilize external type information from a given type-ontology if it is explicitly provided, or can generate an approximate type-ontology *implicitly* from the KG data if no explicit type information is available. We focus on the second scenario and show that even in the absence of explicit type information, using heuristically generated labels improves the performance of a number of bilinear models.

Even though the literature on KG completion has focused mostly on model archi-

tectures training choices and hyperparameters have a very large impact on the final performance (see Section 2.6 for details). We re-implement several bilinear models and present a detailed analysis on the effects of training with different hyperparameter settings and loss functions. Our findings confirm that the hyperparameter settings for obtaining competitive results with these models require a large memory footprint, which becomes prohibitive for very large KGs. We compare the effects of different hyperparameter choices on the baseline models and on our joint framework, and show that the latter is more robust to different hyperparameter choices, especially those that reflect the training conditions of learning embeddings for very large KGs.

3.2 Modelling Type Information for Knowledge Graphs

The core motivation behind our framework JoBi is the observation that there are two distinct types of false triples. Consider the two triples (*Lyon*, `isCapitalOf`, *Germany*) and (*Lyon*, `isCapitalOf`, *George_Orwell*). Even though both these triples express facts that we know to be incorrect, we argue that there is a qualitative difference between the two. Both the head and the tail entities in the first triple obey the implicit semantic type restrictions the relation has for its arguments, but this is not the case for the second one: a triple with the relation `isCapitalOf` does not make much sense when its tail entity corresponds to a person.

3.2.1 Type-Consistency

Formally we consider the scenario where for a KG $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{F})$ with the set of entities $\mathcal{E}$, set of relations $\mathcal{R}$ and set of facts $\mathcal{F}$, we are given a type-ontology $O = (\mathcal{T}, T_E, T_{head}, T_{tail})$ where $\mathcal{T}$ is the set of type labels, $T_E : \mathcal{E} \mapsto \mathcal{P}(\mathcal{T})$ is a function that assigns a set of type labels to each entity, and $T_{head}, T_{tail} : \mathcal{R} \mapsto \mathcal{T}$ are functions that map each relation to the required type label for its head and tail entity respectively. An example KG with all components $(\mathcal{E}, \mathcal{R}, \mathcal{F})$ and the associated $O = (\mathcal{T}, T_E, T_{head}, T_{tail})$ is given in Figure 3.1.

We will call a candidate triple (h, r, t) *type-consistent* if the set of type labels associated with h include the restriction on the head of r, and the type labels associated with t include the restriction on the tail of r. Formally, this is defined as follows:

Definition 1. *Given a knowledge graph* $(\mathcal{E}, \mathcal{R}, \mathcal{F})$ *and a type-ontology* $O = (\mathcal{T}, T_E,$

$$\mathcal{E} = \{Joe\_Biden,\ Emmanuel\_Macron,\ United\_States\_of\_America,\ France\}$$

$$\mathcal{R} = \{\texttt{isPresidentOf},\ \texttt{metWith}\}$$

$$\mathcal{F} = \left\{ \begin{array}{c} (Joe\_Biden,\ \texttt{isPresidentOf},\ United\_States\_of\_America), \\ (Emmanuel\_Macron,\ \texttt{isPresidentOf},\ France), \\ (Joe\_Biden,\ \texttt{metWith},\ Emmanuel\_Macron), \\ (Emmanuel\_Macron,\ \texttt{metWith},\ Joe\_Biden) \end{array} \right\}$$

$$\begin{aligned} T_E(Joe\_Biden) &= \{\text{person},\ \text{politician}\} \\ T_E(Emmanuel\_Macron) &= \{\text{person},\ \text{politician}\} \\ T_E(United\_States\_of\_America) &= \{\text{geographic\_area},\ \text{administrative\_region},\ \text{country}\} \\ T_E(France) &= \{\text{geographic\_area},\ \text{administrative\_region},\ \text{country}\} \end{aligned}$$

$$T_{head}(\texttt{isPresidentOf}) = \text{politician} \qquad T_{head}(\texttt{metWith}) = \text{person}$$

$$T_{tail}(\texttt{isPresidentOf}) = \text{administrative\_region} \qquad T_{tail}(\texttt{metWith}) = \text{person}$$

Figure 3.1: An example knowledge graph $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{F})$ where $\mathcal{E}$ is the set of entities, $\mathcal{R}$ is the set of relations, $\mathcal{F}$ is the set of facts, together with a type ontology $O = (\mathcal{T}, T_E, T_{head}, T_{tail})$ for $\mathcal{G}$.

*T_{head}, T_{tail}), a triple $(h,r,t) \in (\mathcal{E} \times \mathcal{R} \times \mathcal{E})$ is **type-consistent** with respect to O if $T_{head}(r) \in T_E(h)$ and $T_{tail}(r) \in T_E(t)$.*

Given the types and type restrictions in the KG in Figure 3.1, both the triples (*Joe_Biden*, `isPresidentOf`, *United_States_of_America*) and (*Joe_Biden*, `isPresidentOf`, *France*) are type-consistent, while (*France*, `isPresidentOf`, *Joe_Biden*) or (*France*, `hasMet`, *Joe_Biden*) are not. Note that being type-consistent in this sense can be considered a prerequisite for correctness.

3.2.2 Joint Framework

In this section we present our joint framework *JoBi*, which produces improved embeddings for KG completion by augmenting the training procedure using type-consistency labels.

Parametrization. Our framework operates on two identically parametrized bilinear models, where the entity embeddings of the two are tied. During training, these two models receive the same batches of triples as inputs, however the supervision provided for the input triples differ between the two. The first model receives the standard supervision for KG completion models: the original triple from the training set is labelled with 1 and the corrupted triples are labelled 0. We refer to this part of the overall model as the the *fact module*. The second model is trained to predict whether the given triple is type-consistent or not. This done by using a type-ontology O that is either provided, or constructed from the training data. The corrupted triples are labelled with 1 if they are type-consistent with respect to O, and 0 otherwise. We refer to this part of the model as the *type module*.

The scoring functions s_{type} and s_{fact} for the two modules are defined as follows when the joint framework is operating on DistMult:

$$s_{\text{fact}}(h,r,t) = \mathbf{h}^\top \text{diag}(\mathbf{r}\text{fact})\mathbf{t}, \quad s_{\text{type}}(h,r,t) = \mathbf{h}^\top \text{diag}(\mathbf{r}\text{type})\mathbf{t} \tag{3.1}$$

If the base model is ComplEx, the scoring functions are defined as:

$$s_{\text{fact}}(h,r,t) = \text{Re}\left(\mathbf{h}^\top \text{diag}(\mathbf{r}\text{fact})\bar{\mathbf{t}}\right), \quad s_{\text{type}}(h,r,t) = \text{Re}\left(\mathbf{h}^\top \text{diag}(\mathbf{r}\text{type})\bar{\mathbf{t}}\right), \tag{3.2}$$

and for SimplE they are defined as:

$$s_{\text{fact}}(h,r,t) = \frac{1}{2}\left(\mathbf{h}_{\text{head}}^\top \text{diag}(\mathbf{r}\text{fact})\mathbf{t}_{\text{tail}} + \mathbf{t}_{\text{head}}^\top \text{diag}(\mathbf{r}\text{fact}^{-1})\mathbf{h}_{\text{tail}}\right) \tag{3.3}$$

$$s_{\text{type}}(h,r,t) = \frac{1}{2}\left(\mathbf{h}_{\text{head}}^\top \text{diag}(\mathbf{r}\text{type})\mathbf{t}_{\text{tail}} + \mathbf{t}_{\text{head}}^\top \text{diag}(\mathbf{r}\text{type}^{-1})\mathbf{h}_{\text{tail}}\right) \tag{3.4}$$

The reason why we tie the weights of the entity embeddings, but let the embeddings for the relations be optimized separately is largely parameter efficiency: the number of relations in standard KGs are often magnitudes less than the number of entities, hence doubling the number of embeddings for relations is a negligible increase in the total number of training parameters. During training, we optimize the two modules jointly, but use only s_{fact} to score candidate triples during test time. Therefore the addition of the type module has no effect on the number of final parameters of the trained model.

Joint training as tensor decomposition. Recall that bilinear models can be thought of as finding a low-rank decomposition of the adjacency tensor $\mathbf{G}$. For example, DistMult can be written as optimizing the following equation:

$$\min_{\mathbf{E},\mathbf{R}} \left\| \sigma\left(\sum_{i=0}^{n} E_{:,i} \circ R_{:,i} \circ E_{:,i} \right) - \mathbf{G} \right\| \tag{3.5}$$

where $\mathbf{E}$ and $\mathbf{R}$ are matrices corresponding to entity and relation embeddings, $\circ$ is the generalized outer product as defined in Equation 2.9, σ is a non-linearity depending on the loss function used, n is the number of fibres i ranges over, and the expression is minimized with respect to the tensor Frobenius norm.

Likewise, our joint framework can be expressed as joint tensor-tensor decomposition which finds low-rank factors for both the adjacency tensor $\mathbf{G}$ and the type tensor $\mathbf{T}$ defined by the type-ontology $O = (\mathcal{T}, T_E, T_{head}, T_{tail})$:

$$T_{i,j,k} = \begin{cases} 1, & \text{if } T_{head}(r_j) \in T_E(e_i) \text{ and } T_{tail}(r_j) \in T_E(e_k) \\ 0, & \text{otherwise.} \end{cases} \tag{3.6}$$

When our base model is DistMult, our framework aims to find low rank matrices $\mathbf{E}, \mathbf{R}^{\text{fact}}, \mathbf{R}^{\text{type}}$ by optimizing the following equation:

$$\min_{E,R^{\text{fact}},R^{\text{type}}} \left(\left\| \sigma_1\Big(\sum_{i=0}^{d} E_{:,i} \circ R^{\text{fact}}_{:,i} \circ E_{:,i} \Big) - \mathbf{G} \right\| + \alpha \left\| \sigma_2\Big(\sum_{i=0}^{d} E_{:,i} \circ R^{\text{type}}_{:,i} \circ E_{:,i} \Big) - \mathbf{T} \right\| \right), \tag{3.7}$$

where σ_1 and σ_2 may be different non-linearities due to different loss functions for the two models, and $0 < \alpha \leq 1$ is a scalar term that weights down the reconstruction error from $\mathbf{T}$ compared to that of $\mathbf{G}$. Our joint framework with ComplEx and SimplE can also be analogously expressed as a joint tensor-tensor decomposition by replacing the factor matrices in the above expression with those that correspond to the appropriate model.

Generating corrupted triples. During training, we generate negative examples by corrupting the positive ones following the *Locally Closed World Assumption*. For each training triple $k = (h,r,t)$, we construct a set of corrupted triples $corr(k)$ by first flipping a fair coin to determine whether to corrupt the head or the tail entity. Then, we pick an entity $e' \in \mathcal{E}$ and add the triple (e',r,t) or (h,r,e') to $corr(k)$ depending on the outcome of the coin flip.

In order to make training more challenging for the model, we sample $e' \in \mathcal{E}$ so that $corr(k)$ is biased towards triples that are type-consistent with respect to the type-ontology $O = (\mathcal{T}, T_E, T_{head}, T_{tail})$. With probability p, our sampling procedure chooses e' uniformly at random from entities that would result in type-consistent triples. These are the entities $\{e' \,|\, T_{head}(r) \in T_E(e')\}$ if the result of the coin flip is heads, and $\{e' \,|\, T_{tail}(r) \in T_E(e')\}$ if it is tails. With probability $1-p$, we sample e' uniformly at random from all of $\mathcal{E}$ as usual.

Labelling corrupted triples. After we generate the set of corrupted triples $corr(k)$ for a given training triple $k = (h,r,t)$, we generate the type-consistency labels for triples in $corr(k)$ according to the type-ontology $O = (\mathcal{T}, T_E, T_{head}, T_{tail})$. Since for each triple $(h',r,t') \in corr(k)$, either $h' = h$ or $t' = t$, the task reduces to checking if the corrupted entity matches the type-restriction on its position for the relation r. Specifically, the labels of corrupted triples in $k' \in corr(k)$ are calculated as follows:

$$\text{if } k' = (e',r,t) \text{ then} \quad y_{\text{type}}(k') = \begin{cases} 1 & \text{if } T_{head}(r) \in T_E(e') \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

$$\text{if } k' = (h,r,e') \text{ then} \quad y_{\text{type}}(k') = \begin{cases} 1 & \text{if } T_{tail}(r) \in T_E(e') \\ 0 & \text{otherwise} \end{cases} \tag{3.9}$$

Calculating the loss. For s_{fact}, the task is to differentiate the correct triple k from all its corrupted versions $k' \in corr(k)$. We accomplish this by applying softmax over scores of all the candidates and then calculating the loss as the negative log likelihood (NLL) of the true triple:

$$\mathcal{L}_{\text{fact}}(k) = -\log\left(\frac{exp(s_{\text{fact}}(k))}{exp(s_{\text{fact}}(k)) + \sum_{k' \in corr(k)} exp(s_{\text{fact}}(k'))}\right) \tag{3.10}$$

For s_{type}, we evaluate the score for each triple $k' \in \{k\} \cup corr(k)$ individually, based

on the the type-consistency labels $y_{\text{type}}(k')$. We calculate the loss as the binary cross entropy between the output and the type-consistency label:

$$\mathcal{L}_{\text{type}}(k) = \frac{1}{N} \sum_{k' \in \{k\} \cup corr(k)} \left(-y_{\text{type}}(k') \log s_{\text{type}}(k') - (1 - y_{\text{type}}(k')) \log(1 - s_{\text{type}}(k')) \right), \tag{3.11}$$

where $N = |corr(k)| + 1$. We combine the two losses via weighted addition and a tunable hyperparameter α:

$$\mathcal{L} = \mathcal{L}_{\text{fact}} + \alpha \mathcal{L}_{\text{type}} \tag{3.12}$$

3.2.3 Generating type-consistency labels from KG triples

In addition to the KG triples from the training data, our framework requires access to a decision procedure for determining whether a given candidate triple is type-consistent. This is straightforward if an explicit type-ontology is provided. In the absence of such additional information, we generate labels heuristically by labelling a candidate triple (h,r,t) type-consistent if there exists triples (h',r,t) and (h,r,t') in the training data. This is achieved by constructing a surrogate ontology $\mathcal{O}' = (\mathcal{T}', T'_E, T'_{head}, T'_{tail})$ by the following procedure:

For each $r \in \mathcal{R}$:

- add two new labels l_r^{head} and l_r^{tail} to the set of type labels $\mathcal{T}$,
- let $T_{head}(r) = l_r^{head}$, and $T_{tail}(r) = l_r^{tail}$.

For each $(h,r,t) \in \mathcal{F}$:

- add the type label l_r^{head} to the set $T_E(h)$,
- add the type label l_r^{tail} to the set $T_E(t)$.

The following lemma shows that the set of triples labelled type-consistent by the ontology constructed by the procedure above would be a subset the triples that are type-consistent according to the true underlying type-ontology.

Lemma 3.2.1. *For a KG $(\mathcal{E}, \mathcal{R}, \mathcal{F})$ and a corresponding ontology $\mathcal{O}$, if all triples in $\mathcal{F}$ are type-consistent with respect to $\mathcal{O}$, then candidate triple (h,r,t) is type-consistent with respect to the surrogate type-ontology $\mathcal{O}'$ only if it is type-consistent with respect to $\mathcal{O}$.*

Dataset	# entities	# relations	# training triples
FB15K	14,951	1,345	483,142
FB15K-237	14,541	237	272,115
WN18RR	40,943	11	93,003
YAGO3-10	123,182	37	1,079,040
FB1.9M	1,892,241	3,247	19,323,513

Table 3.1: Statistics of datasets used in experiments.

Proof. If a candidate triple (h,r,t) is type-consistent with respect to O', then $T'_{head}(r) = l_r^{head} \in T'_E(h)$ and $T'_{tail}(r) = l_r^{tail} \in T'_E(t)$. This means that there exists triples (h,r,t'), $(h',r,t) \in \mathcal{F}$. Since triples in $\mathcal{F}$ are type-consistent with respect to O by the assumption, it must be that $T_{head}(r) \in T_E(h)$ and $T_{tail}(r) \in T_E(t)$, which proves that (h,r,t) is type-consistent with respect to O. □

3.3 Experiments

In this section, we present empirical evaluation of our framework JoBi on the task of knowledge graph completion. We first re-implement three bilinear models: DistMult, ComplEx and SimplE, and provide a detailed analysis of the effects of different loss functions and hyperparameter settings. We then present the performance of JoBi compared to that of the baselines. We focus on implementations that could scale to very large KGs. Hence, we evaluate our models with training settings that don't require excessively large memory usage. We show that training with JoBi not only improves the performance on all baseline models, but also makes models more robust to hyperparameter choices required for a small memory footprint.

Datasets. We perform baseline experiments on standard datasets FB15K-237 (Toutanova et al., 2015), YAGO3-10 and WN18RR (Dettmers et al., 2018). For evaluating JoBi, we use FB15K (Bordes et al., 2013), FB15K-237, YAGO3-10 and a new large-scale dataset FB1.9M containing a subset of Freebase which we construced from FB3M (Xu and Barbosa, 2018).[1] Statistics for these datasets can be found in Table 3.1.

[1]We do not perform experiments on WordNet derived datasets WN18 or WN18RR because modelling a type-ontology wouldn't provide any information – all entities are synsets and almost all can occur as an object or subject to all the possible relations.

Evaluation. We evaluate all models by ranking the model score for each test triple against the model score for all its head or tail corrupted versions. We report the mean reciprocal rank (MRR) of the correct triple as well as average hits@n for $n = 1, 3, 10$, where hits@n is 1 if the score for the correct triple is ranked in the top n, and 0 otherwise.

Specifically, for each triple (h, r, t) in the test set, we construct two queries: $(h, r, ?)$ and $(?, r, t)$. For the query $(h, r, ?)$ we obtain and sort the model scores $s(h, r, t')$ for all $t' \in \mathcal{E}$ and retrieve the rank of the correct triple amongst the corrupted ones. For the query $(?, r, t)$ we similarly rank $s(h, r, t)$ against $s(h', r, t)$ for all $h' \in \mathcal{E}$. To avoid penalizing the model for ranking another correct triple above the given test triple, we employ *filtered evaluation* suggested by Bordes et al. (2013), and remove any triple k from the set of corrupted triples before performing the ranking if k appears in training, validation or test set.

3.3.1 Re-implementation of Baselines

We re-implement three bilinear models: DistMult, ComplEx and SimplE, and investigate the effects of the choice of loss function, the size of the embeddings, the number of sampled negatives, and the size of training batches on the performance of these models.

Both for embedding sizes and the number of generated negatives examples, setting the hyperparameter values much larger than those contained in our grid has been shown to improve results. For embedding sizes, Kadlec et al. (2017) and Lacroix et al. (2018) report results that were state-of-the-art at the time of publishing with 2000 and 4000 dimensional embeddings. Likewise for generated negatives, it has been shown that state of the art results can be reached with 1-to-all scoring, where each training triple is scored with its head or tail entity replaced with all the entities in the KG at once (Joulin et al., 2017; Dettmers et al., 2018; Lacroix et al., 2018). These settings however cannot scale to very large KGs which might contain magnitudes more entities than the standard benchmarks. Because our focus is on scalability, we restrict our experiments to a maximum of 200 dimensional embeddings and to sampled negatives.

Experiment design We re-implement all the models in PyTorch, where the implementation of different models share everything except the parametrization of the scoring function to ensure that any difference in performance is due to nothing but the

models themselves. We test all models with three different loss functions: max-margin loss, logistic loss (also called binary cross-entropy loss) and negative-log-likelihood of softmax (also called cross-entropy loss). For all datasets, models and loss functions we perform grid search on:

- Number of generated negatives per training triple $n_{neg} = \{1, 5, 25, 100, 500\}$
- Embedding dimensions $d = \{50, 100, 200\}$
- Batch size $b = \{25, 100, 500, 1000\}$

We report best mean reciprocal rank (MRR) on the validation set of each dataset, model and loss-function. We also illustrate the effects of different parameter choices by plotting the best MRR for each model and dataset when fixing one hyperparameter value at a time.

3.3.1.1 Implementation Details

For all runs, we initialize embeddings with Xavier initialization (Glorot and Bengio, 2010), and use Adam optimizer (Duchi et al., 2001) for adjusting learning rate throughout training. We set the initial learning rate to 0.01 since different settings for this hyperparemeter did not make a significant difference in our preliminary experiments.

For each training triple $k = (h, r, t)$, we generate n_{neg} number of corrupted triples by sampling $e'_1, \ldots, e'_{n_{neg}} \in \mathcal{E}$ uniformly at random. For each e'_i we construct a corrupted triple by replacing the head or tail entity of k with equal probability, creating $corr(k)$ so that each $k'_i \in corr(k) = (e_i, r, t)$ or (h, r, e_i).

For all losses, the loss corresponding to a batch is calculated as the mean of the losses for each triple, which is calculated with the model score $s(k)$ for each training triple k, and scores $s(k'_i)$ for the corrupted triples $k'_i \in corr(k)$.

We calculate *max-margin loss* value for a triple k as follows:

$$\mathcal{L}_{margin}(k) = \sum_{k' \in corr(k)} \max[1 + s(k) - s(k'), 0] \tag{3.13}$$

We set the margin to be 1 following previous work, and project the embeddings back to the unit sphere after every gradient update.

For *log-loss*, we label the correct triples with 1 and the corrupted triples with -1. We

	FB15K-237			WN18RR			YAGO3-10		
	margin	*log*	*softmax*	*margin*	*log*	*softmax*	*margin*	*log*	*softmax*
DistMult	0.185	0.19	0.281	0.406	0.406	0.417	0.093	0.449	0.402
ComplEx	0.198	0.203	0.288	0.423	0.411	0.442	0.101	0.444	0.429
SimplE	0.191	0.212	0.276	0.426	0.415	0.437	0.096	0.47	0.415

Table 3.2: Best MRR of three bilinear models on the validation set of three datasets with different loss functions

then use the Pytorch implementation of log-loss which calculates the following:

$$\mathcal{L}_{log}(k) = \sum_{k' \in \{k\} \cup corr(k)} \log(1 + \exp(-y_{k'} \times s(k'))) \tag{3.14}$$

Where y'_k is the label assigned to the triple k'.

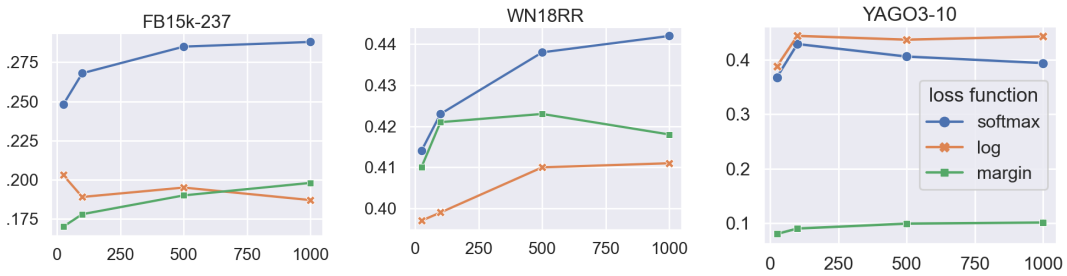
For the implementation of *NLL of softmax*, we use the PyTorch implementation of cross entropy loss that implements a softmax layer and a NLL calculation. For a training triple k the loss is calculated as follows:

$$\mathcal{L}_{softmax}(k) = -\log \frac{\exp(s(k))}{\exp(s(k)) + \sum_{k' \in corr(k)} \exp(s(k'))} \tag{3.15}$$

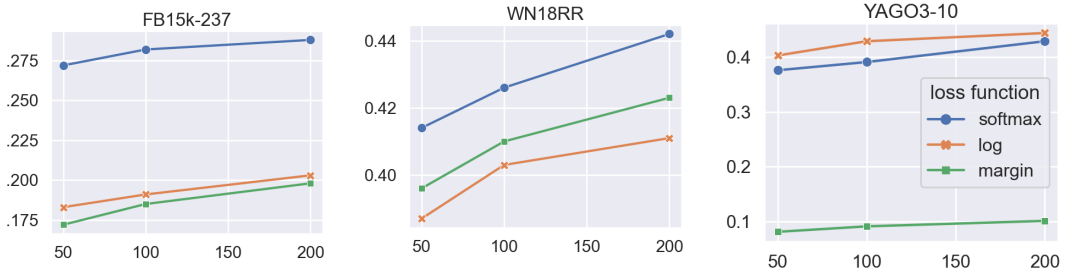
We use early stopping by evaluating the model on the validation set every five epochs, and stop the training when the performance on the validation set decreases. We use the filtered setting both for early stopping and for reporting the results. That is, when ranking the correct triple for early stopping, we remove a corrupted triple from the candidate set if it appears in training or validation set. When we report results on the test set, we add the test set to known triples after the training is completed.

3.3.1.2 Baseline Results and Discussion

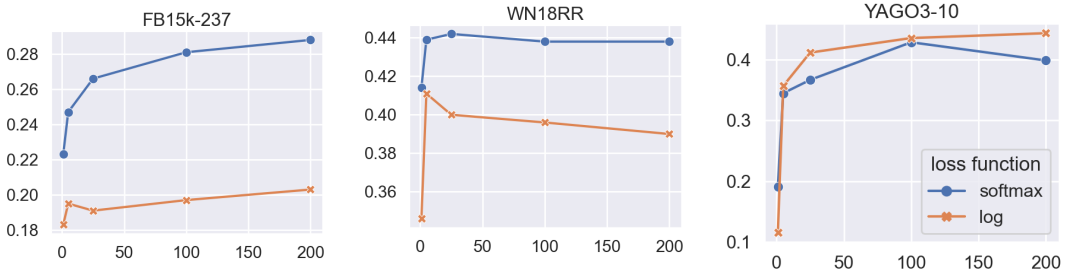
In Table 3.2 we present the best MRR on validation set that the models achieved with different loss functions. The first thing to note here is that changing the loss function seems to have a greater effect on the results than changing the parametrization from one bilinear model to another. The best choice for the loss functions seems to depend on dataset rather than the model, with NLL of softmax performing slightly better in WN18RR and significantly better in FB15K-237 than the other two. Log-loss performs slightly better in YAGO3-10, but significantly worse in FB15K-237 compared to softmax. Max-margin loss seems to be trailing behind on all datasets, but is notably bad in



(a) Batch size vs. MRR



(b) Embedding size vs. MRR



(c) Negatives ratio vs. MRR

Figure 3.2: Analysis of how MRR changes with different choices of hyperparameters. We fix one hyperparameter value at a time and report results with the best settings of other hyperparameters.

