Complex-Valued Embedding Models for Knowledge Graphs

Théo Trouillon

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE LA COMMUNAUTÉ UNIVERSITÉ GRENOBLE ALPES

Spécialité : Mathématiques et Informatique

Arrêté ministériel : 25 mai 2016

Présentée par

Théo TROUILLON

Thèse dirigée par **Eric GAUSSIER**, Professeur, Université Grenoble Alpes,

préparée au sein du **Laboratoire d'Informatique de Grenoble** dans **l'École Doctorale Mathématiques, Sciences et technologies de l'information, Informatique**

Modèles d'embeddings à valeurs complexes pour les graphes de connaissances

Complex-Valued Embedding Models for Knowledge Graphs

Thèse soutenue publiquement le **29 septembre 2017**, devant le jury composé de :

Monsieur MOHAMED NADIF
PROFESSEUR, UNIVERSITE PARIS 5, Président du Jury
Monsieur YVES GRANDVALET
DIRECTEUR DE RECHERCHE, CNRS NORD-PAS DE CALAIS ET PICARDIE, Rapporteur
Monsieur VOLKER TRESP
PROFESSEUR, UNIVERSITE LOUIS ET- MAXIMILIEN DE MUNICH, Rapporteur
Monsieur ANDREW MCCALLUM
PROFESSEUR, UNIV. DU MASSACHUSETTS A AMHERST - USA, Examinateur
Monsieur CHRISTOPHER DANCE
CHERCHEUR, XEROX RESEARCH CENTRE EUROPE - MEYLAN, Examinateur
Monsieur ERIC GAUSSIER
PROFESSEUR, UNIVERSITE GRENOBLE ALPES, Examinateur

UNIVERSITÉ GRENOBLE ALPES

Complex-Valued Embedding Models for Knowledge Graphs

by

Théo Trouillon

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the
Laboratoire d'Informatique de Grenoble
Xerox Research Centre Europe

December 2017

"Good tests kill flawed theories; we remain alive to guess again."

Karl Popper

"Un homme doit savoir s'assoir sur un banc, manger du fromage, et être heureux."

Auteur Inconnu

UNIVERSITÉ GRENOBLE ALPES

Abstract

Laboratoire d'Informatique de Grenoble
Xerox Research Centre Europe

Doctor of Philosophy

Complex-Valued Embedding Models for Knowledge Graphs

by Théo Trouillon

The explosion of widely available relational data in the form of knowledge graphs enabled many applications, including automated personal agents, recommender systems and enhanced web search results. The very large size and notorious incompleteness of these databases calls for automatic knowledge graph completion methods to make these applications viable. Knowledge graph completion, also known as link prediction, deals with automatically understanding the structure of large knowledge graphs—labeled directed graphs—to predict missing entries—labeled edges. An increasingly popular approach consists in representing a knowledge graph as a 3^{rd}-order tensor, and using tensor factorization methods to predict their missing entries.

State-of-the-art factorization models propose different trade-offs between modeling expressiveness, time and space complexity, and generalization abilities. We introduce a new model, ComplEx—for Complex Embeddings—to reconcile expressiveness, complexity and generalization through the use of complex-valued factorization. We corroborate our approach theoretically and show that all possible knowledge graphs can be exactly decomposed by the proposed model. Our approach based on complex embeddings is arguably simple, as it only involves a complex-valued trilinear product, whereas other methods resort to more and more complicated composition functions to increase their expressiveness. The proposed ComplEx model is scalable to large data sets as it remains linear in both space and time, while consistently outperforming alternative approaches on standard link-prediction benchmarks.[1] We also demonstrate its ability to learn useful vectorial representations for other tasks, by enhancing word embeddings that improve performances on the natural language problem of entailment recognition between pair of sentences.

In the last part of this thesis, we explore factorization models ability to learn relational patterns from observed data. By their vectorial nature, it is not only hard to interpret why this class of models works so well, but also to understand where they fail and how they might be improved. We conduct an experimental survey of state-of-the-art models, not towards a purely comparative end, but as a means to get insight about their inductive abilities. To assess the strengths and weaknesses of each model, we create simple tasks that exhibit first, atomic properties of knowledge graph relations, and then, common inter-relational inference through synthetic genealogies. Based on these experimental results, we propose new research directions to improve on existing models, including ComplEx.

[1] Code is available at: `https://github.com/ttrouill/complex`

UNIVERSITÉ GRENOBLE ALPES

Résumé

Laboratoire d'Informatique de Grenoble
Xerox Research Centre Europe

Thèse de Doctorat

Modèles d'Embeddings à Valeurs Complexes pour les Graphes de Connaissances

by Théo Trouillon

L'explosion de données relationnelles disponibles sous la forme de graphes de connaissances a permis le développement de multiples applications, dont les agents personnels automatisés, les systèmes de recommandation et l'amélioration des résultats de recherche en ligne. La grande taille et l'incomplétude de ces bases de données nécessite le développement de méthodes de complétion automatiques pour rendre ces applications viables. La complétion de graphes de connaissances, aussi appelée prédiction de liens, se doit de comprendre automatiquement la structure de larges graphes de connaissances (graphes dirigés labellisés) pour prédire les entrées manquantes (les arêtes labellisées). Une approche populaire consiste à représenter un graphe de connaissances comme un tenseur d'ordre 3, et à utiliser des méthodes de décomposition de tenseur pour prédire leurs entrées manquantes.