YAGO3-10. It is also interesting to note that the difference between the performance of the different losses are much more pronounced for FB15K-237 and YAGO3-10 than for WN18RR. This might be due to WN18RR being derived from a lexical resource rather than a general knowledge graph like the other two, and hence having a different graph topology.

We present detailed graphs of how different hyperparameter settings affect the performance of ComplEx on different datasets in Figure 3.2. In Figure 3.2a it can be seen that larger batch sizes tend to have a positive effect on performance except for log-loss on FB15K-237, although the extent of this changes with dataset and loss function used. The effect of changing embedding size can be seen in Figure 3.2b, and it can

be observed that performance increases with the embedding size. The widest variance in performance is due to different values of the number of generated negatives per positive example during training (Figure 3.2c), with performance improving with increased number of negatives on all dataset except WN18RR, where the performance of log-loss drops more sharply than the performance of softmax. Note that margin-loss is not included in this comparison since it is inherently 1-1 positive to negative example. These results show that hyperparameter settings that seem to work the best are also the ones that have the largest memory footprint.

3.3.2 Experiments with Joint Framework

In this section we analyze the effects of using our joint framework JoBi with DistMult, ComplEx and SimplE on three standard datasets FB15K, FB15K-237 and YAGO3-10, and an additional very large dataset FB1.9M we that construct by iteratively removing from FB3M all entities that occur less than 5 times until no such entities remain. For all datasets we construct a corresponding type-ontology directly from the training set, with the method described in Section 3.2.3. The statistics for these datasets can be found in Table 3.1

For analysis and ablation studies we focus on YAGO3-10 since it is 10 times larger than FB15K or FB15K-237, so we expect it to better reflect how the performance of the models scale.

Baselines We use our re-implementations of DistMult, ComplEx and SimplE as our primary baselines. We also compare our results with the results reported in Jain et al. (2018b) for their model TypeComplex, which is designed to improve on ComplEx by using implicit type information within a KG. TypeComplex does this by amending the scoring function of ComplEx by additional parameters to capture the type compatibility between the entity-relation tuples within the triple. It learns two embeddings $\mathbf{u}_e \in \mathbb{C}^k$ and $\mathbf{a}_e \in \mathbb{C}^d$ for each entity $e \in \mathcal{E}$ and three embeddings $\mathbf{v}_r, \mathbf{w}_r \in \mathbb{C}^k$ and $\mathbf{b}_r \in \mathbb{C}^d$ for each relation $r \in \mathcal{R}$. The scoring function for TypeComplex is defined as:

$$s_c(h,r,t) = \mathrm{Re}\left(\mathbf{a}_h^\top \mathrm{diag}(\mathbf{b}_r)\overline{\mathbf{a}_t}\right) \tag{3.16}$$

$$s(h,r,t) = \sigma\big(s_c(h,r,t)\big) \cdot \sigma\left(\mathbf{u}_h^\top \mathbf{v}_r\right) \cdot \sigma\left(\mathbf{u}_t^\top \mathbf{w}_r\right), \tag{3.17}$$

where $\overline{\mathbf{a}}$ is the complex conjugate of $\mathbf{a}$.

3.3.2.1 Implementation Details

We optimize all models with stochastic gradient descent using Adam (Kingma and Ba, 2015), and perform early stopping with hits@10 on the validation set, where evaluation is performed every five epochs. For all our experiments, we fix the initial learning rate to 0.001. For experiments on FB15K, FB15K-237 and YAGO3-10, we perform a grid search over batch sizes: $\{500, 1000\}$, negative ratios $n_{neg} : \{50, 100\}$, pair-loss weight $\alpha : \{0.5, 1\}$ where applicable, and fix embedding size to 200 and biased sampling probability p to 0.3. We choose the hyperparameters that give the highest hits@10 on the validation set, and use these hyperparameters to report the final results on the test set. For FB1.9M, we use the best hyperparameters from YAGO3-10. Due to memory constraints, we set the embedding size to 100.

For most of our experiments, we choose to use ComplEx as the base for our model (**JoBi ComplEx**), since this configuration consistently outperformed others in preliminary experiments. To test the effect of our techniques on different bilinear models, we report results with DistMult (**JoBi DistMult**) and SimplE (**JoBi SimplE**) on FB15K-237. We also run experiments for explicit comparison of our model with TypeComplex and report it in Table 3.6.

3.3.3 Discussion

It could be seen in Table 3.3 that JoBi ComplEx outperforms both ComplEx and DistMult on all three standard datasets, on all the metrics we consider. For Hits@1, JoBi Complex outperforms baseline ComplEx by 4% on FB15K-237, 6.4% on FB15K and 5.6% on YAGO3-10.

Moreover, results in Table 3.3 demonstrate that JoBi improves performance on DistMult and SimplE. It should be noted that on FB15K-237, all JoBi models outperform all the baseline models, regardless of the base model used.

Lastly, results on FB1.9M (Table 3.4) demonstrate that JoBi improves performance on this very large dataset, where it is not possible to perform softmax over the entire set of entities, or have very large embedding sizes due to memory constraints.

Although one epoch for JoBi takes slightly longer than the baseline, JoBi converges in fewer epochs, resulting in shorter running time overall. We report running times on FB1.9M in Table 3.5.

FB15K-237	*h@1*	*h@3*	*h@10*	*MRR*
SimplE	0.160	0.268	0.430	0.248
DistMult	0.158	0.271	0.432	0.247
ComplEx	0.159	0.275	0.441	0.25
JoBi SimplE	0.188	0.301	0.461	0.277
JoBi DistMult	**0.205**	0.316	0.466	0.29
JoBi ComplEx	0.199	**0.319**	**0.479**	**0.29**
FB15K	*h@1*	*h@3*	*h@10*	*MRR*
DistMult	0.587	0.785	0.867	0.697
ComplEx	0.617	0.803	0.874	0.72
JoBi ComplEx	**0.681**	**0.824**	**0.883**	**0.761**
YAGO3-10	*h@1*	*h@3*	*h@10*	*MRR*
DistMult	0.252	0.407	0.568	0.357
ComplEx	0.277	0.44	0.589	0.383
JoBi ComplEx	**0.333**	**0.477**	**0.617**	**0.428**

Table 3.3: Performance on different datasets against baselines, where h@k denotes hits at k. Results are reported on test sets with the best parameters found in grid search for each model.

ComplEx	*h@1*	*h@3*	*h@10*	*MRR*
Baseline	0.424	0.598	0.721	0.530
JoBi	**0.452**	**0.615**	**0.726**	**0.550**

Table 3.4: Performance on the large-scale FB1.9M dataset, measured against the best performing baseline.

	# epochs	**training time**
ComplEx	70	5 days 5 hours 8 minutes
JoBi ComplEx	30	4 days 19 minutes

Table 3.5: Runtimes of ComplEx and JoBi Complex on FB1.9M.

Comparison with TypeComplex For results of TypeComplex, Jain et al. (2018a) use a wider set of negative ratios in their grid search than we do. To isolate the effects of the different models from hyperparameter choices, we set the negative ratio for our model

FB15K-237	*h@1*	*h@3*	*h@10*	*MRR*
ComplEx	0.209	0.347	0.535	0.314
TypeComplex	**0.296**	-	0.575	**0.389**
JoBi ComplEx	0.276	0.416	**0.587**	0.377
FB15K	*h@1*	*h@3*	*h@10*	*MRR*
ComplEx	0.630	0.818	0.895	0.734
TypeComplex	0.663	-	0.885	0.754
JoBi ComplEx	**0.702**	0.847	**0.906**	**0.782**
YAGO3-10	*h@1*	*h@3*	*h@10*	*MRR*
ComplEx	0.412	0.587	0.701	0.516
TypeComplex	**0.516**	-	0.702	0.587
JoBi ComplEx	0.507	0.647	**0.742**	**0.591**

Table 3.6: Comparison with TypeComplex where the scores are calculated ranking only the tail entities. Results for TypeComplex are taken from Jain et al. (2018a). h@k denotes hits at k.

.

to be 400 to match the setting on their best performing models. We keep the other hyperparameters the same as the best performing models for the previous experiments.

Jain et al. (2018a) use a modified version of the ranking evaluation procedure to report their results, where they only rank the tail entity against all other entities. To be able to compare our model to theirs, we also report the performance of our framework on this modified metric. The results for these experiments can be found in Table 3.6.

Our model generally outperforms TypeComplex by a large margin on hits@10. It also outperforms TypeComplex on MRR by a moderate margin except on FB15K-237, the smallest dataset. On the other hand, TypeComplex outperforms our model on hits@1 in two out of the three datasets. In fact for FB15K, TypeComplex does worse on hits@10 compared to the baseline model. This suggests that TypeComplex may be compromising on hits@k where k is larger to improve the hits@1 metric, which might be undesirable depending on the application.

Qualitative analysis. We analyze correct predictions made by JoBi ComplEx but not regular ComplEx. Among relations in YAGO3-10, a major performance gain can be observed for `hasGender`. The improvement comes solely from tail-entity predictions,

	h@1	*h@3*	*h@10*	*MRR*
Baseline	0.277	0.44	0.589	0.383
BiasedNeg	0.276	0.427	0.568	0.375
Joint	0.287	0.447	0.601	0.392
JoBi	**0.333**	**0.477**	**0.617**	**0.428**

Table 3.7: Results of ablation study on ComplEx model.

with hits@1 increasing from 0.22 to 0.86. Furthermore, we found that the errors made by ComplEx are exactly of the kind that can be mitigated by enforcing plausibility: ComplEx predicts an object that is not a gender (e.g. a sports team or a person) 65% of the time; JoBi makes such an obvious mistake only 2% of the time.

3.3.4 Ablation Studies

We compare joint training without biased sampling (**Joint**) and biased sampling without joint training (**BiasedNeg**) to the full JoBi model on YAGO3-10. We optimize hyperparameters by grid search over batch sizes: $\{200, 500, 1000\}$, negative ratios $n_{neg} : \{50, 100\}$, biased sampling probability $p : \{0.1, 0.2, 0.3\}$ and pair-loss weight $\alpha : \{0.25, 0.5\}$ where applicable.

Discussion. In Table 3.7 it can be seen that Joint on its own gives a slight performance boost over the baseline, and BiasedNeg performs slightly under the baseline on all measures. However, combining our two techniques in JoBi gives 5.6% points improvement on hits@1. This suggests that biased negative sampling increases the efficacy of joint training greatly, but is not very effective on its own.

The reason behind BiasedNeg performing worse on its own but better with Joint could be the choice of binary cross entropy loss for the pair module. We speculate that as the negative ratio increases, the ratio of negative to positive examples for this module becomes more skewed. Biasing the negative triples in the training alleviates this problem by making the classes more balanced, and allows the joint training to be more effective.

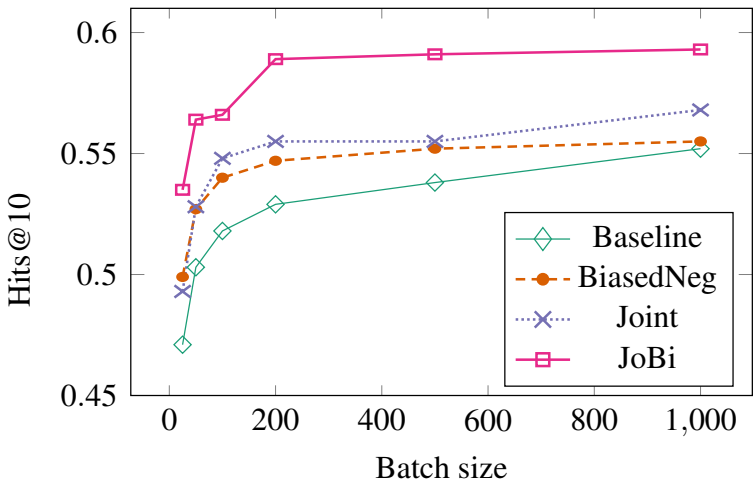


Figure 3.3: Performances on YAGO3-10 with different batch sizes

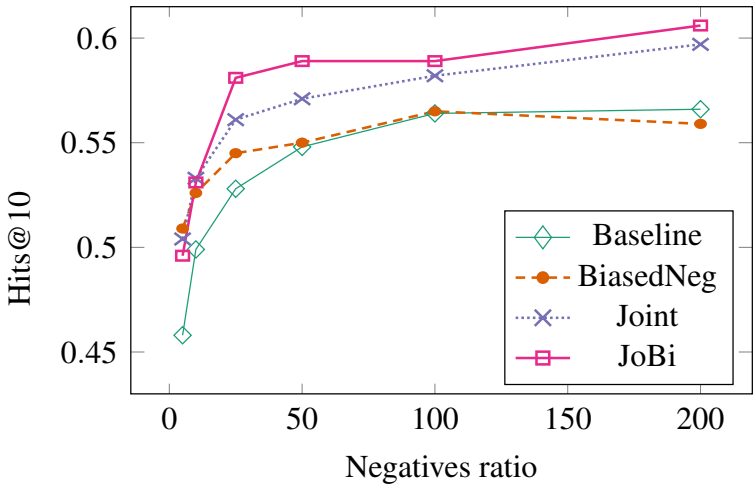


Figure 3.4: Performances on YAGO3-10 with different negative ratios

3.3.5 Robustness to Hyperparameter Settings

We also conduct experiments to isolate the effect of our techniques on varying batch sizes and negative ratios. For these experiments, we keep everything but the batch size constant ($n_{neg} = 25$, $\alpha = 0.5$, $p = 0.3$) and plot the change in hits@10 as the batch size varies in $\{25, 50, 100, 200, 500, 1000\}$. For demonstrating the effects of varying negative ratios, we keep everything but n_{neg} constant (batch size = 200, $\alpha = 0.5$, $p = 0.3$) and plot hits@10 as n_{neg} takes values in $\{5, 10, 25, 50, 100, 200\}$.

Discussion. The results for this experiment are presented in Figures 3.3 and 3.4. These show that JoBi not only consistently performs the best over the entire range of parameters, but also delivers a performance improvement that is especially large when the batch size or the negative ratio is small. This setting was designed to reflect the training conditions on very large datasets. It can be seen that BiasedNeg is more robust to low values of negative ratios, and both BiasedNeg and Joint alone show less deterioration in performance as the batch size decreases. When these two methods are combined in JoBi, the training becomes more robust to different choices on both these parameters.

3.3.6 Limitations of Experimental Design

As presented in Section 2.6, training choices have a large effect on final performance of KG completion models, which can make it difficult to determine whether the increase in scores is due to the main contributions presented in a paper or other slight differences in how the models are trained. For this reason, we re-implement all of the baseline models we consider and present an in-depth analysis of different loss functions and hyperparameter settings. From these experiments we conclude that NLL of softmax is the best choice of loss function, and analyze the effect of our Joint framework on hyperparameters in this context. However, an important detail in the implementation of softmax loss is whether softmax is applied to predictions from a mixed set of head-corrupted and tail-corrupted triples, or whether it is applied separately to each type of corrupted triple separately and losses are combined afterwards. Our implementation features the former, whereas the latter more closely resembles the evaluation and is likely to result in better scores. Hence it is possible that changing this in the experimental setting would negate some of the benefits obtained by joint training with biased sampling reported in this work.

3.4 Related Work

Using pair occurrences for embedding models has been considered before, both as explicit model choices and as negative sampling strategies. Chang et al. (2014) and Krompaβ et al. (2015) use pair occurrences to constrain the set of triples to be used in the optimization procedure. For methods that rely on SGD with contrastive training, this translates to a special case of our biased sampling method where $p = 1$. Garcia-Durán et al. (2016) present *TATEC*, a model that combines bigram and trigram interactions. The trigram model uses a full matrix representation for relations, and hence has many more parameters compared to our model. Jain et al. (2018a) present *JointDM* and *JointComplex*, which could be viewed as a simplification of *TATEC*. Unlike our model, both of these methods use the bigram terms both in training and evaluation, do not share any of the embeddings between two models, and do not provide supervision based on pair occurrences in the data.

Chapter 4

Learning Entity Embeddings from Knowledge Graph and Corpus

4.1 Introduction

The standard datasets and evaluation frameworks for knowledge graph completion assume that the query triples come from the same distribution as the underlying KG, which translates to the assumption that the more the system knows about an entity, the more likely it is to get queried. This down-weighting of importance for rare entities might not always correspond to scenarios in real-life applications: one can easily imagine that the less you know about an entity, the more likely you are to query it. In this chapter, we aim to address this issue and to identify a method for improving the accuracy of knowledge graph completion algorithms on entities rarely or never seen in training data, while leveraging large amounts of unlabeled data from an entity-linked corpus.

We use the framework *coupled matrix tensor factorization* (CMTF; Acar et al. 2013) to embed words from a corpus and entities from a knowledge graph in the same space. This allows us to leverage easily available, abundant textual data to provide additional context to the entity embeddings. This is especially helpful in the case of rare or out-of-vocabulary KG entities because the model can still leverage the distances between the entities according to the unlabeled data, where they might be less rare, to make predictions for the KG completion task. For example, seeing that *Switzerland*, *Austria* and *Liechtenstein* frequently occur in similar contexts in the unlabeled data, where the

latter does not occur in the knowledge graph, might assist inferring from a fact in the KG such as $(Austria, Borders, Switzerland)$, that $(Liechtenstein, Borders, Switzerland)$ holds as well.

The two components in our use of CMTF corresponds to DistMult for the knowledge graph completion and GloVe for word embeddings. Both of these models are amenable to our formulation, as they can be viewed as performing tensor and matrix factorizations. We propose two ways to exploit the CMTF framework with these two models: through the addition of *hard constraints* to tie the embeddings learned from the knowledge graph and the embeddings learned through their occurrence in the text and through *soft constraints*, which let the two diverge to some extent. Figure 4.1 schematically presents the way we tie the embeddings from the knowledge graph and the embeddings from unlabeled text.

For our experiments, we use the standard knowledge graph completion datasets coupled with a corpus from English Wikipedia. For FB15K, FB15K-237 and YAGO3-10, we link the KG to the corpus using two different methods, with one yielding many more links per entity than the other. This allows us to analyze the effects of the link frequency on the suggested models.

In addition to evaluating our methods on datasets with the original training-validation-test set splits, we also report results on modified versions of these datasets where most or all the triples containing some portion of the entities are removed from the training set. This is done in order to evaluate the model performance on few and zero-shot settings.

For modified datasets, we make the distinction between evaluating on entities with very few known facts (*rare*), and evaluating on entities with no known facts (*zero-shot*). For both these versions we make the further distinction between *original* and *artificial* settings. This distinction aims to capture two different possible sources for having little or no data about a given entity. In the original setting, there is the implicit assumption that each link in the underlying, complete KG has equal probability of getting sampled in the training data. The intuition is that rare entities in this setting have less data because they intrinsically have less number of correct facts associated with them. In the artificial setting, the entities the model is evaluated on are rare in the training set not due to their rarity in the underlying KG, but rather due to an imbalance in the sampling procedure. In other words, the KG defined by the training set knows

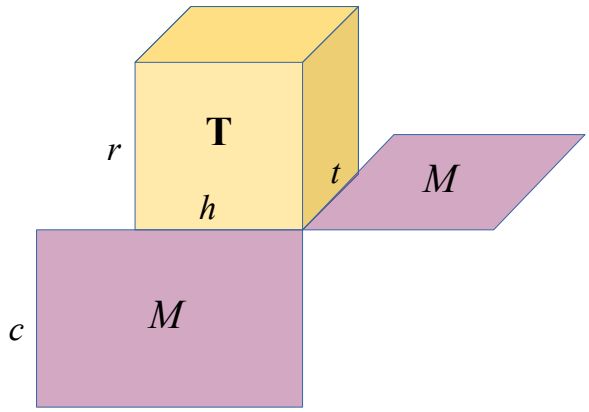


Figure 4.1: A diagram describing CMTF in the context of our work. The tensor $\mathbf{T}$ denotes the knowledge graph which needs to be completed, with the dimensions corresponding to entities (t and h) and relations (r). The entity dimensions are shared with a co-occurrence matrix M (with a dimension for c, the context of a word) that also includes common words (and as such, the co-occurrence matrix has a larger dimension than the entity dimension).

less about these entities than others.

Our results show that there is a significant gap in performance for all models between original and artificial versions. Furthermore, some of our models work better than the baseline on one version but not the other. In practical applications we expect both of these causes to play some part in the rarity of some entities in KGs, and that the KG can resemble the original or the artificial setting more depending on the way it was constructed and is being utilized.

4.2 Background and Notation

The main framework under which we operate is that of *coupled matrix and tensor factorizations* (CMTF; Acar et al. 2013), which is introduced in Section 2.2.3. This is a framework for *data fusion*, where two or more datasets that can be represented as matrices or tensors are coupled in one of the modes and factorized jointly. The factors of the coupled mode then uncover latent structure that both the datasets share.

The work of Acar et al. (2013) contrasts the use of an Alternating Least Square (ALS) algorithm to a gradient-update method to find the joint factorization. Their finding shows that the ALS algorithm, in which each factor is optimized in a coordinate-descent style, does not perform as well as the gradient-based method. As such, we

choose to use Stochastic Gradient Descent (SGD) with Adam for our optimization based on automatic differentation with TensorFlow.

4.2.1 GloVe Word Embeddings

Word embeddings are low-dimensional, dense vectors that capture the meaning of each word in terms of its similarity to other words in the vocabulary. This means that in a good embedding representation, two words that have similar meanings should be represented by points that are close to one another in the embedding space.

GloVe (Pennington et al., 2014) is a model for learning word embeddings that operates on the co-occurrence matrix. This is a sparse matrix $\mathbf{X} \in \mathbb{R}^{d \times d}$ where d is the size of the vocabulary. $\mathbf{X}$ is obtained from a corpus, and the entry $X_{i,j}$ contains the count of the ith word and the jth word occurring in the same context. GloVe considers the context of a word to be n tokens before and after the word within a sentence, where n is a hyperparameter that ranges from 2 to 10. It also discounts the co-occurrence according to the distance between the two tokens, where if the words i and j occur d tokens apart, this contributes $1/d$ to the total count $X_{i,j}$.

GloVe uses a weighted least squares objective to factorize the logarithm of the co-occurrence matrix. It is closely related to an earlier factorization model *Latent Semantic Analysis* (Deerwester et al., 1990), however GloVe employs a weighting scheme that downweights the rare co-occurrances without causing the frequent co-occurrences to overwhelm the final representation. For an entry x in the co-occurrence matrix, the weighting is achieved with the following function:

$$f(x) = \begin{cases} (x/x_{\max})^{\alpha} & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases} \tag{4.1}$$

,

Where $x_{\max}$ and α are hyperparameters.

GloVe learns a focal embedding $\mathbf{w}$ and a context embedding $\tilde{\mathbf{w}}$ for each word, which correspond to the columns of the left and the right factor matrices. It also learns a focal and a context bias b and $\tilde{b}$ for each word to ensure that the factorization is symmetric with respect to context and focal embeddings. The loss function w.r.t the cooccurrence

$X_{i,j}$, the focal word w_i and the context word $\tilde{w}_j$ is given by:

$$\ell_{\text{GloVe}}(X_{ij}, w_i, \tilde{w}_j) = f\left(X_{ij}\right)\left(\mathbf{w_i}^\top \tilde{\mathbf{w}}_\mathbf{j} + b_i + \tilde{b}_j - \log X_{ij}\right)^2 \tag{4.2}$$

.

4.3 Our Joint Model

Our main model couples together the matrix factorization that GloVe performs on the co-occurrence table from unlabeled text and the tensor factorization that DistMult performs on the adjacency tensor of the KG. While this overall coupling falls under the framework of CMTF, we deviate from the literature that uses L2 loss to minimize the reconstruction error and instead use softmax over sampled negatives when calculating the loss for DistMult:

$$\ell_{\text{DistMult}}(h,r,t) = -\log\left(\frac{\exp(s(h,r,t))}{\sum_{t'}\exp(s(h,r,t')) + \sum_{h'}\exp(s(h',r,t))}\right) \tag{4.3}$$

and then define the objective function

$$\mathcal{L}_{\text{DistMult}} = \frac{1}{n_1}\sum_{i=1}^{n_1} \ell_{\text{DistMult}}(h^{(i)}, r^{(i)}, t^{(i)}), \tag{4.4}$$

where the set $\{(h^{(i)}, r^{(i)}, t^{(i)}) \mid i \in [n_1]\}$ composes our knowledge graph training set. Note that the parameters controlled by this objective are the relation and entity embeddings.

The key idea of our model is to pair tensor factorization for learning entity and relation embeddings from a knowledge graph, with matrix factorization to learn embeddings of the words corresponding to these entities from a corpus. We choose to use GloVe as the matrix factorization method for word embeddings. Hence, the loss from the co-occurrence matrix is defined by Equation 4.2.[1] The final objective $\mathcal{L}_{\text{GloVe}}$ is an average of the loss function across all pairs of tokens i and j where $X_{ij} > 1$.

Since DistMult could be cast as a tensor factorization model, and GloVe explicitly factorizes the log co-occurrence matrix, it is possible to couple these heterogeneous data sources using the CMTF framework. We try two ways of coupling the two sources.

[1] We set $\alpha = 3/4$ and $x_{\max} = 100$ as the hyperparameters to the weighting function, as in the original GloVe model.

In the *hard constraint* version of our model, we simply constrain the embeddings of entities obtained through factorizing the KG tensor, and the embeddings of the corresponding words in the corpus, to be equal. The objective in this case is the weighted addition of the GloVe objective in Eq. 4.2 and the DistMult objective in Eq. 4.3:

$$\mathcal{L}_{\text{hard}} = \mathcal{L}_{\text{GloVe}} + \lambda \mathcal{L}_{\text{DistMult}} \tag{4.5}$$

The hard constraint version of our model does not allow the embeddings from the unlabeled text and the embeddings for the entities to deviate at all – they share the same space. However, such a constraint might be overly prohibitive. As such, we introduce a *soft constraint* version of our factorization where we allow the embeddings of entities for the corpus and for the KG to diverge, while adding a regularization-like term that penalizes their divergence. The loss for the soft constraint version is:

$$\mathcal{L}_{\text{soft}} = \mathcal{L}_{\text{GloVe}} + \lambda_1 \mathcal{L}_{\text{DistMult}} + \lambda_2 \sum_{i=1}^{N} \|\mathbf{e_i} - \mathbf{w_i}\|^2 \tag{4.6}$$

We note that the training sets for the KG triples and the unlabeled text need to be aligned with respect to the entities, as described in Figure 4.1. Otherwise, there is no way to enable sharing in the embedding spaces of the two objectives through the entity mentions in each. This means we need to use an entity linker on the unlabeled text, so that we can identify mentions of entities in the corpus. Our approach to that is described in Section 4.4.3.

4.3.1 Learning

We optimize the objectives in Eq. 4.5 and Eq. 4.6 by using mini-batch Stochastic Gradient Descent with Adam (Kingma and Ba, 2015). Two important issues were needed to be resolved during such optimization, both related to imbalances in the data. The first imbalance is between the amount of text and the number of available facts during training in the knowledge graph. The second is an imbalance between the number of mentions of entities in the unlabeled text and mentions of general content words. We detail below our approach to these imbalances.

Scheduling training between text and knowledge graph In preliminary experiments, we discovered that the two objectives $\mathcal{L}_{\text{DistMult}}$ and $\mathcal{L}_{\text{GloVe}}$ are imbalanced with

respect to their underlying number of examples (there is more unlabeled text than KG facts). Even when each GloVe batch is set to have several magnitudes more examples than a DistMult batch, this causes the training to run through many DistMult epochs for each GloVe epoch, and causes the training of DistMult to be finished before a single epoch of the training set for GloVe could be completed.

In order to balance the two objectives, we use scheduling during training. At each step of the algorithm, we flip a biased coin to decide whether to process a minibatch from the corpus or the knowledge graph, where the probability for taking an optimization step on the corpus data decreases as the training progresses. This allows a continuous transition from pre-training to fine-tuning. More specifically, let t be the number of mini-batches processed so far. Then the probability $p(t)$ of processing a mini-batch from the KG is calculated as $p(t) = 1 - \exp(\alpha t)$. This is different than the original formulation of CMTF optimization (Acar et al., 2013), in which the coupled matrix and tensor are optimized all-at-once, will full batch gradient descent.

Upsampling Entity Examples We also discovered that entity occurrences in the corpus are relatively rare compared to the overall vocabulary. This again created an imbalance in the GloVe objective for datasets that use the entity annotations, this time between entities and common words. Since the KG objective focuses on entities, in these datasets we *upsample* entities when creating minibatches: if a co-occurrence pair includes an entity, we increase the chance that it will be included in a given minibatch five-fold compared to word-word co-occurrences.

4.4 Experiments

4.4.1 Baseline Models

As a strong baseline, we consider a linear regression model inspired by cross-lingual embeddings. This model learns GloVe and DistMult embeddings seperately, and then learns a linear mapping between the two.

For training the linear regression, we first train GloVe and DistMult seperately. After this, we divide the set of all entities that occur both in the KG and corpus into training and validation sets (90% and 10%). We train a linear regression model on the training set using gradient descent with Adam and with early stopping on validation set. Let

	FB15K-237	FB15K	WN18RR	YAGO3-10
Frequency cutoff	15	21	2	9
number of rare entities	1450	1495	4056	12313

Table 4.1: Frequency cutoff of rare entities, and the number of rare entities in the training set for each dataset

$\mathbf{e_g}$ denote the DistMult embedding for the entity e, $\mathbf{e_w}$ the GloVe embedding and f the learned mapping from GloVe embedding space to DistMult embedding space. We use f in two ways:

1. **Hard threshold:** For all rare or unseen entities e in the knowledge graph dataset, we replace $\mathbf{e_g}$ with $f(\mathbf{e_w})$.

2. **Soft threshold:** For each entity e in the KG dataset, we take the weighted average of $\mathbf{e_g}$ and $f(\mathbf{e_w})$, where the weight is controlled by a logistic function dependent on the frequency of the entity in the dataset, giving most of the weight to $\mathbf{e_g}$ when e is frequent in the KG, less weight if it is rare.

$$\mathbf{e} = \mu\mathbf{e_g} + (1-\mu)f(\mathbf{e_w}) \tag{4.7}$$

$$\mu = (1+\exp(-k\ \text{freq}_{KG}(e) - x_0))^{-1} \tag{4.8}$$

Where we set x_0 to be the largest frequency in the rarest 10% of the entities, and $k = \frac{7}{x_0}$ so that $\mu \approx 0.001$ when $\text{freq}_{KG}(e) = 0$ and $\mu = 0.5$ when $\text{freq}_{KG}(e) = x_0$.

During test time, the embeddings for entities are updated and the evaluation is done by applying the scoring function to updated embeddings and ranking them as standard.

4.4.2 Datasets

For our experiments, we use the datasets FB15K (Bordes et al., 2013), FB15K-237 (Toutanova and Chen, 2015), YAGO3-10 and WN18RR (Dettmers et al., 2018). All but FB15K have been constructed to avoid test set leakage observed in previous standard datasets, where for most of the the triples in the test, a reverse triple expressing the same fact can be found in the training set.

Since our model focuses on rare entities, we construct modified datasets to capture the performance of each model on rare or unseen entities. More specifically, we are interested in the cases where the rare or unseen entity is *queried*. Therefore, for every

triple (h,r,t) in the evaluation set, if h is a rare or unseen entity then we consider the performance of the model on the query $(h,r,?)$, and if t is a rare or unseen, we likewise consider the performance on $(?,r,t)$.

4.4.2.1 Motivation

We construct our modified datasets to evaluate our model both on entities that are observed rarely in the original training sets, and also on entities that occur frequently on the original training set, but have been mostly or completely removed from the modified dataset. The former corresponds to *rare-original* and *zero-shot original* settings, and the latter corresponds to *rare-artificial* and *zero-shot-artificial* settings.

The training/validation/test splits of the original datasets are performed by uniform sampling: a triple (h,r,t) has the same probability of being added to the test set regardless of the properties of h, r or t. If we consider the combination of all the training, validation and test triples of a dataset to define a complete KG, then we can say that all entities are missing approximately the same proportion of the correct facts associated with them in the training set. So in this sense, the uniform split of training/validation/test sets evaluates models on the scenario where the model is equally knowledgeable, or equally ignorant about all entities.

Rare-original setting does not modify this scenario of uniformly missing triples, but evaluates the model performance on entities which have relatively little correct facts associated with them in the complete KG. *Zero-shot original* setting modifies this assumption minimally by removing the rare entities from the training set completely, and evaluates the model on its performance on these entities. In this scenario, the model is uniformly ignorant about all entities except those that have very few correct facts in the complete KG, for which it knows nothing about.

Rare-artificial considers a different scenario where the model is much more ignorant about a subset of entities than all others. These entities $e' \in \mathcal{E}' \subset \mathcal{E}$ have many correct facts associated with them in the complete KG, but have varying probabilities $p_{e'}$ for a fact associated with them to be missing from the training set, where $p_{e'} \gg p_e$ for $e \notin \mathcal{E}'$. The model is then evaluated on these entities it is much less knowledgeable about compared to others. *Zero-shot artificial* takes this scenario to an extreme, and evaluates the performance of the model on entities that have many facts associated with them in the complete KG, but which the model knows nothing about.

4.4.2.2 Construction

While constructing our moditified datasets, we take rare entities to be the bottom 10% when all entities are ordered according to their frequency. The four modified versions of each original dataset is described below:

Rare-original For this dataset, we leave the training and validation sets as they were, and train the model on the training set with early stopping on the validation set. As results on rare entities we report mean reciprocal rank of all queries $(h_{rare}, r, ?)$ and $(?, r, t_{rare})$ where (h_{rare}, r, t) and (h, r, t_{rare}) are the triples in the validation set where either the head entity or the tail entity has occurred rarely in the training set.

Rare-artificial For this dataset we leave the validation set as is, but modify the training and test sets with the following procedure:

We first split the entities into two sets: frequent entities and rare entities, with rare entities spanning the least frequent 10% of the original training set. We then randomly select n entities from the frequent set, where n is the number of the original rare entities that fall into the least 10%. The selected entities are then pruned from the training set so that their frequency distribution mirrors the original rare entities. We do this by matching one rare entity with one selected frequent entity and removing all but k triples of this entity from the training set, where k is the frequency of the rare entity it has been matched with.

We train the model on the modified training set with early stopping on the original validation set. We then report results on the triples we have removed from the training set.

Zero-shot-original For this dataset we remove all rare entities from the training set, and report results on the triples that have been removed.

Zero-shot-artificial For this dataset we randomly choose 10% of the entities and remove all triples corresponding to them from the training set. We report the results on the removed triples.

	FB15K		FB15K-237		WN18RR		YAGO3-10	
	train	*eval*	*train*	*eval*	*train*	*eval*	*train*	*eval*
Rare original	483142	418	272115	213	86835	170	1079040	38
Zero-shot original	475877	7265	268202	3913	82795	4040	1058353	20687
Rare artificial	387020	96122	219816	52299	71363	15472	886405	192635
Zero-shot artificial	386363	96779	205973	66142	69469	17366	852198	226842

Table 4.2: Number of triples in training and evaluation sets on all the modified datasets

		FB15K		FB15K-237		YAGO3-10	
		rare	*all*	*rare*	*all*	*rare*	*all*
mean frequency in corpus	*FACC*	20	43	25	44	7	51
	anchors	2361	2590	2361	2590	193	331
ratio of entities linked	*FACC*	0.52	0.45	0.54	0.45	0.10	0.09
	anchors	0.91	0.93	0.91	0.93	0.95	0.77

Table 4.3: Statistics of linking KG datasets to the corpus. Mean frequency in the corpus and % of entities linked is reported on the entire training set (*all*) and the least frequent 10% of the entities (*rare*) for each dataset, and for linking performed based on FACC annotations, and with Wikipedia anchors.

4.4.3 Linking Datasets to Corpus

We use three different methods to obtain a corpus where the entities are linked to the KG completion datasets. For FB15K, FB15K-237 and YAGO3-10, we try two different linking methods on similar corpora. For the first method (*FACC*), we use the FACC annotations (Gabrilovich et al., 2013) for obtaining entity linking for the English Wikipedia section of ClueWeb09 (Callan et al., 2009). For the second method, we use an entity-linking procedure based on the code of Yamada et al. (2020) that uses Wikipedia anchors to detect mentions of entities in a recent dump of English Wikipedia. The comparison of the resulting linking statistics from the two methods can be found in Table 4.3. We link WN18RR to the English Wikipedia section of ClueWeb09 after POS-tagging in order to aid the heuristic matching of the corpus tokens with WordNet synsets. We describe all our methods in detail in Appendix A.

4.4.4 Implementation Details

As the vocabulary for the co-occurrence table, we take the most frequent 100,000 words, plus any entity annotations if it occurred more than 5 times. We consider a window of size 10 for counting co-occurrences, and weight the co-occurrences by $1/i$ if the target and the context word occurred i positions apart. We initialize GloVe with pre-trained embeddings.[2]

We tune all hyperparameters on FB15K-237. For embedding dimension and number of negatives per positive example, we perform gridsearch on DistMult only, with the grid: negatives: [50, 100, 200] and embedding dimension: [50, 100, 200] and use the same parameters on our joint model for fair comparison. We use Adam optimizer to dynamically adjust the learning rate, and fix the initial learning rate to be 0.01. For scheduling, we experimentally set α to be 5×10^{-4} so that the training goes through approximately 30 epochs of GloVe objective before DistMult converges. We experimentally set the DistMult batch size to be 2000 and GloVe batch size to be 80000. This setting allows the joint training to fit in the memory of a regular GPU, and partially mitigates the issues arising by the imbalanced dataset sizes. For the soft-constraint experiments, we perform a gridsearch for the optimal λ_2 with the grid: [5, 1, 0.5, 0.1].

4.5 Results and Discussion

DistMult on modified datasets. We first present the results of *DistMult* on rare entities in Table 4.4, where no external information from an entity linked corpus is used. As expected, scores on the original rare entities is much higher than when some entities are rarefied from the training set artificially and at random. The model does better overall than on rare entities in WN18RR and YAGO3-10 as expected. On FB15K the model performs drastically better on the original rare entities from the validation set than the entire test set. The difference is especially salient at *hits@1* where, when it is a rare entity the model is queried against, it returns the other entity correctly 68% of the time, which is about 10% higher than its performance on the original test set. The same effect can be observed on FB15K-237, albeit to a lesser extent. Recall that FB15K contains many triples (h,r,t) in the validation and test sets where (t,r^{-1},h) is observed in the training set, and FB15K-237 was constructed to fix this problem. The surprisingly accurate predictions of *DistMult* on rarely seen entities suggest that

[2] http://nlp.stanford.edu/data/glove.6B.zip

FB15K-237	*MRR*	*h@1*	*h@3*	*h@10*
Original	0.288	0.203	0.315	0.459
Rare-original	0.317	0.233	0.345	0.466
Rare-artificial	0.173	0.117	0.191	0.283
FB15K	*MRR*	*h@1*	*h@3*	*h@10*
Original	0.690	0.583	0.770	0.866
Rare-original	0.745	0.680	0.814	0.837
Rare-artificial	0.199	0.140	0.218	0.311
WN18RR	*MRR*	*h@1*	*h@3*	*h@10*
Original	0.415	0.388	0.428	0.462
Rare-original	0.224	0.188	0.238	0.304
Rare-artificial	0.073	0.065	0.077	0.087
YAGO3-10	*MRR*	*h@1*	*h@3*	*h@10*
Original	0.320	0.215	0.373	0.531
Rare-original	0.223	0.143	0.286	0.357
Rare-artificial	0.059	0.043	0.058	0.088

Table 4.4: Results of DistMult on rare entities

FB15K might have other similar issues that were not completely fixed in FB15K-237.

Performance of Joint and Linear Mapping models. The performance of *DistMult*, linear regression model with hard mapping (*LR-H*) and soft mapping (*LR-S*), and joint model with hard constraint (*Joint-H*) and soft constraint (*Joint-S*) on the original FB15K, FB15K-237 and YAGO3-10 datasets can be found in Table 4.5. The results using linking obtained by Wikipedia anchors (**Anchor linking**) and by FACC annotations (**FACC linking**) are presented separately. All models using anchor links perform poorer or comparable to baseline *DistMult* on FB15K and FB15K-237, however *Joint* models show less deterioration in performance than the *LR* models. Surprisingly for YAGO3-10, *LR* models show slight a improvement over DistMult when using FACC links.

The results on **rare-original** versions of the same datasets are presented in Table 4.6. With anchor linking, the *Joint* models are performing better than or comparable to *DistMult*, while *LR* models have slightly worse performance on FB15K and YAGO3-10, and are slightly better on FB15K-237 than *DistMult*. Of the two *Joint* models,

		Original datasets								
		FB15K			**FB15K-237**			**YAGO3-10**		
		MRR	*h@1*	*h@10*	*MRR*	*h@1*	*h@10*	*MRR*	*h@1*	*h@10*
DistMult		**0.690**	**0.583**	**0.866**	0.288	0.203	0.459	0.320	0.215	0.531
Anchor links	*LR-H*	0.600	0.492	0.785	0.271	0.192	0.431	0.276	0.186	0.454
	LR-S	0.602	0.493	0.793	0.276	0.196	0.438	0.291	0.200	0.468
	Joint-H	0.630	0.502	0.850	**0.295**	**0.210**	**0.469**	0.286	0.185	0.489
	Joint-S	0.661	0.545	0.857	0.290	0.204	0.461	0.254	0.160	0.444
FACC links	*LR-H*	0.596	0.492	0.776	0.249	0.171	0.409	0.344	**0.248**	0.535
	LR-S	0.607	0.502	0.787	0.256	0.179	0.419	**0.346**	**0.249**	**0.537**
	Joint-H	0.664	0.548	0.858	0.288	0.205	0.459	0.185	0.111	0.332
	Joint-S	0.669	0.556	0.861	0.290	0.204	0.463	0.249	0.155	0.440

Table 4.5: Mean reciprocal rank (MRR), hits@1 and hits@10 obtained on the original datasets for DistMult, linear regression models with hard mapping (*LR-H*) and soft mapping (*LR-S*), and joint models with hard constraint (*Joint-H*) and soft constraint (*Joint-S*). Performance of linear regression and joint models are reported both with Wikipedia links based on hyperlink anchors, and entity linking provided in the FACC dataset.

mean reciprocal rank (MRR) for *Joint-H* is 10 points above *DistMult*, suggesting that it is particularly suitable for enriching embeddings of rare entities in this dataset. With FACC linking, *LR* models demonstrate drastically lower performance, but *Joint* models are comparable to *DistMult*, and *Joint-H* shows significant improvement on FB15K-237 even with the sparse linking that FACC provides.

On **rare-artificial** datasets (also shown in Table 4.6), *Joint-H* is consistently performing better than other models. However in this setup *LR* models also offer modest improvements on FB15K and FB15K-237, and the difference between *Joint* and *LR* models are significantly less than what is observed in the rare-original datasets. With FACC linking however, we observe the same pattern where the *LR* models perform below baseline *DistMult*, but *Joint* models perform comparably, with *Joint-H* showing 1.3% improvement in MRR for FB15K-237.

The performance of the models on **zero-shot-original** and **zero-shot-artificial** datasets in Table 4.7 reflect the pattern observed on the rare versions. With anchor links, *Joint-H* does significantly better than all other models on zero-shot-original versions of FB15K and FB15K-237, however for the zero-shot-artificial setting it does only slightly better

Rare-original datasets

		FB15K			FB15K-237			YAGO3-10		
		MRR	*h@1*	*h@10*	*MRR*	*h@1*	*h@10*	*MRR*	*h@1*	*h@10*
DistMult		0.745	0.680	0.848	0.297	0.233	0.431	**0.249**	0.171	**0.366**
Anchor links	*LR-H*	0.631	0.56	0.739	0.308	0.245	0.456	0.157	0.119	0.262
	LR-S	0.661	0.605	0.757	0.316	0.263	0.453	0.147	0.095	0.238
	Joint-H	0.734	0.649	**0.871**	**0.395**	**0.319**	**0.552**	0.233	0.167	0.333
	Joint-S	**0.750**	**0.683**	0.846	0.295	0.207	0.461	0.192	0.119	0.285
FACC links	*LR-H*	0.085	0.057	0.120	0.074	0.052	0.108	0.000	0.000	0.000
	LR-S	0.092	0.066	0.127	0.083	0.060	0.125	0.000	0.000	0.000
	Joint-H	0.738	0.669	0.844	0.347	0.280	0.478	0.226	0.146	0.341
	Joint-S	0.735	0.662	0.848	0.294	0.224	0.453	0.244	**0.195**	0.341

Rare-artificial datasets

		FB15K			FB15K-237			YAGO3-10		
		MRR	*h@1*	*h@10*	*MRR*	*h@1*	*h@10*	*MRR*	*h@1*	*h@10*
DistMult		0.177	0.124	0.277	0.181	0.121	0.297	0.058	0.043	0.086
Anchor links	*LR-H*	0.204	0.144	0.319	0.186	0.124	0.306	0.042	0.027	0.068
	LR-S	0.202	0.143	0.318	0.185	0.127	0.303	0.039	0.025	0.064
	Joint-H	**0.219**	**0.149**	**0.352**	**0.199**	**0.131**	**0.331**	0.074	0.051	0.117
	Joint-S	0.199	0.141	0.313	0.171	0.113	0.289	0.059	0.042	0.09
FACC links	*LR-H*	0.054	0.035	0.089	0.069	0.049	0.104	0.004	0.003	0.007
	LR-S	0.053	0.034	0.088	0.063	0.042	0.100	0.038	0.028	0.056
	Joint-H	0.177	0.124	0.276	0.194	0.131	0.315	0.068	0.048	0.106
	Joint-S	0.173	0.122	0.271	0.182	0.121	0.299	**0.082**	**0.057**	**0.131**

Table 4.6: Results on the rare entities in the original datasets (**Rare-original**), and where some entities were picked at random and rarefied in the training set (**Rare-artificial**). Mean reciprocal rank (MRR), hits@1 and hits@10 on rare entities in is reported for DistMult, linear regression models with hard mapping (*LR-H*) and soft mapping (*LR-S*), and joint models with hard constraint (*Joint-H*) and soft constraint (*Joint-S*). Performance of linear regression and joint models are reported both with Wikipedia links based on hyperlink anchors, and entity linking provided in the FACC dataset.

Zero-shot original datasets

		FB15K			FB15K-237			YAGO3-10		
		MRR	*h@1*	*h@10*	*MRR*	*h@1*	*h@10*	*MRR*	*h@1*	*h@10*
DistMult		0.006	0.001	0.008	0.056	0.046	0.074	0.000	0.000	0.000
Anchor links	*LR-H*	0.030	0.015	0.054	0.149	0.108	0.227	0.081	0.059	0.124
	LR-S	0.028	0.015	0.051	0.160	0.116	0.240	**0.087**	**0.067**	**0.127**
	Joint-H	**0.132**	**0.098**	**0.198**	**0.292**	**0.237**	**0.402**	0.014	0.006	0.025
	Joint-S	0.068	0.054	0.093	0.160	0.116	0.240	0.000	0.000	0.000
FACC links	*LR-H*	0.089	0.070	0.115	0.123	0.110	0.146	0.007	0.001	0.024
	LR-S	0.108	0.074	0.156	0.130	0.114	0.158	0.009	0.001	0.024
	Joint-H	0.019	0.005	0.051	0.111	0.091	0.144	0.000	0.000	0.000
	Joint-S	0.009	0.004	0.014	0.038	0.025	0.063	0.000	0.000	0.000

Zero-shot artificial datasets

		FB15K			FB15K-237			YAGO3-10		
		MRR	*h@1*	*h@10*	*MRR*	*h@1*	*h@10*	*MRR*	*h@1*	*h@10*
DistMult		0.016	0.0009	0.028	0.014	0.009	0.021	0	0	0.001
Anchor links	*LR-H*	0.023	0.013	0.04	0.134	**0.086**	0.228	0.018	0.011	0.03
	LR-S	0.026	0.015	0.044	**0.136**	**0.087**	**0.232**	0.018	0.01	0.032
	Joint-H	**0.085**	**0.057**	**0.136**	0.119	0.075	0.2	0.008	0.004	0.016
	Joint-S	0.004	0.001	0.009	0.014	0.009	0.024	0	0	0.001
FACC links	*LR-H*	0.061	0.042	0.097	0.066	0.048	0.100	0.002	0.001	0.003
	LR-S	0.062	0.042	0.099	0.064	0.045	0.102	0.002	0.001	0.003
	Joint-H	0.023	0.013	0.037	0.059	0.039	0.095	0.001	0.000	0.001
	Joint-S	0.016	0.012	0.022	0.022	0.013	0.039	0.009	0.008	0.011

Table 4.7: Results on the modified datasets where the original rare entities (**zero-shot-original**) or randomly chosen frequent entities (**zero-shot-artificial**) were completely removed from the training set. Mean reciprocal rank (MRR), hits@1 and hits@10 on removed triples are reported for DistMult, linear regression models with hard mapping (*LR-H*) and soft mapping (*LR-S*), and joint models with hard constraint (*Joint-H*) and soft constraint (*Joint-S*). Performance of linear regression and joint models are reported both with Wikipedia links based on hyperlink anchors, and entity linking provided in the FACC dataset.

WN18RR

	Original			**Rare-original**			**Rare-artificial**		
	MRR	*h@1*	*h@10*	*MRR*	*h@1*	*h@10*	*MRR*	*h@1*	*h@10*
DistMult	**0.423**	**0.396**	**0.472**	**0.224**	**0.188**	**0.304**	**0.073**	**0.065**	**0.087**
LR-H	0.271	0.257	0.292	0.029	0.028	0.028	0.002	0.002	0.003
LR-S	0.347	0.318	0.405	0.029	0.028	0.028	0.008	0.004	0.015
Joint-H	0.402	0.379	0.441	0.179	0.122	0.26	0.069	0.065	0.087
Joint-S	0.27	0.174	0.437	**0.223**	**0.188**	0.293	**0.073**	**0.065**	**0.088**

	Zero-shot original			**Zero-shot artificial**		
	MRR	*h@1*	*h@10*	*MRR*	*h@1*	*h@10*
DistMult	0.002	0	0.004	0.001	0	0.002
LR-H	0.009	0.004	0.016	0.005	0.003	0.007
LR-S	0.003	0.001	0.005	0.001	0	0.002
Joint-H	0.008	0.002	0.019	0.01	0.004	0.02
Joint-S	0.004	0.003	0.005	0.001	0	0.002

Table 4.8: Results on modified WN18RR datasets.

on FB15K compared to the *LR* models, and lags behind them on FB15K-237. This suggests that *Joint* models might be more suitable for modelling the situation captured by the *original* settings, where rare entities have a few facts in the underlying complete KG. For YAGO3-10, while *LR* models seems to do slightly better than *Joint* for both zero-shot settings, performance is low for all models, and it is unclear whether the difference is significant. A surprising observation is that with the sparse FACC linking, *LR* models perform better than *Joint* models, and obtain higher scores on FB15K compared to *LR* results with anchor linking.