Les modèles de factorisation existants proposent différents compromis entre leur expressivité, leur complexité en temps et en espace, et leur capacités de généralisation. Nous proposons un nouveau modèle appelé ComplEx, pour "Complex Embeddings", pour réconcilier expressivité, complexité et généralisation par l'utilisation d'une factorisation en nombre complexes. Nous corroborons notre approche théoriquement en montrant que tous les graphes de connaissances possibles peuvent être exactement décomposés par le modèle proposé. Notre approche, basée sur des embeddings complexes reste simple, car n'impliquant qu'un produit trilinéaire complexe, là où d'autres méthodes recourent à des fonctions de composition de plus en plus sophistiquées pour accroître leur expressivité. Le modèle proposé ayant une complexité linéaire en temps et en espace est passable à l'échelle, tout en dépassant les scores de prédiction des approches existantes sur les jeux de données de référence pour la prédiction de liens.[2] Nous démontrons aussi la capacité de ComplEx à apprendre des représentations vectorielles utiles pour d'autres tâches, en enrichissant des embeddings de mots, qui améliorent les prédictions sur le problème de reconnaissance d'implication entre paires de phrases.

Dans la dernière partie de cette thèse, nous explorons les capacités des modèles de factorisation à apprendre les structures relationnelles à partir d'observations. De part leur nature vectorielle, il est non seulement difficile de comprendre pourquoi cette classe de modèles fonctionne aussi bien, mais aussi où ils échouent et comment ils peuvent être améliorés. Nous conduisons une étude expérimentale de modèles de l'état de l'art, non pas simplement pour les comparer, mais pour comprendre leurs capacités d'induction. Pour évaluer les forces et faiblesses de chaque modèle, nous créons d'abord des tâches simples représentant des propriétés atomiques des propriétés des relations des graphes de connaissances ; puis des tâches représentant des inférences multi-relationnelles communes au travers de généalogies synthétisées. À partir de ces résultats expérimentaux, nous

[2]Le code est mis à disposition: `https://github.com/ttrouill/complex`

proposons de nouvelles directions de recherche pour améliorer les modèles existants, y compris ComplEx.

Acknowledgements

My foremost gratitude goes to my directors, whose diversity of mind completed each other, and yielded this fertile collaboration. To Éric Gaussier, who followed me during these three years, and whose continued guidance and patient explanations carried me through this journey. To Guillaume Bouchard and his inexhaustible scientific creativity and continued enthusiasm, who pushed me to explore always new ideas. To Christopher Dance, whose engagement and scientific exigence in every single aspect made the researcher I am today.

I would like to thank Sebastian Riedel and the whole Machine Reading Group at UCL, with which I collaborated successfully in diverse aspects of my thesis, for welcoming me during these two months; with a special mention to Johannes Welbl and Pontus Stenetorp for the several fruitful conversations we had.

I sincerely thank Maximilian Nickel, who transmitted me all his implementation tips and tricks when we first met, and for the many enlightening exchanges we had since then.

Many people contributed to the concretization of the ideas presented here through their instructive conversation, among them my very talented colleagues at Naver Labs Europe—formerly Xerox Research Centre Europe—and in particular Ariadna Quattoni, Stéphane Clinchant, Jean-Marc Andreoli, Julien Perez, Sofia Michel, and Diana Popa; but also Pierre Comon, Massimiliano Pontil, Léo Hubert and Alejandro Blumentals.

I am grateful to all my friends who endured listening about my academic troubles during all these years, Julien, Antonin, and my roomates, with a special mention to Alice who accomplished this on a daily basis; and to Florence for her support during my defence preparation.

Needless to say that this reality has been made possible by my family, my parents who opened us all opportunities, my brother Julian and my sister Léa who are part of the person I am.

This thesis was supported in part by the Association Nationale de la Recherche et de la Technologie through the CIFRE grant 2014/0121, and in part by the former Xerox Research Centre Europe.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Web-scale knowledge graphs provide a structured representation of world knowledge, with projects such as the Google Knowledge Graph [Google Blog, 2012]. They enable a wide range of applications including recommender systems [Koren, 2008], question answering [Bordes et al., 2014b], automated personal agents [Ma et al., 2015] and enhanced search results [Google Blog, 2012] (Figure 1.1). The incompleteness of these knowledge graphs—also called knowledge bases—has stimulated research into predicting missing entries, a task known as *link prediction* or *knowledge-graph completion*. The need for high quality predictions made it progressively become the main problem in statistical relational learning [Getoor and Taskar, 2007], a research field involving the study of relational-data representation and modeling.