We present results on all versions of WN18RR separately in Table 4.8. All models are doing comparable or worse than *DistMult* on **original** and both the **rare** versions, and do not seem to perform notably better than chance on the **zero-shot** settings. This suggests that the semantic content GloVe embeddings capture from our POS-tagged corpus is different enough from the information needed to predict WordNet relations such as *hyponym* and *hypernym* that integrating this information interferes with link prediction.

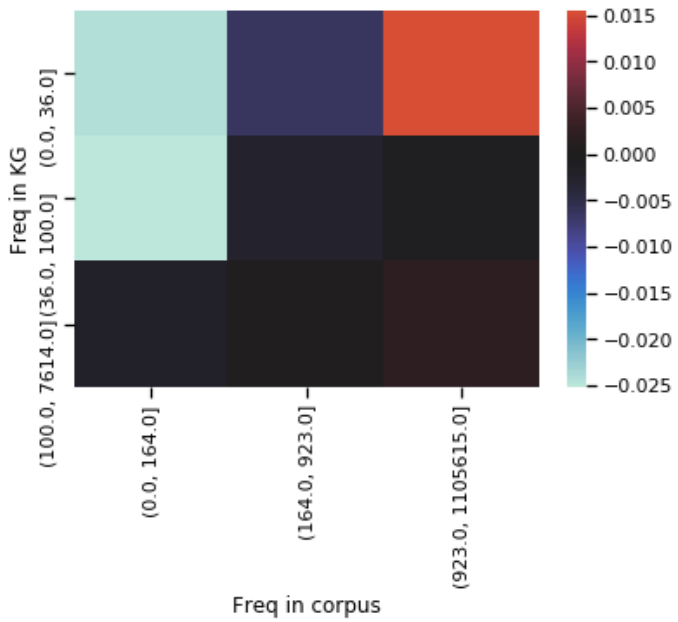


Figure 4.2: Difference between MRR of *Joint-H* and *DistMult* grouped by the frequency of the query entity in the KG and in the corpus on FB15K-237 with anchor linking. Values higher than 0 (red) mean Joint-H is performing better, and lower (blue) mean DistMult is performing better.

Effects of the interaction between corpus and KG frequency on Joint model. As further analysis, in Figure 4.2 we visualize the difference in performance between *DistMult* and *Joint-H* for entities grouped by both their frequency in the KG training set, and in the corpus for the original FB15K-237 with anchor linking. We can observe that the *Joint-H* improves performance on entities that are in the top one-third for frequency in the corpus, but in bottom one-third for frequency in the KG data. Likewise, it shows worse performance on entities which are infrequent in the corpus. It is interesting that the main difference in performance is observed on entities that are infrequent in the KG. This shows that *Joint-H* is largely successful at adjusting the effect of corpus information on entity representations based on the scarcity of facts for that entity. It however also suggests that the model can be improved by better taking into account the frequency of the entity in the corpus.

4.6 Conclusion and Future Work

In this chapter, we cast the task of learning embeddings that contain information both from the knowledge graph and an entity-linked corpus as a joint matrix-tensor factor-

ization model, and show that this can improve performance on rare entities in knowledge graph completion. Our experiments feature *DistMult* as the baseline bilinear model, however our framework is applicable for any parametrization that admits the interpretation of a tensor decomposition. Specifically, *TuckER* (Balazevic et al., 2019) provide a simple bilinear scoring function that achieves state-of-the-art results on standard datasets. Whether our joint training framework also provides performance improvements on rare-entities when the KG component is modelled by other bilinear models such as *TuckER* is worth exploring in future work.

Chapter 5

Tensors over Semirings for Weighted Logic Programs

While research on knowledge graphs in the Natural Language Processing community has focused mostly on learning low-dimensional embeddings for entities and relations with heavy emphasis on link prediction, in other fields, most notably Semantic Web and Databases, there has also been significant research effort on efficient ways to perform *reasoning* over knowledge graphs.

One prominent approach to deduction in knowledge bases is *Datalog*, which is a logic programming paradigm modelled after Prolog, and developed specifically for implementing *deductive databases*. This type of database consists of a set of ground facts similar to the triples in knowledge graphs, together with a set of deduction rules. A given candidate fact is deemed correct by the database if it can be derived from the ground facts by applying the deduction rules.

There has been work on unifying deduction in logic programming with KG embeddings. For example, Rocktäschel and Riedel (2017) modify the unification process employed by Prolog for proof search to take into account embedding representations of entities and relations, and develop a neural model that concurrently learns KG embeddings and mines logical rules. Mei et al. (2020) add embedding representations to Datalog programs in order to integrate learning with logical reasoning in temporal knowledge bases.

Techniques from logic programming have also been incorporated into NLP, especially in parsing through the *parsing as deduction* (Pereira and Warren, 1983) paradigm

which uses the framework of logical deduction to abstract over implementation details of parsing algorithms. In this view, production rules correspond to *axioms*, a string is analogous to a *theorem* and a correct parse is analogous to a *proof*. Hence, proof search techniques from logic programming have found applications as tools for inferring new facts from known ones in deductive databases, for integrating logical deduction in KG embedding models, and for implementing grammatical parsing algorithms in NLP.

The nature of parsing natural language, however, differs from that of deduction in knowledge bases in an important aspect. Both for automatic deduction and for parsing in NLP, there might be many correct proofs/parses for a given input theorem/string. For deduction, any valid proof tree is enough to prove a theorem. However when parsing natural language sentences, the goal is often to pick the parse tree that matches human judgement among many valid ones. This difference motivates probabilistic grammar formalisms and probabilistic logic programs, where a rule has an associated probability value, and the probability of a parse/proof tree can be decomposed into the probabilities of its constituent rules.

Weighted grammars and *weighted logic programs (WLPs)* relax the requirement that the values associated with the rules must be probabilities. In a WLP, the axiom weights can take values from any set that can be cast as a general algebraic structure called a *semiring*. As in probabilistic logic programs, WLP proofs are assigned weights by combining the weights of the axioms used in the proof, and the weight of a theorem is in turn calculated by combining the weights of all its possible proof paths. The combinatorial nature of this procedure makes weighted logic programs highly suitable for specifying dynamic programming algorithms.

Goodman (1999) presents an elegant abstraction for specifying and computing parser values based on WLP where the values could be drawn from any complete semiring. This generalizes the case of Boolean decision problems, probabilistic grammars with Viterbi search and other quantities of interest such as the best derivation or the set of all possible derivations. With semiring weights, it is guaranteed that dynamic programming algorithms can be used to correctly calculate important quantities aggregated over sub-trees, such as the *inside* and *outside* values, regardless of the particular semiring chosen.

In this chapter we further generalize the WLP framework so that weights can take values which are *tensors with semiring entries*, while fully maintaining the guarantees for

the efficient calculation of the quantities of interest. Tensor WLPs are applicable both for reasoning on knowledge graph embeddings, and for latent variable parsing. Although we discuss both in our exposition, in the development of our theory we closely follow the work of Goodman (1999) on semiring parsing. Hence our theoretical results are geared more towards characterizing tensor WLPs in the context of parsing. Nevertheless, this work also provides a sound foundation for integrating logical reasoning within embedding frameworks for knowledge graphs, which we discuss in the context of our motivation applications.

5.1 Motivating Applications

We present two motivating examples for our extension: parsing latent variable grammars, and calculating the aggregate value of paths between two entities in a knowledge graph. Both of these employ dynamic programming algorithms that operate on tensor weights, and casting them as tensor WLPs enables useful abstractions.

5.1.1 Dynamic Programs for Latent Variable Parsing

Latent variable models have been an important component in the NLP toolbox. The central assumption in latent variable models is that the correlations between observed variables in the training data could be explained by unobserved, hidden variables. Latent variables have been used with grammars such as Probabilistic Context-Free Grammars (PCFGs), where each node in the parse tree is represented using a vector of latent state probabilities that further extend the expressiveness of the grammar (Matsuzaki et al., 2005).

In a PCFG, the weight for the rule $A \to BC$, is $p(A \to BC\,|\,A)$: the probability that the production rule $A \to BC$ is applied given that the current non-terminal is A. Latent Variable Probabilistic Context-Free Grammars (L-PCFGs) as presented by Cohen et al. (2014) allow the non-terminals in this rule to be decorated with latent states h_i^A, h_j^B, h_k^C where $i \in [d_A], j \in [d_B], k \in [d_C]$ range over the number of hidden variables for A, B and C respectively, so that independent probability values can be assigned to $p\left(A(h_i^A) \to B(h_j^B)\,C(h_k^C) \,\middle|\, A(h_i^A)\right)$ for all i, j, k. This can be more succinctly expressed by assigning a three-way tensor $\mathbf{T} \in [0,1]^{d_A \times d_B \times d_C}$ of probability values as the weight for $A \to BC$, where

$$T_{i,j,k} = p\left(A(h_i^A) \to B(h_j^B)\,C(h_k^C) \,\middle|\, A(h_i^A)\right).$$

More generally, probabilities associated with rules containing n non-terminals can be expressed as tensors of order-n.

The approach of adding latent variables to formal grammars has proven to be a fruitful one: in the context of PCFG parsing, Matsuzaki et al. (2005) show that L-PCFGs perform on par with models hand-annotated with linguistically motivated features. Cohen et al. (2013) report that on the Penn Treebank dataset, L-PCFGs trained with either EM or a spectral algorithm provide a 20% increase in F1 over PCFGs without latent states. Gebhardt (2018) shows that the benefits of latent variables are not limited to PCFGs by successfully enriching both Linear Context-Free Rewriting Systems and Hybrid Grammars with latent variables, and demonstrates their applicability on discontinuous constituent parsing.

The advantage of expressing parsing algorithms as WLPs is that the structure of the dynamic programming algorithm is disentangled from the particular calculations for obtaining the values of chart items. In Figure 5.1 we present a procedural formulation of probabilistic CKY, a standard parsing algorithm that calculates the total probability mass a PCFG assigns to a string. As a running example, consider the grammar with the rules, and the associated probabilities:

$$\begin{aligned} p(S \to A\ A) &= p_1 \\ p(A \to A\ A) &= p_2 \\ p(A \to a) &= p_3, \end{aligned}$$

and a string *aaa*. CKY starts by assigning the chart items $chart[1,A,2]$, $chart[2,A,3]$, $chart[3,A,4]$ all the probability p_3, since rule 3 is the only one that can generate the terminal a. Then it calculates $chart[1,A,3]$ and $chart[2,A,4]$ as $p_2 \times p_3 \times p_3$ from these values using the update rule on lines 11 and 12 in Figure 5.1. Finally, the goal item $chart[1,S,4]$ that corresponds to the probability of the start symbol S spanning the entire string can be calculated as:

$$\begin{aligned} &p_1 \times chart[1,A,2] \times chart[2,A,4] \\ &+ p_1 \times chart[1,A,3] \times chart[3,A,4]. \end{aligned}$$

Note that the two summands in this expression correspond to the two possible derivations of *aaa* using the grammar rules. The first summand calculates the probability given by the right branching derivation tree where the rule $A \to a$ is applied before

Input: A CFG $\langle N, \Sigma, \mathcal{R}, S \rangle$ in Chomsky normal form, a function $\omega : \mathcal{R} \mapsto [0,1]$ where $\omega(A \to x) = p(A \to x|A)$ and $\omega(A \to BC) = p(A \to BC|A)$ for all $A, B, C \in N$ and $x \in \Sigma$, and a string $\alpha_1 \alpha_2 \ldots \alpha_n$ where $\alpha_i \in \Sigma$

Output: $p(\alpha_1 \alpha_2 \ldots \alpha_n)$

```
chart ← zeros(n, |N|, n+1)
for i ← 1 to n do
    for (A → α_i) ∈ R do
        chart[i, A, i+1] ← ω(A → α_i)
    end
end
for l ← 2 to n do
    for s ← 1 to n-l+1 do
        for t ← 1 to l+1 do
            for (A → BC) ∈ R do
                chart[s, A, s+l] ← chart[s, A, s+l] +
                        ω(A → BC) × chart[s, B, s+t] × chart[s+t, C, s+l]
            end
        end
    end
end
return chart[1, S, n+1]
```

Figure 5.1: PROBABILISTIC CKY (INSIDE) calculates the probability a PCFG assigns to a string

Input: A CFG $\langle N, \Sigma, \mathcal{R}, S \rangle$ in Chomsky normal form,
a set of latent variables $H = \{h_1, h_2, \ldots, h_m\}$,
a mapping ω where $\omega(A \to x) \in [0,1]^{d_A}$ and $\omega(A \to x)_i = p(A(h_i) \to x | A(h_i))$,
$\omega(A \to BC) \in [0,1]^{d_A \times d_B \times d_C}$ and
$\omega(A \to BC)_{i,j,k} = p(A(h_i) \to B(h_j)C(h_k) | A(h_i))$,
$\omega(S) \in [0,1]^{d_S}$ and $\omega(S)_i = p(S(h_i) \text{ is root})$,
and a string $\alpha_1 \alpha_2 \ldots \alpha_n$ for $\alpha_i \in \Sigma$
Output: $p(\alpha_1 \alpha_2 \ldots \alpha_n | h_i)$ for $i \in [m]$

for $A \in N$ **do**
  $chart(A) \leftarrow zeros(n, n+1, d_A)$
end
for $i \leftarrow 1$ ***to*** n **do**
  for $(A \to \alpha_i) \in \mathcal{R}$ **do**
    $chart(A)[i, i+1, :] \leftarrow \omega(A \to \alpha_i)$
  end
end
for $l \leftarrow 2$ ***to*** n **do**
  for $s \leftarrow 1$ ***to*** *n-l+1* **do**
    for $t \leftarrow 1$ ***to*** *l+1* **do**
      for $(A \to BC) \in \mathcal{R}$ **do**
        $chart(A)[s, s+l, :] \leftarrow chart(A)[s, s+l, :] \quad +$
        $\omega(A \to BC)\Big(\mathbf{I}, chart(B)[s, s+t, :], chart(C)[s+t, s+l, :]\Big)$
      end
    end
  end
end
return $\omega(S)^\top chart(S)[1, n+1, :]$

Figure 5.2: INSIDE ALGORITHM FOR L-PCFGS to calculate the probability of a string for each hidden variable of the start symbol S.

$A \to AA$, while the second one corresponds to the probability given the the left branching derivation tree where the rule $A \to AA$ is applied first.

The inside algorithm for L-PCFGs is presented in Figure 5.2. To see how it differs from CKY, consider the same string and the grammar rules as the previous example, but with the rules associated with distributions over latent states rather than probabilities:

$$w(S \to A\ A) = w_1 \in [0,1]^{d_S, d_A, d_A}$$
$$w(A \to A\ A) = w_2 \in [0,1]^{d_A, d_A, d_A}$$
$$w(A \to a) = w_3 \in [0,1]^{d_A}.$$

The algorithm initializes the chart items in the same order, assigning $\mathit{chart}(A)[1,2,:]$, $\mathit{chart}(A)[2,3,:]$ and $\mathit{chart}(A)[3,4,:]$ the vector w_3. It then traverses the chart items as CKY does, the only difference being the calculation of the values. For $\mathit{chart}(A)[1,3,:]$ and $\mathit{chart}(A)[2,4,:]$ this translates to the tensor contraction $w_2(\mathbf{I}, w_3, w_3)$, which is a vector in $[0,1]^{d_A}$. For $\mathit{chart}(S)[1,4,:]$ the value is given by:

$$w_1(\mathbf{I}, \mathit{chart}(A)[1,2,:], \mathit{chart}(A)[2,4,:])$$
$$+\, w_1(\mathbf{I}, \mathit{chart}(A)[1,3,:], \mathit{chart}(A)[3,4,:]).$$

Considering the almost identical structure of these two algorithms, one would expect that it is also possible to unify them by abstracting over the calculation of the values. However, semiring WLPs do not extend to latent variable models because latent variables are often represented as vectors, matrices and higher-order tensors, and these taken together no longer form a semiring. This is because in the semiring framework, values for deduction items and rules must all come from the same set, and the semiring operations must be defined over all pairs of values from this set. This does not allow for letting different grammar nonterminals be represented by vectors of different sizes. More importantly, it does not allow for the value of a rule to be a tensor whose dimensionality depends on the arity of the rule, as is generally the case in latent variable frameworks.

In this chapter, we start with a broad interpretation of latent variables as tensors over an arbitrary semiring. While a set of tensors over semirings is no longer a semiring, we prove that if the tensors have certain matching dimensions for the grammar rules they are assigned to, then they fulfill all the desirable properties relevant for the semiring parsing framework. This paves the way to use WLPs with latent variables, naturally

improving the expressivity of the statistical model represented by the underlying WLP. Introducing a semiring framework like ours makes it easier to seamlessly incorporate latent variables into any execution model for dynamic programming algorithms (or software such as Dyna (Eisner et al., 2005) and other Prolog-like/WLP-like solvers).

We focus on CFG parsing, however the same latent variable techniques can be applied to any weighted deduction system, including systems for parsing TAG, CCG and LCFRS, and systems for Machine Translation (Lopez, 2009). The methods we present for inside and outside computations can be used to learn latent refinements of a specified grammar for any of these tasks with EM (Dempster et al., 1977; Matsuzaki et al., 2005), or used as a backbone to create spectral learning algorithms (Hsu et al., 2012; Bailly et al., 2009; Cohen et al., 2014).

5.1.2 Path Representations in Knowledge Graphs

There is a wealth of work exploring the effectiveness of using path information for knowledge graph completion (see Section 2.5.3 for a review). Many of these models estimate probabilities associated with random walks on the graph to use as features. Others define embeddings of paths from the embeddings of the component relations and entities, and use these as regularizers (Lin et al., 2015b), or to score additional *(entity, path, entity)* triples during training (Guu et al., 2015). Compared to models that treat triples as isolated training instances, this approach provides richer structure for the model. However, it also comes with the added cost of calculating over possible paths, which are potentially exponential in the size of the edges.

The model suggested by Toutanova et al. (2016) is of particular interest because it uses a dynamic programming algorithm to calculate the aggregate representation of all paths up to a certain length between the entities in a KG, and uses it to extend a bilinear scoring function. The representation $\Phi(\pi)$ of a path $\pi : r_1 r_2 \dots r_n$ consisting of the relations r_i is defined as the matrix multiplication of the embeddings of its components, scaled by a scalar weight w_e corresponding to the intermediate entity at each step.

$$\Phi(\pi) = \mathbf{R_1}\tanh(w_{e_1})\mathbf{R_2}\tanh(w_{e_2})\dots\mathbf{R_n}\tanh(w_{e_n}). \tag{5.1}$$

Based on this path representation, the desired value associated with all possible paths π between entities h and t is defined as follows:

$$F(h,t) = \sum_{\pi} w_{|\pi|} p(t|h,\pi)\Phi(\pi), \tag{5.2}$$

where $w_{|\pi|}$ is a parameter that weights the contribution of π based on its length. The scoring function for a triple (h,r,t) is then defined as:

$$s(h,r,t) = \mathbf{h}^\top \mathbf{R}\mathbf{t} + vec(F(h,t))^\top vec(\mathbf{R}) \tag{5.3}$$

The algorithm for calculating $F_l(h,t)$, the aggregate representation for all paths of length l between entities h and t, is presented in Figure 5.3. Note that the structure of

---

Input: A KG $(\mathcal{E},\mathcal{R},\mathcal{F})$, embeddings $\mathbf{R} \in \mathbb{R}^{d\times d}$ for $r \in \mathcal{R}$, scalar weights w_r for $r \in \mathcal{R}$, set of neighbors N_e for $e \in \mathcal{E}$ and maximum path length L.

Output: $F(h,t)$ for all $h,t \in \mathcal{E}$

for $h,t \in \mathcal{E}$ **do**
  $F_1(h,t) \leftarrow \mathbf{0}$
end
for $(h,r,t) \in \mathcal{F}$ **do**
  $F_1(h,t) \leftarrow F_1(h,t) + \tanh(w_r)p(t|h,r)\mathbf{R}$
end
for $l = 2$ ***to*** L **do**
  for $h,r \in \mathcal{E}$ **do**
    $F_l(h,t) \leftarrow \mathbf{0}$ **for** $e \in N_h$ **do**
      $F_l(h,t) \leftarrow F_l(h,t) + F_1(h,e)F_{l-1}(e,t)$
    end
  end
end
return $F(h,t)$ *for all* $h,t \in \mathcal{E}$

Figure 5.3: PATH-SUM calculates the aggregate path representation $F_l(h,t)$ for all length-l paths between h and t

---

the algorithm corresponds to a forward calculation for a hidden Markov model (HMM).

Unlike in the previous example for L-PCFGs, here the chart values are all matrices $\mathbf{R} \in \mathbb{R}^{d\times d}$ with the same dimensions. This means that they admit characterization both as semirings, and as tensors over semirings. However, casting the algorithm as a WLP with tensor weights enables interesting extensions to the estimated values, which aren't possible with semiring WLPs. For example, a tensor WLP would allow chart items to

be matrices in $\mathbb{R}^{d_1 \times d_2}$ where $d_1 \neq d_2$, or for the combination of $F_1(s,e)$ and $F_l(e,t)$ to be modulated by additional multilinear parameters.

Further extensions that integrate logical rules in bilinear embeddings are also possible with tensor WLPs. Recent work by Niu et al. (2020) mines rules of the form $R_1 \rightarrow R_2$ and $R_1\, R_2 \rightarrow R_3$ with the rule mining tool AIME+ (Galárraga et al., 2015), and defines a scoring function that only takes into account the paths that can be generated by the application of the rules. While their model is based on TransE, it can be adapted to bilinear models by observing that these rules define a context free grammar. It is then possible to replace the tensor WLP for the *forward algorithm* corresponding to Algorithm 5.3 with a tensor WLP for an *inside algorithm* to calculate the aggregate value of all derivable paths between entities h and t given the rules. These demonstrate some ways in which the theory we develop here can be used in KG embedding applications. However for the rest of the chapter, we focus on theoretical properties of tensor WLPs, and leave exploring their applications for future work.

5.2 Main Results Takeaway

We present a strict generalization of semiring weighted logic programming, with a particular focus on parser descriptions in WLP for context-free grammars. Throughout, we utilize the correspondence between axioms and grammar rules, deductive proofs and grammar derivations, and derived theorems and strings.

We assume that axioms/grammar rules come equipped with weights in the form of tensors over semiring values. The main issue with going from semirings to tensors over semiring values is that these weights need to be *well defined* in that any valid derivation should correspond to a sequence of well defined semiring operations. For CFGs, we give a straightforward condition that ensures this is the case. This essentially boils down to making sure that each non-terminal corresponds to a fixed vector space dimension. For example, if A corresponds to a space $\mathbb{S}^{d_1}$, B to $\mathbb{S}^{d_2}$ and C to $\mathbb{S}^{d_3}$, then a rule $A \rightarrow B\, C$ would have a tensor weight in $\mathbb{S}^{d_2 \times d_3 \times d_1}$.

As long as the weights are well defined, the standard definitions for the value of a grammar derivation and a string according to a semiring weighted grammar extend to the case of tensors of semirings. WLPs provide the means to declaratively specify dynamic programming algorithms to obtain these values of interest. In line with Sikkel (1998) and Goodman (1999) we present precise conditions for when a tensor WLP

describes a correct parser.

The value of the WLP formulation of parsing algorithms is that it provides a unified fashion in which dynamic programming algorithms can be extracted from the program description. This relies on the ability of a WLP to decompose the value of a proof to a combination of the values of the sub-proofs. Specifically, given a derivation tree, a WLP description automatically provides algorithms for calculating the inside and outside values. We provide analogous algorithms for calculating the inside and outside values for tensor WLPs. Our outside formulation addresses the non-commutative nature of tensors themselves, and could be extended to cases where the underlying semiring is non-commutative using the techniques presented by Goodman (1998).

5.3 Related Work

Parsing as deduction (Pereira and Warren, 1983) is an established framework that allows a number of parsing algorithms to be written as declarative rules and deductive systems (Shieber et al., 1995), and their correctness to be rigorously stated (Sikkel, 1998). Goodman (1999) extends this framework to arbitrary semirings and shows that various different values of interest can be computed using the same algorithm, by changing the semiring. This has led to the development of Dyna, a toolkit for declaratively specifying weighted logic programs, allowing concise implementation of a number of NLP algorithms (Eisner et al., 2005).

The semiring characterization of possible values to assign to WLPs gave rise to the formulation of a number of novel semirings. One novel semiring of interest for purposes of learning parameters is the *generalized entropy semiring* (Cohen et al., 2008) which can be used to calculate the KL-divergence between the distribution of derivations induced by two weighted logic programs. Other two semirings of interest are *expectation* and *variance* semirings introduced by Eisner (2002) and Li and Eisner (2009). These utilize the algebraic structure to efficiently track quantities needed by the expectation-maximization algorithm for parameter estimation. Their framework allows working with parameters in the form of vectors in $\mathbb{R}^n$ for a fixed n, coupled with a scalar in $\mathbb{R}_{\geq 0}$. The semiring value of a path is roughly calculated by the multiplication of the scalars and (appropriately weighted) *addition* of the vectors. This is in contrast with our framework where weights could be tensors of arbitrary order rather than only vectors, and the values of paths are calculated via tensor multiplication.

Finally, Gimpel and Smith (2009) extended the semiring framework to a more general algebraic structure with the purpose of incorporating non-local features. Their extension comes at the cost that the new algebraic structure does not obey all the semiring axioms. Our framework differs from theirs in that under reasonable conditions, tensors of semirings do behave fully like regular semirings.