Knowledge graphs were born with the advent of the Semantic Web, pushed by the World Wide Web Consortium (W3C) recommendations. Namely, the Resource Description Framework (RDF) standard, that underlies knowledge graphs' data representation, provides for the first time a common framework across all connected information systems to share their data under the same paradigm. Being more expressive than classical relational databases, all existing relational data can be translated into RDF knowledge graphs [Sahoo et al., 2009]. Through these data-representation standards glimpses the hope for a future, freely accessible, global database storing all of humanity's knowledge, that could be automatically completed by reliable link-prediction methods.

In artificial intelligence, many tasks require what is called *commonsense knowledge* to be solved perfectly. The ensemble of facts and information about the world that any person is expected to know constitutes the commonsense knowledge. Such tasks are considered AI-complete, that is, they are considered as hard as developing an artificial general intelligence (AGI). These tasks include natural language understanding and image understanding [Yampolskiy, 2012]. The existence of such a complete knowledge

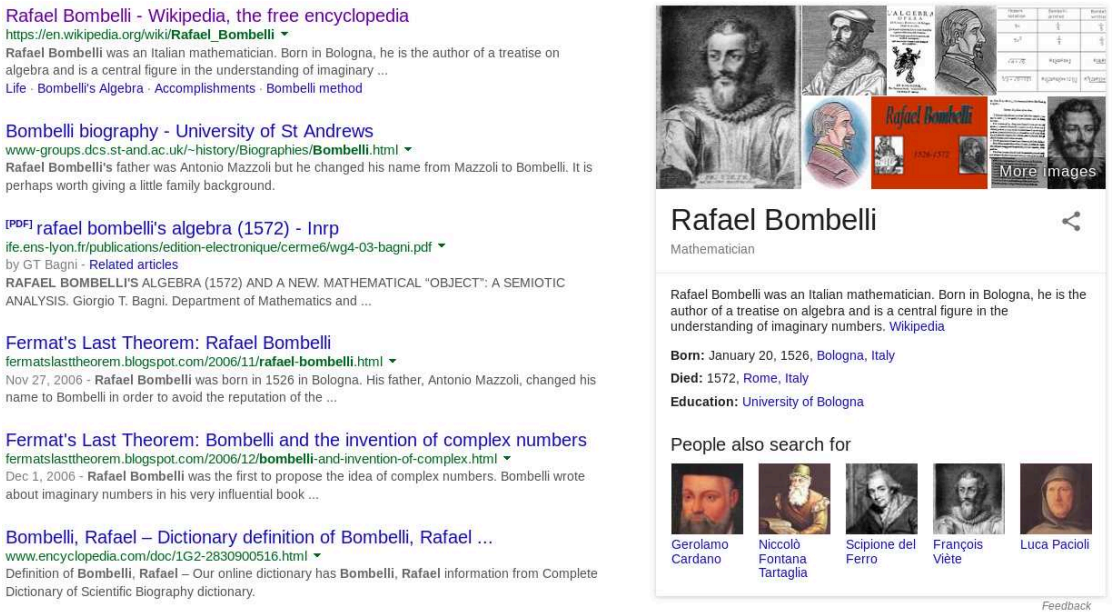


FIGURE 1.1: Example of a Google search result for the query "Rafael Bombelli", enhanced by the Google Knowledge Graph (right-side block), that provides extra-information such as birth and death place and time, and education.

base of commonsense knowledge, as pursued by the the Cyc project [Lenat, 1995], would help solve hard artificial intelligence problems, and open a path to AGI.

Formally, knowledge graphs express data as a directed graph with labeled edges (relations) between pairs of nodes (entities): relations are binary predicates. Natural redundancies between the recorded relations often make it possible to fill in the missing entries of a knowledge graph. As an example, the relation `livesInCountry` could not be recorded for all entities, but it can be inferred if the relation `livesInCity` is known. The goal of link prediction is the automatic discovery of such regularities. However, inference between relations is often non-deterministic: the combination of the two facts `livesInCity(John,Athens)` and `isInCountry(Athens,Greece)` does not always imply the fact `hasNationality(John,Greece)`. Hence, it is natural to handle inference probabilistically, and jointly with other facts involving these relations and entities. To this end, an increasingly popular method is to state the knowledge graph completion task as a 3D binary tensor completion problem, where each tensor slice is the adjacency matrix of one relation in the knowledge graph, and compute a decomposition of this partially-observed tensor from which its missing entries can be completed.

Decomposition models, also known as factorization models, or latent factor models, or low-rank embedding models; were popularized by the Netflix challenge [Koren et al., 2009]. A partially-observed matrix or tensor is decomposed into a product of embedding matrices with much smaller dimensions, resulting in fixed-dimensional vector representations for each entity and relation in the graph, that allow completion of the missing entries. For a

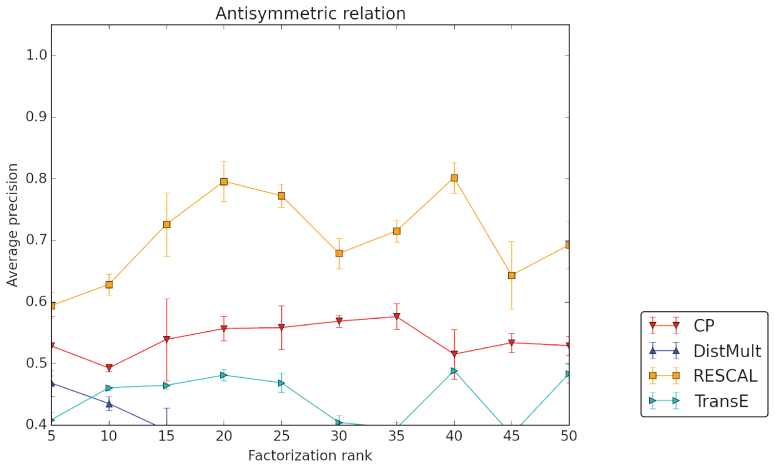


FIGURE 1.2: Cross-validated average precision on a synthetic antisymmetric relation with 30 entities, for each rank ranging from 5 to 50 on different state-of-the-art models. RESCAL is working best but does not exceed 0.8 of average precision and has quadratic complexity. See the full experiment description in Section 4.1.2.

given fact *r(s,o)* in which the subject entity s is linked to the object entity o through the relation r, a score for the fact can be recovered as a multilinear product between the embedding vectors of s, r and o, or through more sophisticated scoring functions [Nickel et al., 2016a].

Binary relations in knowledge graphs exhibit various types of patterns: hierarchies and compositions like `fatherOf`, `olderThan` or `isPartOf`, with strict/non-strict orders or preorders, and equivalence relations like `isSimilarTo`. These characteristics map to different combinations of the following properties: reflexivity/irreflexivity, symmetry/antisymmetry and transitivity. As described in Bordes et al. [2013a], a relational model should (i) be able to learn all combinations of such properties, and (ii) be linear in both time and memory in order to scale to the size of present-day knowledge graphs, and keep up with their growth.

A natural way to handle any possible set of relations is to use the classic canonical polyadic (CP) tensor decomposition [Hitchcock, 1927], which yields two different embeddings for each entity and thus poor generalization performance as shown in Chapter 4. With unique entity embeddings, multilinear products scale well and can naturally handle both symmetry and (ir)reflexivity of relations. However, dealing with antisymmetric and more generally asymmetric relations has so far almost always implied scoring functions with superlinear time and space complexity [Nickel et al., 2011; Socher et al., 2013], making models prone to overfitting and not scalable.

Finding the best trade-off between expressiveness, generalization and complexity is the keystone of embedding models. Following an empirical approach to the problem, we

designed different synthetic tasks that each targets different types of inference abilities—among them learning the basic binary-relation properties. From the observation that no existing factorization model could correctly learn an antisymmetric relation, as shown in Figure 1.2, we explore matrix and tensor decompositions in the *complex* space. Indeed, antisymmetric—or skew-symmetric—matrices are known to have complex eigenvalues [Horn and Johnson, 2012]. Through the use of complex linear algebra, we aimed at:

1. Correctly modeling all basic properties of binary relations.
2. Building a scoring function with linear time and space complexity.
3. Ensuring good generalization by keeping unique representations of entities.

Structure of the Thesis

The resulting model, based on unitary-diagonalization properties, is presented in Chapter 3. We discuss its existence and rank bounds first in the single-relation case, and then extend it to the multi-relational, tensor case. We present a stochastic gradient algorithm to learn the decomposition of partially-observed tensors. Experimental results with this model, and its different applications are described in Chapter 4. We first assess its ability to model jointly symmetric and antisymmetric relations on synthetic data, and then compare it to state-of-the-art models on established link-prediction benchmarks. We also show the flexibility of the knowledge graph decomposition approach to learn reusable vectorial representations of entities, by learning word embeddings that improve on entailment recognition. Finally, we conduct an experimental survey to assess state-of-the-art latent factor models ability to learn from data in Chapter 5. We design synthetic experiments that exhibit binary-relation properties, as well as common multi-relational inference through genealogical relations. Results give insights about each parametrization's pros and cons, and open to different future research directions. We conclude this thesis contributions and perspectives in Chapter 6. Appendix A presents an partially-related contribution of this thesis on online learning to sample training data for stochastic gradient descent. We demonstrate the benefits on different matrix factorization problems.

We made our implementation of the proposed model available[1], as well as the synthetic data used in the last chapter[2].

[1] `https://github.com/ttrouill/complex`
[2] `https://github.com/ttrouill/induction_experiments`

Publications Written During this Thesis

- Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., and Bouchard, G. (2016b). Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, volume 48, pages 2071–2080.
- Trouillon, T., Dance, C. R., Welbl, J., Riedel, S., Gaussier, É., and Bouchard, G. (2017a). Knowledge graph completion via complex tensor factorization. *arXiv preprint arXiv:1702.06879*. To appear in the Journal of Machine Learning Research.
- Trouillon, T. and Nickel, M. (2017). Complex and holographic embeddings of knowledge graphs: a comparison. *International Workshop on Statistical Relational AI*.
- Trouillon, T., Gaussier, É., Dance, C. R., and Bouchard, G. (2017b). On inductive abilities of latent factor models. Submitted to the Journal of Artificial Intelligence Research.
- Bouchard, G., Singh, S., and Trouillon, T. (2015a). On approximate reasoning capabilities of low-rank vector spaces. *AAAI Spring Symposium on Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches*.
- Bouchard, G., Trouillon, T., Perez, J., and Gaidon, A. (2015b). Online learning to sample. *arXiv preprint arXiv:1506.09016*.
- Trouillon, T., Dance, C. R., Gaussier, É., and Bouchard, G. (2016a). Decomposing real square matrices via unitary diagonalization. *arXiv:1605.07103*.