5.4 Background and Notation

Our formalism could be used to enrich any WLP that implements a dynamic programming algorithm, but for simplicity, we follow Goodman (1999) and focus our presentation on parsers with a context-free backbone.[1]

5.4.1 Context-free Grammars

Formally, a Context-Free Grammar (CFG) is a 4-tuple $\langle N, \Sigma, \mathcal{R}, S \rangle$. The set N consists of non-terminal symbols which will be denoted by uppercase letters A, B etc., and S is a non-terminal that is the special start symbol. The set Σ consists of terminal symbols, which will be denoted by lowercase letters a, b etc. $\mathcal{R}$ is the set of rules of the form $A \to \alpha$ consisting of one non-terminal on the left hand side (lhs), and a string $\alpha \in (N \cup \Sigma)^*$ on the right hand side (rhs). We will use $\alpha \Rightarrow \beta$ if β could be derived from α with the application of one grammar rule. We will say that a sentence $\sigma \in \Sigma^+$ could be derived from the non-terminal A if σ could be generated by starting with A and repeatedly applying rules in $\mathcal{R}$ until the right hand side contains only terminals, and denote this as $A \stackrel{*}{\Rightarrow} \sigma$. We will denote the language that a grammar G defines by $\mathcal{L}(G) = \{\sigma | S \stackrel{*}{\Rightarrow} \sigma\}$.

CFG derivations can naturally be represented as trees. We will use the notation $\langle r : T_1 \dots T_k \rangle$ to represent a tree that has the node r as its root and $T_1, \dots, T_k$ as its direct subtrees. We will use $\mathcal{D}_G$ to denote the set of all derivation trees that can be constructed with the grammar G, and $\mathcal{D}_G(\sigma)$ to denote all valid derivation trees that generate the sentence σ in G.

[1] Note that given a grammar G in a formalism F and a string α, it is possible to construct a CFG grammar $c(G, w)$ from G and α (Nederhof, 2003). This construction is possible even for range concatenation grammars (Boullier, 2000) which span all languages that could be parsed in poly-time.

5.4.2 Semirings

A semiring is an algebraic structure similar to a ring, except that it does not require additive inverses.

Definition 2. *A **semiring** is a set $\mathbb{S}$ together with two operations $+$ and $\times$, where $+$ is commutative, associative and has an identity element 0. The operation of $\times$ is associative, has an identity element 1 and distributes over $+$.*

The set of non-negative integers together with the usual $\times, +, 0, 1$ is a semiring, and so are probability values in $[0,1]$. Booleans $\{$TRUE, FALSE$\}$ also form a semiring with $\times := \vee$, $+ := \wedge$, $0 :=$ FALSE and $1 :=$ TRUE.

There are a few less common semirings that provide useful values in parsing. The *Viterbi* semiring calculates the probability of the best derivation. It has values in $[0,1]$, $+ :=$ max and $\times, 0, 1$ as standard. The *Derivation forest*, *Viterbi derivation* and *Viterbi n-best* semirings calculate the set of all derivations, the best derivation and the *n*-best derivations respectively. Unlike the previous examples, the $\times$ operation of these semirings is not commutative. In general, if the $\times$ operation in a semiring is commutative, we refer to it as a *commutative semiring*, and otherwise it is referred to as *non-commutative*. For precise definitions and detailed descriptions of these semirings see Goodman (1999).

5.4.3 Weighted Logic Programming

A logic program consists of *axioms* and *inference rules* that could be applied iteratively to prove theorems. Inference rules are expressed in the form $\frac{A_1 \ldots A_k}{B}$ where $A_1 \ldots A_k$ are antecedents from which B can be concluded. Axioms are inference rules with no antecedents.

One way to express dynamic programming algorithms such as CKY is as logic programs. This approach takes the point of view of *parsing as deduction*: terms consist of grammar rules and *items* in the form of $[i, A, j]$ that correspond to the intermediate entries in the chart. Grammar rules are taken to be axioms, and the description of the parser is given as a set of inference rules. These can have both grammar rules and items as antecedents and an item as the conclusion. A logic program in this form includes a special designated goal item that stands for a successful parse. In CKY this would correspond to the chart item $[0, S, n+1]$ where S is the start symbol from the CFG and

n is the length of the string to be parsed.

Continuing with the example of CKY, consider the procedural description for how to obtain a chart item from smaller chart items if we have the rule $A \to B\,C$ in the grammar:

$$chart[i,A,j] \leftarrow chart[i,A,j] \vee (chart[i,B,k] \wedge chart[k,C,j]).$$

The corresponding inference rule in a logic program would be:

$$\frac{A \to B\,C \qquad [i,B,k] \qquad [k,C,j]}{[i,A,j]}.$$

Note that in the inference rule above, $A \to B\,C$ is a rule template with free variables A,B,C. In general, the terms in inference rules can contain free variables, however for a logic program to describe a valid dynamic programming algorithm, every free variable in the conclusion of an inference rule must appear in its antecedents as well. The apparent discrepancy of the term $chart[i,A,j]$ appearing twice in the procedural description but just once in the inference rule is that in a logic program, once we derive $[i,A,j]$, it is retained in the set of items we know to be true, whereas in the procedural description this must be stated explicitly within the calculation.

A *weighted* logic program is a logic program where terms are assigned values from a semiring. When paired with semiring operations, inference rules provide the description of how to compute the value of the conclusion given the values of the antecedents. The result of an application of a particular inference rule is the semiring multiplication of all the antecedents. The value of a term B is then calculated as the semiring sum of values obtained from inference rules that have B as their conclusion. As an example, consider the *path-sum* algorithm given in Figure 5.3. The line that corresponds to the calculation of the representation of a path of length l from the length of paths $l-1$ is:

$$F_l(h,t) \leftarrow F_l(h,t) + F_1(h,e)F_{l-1}(e,t)$$

The corresponding inference rule in the WLP would be:

$$\frac{F_1(h,e) \qquad F_{l-1}(e,t)}{F_l(h,t)}.$$

Notice that even though the procedural description for CKY uses Boolean operations and the procedural description for *path-sum* uses addition and multiplication, this distinction is no longer apparent in their respective logic program descriptions. In the

following section, we present how this abstraction can be formalized using the language of semirings.

5.4.4 Semiring Parsing

In the context of parsing, Goodman (1999) presents a framework where a grammar G comes equipped with a function w that maps each rule in G to a semiring value. Then, a grammar derivation string E consisting of the successive applications of rules $e_1, \ldots, e_n$ is defined to have the value $V_G(E) = \prod_{i=1}^{n} w(e_i)$, and the value of a sentence $\sigma \in \mathcal{L}(G)$ is defined as the sum $V_G = \sum_{j=1}^{k} V_G(E_j)$ where $E_1, E_2, \ldots, E_k$ are the derivations of σ in G.

A parser specification is given in the form of a weighted logic program, referred to as an *item-based description*. The description includes the form of the intermediate items, the designated goal item and the inference rules, and takes as input a PCFG and a string to be parsed. From these, the value of a derivation D is calculated recursively as follows:

$$V(D) = \begin{cases} w(D) & \text{if } D \text{ is a rule} \\ \prod_{i=1}^{k} V(D_i) & \text{if } D = \langle b : D_1, \ldots, D_k \rangle, \end{cases} \tag{5.4}$$

where $\prod$ is the semiring product.

Let $inner(x)$ represent the set of all derivation trees headed by the item x. Then the value of x is:

$$V(x) = \sum_{D \in inner(x)} V(D), \tag{5.5}$$

where $\sum$ is the semiring addition. The value of a sentence is then equal to $inner(goal)$, where *goal* is the designated goal item in the logic program.

One of the main contributions of Goodman (1999) is to provide conditions that characterize when and item-based description defines a correct parser. A semiring parser is said to be correct if the value of an input string according to the grammar equals the value of the string according to the parser. Given the definitions for these values, the conditions of correctness for an item based description is as follows:

Theorem 5.4.1. *(Goodman 1999, Theorem 1; informal) An item-based description I is correct if for every grammar G there exists a one-to-one correspondence between the set of grammar derivations and the set of item derivations, and these derivations have the the same value regardless of the weight function used.*

One caveat with calculating based on item-based derivations is that there is an ordering of items: we cannot compute the value of an item unless the values of all its children are computed already. For this, Goodman (1999) assumes that each item is assigned to a *bucket* so that if an item b depends on a, then $bucket(a) \leq bucket(b)$. If a bucket depends on itself, then it is considered a special *looping* bucket.[2] For all the formulas we present in our exposition in this chapter we assume that the items belong to non-looping buckets. The formulas for looping buckets are provided in Appendix B.

For an item x, calculating its value might require summing over exponentially many derivation trees. To address this, it is possible to provide a general formula that efficiently computes the inner value for an item (Goodman 1999, Theorem 2):

$$V(x) = \sum_{a_1,\ldots,a_k \text{ s.t. } \frac{a_1,\ldots,a_k}{x}} \prod_{i=1}^{k} V(a_i). \tag{5.6}$$

The other important value associated with an item x is its *outside* value $Z(x)$, which is the sum of values of derivation trees, modified so that x is removed with all its subtrees. This value is complementary to the inside values $V(x)$ (Goodman 1999, Theorem 4):

$$V(x) \times Z(x) = \sum_{D \text{ a derivation}} V(D)C(D,x). \tag{5.7}$$

where $C(D,x)$ is the count of the occurrences of item x in derivation D.

$Z(x)$ can likewise be calculated using a recursive formula if the values are from a commutative semiring (Goodman 1999, Theorem 5):

$$Z(x) = \sum_{\substack{j,a_1,\ldots,a_k,b \text{ s.t.} \\ \frac{a_1\ldots a_k}{b} \text{ and } x=a_j}} Z(b) \times \prod_{i=1}^{j-1} V(a_i) \times \prod_{i=j+1}^{k} V(a_i). \tag{5.8}$$

5.4.5 Tensor Notation

We use the term *tensor* to refer to an n-dimensional array of semiring values. We use $\mathbb{S}$ to denote a semiring and $\mathbf{A}, \mathbf{B}$ etc. to denote tensors. The element $\mathbf{A} \in \mathbb{S}^{a_1 \times a_2 \times \ldots \times a_n}$ will denote that $\mathbf{A}$ is an nth order tensor of values drawn from $\mathbb{S}$, with the ith order having dimension a_i. As usual, the semiring entry with index $k_1, \ldots, k_n$ will be denoted with subscripts $A_{k_1,\ldots,k_n}$.

[2] Looping buckets can happen in practice even when there is no trivial derivation rules e.g. $A \rightarrow A$ in the input grammar. For an explicit example with a small CFG and the Earley parser see Goodman (1998, p. 28)

5.5 Tensor Weighted Logic Programs

In this section we present a framework that generalizes that of Goodman (1999), and is able to capture tensors over semirings as weights. Note that this includes scalars as a special case.

5.5.1 Semiring Operations

The main reason why tensors over semirings are not semirings is that with tensor weights, $\oplus$ and $\otimes$ become partially defined – not all elements can naturally be added or multiplied to any other element anymore. We refer to these structures as *partial semirings*. With some reasonable constraints, we show that $\oplus$ and $\otimes$ obey the semiring axioms in cases that are relevant for the semiring parsing framework.

Let $\mathbb{S}$ be the chosen underlying semiring, $+, \times$ to be the semiring operations and $\mathbf{0}, \mathbf{1}$ be the additive and multiplicative identity of the semiring respectively. The set of possible weights are defined as $\{\mathbb{S}^{d_1 \times \ldots \times d_n}\}$ for $n \in \mathbb{N}$, and $d_i \in \mathbb{N}$ for all $i \leq n$. $\oplus$ is a partial addition that is defined on two tensors $\mathbf{A}, \mathbf{B} \in \mathbb{S}^{d_1 \times \ldots \times d_n}$ as long as the dimensions of each of their orders match. Then, the addition is defined component-wise:

$$(\mathbf{A} \oplus \mathbf{B})_{i_1,\ldots,i_n} := \mathbf{A}_{i_1,\ldots,i_n} + \mathbf{B}_{i_1,\ldots,i_n}. \tag{5.9}$$

The additive identity is now a class of tensors, one for each unique list of tensor dimensions. The additive identity for any $\mathbf{A} \in \mathbb{S}^{d_1 \times \ldots \times d_n}$ is the tensor $\mathbf{Z} \in \mathbb{S}^{d_1 \times \ldots \times d_n}$ with $\mathbf{0}$ in every entry.

Multiplication is defined as the contraction of an index between two tensors. Specifically, we consider the family $\otimes_{[k;l]}$ which contracts the kth order of the first tensor with the lth order of the second tensor. This is only defined if the two indices to be contracted have the same dimension, as follows:

$$\left(\mathbf{A} \otimes_{[k;l]} \mathbf{B}\right)_{\substack{i_1,\ldots,i_{k-1},j_1,\ldots,j_{l-1},\\ j_{l+1},\ldots,j_m,i_{k+1},\ldots,i_n}} := \sum_{i_k,j_l} \delta(i_k, j_l) \mathbf{A}_{i_1,\ldots,i_n} \times \mathbf{B}_{j_1,\ldots,j_m}, \tag{5.10}$$

where δ is the identity function that is equal to $\mathbf{1}$ if $i_k = j_l$ and $\mathbf{0}$ otherwise. Note that the indices of $\mathbf{B}$ which are not contracted go in between the indices of $\mathbf{A}$, replacing the contracted index of $\mathbf{A}$. We will use $\otimes_j$ as a shorthand of $\otimes_{[j;1]}$, and in cases where $j = l = 1$, we will omit the subscript on $\otimes$ altogether.

More generally, we will allow multiplication operations that contract multiple consecutive dimensions. $\mathbf{A} \otimes^r_{[k;l]} \mathbf{B}$ will denote contracting order k of $\mathbf{A}$ with order l of $\mathbf{B}$, order

$k+1$ of $\mathbf{A}$ with order $l+1$ of $\mathbf{B}$ and so forth until order $k+r-1$ of $\mathbf{A}$ and $l+r-1$ of $\mathbf{B}$. Formally:

$$\left(\mathbf{A} \otimes^{r}_{[k;l]} \mathbf{B}\right)_{\substack{i_1,\ldots,i_{k-1},j_1,\ldots,j_{l-1},\\ j_{l+r},\ldots,j_m,i_{k+r},\ldots,i_n}} := \sum_{\substack{i_k,\ldots,i_{k+r-1}\\ j_l,\ldots,j_{l+r-1}}} \left(\prod_{p=0}^{r-1} \delta\left(i_{k+p}, j_{l+p}\right)\right) \mathbf{A}_{i_1,\ldots,i_n} \mathbf{B}_{j_1,\ldots,j_m}. \tag{5.11}$$

We will use the notation $\mathbf{A} \otimes^{*} \mathbf{B}$ as a shorthand for $\mathbf{A} \otimes^{rank(\mathbf{A})} \mathbf{B}$ if $rank(\mathbf{A}) < rank(\mathbf{B})$ and $\mathbf{A} \otimes^{rank(\mathbf{B})} \mathbf{B}$ otherwise.

To make the presentation clearer, we will also use the notation $X \otimes [A_1, A_2, \ldots, A_k]$ to denote contraction of A_1 with the first order of X, A_2 with the second and so forth. In other words $X \otimes [A_1, \ldots, A_n]$ is equivalent to $X \otimes_n A_n \otimes_{n-1} A_{n-1} \ldots \otimes_1 A_1$.

The multiplicative identity for $\mathbf{A} \in \mathbb{S}^{d_1 \times \ldots \times d_n}$ and $\otimes_k$ is the identity matrix $\mathbf{I} \in \mathbb{S}^{d_k \times d_k}$ where the diagonal entries are the multiplicative identity from the underlying semiring, and the non-diagonals are the additive identity. For $\mathbf{A} \in \mathbb{S}^{d_1 \times \ldots \times d_n}$ and $\otimes^{r}_{k}$ the multiplicative identity is a rank-$2r$ tensor $\mathbf{I} \in \mathbb{S}^{d_k \times \ldots \times d_{k+r-1} \times d_k \times \ldots \times d_{k+r-1}}$ and is defined as follows:

$$\mathbf{I}_{d_1,\ldots,d_r} = \prod_{i=0}^{\frac{n}{2}} \delta\left(d_i, d_{\frac{r}{2}+i}\right). \tag{5.12}$$

Lastly, as the higher order analogue of the transpose operator, we will define a permutation operator $\mathbf{A}^{\pi}$ where $\pi = [\pi_1, \pi_2, \ldots, \pi_r]$ is a permutation of $[1 \ldots r]$ and r is the order of $\mathbf{A}$. The π_ith order of $\mathbf{A}^{\pi}$ is equal to the ith order of $\mathbf{A}$.

The key property of semirings for purposes of efficient calculation of item values is the distributive property. This property also holds for tensors over semirings.

Lemma 5.5.1. *For any k, l, $\otimes_{[k;l]}$ distributes over $\oplus$.*

A proof can be found in Appendix A.

5.5.2 Grammar Derivations

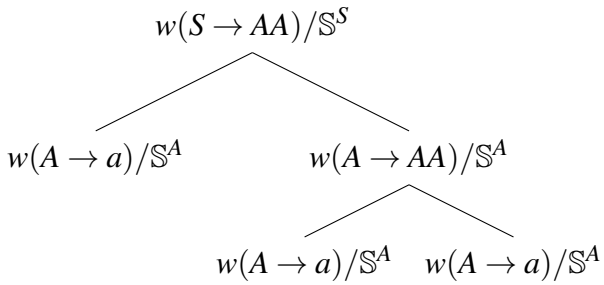
For a grammar G with a function w that provides a mapping from rules to tensor weights, we will define a value of a derivation via the derivation tree:

Definition 3. *Given a grammar G and a weight function w, the value of a derivation*

Tensor dimensions of grammar rules:

$$w(S \to AA) \in \mathbb{S}^{A \times A \times S} \quad w(A \to AA) \in \mathbb{S}^{A \times A \times A} \quad w(A \to a) \in \mathbb{S}^{A}$$

Grammar derivation tree:



The value of the tree is given by the equation:

$$w(S \to AA) \otimes (w(A \to a), (w(A \to AA) \otimes (w(A \to a), w(A \to a))))$$

Grammar derivation string:

$$S \underset{\mathbb{S}^{A \times A \times S}}{\overset{S \to AA}{\Longrightarrow}} AA \underset{\mathbb{S}^{A \times S}}{\overset{A \to a}{\Longrightarrow}} aA \underset{\mathbb{S}^{A \times A \times S}}{\overset{A \to AA}{\Longrightarrow}} aAA \underset{\mathbb{S}^{A \times S}}{\overset{A \to a}{\Longrightarrow}} aaA \underset{\mathbb{S}^{S}}{\overset{A \to a}{\Longrightarrow}} aaa$$

The value of the string is given by the equation:

$$w(S \to AA) \otimes w(A \to a) \otimes (A \to AA) \otimes (A \to a) \otimes (A \to a)$$

Figure 5.4: Example derivation for the string “aaa”. We illustrate the initial dimensions of the tensor values for the rules and also show the intermediate tensor dimensions during the calculation of the value of the grammar tree and the grammar string.

tree T is:

$$V_G^w(T) = \begin{cases} w(r) & \text{if } T = \langle r \rangle \\ w(r) \otimes [V_G^w(T_1), \ldots, V_G^w(T_k)] & \text{if } T = \langle r : T_1, \ldots, T_k \rangle \end{cases}.$$

Note that there is no guarantee that this equation is defined for any arbitrary w. We will call a weight function *w well defined* for a grammar G if for all valid derivation trees T in G, $V_G^w(T)$ is defined. For CFGs there is a straightforward method to ensure that w is well defined:

Lemma 5.5.2. *A weight function w for a given CFG is well defined if there exist consistent dimensions d_i for each nonterminal A_i such that for all grammar rules $R : A_n \rightarrow \alpha_1 A_1 \alpha_2 \ldots \alpha_{n-2} A_{n-1} \alpha_n$, $w(R) \in \mathbb{S}^{d_1 \times \ldots \times d_n}$.*

Proof is given together with Lemma 5.5.3.

Given a grammar derivation tree T, let the list of derivation rules $E : R_1, R_2, \ldots, R_n$ appearing in T ordered via depth-first, left-to-right manner be referred to as a **grammar derivation string**.

Definition 4. *Given a CFG with tensor weights w, the **value of a grammar derivation string** is defined as:*

$$V_G^w(E) = \bigotimes_i w(R_i),$$

where the application of $\otimes$ proceeds from left to right as is standard.

For semirings, since the bracketing does not affect the final value of an expression, it is straightforward to show that the value of a grammar derivation tree corresponds to that of a grammar derivation string. With tensors over semirings this might fail with an arbitrary formalism F, and in general we require the value of a derivation to be calculated with the bracketing induced by the derivation tree. However, for the special case of CFGs, the value of the grammar derivation tree and the value of its corresponding grammar derivation string are always equal. This means that for the computation of the value of the derivation, it is possible to replace the bracketing induced by the derivation tree by left-to-right bracketing without affecting the final value. Figure 5.4 demonstrates the calculation of the value of the tree and the string for the same derivation together with how the tensor dimensions of the intermediate results evolve with each step of the calculation.

Lemma 5.5.3. *Given a CFG G and a weight function w that fulfills the condition in Lemma 5.5.2, then w is well defined and $V_G^w(T) = V_G^w(E)$ for any grammar derivation tree T and corresponding grammar derivation string E.*

Proof. We will proceed by induction on the derivation tree. If T consists of only one rule r, then $V_G^w(T) = V_G^w(E)$. Furthermore, r does not have any non-terminals on its rhs, so $V_G^w(T) \in \mathbb{S}^{d_0}$ with $\mathbb{S}^{d_0}$ corresponding to the lhs non-terminal in r.

Otherwise, T has a labeled node r and the subtrees $T_1, \ldots, T_k$. Notice that if $A_0 \in \mathbb{S}^{d_1 \times \ldots \times d_n \times d_0}$, $A_1 \in \mathbb{S}^{d_2}$,..., $A_n \in \mathbb{S}^{d_n}$, then $A_0 \otimes [A_1, \ldots, A_n] = A_0 \otimes A_1 \otimes \ldots \otimes A_n$ due to all arguments within $[\ldots]$ being rank-1.

Because w fulfills the condition in Lemma 5.5.2, $w(r) \in \mathbb{S}^{d_1 \times \ldots \times d_k \times d_0}$ for some d_i where $\mathbb{S}^{d_0}$ is the space corresponding to the non-terminal on the lhs of r, and $\mathbb{S}^{d_i}$ is the space corresponding to the ith non-terminal appearing in the rhs of r for $i = 1, \ldots, k$. Then to complete the proof, it suffices to show that $V_G^w(T_i) \in \mathbb{S}^{d_i}$ for all subtrees T_i. This already holds for the base case. For each $T_i : \langle r_i : T_1', .., T_k' \rangle$, if $w(r_i) \in \mathbb{S}^{d_1^i \times \ldots \times d_k^i \times d_0^i}$ then by induction $V_G^w(T_i) \in \mathbb{S}^{d_0^i}$, where $\mathbb{S}^{d_0^i}$ is the space corresponding to the non-terminal in the lhs of R_i. For the derivation to be valid, this non-terminal needs to match the ith non-terminal in the rhs of R, hence $\mathbb{S}^{d_0^i} = \mathbb{S}^{d_i}$ □

5.5.3 Path Values

Similar to grammar derivation values, we can calculate the tensor values for paths in a knowledge graph where the edges are associated with the tensor weight of the relation:

Definition 5. *Let $P = \langle r_0, r_1, \ldots, r_n \rangle$ be a path that connects two entities in a given KG, and $w(r)$ be the function mapping each relation to its tensor embedding. Then, the value of the path P is:*

$$V_{KG}^w(P) = \begin{cases} w(r) & \text{if } P = \langle r \rangle \\ w(r) \otimes V_{KG}^w(Q) & \text{if } P = \langle r, Q \rangle \end{cases}.$$

This means that for a path $P = \langle r_0, r_1, \ldots, r_n \rangle$, the value of the path is given by $r_0 \otimes r_1 \otimes \ldots \otimes r_n$.

Comparing Definition 5 above with Definition 3, it is clear that the value of a path is a special case of the value of a grammar derivation. Lemma 5.5.2 translates to the KG

$$\frac{w(A \to w_i)}{[i,A,j]}$$

$$\frac{w(A \to BC) \;\; [i,B,k] \;\; [k,C,j]}{[i,A,j]}$$

Figure 5.5: Item-based description for CKY

$$\frac{w(r) \;\; (h,r,t)}{F_1(h,t)}$$

$$\frac{F_1(h,e) \;\; F_{l-1}(e,t)}{F_l(h,t)}$$

Figure 5.6: Item-based description for the path-sum algorithm

case as follows:

Corollary 5.5.3.1. *A weight function w for a given KG is well defined if there exists consistent dimensions d_e for each entity e such that for all triples (h,r,t) in the KG, $w(r) \in \mathbb{S}^{d_t \times d_h}$*

Note that this condition allows considerable freedom on how to represent relations. In particular, if one is given an ontology over the entities as explored in Chapter 3, it is possible to learn embeddings for different types of entities in vector spaces of different dimensions. As long as the embedding dimensions of the relations agree with the embedding spaces for their head and tail types, it is possible to calculate the path representations.

5.5.4 Item-based Descriptions

Item-based descriptions, when used for describing parsers, consist of a set of deduction rules of the form $\frac{T_1 \ldots T_k}{Q} P_1 \ldots P_j$ where upper case letters could either be grammar rule templates (e.g. if $T_1 : A \to B\,C$ then any non-terminals from the grammar can be substituted for A,B,C) or for **items**. $T_1 \ldots T_k$ are referred to as antecedents, Q as the conclusion and $P_1 \ldots P_j$ are side conditions that the parser requires to execute the rule, but doesn't use the values of. Items correspond to chart elements in procedural descriptions of parsers, and are placeholders for intermediate results which can be combined to obtain the final result. The item-based description also provides a special **goal item** which is variable-free, and does not occur as a condition of any other inference rules.