Chapter 2

Related Work

Before focusing on state-of-the-art models and methods for link prediction in knowledge graphs, let us put this problem back into its context. Link prediction is one of the main tasks of statistical relational learning (SRL) [Getoor and Taskar, 2007], a sub-field of machine learning concerned with the *representation* and *modeling* of relational data. We then formally define the link-prediction problem and review the literature, with an emphasis on latent factor models on which this thesis focuses. Finally we discuss related factorization problems and methods.

2.1 Relational Learning

Data is said to be relational when its *representation* is expressed as links, or relations, between the underlying objects of the database: the *entities*. This linked nature between the entities can be expressed in different but equivalent formalisms such as relational tables, as classically used in relational database management systems [Codd, 1970]; ground predicates in first-order logic where predicates are the relations and ground terms the entities [De Raedt, 2008; Richardson and Domingos, 2006]; and n-tuples through set theory where relations and entities are mixed in the tuples [Nickel, 2013]. In this work we will focus on specific relational data expressed as triples. Collections of such triples are known as *knowledge graphs*.

A knowledge graph stores data about a set of entities $\mathcal{E}$ and a set of relations $\mathcal{R}$, where relations link pairs of entities in the form of facts $r(s, o)$—for example `isCapitalOf(Ulaanbaatar,Mongolia)`—that we also write as triples (r, s, o), where the relation $r \in \mathcal{R}$ and the subject and object entities $s, o \in \mathcal{E}$. It is thus naturally represented as a labeled directed graph: a directed graph which has labeled edges that connect

subject entities to object entities, where the labels are the different relations $r \in \mathcal{R}$ (see Figure 2.1). We denote the set of all possible triples for a given entity set and relation set by $\mathcal{T} = \mathcal{R} \times \mathcal{E} \times \mathcal{E}$. A knowledge graph is hence a subset of $\mathcal{T}$: the set of observed triples (r, s, o) among all the possible ones, that we write $(r, s, o) \in \mathcal{T}_\Omega \subseteq \mathcal{T}$.

This representation has been driven by the coming of the Semantic Web, through the recommendations of the W3C, and namely, the Resource Sescription Framework (RDF) [Cyganiak et al., 2014]. Databases that follows this representation of data as triples are called knowledge graphs or knowledge bases. Many such knowledge graphs have been collaboratively or automatically created in recent years such as DBpedia [Auer et al., 2007], Freebase [Bollacker et al., 2008] and the Google Knowledge Vault [Dong et al., 2014].

From the very existence of these knowledge graphs and the applications they enable arise different tasks, such as predicting the missing triples in it—the task on which this thesis focuses—but also finding entities that are different instances of the same underlying object [Köpcke and Rahm, 2010], or grouping similar entities together [Fortunato, 2010]. Tackling these tasks require inferential abilities about the data, that is, a *model* of the knowledge graph considered. Many different formalisms for modeling relational data have been proposed, including first-order logic [Muggleton, 1995; Lisi, 2010; Galárraga et al., 2015], probabilistic graphical models [Ngo and Haddawy, 1997; Wellman et al., 1992; Kersting and De Raedt, 2001], latent space models [Nickel et al., 2011; Bordes et al., 2013b; Riedel et al., 2013], and different combinations of those [Richardson and Domingos, 2006; Rocktaschel et al., 2015].

2.1.1 Knowledge Graphs

Knowledge graphs differ largely in the way they are constructed, and in the domain they store data about.

Construction methods Some graphs are manually curated by experts, and have very accurate data such as WordNet [Fellbaum, 1998], but are generally restricted to small knowledge graphs as expert annotation is expensive. Other graphs such as Freebase [Bollacker et al., 2008] are created collaboratively on an open-access platform, following the model of Wikipedia. This construction model allows for a much larger scalability, while keeping a good data reliability, as Freebase has been estimated to be 99% accurate [Giannandrea, 2011].

More and more knowledge graphs resort to automatic triple extraction from data, either structured or not. The DBpedia project [Auer et al., 2007] extracts information from

Knowledge graph	Number of Entities $\lvert\mathcal{E}\rvert$	Relations $\lvert\mathcal{R}\rvert$	Facts $\lvert\Omega\rvert$
WordNet	155 K	116	9 M
NELL	5 M	306	0.5 M
YAGO2	10 M	114	447 M
DBpedia	5 M	1,367	538 M
Freebase	40 M	35,000	637 M
Google Knowledge Vault	45 M	4,469	1,600 M

TABLE 2.1: Number of entities $|\mathcal{E}|$, relations $|\mathcal{R}|$, and observed triples $|\Omega|$ of some knowledge graphs.

Wikipedia, and re-frames it as a knowledge graph. Similarly YAGO [Suchanek et al., 2007; Hoffart et al., 2013] also uses other sources of structured data. Other projects also make use of unstructured data such as the Never-Ending Language Learning system [Carlson et al., 2010] and the Google Knowledge Vault [Dong et al., 2014]. Both crawl the web and extract triples directly from its content, including text, tabular data and page structure. Knowledge graphs constructed this way are much bigger than the humanly created ones, but also less reliable as NELL is estimated to be 87% accurate [Lohr, 2010]. Sizes of the aforementioned knowledge graphs are summarized in Table 2.1.

Data domain Most of the knowledge graphs above store general knowledge about the world, akin to Wikipedia—but as triples. There are also projects dedicated to specific types of data. WordNet [Fellbaum, 1998] is a lexical database of English, its entities are word meanings, grouped in synsets, each representing a different concept. For polysemous words for example, each of their different meanings will be represented by a different entity. The entities are interlinked together by conceptual-semantic and lexical relations, such as hypernymy, meronymy or being part of another synset. WordNet resource has proven useful in many natural language processing tasks, such as word-sense identification [Leacock and Chodorow, 1998], text classification [Scott and Matwin, 1998] and information retrieval [Varelas et al., 2005].

In biology, many knowledge graphs arise such as Bio2RDF [Belleau et al., 2008] and LinkedLifeData [Momtchev et al., 2009]. Both projects aim at unifying many existing bioinformatics databases in a single knowledge graph. The IntAct database [Kerrien et al., 2011] describes interactions between pairs of molecules. Beyond projects that are explicitly storing their data as triples, many data sets that represent networks can be naturally expressed in the same triple formalism. Among them are the CORA [McCallum et al., 2000] and Citeseer [Lawrence et al., 1999] data sets, that represent citations network between scientific articles. There are also the Kinships data set [Denham, 1973] that describes kinship relations between individuals of an aboriginal tribe from Australia, the Nations data set that features diplomatic relations between countries [Rummel, 1976],

and the unified medical language system (UMLS) data set [McCray, 2003] that links medical concepts through their interactions.

2.1.2 Tasks and Applications

Knowledge-graph learning problems essentially inherit the classical problems coming from both databases, and machine learning. They thus have their own classification and clustering problems, namely *collective classification* and *link-based clustering*. But also classical databases problems such as avoiding duplicates and being as complete as possible, that is *entity resolution*, and our problem of interest, *link prediction*.

Collective classification When data naturally exhibits an interlinked nature, as is the case for social networks, or biological networks for example, the classical attribute-based classification model does not exploit this relational information properly. In this case, data can be naturally framed as a knowledge graph, and the classification of its entities among a set of classes, based on the links between entities—and their attributes when they exist—is known as *collective classification*. Methods that explicitly take into account such networked information have proven to be more accurate than those that do not [Sen et al., 2008; Neville and Jensen, 2003]. Collective classification applications include document classification [Chakrabarti et al., 1998], part-of-speech tagging [Lafferty et al., 2001], and counter-terrorism analysis [Macskassy and Provost, 2005].

Link-based clustering Similarly to collective classification, *link-based clustering* methods are clustering methods tailored for interlinked data, and make use of the relational patterns between entities. Such methods are widely used in social network analysis for community detection [Fortunato, 2010], for example on mobile phone communications [Blondel et al., 2008], e-mail exchanges [Tyler et al., 2005], and Facebook "friendship" networks [Traud et al., 2009].

Entity resolution Knowledge graphs that aggregate data from multiple sources of data, structured or unstructured, face the problem of duplicate entities. This is especially true when data is harvested from raw text, where for example, the same person name can be written either fully, or only with first name initial, or with middle name initial, and so forth. Resolving these duplicates is known as *entity resolution*, or more generally as *record linkage* [Köpcke and Rahm, 2010]. Approaches to solve this problem can be either fully automatic [Dredze et al., 2010; Bhattacharya and Getoor, 2007], or involve interactive interfaces that suggests possible conflicts to users [Bilgic et al., 2006]. As the

number of duplicates consequently affects the quality of the models that will be built on the knowledge graph to solve other tasks, this task arises quite naturally from the existence of knowledge graphs. But it also has its own direct applications in government data, public health systems, comparison shopping engines, and generally any information system that gather/store data from/in multiple databases [Christen, 2012].

Knowledge graphs are notoriously largely incomplete and predicting their missing entries is thus one of the main problems of relational learning. This problem is known as *link prediction*, or *knowledge graph completion*. Beyond search-results enhancement (Figure 1.1), link prediction has various applications including question answering [Bordes et al., 2014b], recommender systems [Rendle and Schmidt-Thieme, 2010] (see Section 2.3.1) and probabilistic querying of knowledge bases [Huang and Liu, 2009; Krompaß et al., 2014].