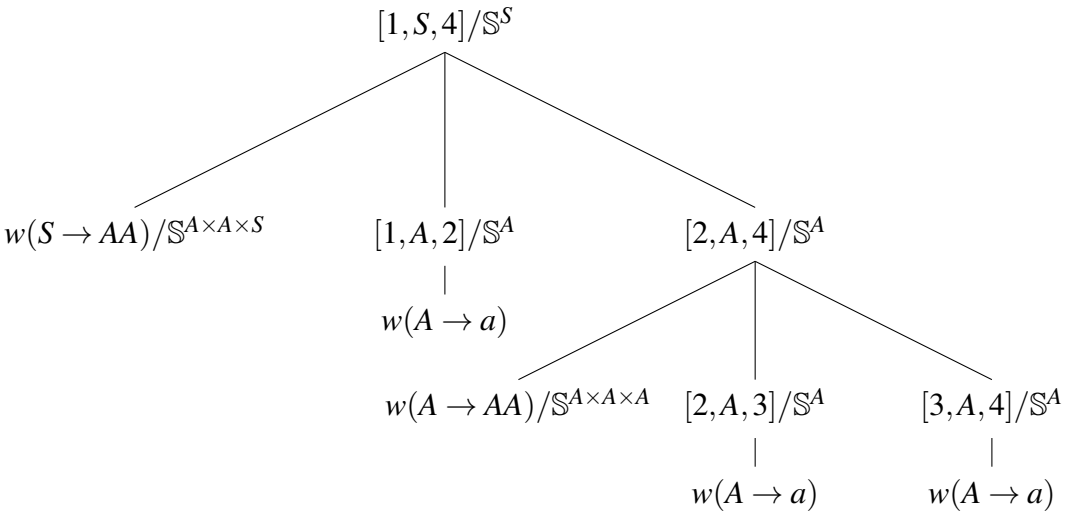


Figure 5.7: Item derivation corresponding to the derivation given in Figure 5.4 using the item-based description of CKY in Figure 5.5.

Definition 6. *Given a grammar G and an item-based description I, a valid* ***item derivation tree*** *is defined as follows:*

- *For all $r \in G$, $\langle r \rangle$ is an item derivation tree.*
- *If $D_{a_1},\ldots,D_{a_k}$ and $D_{c_1},\ldots,D_{c_j}$ are derivation trees headed by $a_1,\ldots,a_k$ and $c_1,\ldots,c_j$ respectively, and $\frac{a_1\ldots a_k}{b}c_1,\ldots,c_j$ is the instantiation of a deduction rule in I, then $\langle b : D_{a_1},\ldots,D_{a_k}\rangle$ is also an item derivation tree.*

$inner_\sigma(x)$ denotes the set of all trees headed by x that occur in parses for σ. Formally, $D \in inner_\sigma(x)$ if D is headed by x and is a subtree of some $D' \in \mathcal{D}_{I(G)}(\sigma)$, where $\mathcal{D}_{I(G)}(\sigma)$ is the set of all valid item-derivation trees for σ. The value of a derivation tree is calculated similarly to that of a grammar tree:

$$V^w_{I(G)}(D) = \begin{cases} w(D) & \text{if } D \text{ is a rule} \\ V^w_{I(G)}(D_1) \otimes [V^w_{I(G)}(D_2),\ldots,V^w_{I(G)}(D_n)] & \text{if } D = \langle b : D_1,\ldots,D_n\rangle \end{cases}.$$

Notice that unlike the definition for semiring WLPs given in 5.4.4, the first antecedent in the inference rule has a special role in the calculation. Intuitively, our framework treats the value of the first antecedent as a *function*, and the trailing ones as the arguments. The interaction between the trailing antecedents is thus moderated through the value of the first antecedent, which corresponds to the requirement that the children nodes be independent of each other given the parent node. The special role of the first antecedent also requires that the tensor weights have dimensions that enable the

calculation defined by any given derivation tree:

Definition 7. *If for any $\sigma \in \mathcal{L}(G)$ and any $T, T' \in inner_\sigma(x)$, $V^w_{I(G)}(T)$ and $V^w_{I(G)}(T')$ are defined and $dim(V^w_{I(G)}(T)) = dim(V^w_{I(G)}(T'))$, then the weights w are well defined.*

For any of our following theorems to hold, we require that the given weights are well defined.

Given an item-based derivation I, a grammar G, a well defined weight function w and a target sentence σ, the value of an item x is defined to be the sum of all its possible derivations. Formally:

$$V^w_{I(G)}(x, \sigma) = \bigoplus_{D \in inner_\sigma(x)} V^w_{I(G)}(D).$$

Definition 8. *For a given grammar G and item-based description I, the value of a sentence σ is equal to the value of the goal item which spans σ:*

$$V^w_{I(G)}(\sigma) = V^w_{I(G)}(goal, \sigma).$$

Definition 9. *An item-based description is **correct** if for all grammars G, complete semirings $\mathbb{S}$, well defined weight functions w and sentences σ, $V^w_{I(G)}(\sigma) = V^w_G(\sigma)$.*

Now we are ready to state the equivalent theorem to Theorem 5.4.1. Let us introduce a special symbol $\perp$ and extend V^w_G and $V^w_{I(G)}$ to any weight function w so that if w is not-well defined for G, then $V^w_G(\sigma) = \perp$ and likewise for $V^w_{I(G)}$.

Theorem 5.5.4. *An item-based description I is correct if*

- *For every grammar G, the mapping $g : \mathcal{D}_{I(G)} \to \mathcal{D}_G$ that maps $d' \in \mathcal{D}_{I(G)}$ to the corresponding $d \in \mathcal{D}_G$ is a bijection with an inverse function f.*
- *For any complete semiring $\mathbb{S}$ and weight function w, g and f preserve the values assigned to a derivation:*

$$V^w_G(d) = V^w_{I(G)}(f(d)) \quad and \quad V^w_{I(G)}(d') = V^w_G(g(d')).$$

Proof proceeds similarly to that in Goodman (1999) and can be found in Appendix A.

5.6 Efficient Calculation of Inside and Outside Values

In the following, we will omit the sentence σ from $inner_\sigma(x)$ and refer to this as $inner(x)$. Let $inner(\frac{a_1,\ldots,a_k}{x})$ be the set of derivation trees where the root note is x,

and the direct children of x are $a_1, \ldots, a_k$.

5.6.1 Inside Calculations

For efficient computation of and item value $V(x)$, we will assume that there is a partial order b on the items so that if the item y depends on x, then $b(x) \leq b(y)$. Given such a partial order the calculation of $V(x)$ can be performed as follows:

Theorem 5.6.1.

$$V(x) = \bigoplus_{\substack{[a_1,\ldots,a_k] \\ s.t. \frac{a_1,\ldots,a_k}{x}}} V(a_1) \otimes [V(a_2), \ldots, V(a_k)] .$$

The proof uses the distributive property and follows that of Goodman (1999). It can be found in Appendix B.

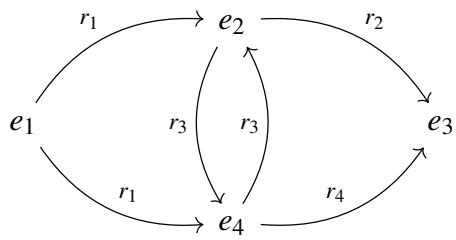
In the context of L-PCFG parsing, this formulation combined with the item based description in Figure 5.5 corresponds to the procedural algorithm given in Figure 5.2. To demonstrate, consider the inside calculation using the parser description for our running example with the grammar in Figure 5.4 and the string *aaa*. The partial order in this case is that $[i, A, j] \leq [i', B, j']$ if $i \leq i'$ and $j \leq j'$. The inside value for the goal item $[1, S, 4]$ is then calculated as follows:

$$\begin{aligned}
V([1,S,4]) =& w(S \to AA) \otimes \big[V([1,A,2]), V([2,A,4])\big] \\
& \oplus w(S \to AA) \otimes \big[V([1,A,3]), V([3,A,4])\big], \\
V([1,A,3]) =& w(A \to AA) \otimes \big[V([1,A,2]), V([2,A,3])\big] \\
V([2,A,4]) =& w(A \to AA) \otimes \big[V([2,A,3]), V([3,A,4])\big] \\
V([1,A,2]) =& V([2,A,3]) = V([3,A,4]) = w(A \to a).
\end{aligned}$$

Now, by iteratively calculating the intermediate item values and performing the appropriate substitutions, we can see that the full expression for the goal item can be obtained without calculating the value of each intermediate item multiple times:

$$\begin{aligned}
V([1,S,4]) =& w(S \to AA) \otimes \big[w(A \to a), w(A \to AA) \otimes [w(A \to a), w(A \to a)]\big] \\
& \oplus w(S \to AA) \otimes \big[w(A \to AA) \otimes [w(A \to a), w(A \to a)], w(A \to a)\big].
\end{aligned}$$

Inference over paths in the KG is can likewise be captured with the inside calculation using the item-based description in Figure 5.6. Consider a KG with entities $\{e_1, e_2, e_3, e_4\}$ and relations $\{r_1, r_2, r_3, r_4\}$:



Assume that we would like to calculate the path-sum of all paths of length 3 from e_1 to e_3. Using the item-based description in Figure 5.6, this would be equivalent to calculating the inside value for the goal item $L_3(e_1, e_3)$[3]:

$$
\begin{aligned}
F_3(e_1,e_3) =& F_1(e_1,e_2) \otimes F_2(e_2,e_3) \oplus F_1(e_1,e_4) \otimes F_2(e_4,e_3) \\
F_2(e_2,e_3) =& F_1(e_2,e_4) \otimes F_1(e_4,e_3) \\
F_2(e_4,e_3) =& F_1(e_4,e_2) \otimes F_1(e_2,e_3) \\
F_1(e_1,e_2) =& F_1(e_1,e_4) = w(r_1) \\
F_1(e_2,e_4) =& F_1(e_4,e_2) = w(r_3) \\
F_1(e_2,e_3) =& w(r_2) \\
F_1(e_4,e_4) =& w(r_4)
\end{aligned}
$$

Substituting the intermediate items with their corresponding weights, we get the value for the goal item:

$$F_3(e_1,e_3) = (w(r_1) \otimes w(r_3) \otimes w(r_4)) \oplus (w(r_1) \otimes w(r_3) \otimes w(r_2))$$

As in the previous example, we can see that the two summands correspond to the two paths between e_1 and e_3 that are of length 3.

Inside calculation with tensors of values from the Viterbi semiring. For the examples above, we did not explicitly specify from which semiring the tensor entries were taken from. For KGs, the standard semiring would be $\mathbb{R}$, and $w(r)$ would be matrix embeddings with various constraints, as explored in previous chapters. However, it is also instructive to consider how the calculation would be realized if the semiring is a different one, and semiring operations correspond to something other than the regular matrix addition and multiplication.

[3]We omit the weight term $\tanh(w_r)p(t|h,r)$ in the item description to simplify the presentation.

One of the semirings Goodman (1999) considers is the Viterbi-derivation semiring, which allows the inside calculation to capture the derivation tree with the highest probability together with its probability weight. This semiring is a product of two semirings. The first one is the Viterbi semiring **Vit** with values in $[0,1]$, addition as the $\oplus$ operator and **max** as the $\otimes$ operator. The other one is the derivation-forest semiring **Deriv**, which has as elements sets of derivation trees, set union as the $\oplus$ operator and pairwise concatenation as the $\otimes$ operator. For the Viterbi algorithm for L-PCFGs, tensors with entries in $(\mathbf{Vit}, \mathbf{Deriv})$ capture the correct calculations.

Similarly, it is possible to define a semiring for KG paths, **Str**. The elements of this semiring are sets of KG paths, **0** corresponds to the empty set $\emptyset$ and **1** is the set with the empty path $\{\langle\rangle\}$. $\oplus$ is defined as the set union, and $\otimes$ is the pairwise concatenation of the paths in each set. Then the Viterbi algorithm for inference over KG paths can be captured using weights from $(\mathbf{Str}, \mathbf{Vit})$. When there are tensor embeddings associated with the KG, the algorithm can be simply extended likewise, by using tensors with weights from this semiring.

5.6.2 Outside Calculations

For the notion of a value of a derivation to extend to outside trees, we will have to do some modifications. This is because an outside tree will have one subtree $\langle b : A_1, \ldots, A_n \rangle$, such that $V(A_1) \otimes [V(A_2), \ldots, V(A_n)]$ will potentially not be defined since one of the subtrees A_k will be missing. Note that the missing A_k will be headed by an item. We will say the a tree $T \in outer(x)$ if T can be obtained by taking a tree T' headed by the goal item and removing any of its subtrees headed by the item x. Outer value $Z(T_k)$ is defined recursively as follows:

If T_k is headed by the goal item then $Z(T_k) = I_{d_S}$. Else, it has a direct parent tree T such that $T = \langle b : T_1, \ldots, T_k, \ldots, T_n \rangle$. In this case,

$$Z(T_k) = \Big(V(T_1) \otimes_k \left[I_{T_k \times d_S}, V(T_{k+1}), \ldots, V(T_n) \right] \Big)^{\pi} \otimes [V(T_2), \ldots, V(T_{k-1})] \otimes^* Z(T),$$

where $I_{T_k \times d_S}$ is the identity tensor for the space $\mathbb{S}^{d_1 \times \ldots \times d_i \times d_S}$, $T_k \in \mathbb{S}^{d_1 \times \ldots \times d_i}$, and d_s is the dimension assigned for the terminal symbol S. The permutation π is defined as follows:

$$[1, 2, \ldots, i, j+1, j+2, \ldots, n, i+1, i+2, \ldots, j],$$

where $i = k + \text{rank}(T_k) - 1$ and $j = k + 2 \times \text{rank}(T_k) + 1$

To understand the function of π it is useful to consider the dimensions of the term before and after it is applied. Let the term $V(T_0) \otimes_k \left[I_{T_k \times d_S}, V(T_{k+1}), \ldots, V(T_n)\right]$ have dimensions:

$$e_1 \times \ldots \times e_{k-1}, d_1 \times \ldots \times d_i \times d_S \\ \times e_k \times d_1 \times \ldots \times d_i \times d_S \times d'_n \times \ldots \times d'_m.$$

Here $e_1, \ldots, e_{k-1}$ are the dimensions that will be contracted with $V(T_1), \ldots, V(T_{k-1})$ with the second multiplication operation, and $d'_n, \ldots, d'_m$ are the dimensions that were either introduced by the contraction with $V(T_{k+1}), \ldots, V(T_n)$ or were trailing dimensions from $V(T_1)$. The result of the contraction with $I_{T_k \times d_S}$ are the dimensions in the middle: $d_1, \ldots, d_i, d_S, e_k, d_1, \ldots, d_i, d_S$. Unlike the original definition of I there is one dimension e_k missing from the beginning of the sequence since it got used up during the contraction operation. What the permutation does is to move one section of the dimensions introduced by I to the very end. The dimensions become:

$$e_1 \times \ldots \times e_{k-1}, d_1 \times \ldots \times d_i \times \\ d'_n \times \ldots \times d'_m \times d_S \times e_k \times d_1 \times \ldots \times d_i \times d_S.$$

Note that this has no effect on the next contraction with $V(T_1), \ldots, V(T_{k-1})$ since the first $k-1$ orders are left in place. However, changing the order of the orders allow the last contraction with $Z(T)$ to be well defined.

Lemma 5.6.2. *Let V and Z be defined on a commutative semiring $\mathbb{S}$ and let $O \in outer_\sigma(x)$ and $T \in inner_\sigma(x)$. If combining O and T in the obvious way results in the complete derivation D,*

$$V(D) = V(T) \otimes^* Z(O).$$

Proof. (Sketch) We proceed by induction on the parse tree. Base case is where $x = goal$, $T = D$ and O is empty. Then $V(T) = V(D)$ and $Z(O) = I_S$. $V(D) \otimes^* I_S = V(D)$ by the definition of I_S which proves the statement.

Otherwise T has a parent tree $T_p = \langle y : T_1, \ldots, T_n \rangle$ where $T = T_k$. Furthermore, $T_p \in \text{inner}_\sigma(y)$, $O_p \in \text{outer}_\sigma(y)$ and by the induction hypothesis $V(D) = V(T_p) \otimes^* Z(O_p)$.

Since $T_p \in \text{inner}_\sigma(y)$ we know that

$$V(T_p) = V(T_1) \otimes [V(T_2), \ldots, V(T_m)], \\ V(D) = (V(T_1) \otimes [V(T_2), \ldots, V(T_m)]) \otimes^* Z(O_p).$$

The proof progresses by calculating the value for $[V(D)]_i$ based on the above term and shows that this is equal to the value of $[V(T) \otimes^* Z(O)]_i$. Full proof can be found in Appendix B. □

In the general case, Goodman (1999) defines the reverse value of x as the sum of all its outer trees.

$$Z(x) = \bigoplus_{T \in outer(x)} Z(T).$$

We will see that for a well defined weight function w, any $D \in outer_\sigma(x)$ will be assigned a value with dimensions $d_1 \times \ldots \times d_n \times d_S$ where d_S is the dimension assigned to the start symbol S, and $d_1, \ldots, d_n$ are the dimensions for $inner_\sigma(x)$.

Lemma 5.6.3. *Let $C(D,x)$ represent the number of times x occurs in a derivation D. Then,*

$$V(x) \otimes^* Z(x) = \bigoplus_{D \in \mathcal{D}(\sigma)} V(D)C(D,x).$$

Proof.

$$\begin{aligned} V(x) \otimes^* Z(x) &= \bigoplus_{T \in inner(x)} V(T) \otimes^* \bigoplus_{O \in outer(x)} Z(O) \\ &= \bigoplus_{T \in inner(x)} \bigoplus_{O \in outer(x)} V(T) \otimes^* Z(O). \end{aligned}$$

By Lemma 5.6.2, $Z(O) \otimes^* V(T) = V(D)$. For an item x, any $O \in outer(x)$ and $T \in inner(x)$ can be combined to form a successful derivation tree containing x, and thus the number $C(D,x)$ corresponds exactly to the number of derivation trees containing x. Hence,

$$\begin{aligned} V(x) \otimes^* Z(X) &= \bigoplus_{\substack{T \in inner(x) \\ O \in outer(x)}} V(T) \otimes^* Z(O) \\ &= \bigoplus_{D \in \mathcal{D}(\sigma)} V(D)C(D,x). \square \end{aligned}$$

Now we are ready to state how to calculate the outside value of an item. Following Goodman (1999) we will extend the notation for the set of outer trees and introduce $outer\left(k, \frac{a_1 \ldots a_n}{b}\right) \subseteq outer(a_k)$ to mean the subset of the outer trees in $outer(a_k)$ where a_k has parent b and the siblings a_i. In other words, this is the set of all outer trees where the rule from which a_k is removed is $\dfrac{a_1 \ldots a_n}{b}$.

Theorem 5.6.4. *If x is the goal item, then $Z(x) = I_s$. Else,*

$$Z(x) = \bigoplus_{j,a_1,..,a_k,b \text{ s.t. } \frac{a_1 \ldots a_k}{b} \text{ and } x=a_j} (V(a_1) \otimes_k [I_{a_k}, V(a_{k+1}), ..., V(a_n)])^{\pi} \otimes [V(a_2), \ldots, V(a_{k-1})] \otimes^* Z(b).$$

Proof. (sketch) $Z(x) = \bigoplus_{D \in outer(x)} Z(D)$. Either x is a goal item, in which case $Z(x) = I_S$.

Otherwise the outer trees $outer(x)$ could be written as the union of outer trees $outer\left(k, \frac{a_1 \ldots a_n}{b}\right)$ for each rule $\frac{a_1 \ldots a_n}{b}$ where $a_k = x$ for some k. Hence:

$$Z(x) = \bigoplus_{\substack{j,a_1,..,a_k,b \text{ s.t.} \\ \frac{a_1 \ldots a_k}{b} \text{ and } x=a_j}} \bigoplus_{D \in outer\left(k, \frac{a_1 \ldots a_n}{b}\right)} Z(D).$$

Using the distributive property of the partial semiring, the inside part of the equation becomes:

$$\bigoplus_{D \in outer\left(k, \frac{a_1 \ldots a_n}{b}\right)} Z(D) = (V(a_1) \otimes_k [I_{a_k}, V(a_{k+1}), \ldots, V(a_n)])^{\pi} \otimes [V(a_2), \ldots, V(a_{k-1})] \otimes^* Z(b).$$

Replacing the inner part of the previous equation with this term gives the desired equality. □

5.7 Conclusion and Future Directions

In this chapter we extend semiring-WLPs to tensor WLPs, where the weights of a logic program are allowed to be tensors of semirings. We provide some results that connect tensor weighted grammars to WLPs that describe parsers, and the general forms of the inside and outside calculations that provide the means to extract dynamic programming algorithms from tensor weighted logic programs. These results are applicable in developing algorithms for knowledge graph reasoning with embedding weights, as well as syntactic parsing.

Generalizing semiring weights to multi-indexed sets of semiring values is one way to incorporate tensor weights into the WLP framework, which we have pursued here. Another theoretical abstraction applicable in this setting would be to use *category theory*

as done in Clark et al. (2008) and Coecke et al. (2010) to model compositionality of tensor representations for the meanings of words into the meaning of a sentence.

In the categorical view of WLP, a grammar specification would define a particular category, and a weight assignment to rules corresponds to a functor between this category and the category of vector spaces and (multi)linear maps. This view is more abstract than the work presented here, and has associated advantages and disadvantages. The main advantage is that the categorical view automatically handles tensors of different arity and the related restrictions on their composition. The disadvantage is that it is further removed from calculations of the values of interest, and hence would likely not provide as clear a connection to dynamic programming algorithms. Nevertheless, the alternative characterization of tensor WLPs as categories and functors between them is an interesting area for further research.

Chapter 6

Conclusion

In this thesis, we explore tensor methods for integrating external information and knowledge graph facts into unified representations. In order to summarize our findings in relation to the overarching motivation of this dissertation, we repeat our thesis statement below:

Thesis Statement. *Tensor representations provide a unified framework for modelling knowledge graphs and different types of related data. Tensor methods can be applied both for learning effective representations of entities and relations through knowledge fusion, and for efficiently reasoning over the learned representations.*

In Chapter 2, we present the connection between tensors, knowledge graphs and the task of knowledge graph completion. Specifically, we show that KGs have a natural representation as high-dimensional, sparse, binary tensors, and bilinear models for KG completion learn embeddings that correspond to a dense, low-dimensional, real-valued decomposition of this tensor.

In Chapters 3 and 4, we show that both information about the types of entities, and cooccurrence information from an entity linked corpus can be effectively represented as a matrix or tensor related to the adjacency tensor of the knowledge graph. Viewing both the KG and the related data as tensors enables us to characterize the enriched embeddings we aim to obtain as the components of a joint matrix-tensor or tensor-tensor decomposition. We provide joint training algorithms that achieve this objective, and assess their effectiveness in improving the model performance on the KG completion tasks through experiments on standard datasets. These two chapters provide support

for our claim that tensor methods can be used to obtain effective embeddings through knowledge fusion of a KG and related data.

In Chapter 5 we address the remaining part of our thesis statement, and show that tensors in general, and knowledge graph embeddings in particular, are suitable structures to be used as weights in logical deduction. We develop the theory for extending semiring weighted logic programs to tensors over semirings, and show that tensor values allow efficient calculation of aggregate representations of possible deduction trees. The theoretical findings presented support our claim that tensors can be integrated into logical reasoning approaches on knowledge graphs.

Limitations. The work presented in this thesis has some limitations that suggest avenues of future work. We list some of these below:

- The embedding learning techniques presented in Chapter 3 and 4 are evaluated only on knowledge graph completion, which is the same task the representations are trained on. The true value of an embedding representation obtained from multiple related data sources would be its broader applicability, and hence evaluating the effectiveness of these embeddings on downstream tasks, and comparing them with other techniques for graph embeddings would provide evidence of whether our methods are effective beyond KG completion.
- Our focus on tensor methods means that the methods we explore for knowledge fusion are limited to multi-linear functions. While this provides simple, effective and efficient algorithms for learning and scoring embeddings and fulfills the requirements for the WLP framework, it is possible that non-linear approximations are more suitable for some cases. Deep, highly non-linear transformer architectures largely superseded the fast and shallow models for word embeddings, and it is possible that similarly complex architectures would provide better embeddings for knowledge graphs with external information compared to tensor methods. It is especially worth exploring how joint training compares to deep learning techniques for knowledge transfer.
- The tensor-WLP framework we present in Chapter 5 provides the theoretical basis for performing reasoning using embeddings for entities, relations, and rules. However the implementation of such a system is left for future work, and some related practical issues remain. For example, tensor weights for rules often end

up as higher-order tensors. This means that a naive implementation is likely to result in prohibitively many parameters both for inference calculations and parameter estimation. Finding an appropriate restriction on the embeddings to limit the number of free parameters is one of the tasks that need to be accomplished in order to enable implementing tensor-WLPs for reasoning with knowledge graph embeddings. Tensor decomposition methods are a promising direction for finding low-dimensional approximations of these tensors, and to make the inference calculations computationally tractable.

Appendix A

Entity-Linking for Experiments in Chapter 4

English Wikipedia from ClueWeb09 with FACC annotations. We use the English Wikipedia section of ClueWeb09 Callan et al. (2009) and use FACC Freebase ID annotations
(Gabrilovich et al., 2013) to link the corpus with FB15K, FB15K-237 and YAGO3-10. Linking FB15K and FB15K-237 is straightforward since entities are represented with Freebase IDs both in the dataset and in the annotations. For YAGO3-10 we convert the YAGO IDs into Freebase IDs with the following heuristic:

The majority of entities in YAGO3-10 are derived from Wikipedia, in which case their YAGO IDs are the same as their Wikipedia identifier. We query the Wikidata API using this identifier, and use the Freebase ID to link the entity to corpus if the ID is found in the Wikidata entry. There is also a small number of entities in YAGO3-10 that are from WordNet. We convert these to their common noun forms if they are single word nouns, and link them with the occurrences of this noun from the corpus. If the name of the entity is multiple words we treat them as entities that do not have links to the corpus.