2.2 Link-Prediction

In this section, we formally define the link-prediction problem in knowledge graphs, as well as the notations that will be used throughout this manuscript. We then introduce in detail a family of state-of-the-art models, the latent factor models, on which this work focuses. We then review other approaches to the problem, including proposals mixing first-order logic and latent space models, as this is also one of interest to us (see Chapter 5).

Let us first introduce some notations. The number of entities is denoted by $N_e = |\mathcal{E}|$, and the number of relations by $N_r = |\mathcal{R}|$. The i^{th} row of a complex matrix $X \in \mathbb{C}^{n \times m}$ is written $x_i \in \mathbb{C}^m$. By a slight abuse of notation, for entities $i \in \mathcal{E}$ and relations $r \in \mathcal{R}$, we will write their corresponding rows in the embedding matrices as x_i or x_r, where $x_i, x_r \in \mathbb{C}^m$.

As previously defined, a knowledge graph is a set of observed triples (r, s, o), denoted by $\mathcal{T}_\Omega$. The link-prediction task consists in predicting some missing triples $(r', s', o') \in \mathcal{T} \setminus \mathcal{T}_\Omega$. Figure 2.1 presents a simple knowledge graph with five entities: `Bombelli`, `Grimaldi`, `Manfredi` (three Italian mathematicians), the city `Bologna` and its university `U. Bologna`; interlinked by two relations, `studied` and `born`. The fact `studied(Bombelli,U. Bologna)` is one of several facts that are missing in this graph. Link prediction's goal is to automatically discover and use redundancies in the graph to predict whether this missing triple is true or not— in order to display it in a search result for example, as shown in Figure 1.1. Here the facts `studied(`s`,U. Bologna)` and `born(`s`,Bologna)` are

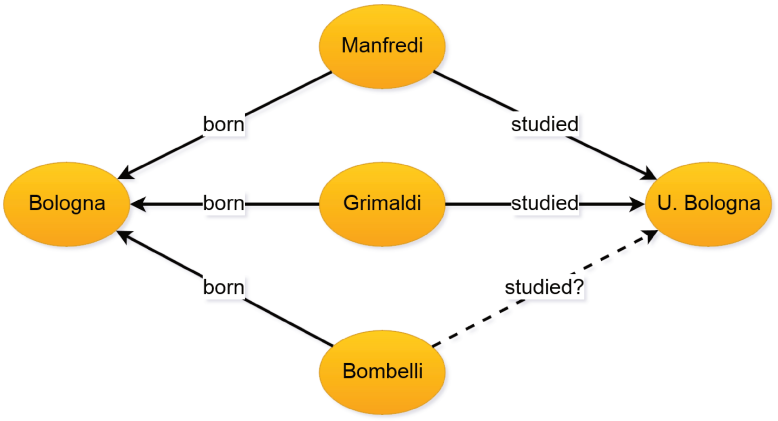


FIGURE 2.1: Example of a knowledge graph with 5 entities (nodes), 2 relations (edge labels) and 5 observed facts (edges). The dotted edge represent a missing fact of interest.

observed for the entities s for which they are observed. However that might not always be true, and link-prediction models thus have to handle inference probabilistically.

There are different possible assumptions concerning the truth of the missing triples. Under the *closed-world assumption* [Nickel et al., 2016a], those missing triples are considered to indicate false facts—then the link-prediction problem is *de facto* solved. This assumption is primarily used with complete data sets that serve as benchmarks for link prediction through cross-validation, such as the Kinships or UMLS data sets [Denham, 1973; McCray, 2003], as shown in Section 4.2. The very existence of the link-prediction problem implies the *open-world assumption*: missing triples are indeed considered as missing, and one has to separate the true facts from the false facts among them [Drumond et al., 2012].

To abstract ourselves from different assumptions, and generalize to all cases, we consider that we observe a set of true and false triples—possibly only true ones—by associating to each observed triple $(r, s, o) \in \mathcal{T}$ its corresponding truth value $y_{rso} \in \{-1, 1\}$. For example, the fact `first_used(Cardano, imaginary_numbers)` is true [Cardano, 1545][1]. It thus has a corresponding truth value $y_{rso} = 1$. To false facts we attribute the value -1. We write the corresponding set of observed facts with their truth values $\Omega = \{((r, s, o), y_{rso}) \,|\, (r, s, o) \in \mathcal{T}_\Omega\}$. Link prediction is then framed as predicting the truth values $y_{r',s',o'}$ of a disjoint set of unobserved triples $(r', s', o') \in \mathcal{T} \setminus \mathcal{T}_\Omega$.

Graphs are naturally represented by their adjacency matrix. Knowledge graphs have labeled edges, and hence have a different adjacency matrix $Y_r \in \{-1, 1\}^{N_e \times N_e}$ for each relation $r \in \mathcal{R}$. These matrices Y_r are only partially observed as we only observe the values y_{rso} for observed triples $((r, s, o), y_{rso}) \in \Omega$. Note that -1 represent the truth

[1] The first person to stumble on imaginary numbers was Heron of Alexandria. However, no mathematician until Cardano—a millenium and a half later—went past the puzzling moment of facing the square root of a negative number. Simply putting "nevertheless we will operate", Cardano provided the first solution to a system of equations that has complex roots, and the first actual use of complex numbers [Cardano, 1545].