POS-tagged English Wikipedia from ClueWeb09. Entities in WN18RR correspond to *synsets* of WordNet. To link these to the English Wikipedia section of ClueWeb09, we first POS tag the corpus using the NLTK library (Bird, 2006). Since each entity consists of a 'sense' and that the corpus is not sense-disambiguated, we link the words to synsets heuristically. Synset labels consist of a string that has one or more words,

the POS tag, and a sense ID. We only link the entities that have sense ID 1 (this tends to correspond to the first sense in which the word is used) and have single word strings, in which case we link the synset with its corresponding word. The simple linking procedure results in about half of the entities to be linked to corpus occurrences.

Wikipedia with entity annotations from hyperlinks To generate this corpus, we use the latest English Wikipedia dump, and use the hyperlinks within Wikipedia to guide the entity linking for FB15K, FB15K-237 and YAGO3-10. Our code is adapted from Yamada et al. (2020) and works as follows:

For a given Wikipedia page, any text that is hyperlinked to another Wikipedia page is considered to refer to the entity of that page, and the text is replaced with the Wikipedia ID for the page. For detecting mentions that are not hyperlinked, the algorithm first collects all pairs of hyperlinks and the hyperlinked tokens throughout the Wikipedia dump.[1] Then for each page, the algorithm constructs a set of candidate entities. These are either the entity of the page or any of the entities that appear in it as hyperlinks. Then a token t is replaced with the ID for one of the candidate entities e if e is the only entity in the candidate set that t has ever been hyperlinked to.

Linking the entity annotations generated this way to YAGO3-10 is straightforward since both the corpus and the KG dataset use Wikipedia IDs as entity labels. We link FB15K and FB15K-237 datasets by querying Wikidata for Wikipedia IDs given the Freebase IDs.

[1] A pair of entity and text (e,t) is added to the collection if the ratio of the number of times t appears linked to any entity to its overall occurrence count is no less than 0.2, and the ratio of the number of times t is linked to e to the total number of times t is linked to an entity is no less than 0.01

Appendix B

Proofs of Theorems in Chapter 5

Lemma 5.5.1. *For any k,l, $\otimes_{[k;l]}$ distributes over $\oplus$*

Proof. We will proceed by showing that:

$$A \otimes_{[k;l]} (B \oplus C) = (A \otimes_{[k;l]} B) \oplus (A \otimes_{[k;l]} C)$$

Firstly, note that for the left hand side of the equation to be defined, B and C needs to be of matching ranks, and that $B \oplus C$ will be the same rank as both B and C. Therefore, if the left hand side is well defined then both $A \otimes_{[k;l]} B$ and $A \otimes_{[k;l]} C$ is defined and has matching ranks. So the right hand side is defined if and only if the left hand side is defined as well.

$$\begin{aligned}
&[A \otimes_{[j;k]} (B \oplus C)]_{i_1,\ldots,i_{k-1},j_1,\ldots,j_{l-1},j_{l+1},\ldots,j_m,i_{k+1},\ldots,i_n} \\
&\quad = \sum_{i_k,j_l} \delta(i_k,j_l) A_{i_1,\ldots,i_n} \times (B \oplus C)_{j_1,\ldots,j_m} \\
&\quad = \sum_{i_k,j_l} \delta(i_k,j_l) A_{i_1,\ldots,i_n} \times (B_{j_1,\ldots,j_m} + C_{j_1,\ldots,j_m}) \\
&\quad = \sum_{i_k,j_l} \delta(i_k,j_l)(A_{i_1,\ldots,i_n} \times B_{j_1,\ldots,j_m}) + \delta(i_k,j_l)(A_{i_1,\ldots,i_n} \times C_{j_1,\ldots,j_m}) \\
&\quad = [(A \otimes_{[k;l]} B) \oplus (A \otimes_{[k;l]} C)]_{i_1,\ldots,i_{k-1},j_1,\ldots,j_{l-1},j_{l+1},\ldots,j_m,i_{k+1},\ldots,i_n}
\end{aligned}$$

□

Theorem 5.4.1. *An item-based description I is correct if*

- *For every grammar G, the mapping* $g : \mathcal{D}_{I(G)} \to \mathcal{D}_G$ *that maps* $d' \in \mathcal{D}_{I(G)}$ *to the corresponding* $d \in \mathcal{D}_G$ *is a bijection with an inverse function* f.
- *For any complete semiring S and weight function w,* g *and* f *preserve the values assigned to a derivation:*

$$V_G^w(d) = V_{I(G)}^w(f(d)) \text{ and } V_{I(G)}^w(d') = V_G^w(g(d'))$$

Proof.

$$V_{I(G)}^w(\alpha) = V_{I(G)}^w(goal, \alpha) = \bigoplus_{D \in \text{inner}_\alpha(goal)} V_{I(G)}^w(D) = \bigoplus_{D \in \mathcal{D}_{I(G)}(\alpha)} V_G^w(g(D))$$

Observe that $D \in \mathcal{D}_{I(G)}(\alpha)$ iff $g(D) \in \mathcal{D}_G(\alpha)$ since the rules that appear in the leaves of D, applied from left to right, determines the grammar derivation tree $g(D)$ uniquely via g, and vice versa. Hence,

$$V_{I(G)}^w(\alpha) = \bigoplus_{g(D) \in \mathcal{D}_G(\alpha)} V_G^w(g(D)) = V_G^w(\alpha)$$

□

Theorem 5.6.1.

$$V(x) = \bigoplus_{\substack{[a_1,\ldots,a_k] \\ s.t. \frac{a_1 \ldots a_k}{x}}} V(a_1) \otimes [V(a_2), \ldots, V(a_k)]$$

Proof. Recall that by definition, $V(x) = \bigoplus_{D \in inner(x)} V(D)$. For any item derivation D, D is either an axiom or there is some $a_1, \ldots, a_k, b$ s.t. $D \in inner(\frac{a_1 \ldots a_k}{b})$. If D is an axiom, then $inner(D)$ is just a single rule a, and so $V(D) = V(a)$. Else, for each rule $\frac{a_1 \ldots a_k}{x}$

$$\bigoplus_{D \in inner(\frac{a_1 \ldots a_k}{x})} V(D) = \bigoplus_{\substack{D_{a_1} \in inner(a_1), \ldots, \\ D_{a_k} \in inner(a_k)}} V(D_{a_1}) \otimes [V(D_{a_2}), \ldots, V(D_{a_k})]$$

$$= \left(\bigoplus_{D_{a_1} \in inner(a_1)} V(D_{a_1}) \right) \otimes \left(\bigoplus_{\substack{D_{a_2} \in inner(a_2), \ldots, \\ D_{a_k} \in inner(a_k)}} \bigotimes_{i=2}^{k} V(D_{a_i}) \right)$$

$$= \left(\bigoplus_{D_{a_1} \in inner(a_1)} V(D_{a_1}) \right) \otimes \left(\bigoplus_{D_{a_2} \in inner(a_2)} V(D_{a_2}), \ldots, \bigoplus_{D_{a_k} \in inner(a_k)} V(D_{a_k}) \right)$$

$$= V(a_1) \otimes [V(a_2), \ldots, V(a_k)]$$

Where the last step holds due to the distributive property of the partial semiring.

Since the set $inner(x) = \bigcup_i D_i$ where $D_i \in inner(\frac{a_1 \ldots a_k}{x})$ for all inference rules $\frac{a_1 \ldots a_k}{x}$, we can write the summation over $D \in inner(x)$ as:

$$\begin{aligned} V(x) &= \bigoplus_{D \in inner(x)} V(D) \\ &= \bigoplus_{\substack{[a_1,\ldots,a_k] \\ \text{s.t.} \frac{a_1,\ldots,a_k}{x}}} \bigoplus_{D \in inner(\frac{a_1 \ldots a_k}{x})} V(D) \\ &= \bigoplus_{\substack{[a_1,\ldots,a_k] \\ \text{s.t.} \frac{a_1,\ldots,a_k}{x}}} V(a_1) \otimes [V(a_2), V(a_3), \ldots, V(a_k)] \end{aligned}$$

Where the last line is obtained by replacing the inner part of the expression with the equality obtained from the previous part of the proof. □

Lemma 5.6.2. *Let V and Z be defined on a commutative semiring $\mathbb{S}$ and let $O \in outer_\alpha(x)$ and $T \in inner_\alpha(x)$. If combining O and T in the obvious way results in the complete derivation D then*

$$V(D) = V(T) \otimes^* Z(O)$$

Proof. To simplify notation of the indices, let $\mathbf{i}$ stand for a list of indices $i_1, \ldots, i_n$ for some n. We will also use $\mathbf{d}^i$ to denote a list $d^i_1, \ldots d^i_{n_i}$ and $\mathbf{d}$ to denote $\mathbf{d}^1, \ldots, \mathbf{d}^n$. $\delta(\mathbf{i}, \mathbf{j}) = \prod_{k=1}^n \delta(i_k, j_k)$.

We will proceed by induction on the parse tree. Base case is where $x = goal$, $T = D$ and O is empty. Then $V(T) = V(D)$ and $Z(O) = I_S$. $V(D) \otimes^* I_S = V(D)$ by the definition of I_S which proves the statement.

Otherwise T has a parent tree $T_p = \langle y : T_1, \ldots, T_n \rangle$ where $T = T_k$. Furthermore, $T_p \in \text{inner}_\alpha(y)$, $O_p \in \text{outer}_\alpha(y)$ and by induction hypothesis $V(D) = V(T_p) \otimes^* Z(O_p)$.

Since $T_p \in \text{inner}_\alpha(y)$ we know that

$$V(T_p) = V(T_1) \otimes [V(T_2), \ldots, V(T_m)]$$

So

$$V(D) = (V(T_1) \otimes [V(T_2), \ldots, V(T_m)]) \otimes^* Z(O_p)$$

The proof progresses by calculating the value for $[V(D)]_i$ based on the above term and shows that this is equal to the value of $[V(T) \otimes^* Z(O)]_i$.

Let:

$$V(T_1) \in \mathbb{S}^{\mathbf{e},\mathbf{f}} \qquad V(T_i) \in \mathbb{S}^{e_i,\mathbf{d}^i}$$
$$Z(O_p) \in \mathbb{S}^{\mathbf{d},\mathbf{f},s} \qquad V(D) \in \mathbb{S}^{s}$$

Then:

$$\begin{aligned} V[(T_p)]_{\mathbf{d},\mathbf{f}} &= [V(T_1) \otimes (V(T_2),\ldots,V(T_m))]_{\mathbf{d},\mathbf{f}} \\ &= \sum_{\mathbf{e},\mathbf{e}'} V(T_1)_{\mathbf{e},\mathbf{f}} \times \prod_{i=2}^{m} \delta(e_i,e_i') V(T_i)_{e_i',\mathbf{d}^i} \end{aligned}$$

$$\begin{aligned} [V(D)]_s = [V(T_p) \otimes^* Z(O_p)]_s = \\ \sum_{\mathbf{e},\mathbf{e}',\mathbf{d},\mathbf{d}'\mathbf{f},\mathbf{f}'} V(T_1)_{\mathbf{e},\mathbf{f}} \times \left(\prod_{i=2}^{m} \delta(e_i,e_i') V(T_i)_{e_i,\mathbf{d}^i} \right) \\ \times \delta(\mathbf{d},\mathbf{d}') \delta(\mathbf{f},\mathbf{f}') Z(O_p)_{\mathbf{d},\mathbf{f},s} \end{aligned}$$

Now we will proceed to prove that this term is equal to $V(T_k) \otimes^* Z(O)$. Let $I_{T_k} \in \mathbb{S}^{e_k',\mathbf{d}^k,s,e_k,\mathbf{d}^k,s}$. We will calculate the value of the outside term in sections. Let

$$A = V(T_1) \otimes_k (I_{T_k}, V(T_{k+1}),\ldots,V(T_n)).$$

Then,

$$\begin{aligned} &A_{e_1,\ldots,e_{k-1},\mathbf{d}^k,s,\hat{e}_k,\hat{\mathbf{d}}_k,\hat{s},\mathbf{d}^{k+1},\ldots,\mathbf{d}^n,\mathbf{f}} = \\ &A^{\pi}_{e_1,\ldots,e_{k-1},\mathbf{d}^k,\mathbf{d}^{k+1},\ldots,\mathbf{d}^n,\mathbf{f},s,\hat{e}_k,\hat{\mathbf{d}}_k,\hat{s}} = \\ &\sum_{\substack{e_k,\ldots,e_n \\ e_k',\ldots,e_n'}} V(T_1)_{\mathbf{e},\mathbf{f}} \times \delta(e_k,e_k') \delta(\mathbf{d}^k,\hat{\mathbf{d}}^k) \delta(s,\hat{s}) \times \prod_{i=k+1}^{m} \delta(e_i,e_i') V(T_i)_{e_i',\mathbf{d}^i} \end{aligned}$$

$$\begin{aligned} &[A^{\pi} \otimes (V(T_2),\ldots,V(T_{k-1}))]_{\mathbf{d},\mathbf{f},s,\hat{e}_k,\hat{\mathbf{d}}^k,\hat{s}} = \\ &\sum_{\mathbf{e},\mathbf{e}'} V(T_1)_{\mathbf{e},\mathbf{f}} \times \prod_{\substack{i=2 \\ i\neq k}}^{n} V(T_i)_{e_i',\mathbf{d}^i} \times \delta(\mathbf{e},\mathbf{e}') \times \delta(e_k,\hat{e}_k) \times \delta(\mathbf{d}^k,\hat{\mathbf{d}}^k) \times \delta(s,\hat{s}) \end{aligned}$$

$$\begin{aligned} [Z(O)]_{\hat{e}_k,\hat{\mathbf{d}}^k,\hat{s}} = \sum_{\substack{\mathbf{e},\mathbf{e}',\mathbf{d},\mathbf{d}' \\ \mathbf{f},\mathbf{f}',s,s'}} V(T_1)_{\mathbf{e},\mathbf{f}} \times \prod_{\substack{i=2 \\ i\neq k}}^{n} V(T_i)_{e_i',\mathbf{d}^i} \times Z(O_p)_{\mathbf{d}',\mathbf{f}',s'} \\ \times \delta(\mathbf{e},\mathbf{e}') \times \delta(e_k,\hat{e}_k) \times \delta(\mathbf{d}^k,\hat{\mathbf{d}}^k) \times \delta(s,\hat{s}) \\ \times \delta(\mathbf{d},\mathbf{d}') \times \delta(\mathbf{f},\mathbf{f}') \times \delta(s,s') \end{aligned}$$

$$
\begin{aligned}
&[V(T_k) \otimes^* Z(O)]_{\hat{s}} = \\
&\quad \sum_{\substack{\mathbf{e},\mathbf{e}',\mathbf{d},\mathbf{d}' \\ \mathbf{f},\mathbf{f}',s,s' \\ \hat{e}_k,\hat{\mathbf{d}}^k,e''_k,\mathbf{d}^{k''}}} V(T_k)_{e''_k,\mathbf{d}^{k''}} \times V(T_1)_{\mathbf{e},\mathbf{f}} \times \prod_{\substack{i=2 \\ i\neq k}}^{n} V(T_i)_{e'_i,\mathbf{d}^i} \times Z(O_p)_{\mathbf{d}',\mathbf{f}',s'} \\
&\qquad \times \delta(\mathbf{e},\mathbf{e}') \times \delta(e_k,\hat{e}_k) \times \delta(\mathbf{d}^k,\hat{\mathbf{d}}^k) \times \delta(s,\hat{s}) \\
&\qquad \times \delta(\mathbf{d},\mathbf{d}') \times \delta(\mathbf{f},\mathbf{f}') \times \delta(s,s') \times \delta(e''_k,\hat{e}_k) \times \delta(\mathbf{d}^{k''},\hat{\mathbf{d}}^k) \\
&= \sum_{\mathbf{e},\mathbf{e}',\mathbf{d},\mathbf{d}'\mathbf{f},\mathbf{f}'} V(T_1)_{\mathbf{e},\mathbf{f}} \times \prod_{i=2}^{m} V(T_i)_{e_i,\mathbf{d}^i} \times Z(O_p)_{\mathbf{d},\mathbf{f},\hat{s}} \\
&\qquad \times \delta(\mathbf{e},\mathbf{e}') \times \delta(\mathbf{d},\mathbf{d}') \times \delta(\mathbf{f},\mathbf{f}')
\end{aligned}
$$

Which completes the proof. The last simplification step is obtained by replacing $\hat{e}_k$ and e''_k with e_k, $\hat{\mathbf{d}}^k$ and $\mathbf{d}^{k''}$ with $\mathbf{d}^k$ and s and s' with $\hat{s}$ since these need to be equal for any term to contribute to the final sum. The commutativity of $\mathbb{S}$ then allows $V(T_k)_{e_k,\mathbf{d}^k}$ to be moved to its place in the sequence.

□

Theorem 5.6.4. *If x is the goal item, then $Z(x) = I_s$. Else,*

$$
\begin{aligned}
Z(x) = \bigoplus_{\substack{j,a_1,..,a_k,b \text{ s.t.} \\ \frac{a_1 \ldots a_k}{b} \text{ and } x=a_j}} &(V(a_1) \otimes_k [I_{a_k}, V(a_{k+1}), \ldots, V(a_n)])^{\pi} \\
&\otimes (V(a_2), \ldots, V(a_{k-1})) \otimes^* Z(b)
\end{aligned}
$$

Proof. by definition $Z(x) = \bigoplus_{D \in outer(x)} Z(D)$. Either x is a goal item, in which case $Z(x) = Z() = I_S$.

Otherwise the outer trees $outer(x)$ could be written as the union of outer trees $outer\left(k, \frac{a_1 \ldots a_n}{b}\right)$ for each rule $\frac{a_1 \ldots a_n}{b}$ where $a_k = x$ for some k. Hence:

$$
Z(x) = \bigoplus_{\substack{j,a_1,..,a_k,b \text{ s.t.} \\ \frac{a_1 \ldots a_k}{b} \text{ and } x=a_j}} \bigoplus_{D \in outer\left(k, \frac{a_1 \ldots a_n}{b}\right)} Z(D)
$$

For the inner part of this equation we have:

$$\bigoplus_{D \in outer\left(k, \frac{a_1 \dots a_n}{b}\right)} Z(D) =$$
$$\bigoplus_{D_b \in outer(b)} \bigoplus_{\substack{D_{a_1} \in inner(a_1), \dots, \\ D_{a_{k-1}} \in inner(a_{k-1})}} \bigoplus_{\substack{D_{a_{k+1}} \in inner(a_{k+1}), \dots, \\ D_{a_n} \in inner(a_n)}}$$
$$\left(V(D_{a_1}) \otimes_k \left[I_{D_{a_k}} \times d_S, V(D_{a_{k+1}}), \dots, V(D_{a_n})\right]\right)^{\pi}$$
$$\otimes \left(V(D_{a_2}), \dots, V(D_{a_{k-1}})\right) \otimes^* Z(D_b)$$

Since $\oplus$ distributes over $\otimes$, this can rewritten as

$$\bigoplus_{D \in outer\left(k, \frac{a_1 \dots a_n}{b}\right)} Z(D) =$$
$$\left(\bigoplus_{\substack{D_{a_1} \in \\ inner(a_1)}} V(D_{a_1}) \otimes_k \left[I_{D_{a_k}}, \bigoplus_{\substack{D_{a_{k+1}} \in \\ inner(a_{k+1})}} V\left(D_{a_{k+1}}\right), \dots, \bigoplus_{\substack{D_{a_n} \in \\ inner(a_n)}} V\left(D_{a_n}\right) \right] \right)^{\pi}$$
$$\otimes \left(\bigoplus_{D_{a_2} \in inner(a_2)} V(D_{a_2}), \dots, \bigoplus_{D_{a_{k-1}} \in inner(a_{k-1})} V(D_{a_{k-1}}) \right)$$
$$\otimes^* \bigoplus_{D_b \in outer(b)} Z(D_b)$$

And since $V(a_i)$ and $Z(D_b)$ are defined as the summation of their inner and outer trees respectively

$$\bigoplus_{D \in outer\left(k, \frac{a_1 \dots a_n}{b}\right)} Z(D) =$$
$$\left(V(a_1) \otimes_k [I_{a_k}, V(a_{k+1}), \dots, V(a_n)]\right)^{\pi} \otimes \left(V(a_2), \dots, V(a_{k-1})\right) \otimes^* Z(b)$$

Replacing the inner part of the previous equation with this term gives us the desired equality, completing the proof. □

Appendix C

Inside and Outside Calculations for Looping Buckets

In computing the inside and outside values with an item-based description, we assume a pre-computed ordering over items in the form of *buckets*. For items x and y, we write $\mathit{bucket}(x) \leq \mathit{bucket}(y)$ if the value of y depends on the value of x. So far we have assumed that items could be simply sorted so that no item directly or indirectly depends on itself, and given the inside and outside formulas accordingly. In this section we give the equivalent formulas for items in *looping buckets*. Items in a looping bucket depend on each other and computing their values might require an infinite sum. Our presentation and proofs both follow that of Goodman (1998).

For an item x in a looping bucket B, let the *generation* of a derivation tree x to be the maximum number of items in B that could appear in a single path from the root to a leaf. This intuitively provides an ordering for processing a potentially infinite number of trees by starting from generation 0 and incrementally adding larger and larger trees. We will denote the set of inner trees of x with generation at most g with $\mathit{inner}_{\leq}(x,B)$ Adding up the values of all inner trees of x that have generation at most g then gives us an approximation for the true inner value of x, and the approximation gets better as g gets larger. Formally, we define a *g generation value* for an item x in bucket B as:

$$V_{\leq g}(x,B) = \bigoplus_{D \in \mathit{inner}_{\leq g}(x,B)} V(D)$$

For ω-continuous semirings, the infinite sum is equal to the supremum of the partial

sums (Kuich 1997, 613), hence (Goodman 1999, 589):

$$V(x) = \bigoplus_{D \in inner(x)} V(D) = \sup_g V_{\leq g}(x, B)$$

Fortunately, tensors of semirings of set dimensions are ω-continuous as long as the underlying semiring is ω-continuous. We give the necessary definitions to establish this property:

Definition 10. *(Kuich 1997, 611) A semiring is **naturally ordered** if there is a partial ordering $\sqsubseteq$ such that $x \sqsubseteq y$ iff there is a z s.t. $x \oplus z = y$.*

Definition 11. *(Kuich 1997, 612) A naturally ordered complete semiring is ω-continuous if for any sequence $x_1, x_2, \ldots$ and for any constant y, if for all n, $\bigoplus_{0 \leq i \leq n} x_i \sqsubseteq y$ then $\bigoplus_i x_i \sqsubseteq y$*

Notice that for the set of tensors in $\mathbb{S}^{\mathbf{d}}$ where $\mathbf{d}$ is an arbitrary list of positive integers, if the underlying semiring has a natural ordering then this could be extended straightforwardly to $\mathbb{S}^{\mathbf{d}}$ by the following rule: $\mathbf{X} \sqsubseteq \mathbf{Y}$ iff $\mathbf{X_i} \sqsubseteq \mathbf{Y_i}$ for all indices $\mathbf{i}$. It is straightforward to check that if the underlying semiring is ω-continuous, then $\mathbb{S}^{\mathbf{d}}$ is ω-continuous as well.

Goodman (1999) gives a formula for $V_{\leq g}(x, B)$ in order to compute or approximate the supremum. Below we give the analogous formula for partial semirings:

Theorem B.1. *For items x in a looping bucket B and the generation $g \geq 1$*

$$V_{\leq g}(x, B) = \bigoplus_{\substack{[a_1, \ldots, a_k] \\ s.t. \frac{a_1, \ldots, a_k}{x}}} K_g(a_1, B) \otimes [K_g(a_2, B), \ldots, K_g(a_k, B)]$$

Where

$$K_g(a, B) = \begin{cases} V(a) & \text{if } a \notin B \\ V_{\leq g-1}(a, B) & \text{if } a \in B \end{cases}$$

Proof.

$$
\begin{aligned}
V_{\leq g}(x,B) &= \bigoplus_{D \in inner_{\leq g}(x,B)} V(D) \\
&= \bigoplus_{\substack{[a_1,\ldots,a_k] \\ \text{s.t.} \frac{a_1,\ldots,a_k}{x}}} \bigoplus_{\substack{D_{a_1} \in inner_{\leq g-1}(a_1,B),\ldots, \\ D_{a_k} \in inner_{\leq g-1}(a_k,B)}} V\left(\langle x : D_{a_1}, \ldots D_{a_k} \rangle\right) \\
&= \bigoplus_{\substack{[a_1,\ldots,a_k] \\ \text{s.t.} \frac{a_1,\ldots,a_k}{x}}} \bigoplus_{\substack{D_{a_1} \in inner_{\leq g-1}(a_1,B),\ldots, \\ D_{a_k} \in inner_{\leq g-1}(a_k,B)}} V(D_{a_1}) \otimes [V(D_{a_2}), \ldots, V(D_{a_k})] \\
&= \bigoplus_{\substack{[a_1,\ldots,a_k] \\ \text{s.t.} \frac{a_1,\ldots,a_k}{x}}} \bigoplus_{D_{a_1} \in inner_{\leq g-1}(a_1,B)} V(D_{a_1}) \\
&\qquad \otimes \left[\bigoplus_{D_{a_2} \in inner_{\leq g-1}(a_2,B)} V(D_{a_2}), \ldots, \bigoplus_{D_{a_k} \in inner_{\leq g-1}(a_k,B)} V(D_{a_k}) \right] \\
&= \bigoplus_{\substack{[a_1,\ldots,a_k] \\ \text{s.t.} \frac{a_1,\ldots,a_k}{x}}} V_{\leq g-1}(a_1,B) \otimes [V_{\leq g-1}(a_2,B), \ldots, V_{\leq g-1}(a_k,B)]
\end{aligned}
$$

Note that if a_i is not in the bucket B then $V_{\leq g-1}(a_i,B) = V(a_i)$, hence $V_{\leq g-1}(a_i,B)$ can be replaced with $K_g(a_i,B)$, completing the proof. □

We will follow a similar strategy for computing the outside values of items that belong to a looping bucket. The only difference is the slight difference in the definition of the generation of of the tree. If $D \in outer(x)$ where x belongs to a looping bucket B, then the generation of D is maximum number of items that could appear in a single path from the root to x, where x is included in the count. Let

$$
Z_{\leq g}(x,B) = \bigoplus_{D \in outer_{\leq g}(x,B)} Z(D)
$$

Theorem B.2. *For items x in a looping bucket B and the generation $g \geq 1$*

$$
\begin{aligned}
Z_{\leq g}(x,B) = \bigoplus_{\substack{j,a_1,..,a_k,b \text{ s.t.} \\ \frac{a_1 \ldots a_k}{b} \text{ and } x=a_j}} &(V(a_1) \otimes_k [I_{a_k}, V(a_{k+1}), \ldots, V(a_n)])^{\pi} \\
&\otimes [(V(a_1), \ldots, V(a_{k-1})] \otimes^* H_g(b,B)
\end{aligned}
$$

Where π is defined as in Theorem 5.6.4 and

$$H_g(b,B) = \begin{cases} Z(b) & \text{if } b \notin B \\ Z_{\leq g-1}(b,B) & \text{if } b \in B \end{cases}$$

Proof.

$$\begin{aligned}
Z_{\leq g}(x,B) &= \bigoplus_{D \in outer_{\leq g}(x,B)} Z(D) \\
&= \bigoplus_{\substack{j,a_1,..,a_k,b \text{ s.t.} \\ \frac{a_1 \ldots a_k}{b} \text{ and } x=a_j}} \; \bigoplus_{D \in outer_{\leq g-1}\left(k, \frac{a_1 \ldots a_n}{b}\right)} Z(D) \\
&= \bigoplus_{\substack{j,a_1,..,a_k,b \text{ s.t.} \\ \frac{a_1 \ldots a_k}{b} \text{ and } x=a_j}} \; \bigoplus_{D_b \in outer_{\leq g-1}(b)} \; \bigoplus_{\substack{D_{a_1} \in inner(a_1),\ldots, \\ D_{a_{k-1}} \in inner(a_{k-1})}} \; \bigoplus_{\substack{D_{a_{k+1}} \in inner(a_{k+1}),\ldots, \\ D_{a_n} \in inner(a_n)}} \\
&\qquad \left(V(D_{a_1}) \otimes_k \left[I_{D_{a_k} \times d_S}, V(D_{a_{k+1}}), \ldots, V(D_{a_n})\right]\right)^{\pi} \\
&\qquad \otimes \left(V(D_{a_2}), \ldots, V(D_{a_{k-1}})\right) \otimes^* Z_{\leq g}(D_b,B) \\
&= \bigoplus_{\substack{j,a_1,..,a_k,b \text{ s.t.} \\ \frac{a_1 \ldots a_k}{b} \text{ and } x=a_j}} \left(\bigoplus_{\substack{D_{a_1} \in \\ inner(a_1)}} V(D_{a_1}) \otimes_k \left[I_{D_{a_k}}, \bigoplus_{\substack{D_{a_{k+1}} \in \\ inner(a_{k+1})}} V(D_{a_{k+1}}), \ldots, \bigoplus_{\substack{D_{a_n} \in \\ inner(a_n)}} V(D_{a_n}) \right] \right)^{\pi} \\
&\qquad \otimes \left(\bigoplus_{D_{a_2} \in inner(a_2)} V(D_{a_2}), \ldots, \bigoplus_{D_{a_{k-1}} \in inner(a_{k-1})} V(D_{a_{k-1}}) \right) \\
&\qquad \otimes^* \bigoplus_{D_b \in outer_{\leq g-1}(b)} Z_{\leq g-1}(D_b,B) \\
&= \bigoplus_{\substack{j,a_1,..,a_k,b \text{ s.t.} \\ \frac{a_1 \ldots a_k}{b} \text{ and } x=a_j}} (V(a_1) \otimes_k [I_{a_k}, V(a_{k+1}), \ldots, V(a_n)])^{\pi} \\
&\qquad \otimes [(V(a_2), \ldots, V(a_{k-1})] \otimes^* Z_{\leq g-1}(b,B)
\end{aligned}$$

Like the inner case, note that for an item b not in the looping bucket b, $Z_{\leq g-1}(b,B) = Z(b)$, hence we can replace $Z_{\leq g-1}(b,B)$ with $H_g(b,B)$, completing the proof. □

Bibliography

Evrim Acar, Morten Arendt Rasmussen, Francesco Savorani, Tormod Næs, and Rasmus Bro. 2013. Understanding data fusion within the framework of coupled matrix and tensor factorizations. *Chemometrics and Intelligent Laboratory Systems*, 129:53–63.

Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Laurent Vermue, Mikhail Galkin, Sahand Sharifzadeh, Asja Fischer, Volker Tresp, and Jens Lehmann. 2020. Bringing light into the dark: A large-scale evaluation of knowledge graph embedding models under a unified framework. *arXiv preprint arXiv:2006.13365*.

Bo An, Bo Chen, Xianpei Han, and Le Sun. 2018. Accurate text-enhanced knowledge graph representation learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 745–755.

Anima Anandkumar, Rong Ge, Daniel Hsu, Sham M Kakade, and Matus Telgarsky. 2014. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15:2773–2832.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: a nucleus for a web of open data. In *The Semantic Web*, pages 722–735. Springer-Verlag.

Raphaël Bailly, François Denis, and Liva Ralaivola. 2009. Grammatical inference as a principal component analysis problem. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 33–40.

Ivana Balazevic, Carl Allen, and Timothy Hospedales. 2019. Tucker: Tensor factorization for knowledge graph completion. In *Proceedings of the 2019 Conference on*

Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 5188–5197.

Ivana Balažević, Carl Allen, and Timothy M Hospedales. 2019. Hypernetwork knowledge graph embeddings. In *International Conference on Artificial Neural Networks*, pages 553–565.

Steven Bird. 2006. NLTK: the natural language toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 69–72.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. 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.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, pages 1–9.

Guillaume Bouchard, Sameer Singh, and Theo Trouillon. 2015. On approximate reasoning capabilities of low-rank vector spaces. *AAAI Spring Syposium on Knowledge Representation and Reasoning (KRR): Integrating Symbolic and Neural Approaches*.

Pierre Boullier. 2000. Range concatenation grammars. In *Proceedings of the Sixth International Workshop on Parsing Technologies*, pages 53–64.

Matthias Bröcheler, Lilyana Mihalkova, and Lise Getoor. 2010. Probabilistic similarity logic. In *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, pages 73–82.

Razvan Bunescu and Raymond Mooney. 2007. Learning to extract relations from the web using minimal supervision. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 576–583.

Liwei Cai and William Yang Wang. 2018. KBGAN: Adversarial learning for knowl-

edge graph embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 1470–1480.

Jamie Callan, Mark Hoy, Changkuk Yoo, and Le Zhao. 2009. Clueweb09. http://www.lemurproject.org/clueweb09/index.php.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, pages 1306–1313.

Kai-Wei Chang, Wen-tau Yih, Bishan Yang, and Christopher Meek. 2014. Typed tensor decomposition of knowledge bases for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1568–1579.

Stephen Clark, Bob Coecke, and Mehrnoosh Sadrzadeh. 2008. A compositional distributional model of meaning. In *Proceedings of the Second Quantum Interaction Symposium (QI-2008)*, pages 133–140.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. *arXiv preprint arXiv:1003.4394*.

Shay B Cohen, Robert J Simmons, and Noah A Smith. 2008. Dynamic programming algorithms as products of weighted logic programs. In *Proceedings of the International Conference on Logic Programming*, pages 114–129.

Shay B Cohen, Karl Stratos, Michael Collins, Dean P Foster, and Lyle Ungar. 2013. Experiments with spectral learning of latent-variable PCFGs. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 148–157.

Shay B Cohen, Karl Stratos, Michael Collins, Dean P Foster, and Lyle Ungar. 2014. Spectral learning of latent-variable PCFGs: Algorithms and sample complexity. *The Journal of Machine Learning Research*, 15(1):2399–2449.

William Cohen, Fan Yang, and Kathryn Rivard Mazaitis. 2020. Tensorlog: A proba-

bilistic database implemented using deep-learning infrastructure. *Journal of Artificial Intelligence Research*, 67:285–325.

Pierre Comon, Xavier Luciani, and André LF De Almeida. 2009. Tensor decompositions, alternating least squares and other tales. *Journal of Chemometrics*, 23(7-8):393–405.

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2017. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 132–141.

Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. 2007. ProbLog: a probabilistic prolog and its application in link discovery. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 2468–2473.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. 2018. Convolutional 2D knowledge graph embeddings. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 1811–1818.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Boyang Ding, Quan Wang, Bin Wang, and Li Guo. 2018. Improving knowledge graph embedding using simple constraints. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 110–121.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy,

Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault. In *Proceedings of the Twentieth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 601–610.

John Duchi, Elad Hazan, and Yoram Singer. 2001. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Lisa Ehrlinger and Wolfram Wöß. 2016. Towards a Definition of Knowledge Graphs. *SEMANTiCS*, 48:1–4.

Jason Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 1–8.

Jason Eisner, Eric Goldlust, and Noah A Smith. 2005. Compiling comp ling: Weighted dynamic programming and the Dyna language. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 281–290.

Norbert Fuhr. 2000. Probabilistic Datalog: Implementing logical information retrieval for advanced applications. *Journal of the American Society for Information Science*, 51(2):95–110.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. FACC1: Freebase annotation of ClueWeb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0).

Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M Suchanek. 2015. Fast rule mining in ontological knowledge bases with AMIE+. *The International Journal on Very Large Data Bases*, 24(6):707–730.

Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. 2013. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 413–422.

Alberto Garcia-Durán, Antoine Bordes, Nicolas Usunier, and Yves Grandvalet. 2016. Combining two and three-way embedding models for link prediction in knowledge bases. *Journal of Artificial Intelligence Research*, 55:715–742.

Alberto Garcia-Duran, Sebastijan Dumančić, and Mathias Niepert. 2018. Learning sequence encoders for temporal knowledge graph completion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4816–4821.

Kilian Gebhardt. 2018. Generic refinement of expressive grammar formalisms with an application to discontinuous constituent parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3049–3063.

Kevin Gimpel and Noah A Smith. 2009. Cube summing, approximate inference with non-local features, and dynamic programming without semirings. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 318–326.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256.

Fréderic Godin, Anjishnu Kumar, and Arpit Mittal. 2019. Learning when not to answer: a ternary reward structure for reinforcement learning based question answering. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers)*, pages 122–129.

Rishab Goel, Seyed Mehran Kazemi, Marcus Brubaker, and Pascal Poupart. 2020. Diachronic embedding for temporal knowledge graph completion. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, volume 34, pages 3988–3995.

Joshua Goodman. 1999. Semiring parsing. *Computational Linguistics*, 25(4):573–606.

Joshua T Goodman. 1998. *Parsing Inside-Out*. Ph.D. thesis, Harvard University Cambridge, Massachusetts.

Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2016. Jointly embedding knowledge graphs and logical rules. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 192–202.

Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2018. Knowledge graph

embedding with iterative guidance from soft rules. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, volume 32, pages 4816–4823.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 318–327.

Xu Han, Zhiyuan Liu, and Maosong Sun. 2018. Neural knowledge acquisition via mutual attention between knowledge graph and text. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, volume 32, pages 4832–4839.

Ming He, Xiangkun Du, and Bo Wang. 2019. Representation learning of knowledge graphs via fine-grained relation description combinations. *IEEE Access*, 7:26466–26473.

Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. 2015. Learning to represent knowledge graphs with gaussian embedding. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, pages 623–632.

Christopher J. Hillar and Lek-Heng Lim. 2013. Most tensor problems are NP-hard. *Journal of the ACM*, 60(6):1–39.

Mohammad Javad Hosseini, Shay B Cohen, Mark Johnson, and Mark Steedman. 2019. Duality of link prediction and entailment graph induction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4736–4746.

Daniel Hsu, Sham M Kakade, and Tong Zhang. 2012. A spectral algorithm for learning hidden markov models. *Journal of Computer and System Sciences*, 78(5):1460–1480.

Prachi Jain, Pankaj Kumar, Mausam, and Soumen Chakrabarti. 2018a. Type-sensitive knowledge base inference without explicit type supervision. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 75–80.

Prachi Jain, Shikhar Murty, Mausam Mausam, and Soumen Chakrabarti. 2018b. Mitigating the effect of out-of-vocabulary entity pairs in matrix factorization for kb inference. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4122–4129.

Prachi Jain, Sushant Rathi, Mausam, and Soumen Chakrabarti. 2020. Temporal knowledge base completion: New algorithms and evaluation protocols. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 3733–3747.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 687–696.

Guoliang Ji, Kang Liu, Shizhu He, and Jun Zhao. 2016. Knowledge graph completion with adaptive sparse transfer matrix. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 985–991.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Maximilian Nickel, and Tomás Mikolov. 2017. Fast linear model for knowledge graph embeddings. In *6th Workshop on Automated Knowledge Base Construction*.

Rudolf Kadlec, Ondřej Bajgar, and Jan Kleindienst. 2017. Knowledge base completion: Baselines strike back. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 69–74.

Seyed Mehran Kazemi and David Poole. 2018. Simple embedding for link prediction in knowledge graphs. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 4289–4300.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, Conference Track Proceedings*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, Conference Track Proceedings*.

Tamara G. Kolda and Brett W. Bader. 2009. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500.

Bhushan Kotnis and Vivi Nastase. 2017. Analysis of the impact of negative sampling on link prediction in knowledge graphs. *arXiv preprint arXiv:1708.06816*.

Denis Krompaβ, Stephan Baier, and Volker Tresp. 2015. Type-constrained representation learning in knowledge graphs. In *Proceedings of the 14th International Conference on The Semantic Web*, volume 9366, pages 640–655.

Werner Kuich. 1997. Semirings and formal power series: Their relevance to formal languages and automata. In Rozenberg Grzegorz and Arto Salomaa, editors, *Handbook of Formal Languages*, volume 1, pages 609–677. Springer.

Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. 2018. Canonical tensor decomposition for knowledge base completion. In *Proceedings of the 35th International Conference on Machine Learning*, pages 2863–2872.

Niels Landwehr, Kristine Kersting, and Luc De Raedt. 2007. Integrating naive Bayes and FOIL. *Journal of Machine Learning Research*, 8:481–507.

Niels Landwehr, Andrea Passerini, Luc De Raedt, and Paolo Frasconi. 2010. Fast learning of relational kernels. *Machine Learning*, 78(3):305–342.

Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 529–539.

Zhifei Li and Jason Eisner. 2009. First and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 40–51.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-hop knowledge graph reasoning with reward shaping. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3243–3253.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015a. Modeling relation paths for representation learning of knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 705–714.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015b. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence*, pages 2181–2187.

Hanxiao Liu, Yuexin Wu, and Yiming Yang. 2017. Analogical inference for multi-

relational embeddings. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 2168–2178.

Adam Lopez. 2009. Translation as weighted deduction. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 532–540.

Farzaneh Mahdisoltani, Joanna Biega, and Fabian M Suchanek. 2013. YAGO3: A knowledge base from multilingual Wikipedias. In *Proceedings of the 7th Biannial Conference on Innovative Data Systems Research*.

Gengchen Mai, Krzysztof Janowicz, Ling Cai, Rui Zhu, Blake Regalia, Bo Yan, Meilin Shi, and Ni Lao. 2020. SE-KGE: A location-aware knowledge graph embedding model for geographic question answering and spatial semantic lifting. *Transactions in GIS*, 24(3):623–655.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 75–82.

Hongyuan Mei, Guanghui Qin, Minjie Xu, and Jason Eisner. 2020. Neural Datalog through time: Informed temporal modeling via logical specification. In *International Conference on Machine Learning*, volume 119, pages 6808–6819.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, pages 3111–3119.

George A. Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Pasquale Minervini, Luca Costabello, Emir Muñoz, V´Nováček, and Pierre-Yves Vandenbussche. 2017. Regularizing knowledge graph embeddings via equivalence and inversion axioms. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 668–683.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of*

the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, pages 1003–1011.

Hatem Mousselly-Sergieh, Teresa Botschen, Iryna Gurevych, and Stefan Roth. 2018. A multimodal translation-based approach for knowledge graph representation learning. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 225–234.

Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. 2019. Learning attention-based embeddings for relation prediction in knowledge graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4710–4723.

Mark-Jan Nederhof. 2003. Weighted deductive parsing and Knuth's algorithm. *Computational Linguistics*, 29(1):135–143.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 156–166.

Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2018. A novel embedding model for knowledge base completion based on convolutional neural network. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 327–333.

Dat Quoc Nguyen. 2020. A survey of embedding models of entities and relationships for knowledge graph completion. In *Proceedings of the Graph-based Methods for Natural Language Processing (TextGraphs)*, pages 1–14.

Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. 2016. STransE: a novel embedding model of entities and relationships in knowledge bases. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 460–466.

Maximilian Nickel. 2013. *Tensor Factorization for Relational Learning*. Ph.D. thesis, Ludwig-Maximilians-Universität München.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016a. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104:11–33.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016b. Holographic embeddings of knowledge graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 1955–1961.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2012. Factorizing YAGO: scalable machine learning for linked data. In *Proceedings of the 21st International Conference on World Wide Web*, pages 271–280.

Nils J Nilsson. 2009. *The quest for artificial intelligence*. Cambridge University Press.

Guanglin Niu, Yongfei Zhang, Bo Li, Peng Cui, Si Liu, Jingyang Li, and Xiaowei Zhang. 2020. Rule-guided compositional representation learning on knowledge graphs. In *Proceedings of the Thirty-Fouth AAAI Conference on Artificial Intelligence*, volume 34, pages 2950–2958.

Thomas Pellissier Tanon, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher. 2016. From Freebase to Wikidata: The great migration. In *Proceedings of the 25th International Conference on World Wide Web*, pages 1419–1428.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Languages Processing*, pages 1532–1543.

Fernando C N Pereira and David H D Warren. 1983. Parsing as deduction. In *Proceedings of the 21st Annual Meeting on Association for Computational Linguistics*, pages 137–144.

Pouya Pezeshkpour, Liyan Chen, and Sameer Singh. 2018. Embedding multimodal relational data for knowledge base completion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3208–3218.

Saige Qin, Guanjun Rao, Chenzhong Bin, Liang Chang, Tianlong Gu, and Wen Xuan. 2019. Knowledge graph embedding based on adaptive negative sampling. In *International Conference of Pioneering Computer Scientists, Engineers and Educators*, pages 551–563.

Peiyuan Qiu, Jialiang Gao, Li Yu, and Feng Lu. 2019. Knowledge embedding with

geospatial distance restriction for geographic knowledge graph completion. *ISPRS International Journal of Geo-Information*, 8(6):254.

J Ross Quinlan. 1990. Learning logical definitions from relations. *Machine learning*, 5(3):239–266.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine learning*, 62(1-2):107–136.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84.

Tim Rocktäschel and Sebastian Riedel. 2017. End-to-end differentiable proving. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 3791–3803.

Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. 2020. You CAN teach an old dog new tricks! on training knowledge graph embeddings. In *International Conference on Learning Representations*.

Gilbert Ryle. 2009. *The concept of mind*. Routledge.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *Proceedings of the European Semantic Web Conference*, pages 593–607.

William Beecher Scoville and Brenda Milner. 1957. Loss of recent memory after bilateral hippocampal lesions. *Journal of Neurology, Neurosurgery, and Psychiatry*, 20(1):11.

Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. 2019. End-to-end structure-aware convolutional networks for knowledge base completion. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, volume 33, pages 3060–3067.

Stuart M Shieber, Yves Schabes, and Fernando C N Pereira. 1995. Principles and implementation of deductive parsing. *The Journal of Logic Programming*, 24(1-2):3–36.

Nicholas D. Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E.

Papalexakis, and Christos Faloutsos. 2017. Tensor decomposition for signal processing and machine learning. *IEEE Transactions on Signal Processing*, 65(13):3551–3582.

Klaas Sikkel. 1998. Parsing schemata and correctness of parsing algorithms. *Theoretical Computer Science*, 199(1-2):87–103.

Ajit P Singh and Geoffrey J Gordon. 2008. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 650–658.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, pages 926–934.

Gustav Sourek, Vojtech Aschenbrenner, Filip Zelezny, Steven Schockaert, and Ondrej Kuzelka. 2018. Lifted relational neural networks: Efficient learning of latent relational structures. *Journal of Artificial Intelligence Research*, 62:69–100.

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

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509.

Kristina Toutanova, Xi Victoria Lin, Wen-tau Yih, Hoifung Poon, and Chris Quirk. 2016. Compositional learning of embeddings for relation paths in knowledge base and text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1434–1444.

Théo Trouillon and Maximilian Nickel. 2017. Complex and holographic embeddings of knowledge graphs: A comparison.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Eric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 2071–2080.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, Nilesh Agrawal, and Partha Talukdar. 2020. InteractE: Improving convolution-based knowledge graph embeddings by increasing feature interactions. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, volume 34, pages 3009–3016.

Neil Veira, Brian Keng, Kanchana Padmanabhan, and Andreas G Veneris. 2019. Unsupervised embedding enhancements of knowledge graphs using textual associations. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 5218–5225.

Mengya Wang, Erhu Rong, Hankui Zhuo, and Huiling Zhu. 2018a. Embedding knowledge graphs based on transitivity and asymmetry of rules. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 141–153.

Peifeng Wang, Shuangyin Li, and Rong Pan. 2018b. Incorporating GAN for negative sampling in knowledge representation learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, volume 32.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743.

Quan Wang, Bin Wang, and Li Guo. 2015a. Knowledge base completion using embeddings and rules. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 1859–1865.

William Yang Wang and William W Cohen. 2016. Learning first-order logic embeddings via matrix factorization. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2132–2138.

William Yang Wang, Kathryn Mazaitis, Ni Lao, and William W Cohen. 2015b. Efficient inference and learning in a large knowledge base. *Machine Learning*, 100(1):101–126.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014a. Knowledge graph

and text jointly embedding. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1591–1601.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014b. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of theTwenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119.

Zhigang Wang and Juanzi Li. 2016. Text-enhanced representation learning for knowledge graph. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 1293–1299.

Zikang Wang, Linjing Li, Qiudan Li, and Daniel Zeng. 2019. Multimodal data enhanced representation learning for knowledge graphs. In *Proceedings of the 2019 International Joint Conference on Neural Networks*, pages 1–8.

Zhuoyu Wei, Jun Zhao, Kang Liu, Zhenyu Qi, Zhengya Sun, and Guanhua Tian. 2015. Large-scale knowledge base completion: Inferring via grounding network sampling over selected instances. In *Proceedings of the Twenty-Fourth International ACM Conference on Information and Knowledge Management*, pages 1331–1340.

Jiawei Wu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2016. Knowledge representation via joint learning of sequential text and knowledge graphs. *arXiv preprint arXiv:1609.07075*.

Yanrong Wu and Zhichun Wang. 2018. Knowledge graph embedding with numeric attributes of entities. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 132–136.

Han Xiao, Minlie Huang, Lian Meng, and Xiaoyan Zhu. 2017. Ssp: semantic space projection for knowledge graph embedding with text descriptions. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3104–3110.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016. From one point to a manifold: knowledge graph embedding for precise link prediction. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 1315–1321.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016a. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, volume 30, pages 2659–2665.

Ruobing Xie, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2017. Image-embodied knowledge representation learning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 3140–3146.

Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2016b. Representation learning of knowledge graphs with hierarchical types. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2965–2971.

Jiacheng Xu, Xipeng Qiu, Kan Chen, and Xuanjing Huang. 2017. Knowledge graph representation with jointly structural and textual encoding. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 1318–1324.

Peng Xu and Denilson Barbosa. 2018. Investigations on knowledge base embedding for relation prediction and extraction. *arXiv preprint arXiv:1802.02114*.

Ikuya Yamada, Akari Asai, Jin Sakuma, Hiroyuki Shindo, Hideaki Takeda, Yoshiyasu Takefuji, and Yuji Matsumoto. 2020. Wikipedia2vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from wikipedia. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 23–30.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *3rd International Conference on Learning Representations, Conference Track Proceedings*.

Fan Yang, Zhilin Yang, and William W Cohen. 2017. Differentiable learning of logical rules for knowledge base reasoning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 2316–2325.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. KG-BERT: BERT for knowledge graph completion. *arXiv preprint arXiv:1909.03193*.

Richong Zhang, Fanshuang Kong, Chenyue Wang, and Yongyi Mao. 2018. Embedding of hierarchically typed knowledge bases. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, volume 32, pages 2046–2053.

Wen Zhang, Jiaoyan Chen, Bibek Paudel, Hai Zhu, Liang Wang, Wei Zhang, Abraham Bernstein, and Huajun Chen. 2019a. Iteratively learning embeddings and rules for

knowledge graph reasoning. In *Proceedings of the 2019 World Wide Web Conference*, pages 2366–2377.

Yongqi Zhang, Quanming Yao, Yingxia Shao, and Lei Chen. 2019b. NSCaching: simple and efficient negative sampling for knowledge graph embedding. In *IEEE Thirty-Fifth International Conference on Data Engineering*, pages 614–625.

Huaping Zhong, Jianwen Zhang, Zhen Wang, Hai Wan, and Zheng Chen. 2015. Aligning knowledge and text embeddings by entity descriptions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 267–272.

W Zhou, S Wang, and C Jiang. 2019. Knowledge graph embedding with interactive guidance from entity descriptions. *IEEE Access*, 7:156686–156693.