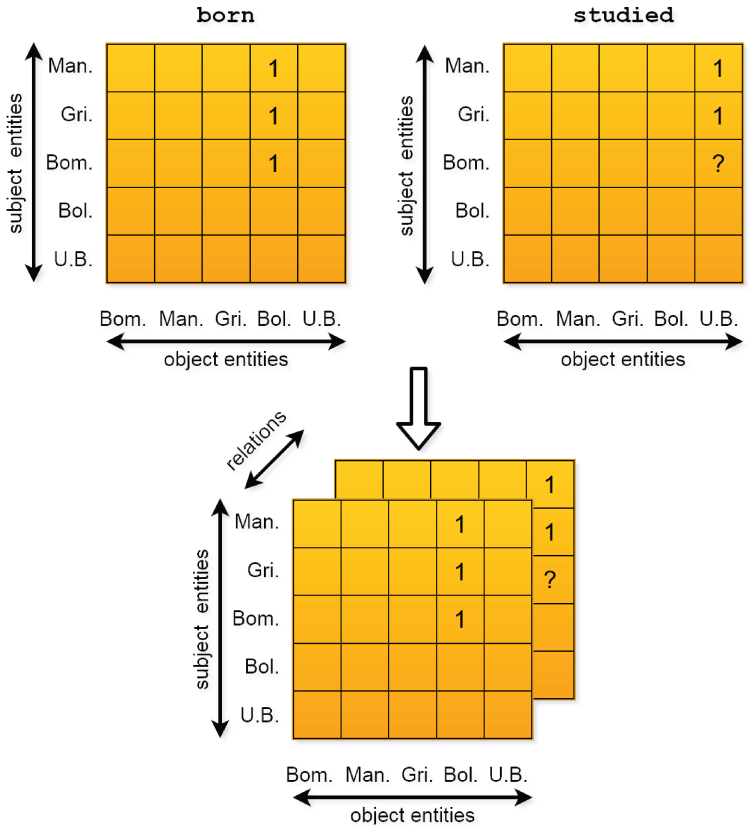


FIGURE 2.2: Adjacency matrices stacked into a 3rd-order partially-observed tensor, corresponding to the knowledge graph in Figure 2.1. The question mark represents a missing fact of interest.

value y_{rso} of the *false* observed triples in Ω, not the missing ones. All those adjacency matrices can be stacked as a 3rd-order, partially-observed tensor $\mathbf{Y} \in \{-1, 1\}^{N_r \times N_e \times N_e}$, and the value at index (r, s, o) is the truth value of the corresponding triple: y_{rso}, for observed triples $((r, s, o), y_{rso}) \in \Omega$. Figure 2.2 shows the adjacency matrices and tensor corresponding to the knowledge graph presented in Figure 2.1.

By representing a knowledge graph as a partially observed tensor, the link-prediction problem becomes a *tensor-completion* problem. To enable completion by factorization methods, the unknown tensor is assumed to have low rank or approximately low rank. In this case, the observed entries constrain the possible values of the missing entries, and the tensor can then be completed by the product of learnt factor matrices with much smaller dimensions. Low-rank matrix and tensor-factorization methods have proven to be very effective for completion problems, and numerous research works have been published on the topic (see Section 2.3.1). We decided to follow this approach to tackle the link-prediction problem. In the next section, we review the state-of-the-art *latent factor models* for link prediction, which are models that learn a factorization of the knowledge graph tensor, and we discuss their known advantages and limitations.

Model	**Scoring Function ϕ**	**Parameters Θ**
CP [Hitchcock, 1927]	$\langle w_r, u_s, v_o \rangle$	$w_r, u_s, v_o \in \mathbb{R}^K$
RESCAL [Nickel et al., 2011]	$e_s^T W_r e_o$	$W_r \in \mathbb{R}^{K^2}$, $e_s, e_o \in \mathbb{R}^K$
TransE [Bordes et al., 2013b]	$-\|\|(e_s + w_r) - e_o\|\|_q$	$w_r, e_s, e_o \in \mathbb{R}^K$
F model [Riedel et al., 2013]	$u_d^\top w_r$	$w_r, u_d \in \mathbb{R}^K$
DistMult [Yang et al., 2015]	$\langle w_r, e_s, e_o \rangle$	$w_r, e_s, e_o \in \mathbb{R}^K$
ComplEx (this thesis)	$\mathrm{Re}(\langle w_r, e_s, \bar{e}_o \rangle)$	$w_r, e_s, e_o \in \mathbb{C}^K$

Table 2.2: Scoring functions of state-of-the-art latent factor models for a given fact $r(s, o)$, along with the representation of their parameters. In the F model, d indexes all possible pairs of entities: $d = (s, o) \in \mathcal{E} \times \mathcal{E}$.

2.2.1 Latent Factor Models

We define each model by its *scoring function* $\phi(r, s, o; \Theta)$, where Θ are the latent parameters of this model—the entity and relation embeddings—and $\phi(r, s, o; \cdot) : \mathbb{C}^{|\Theta|} \to \mathbb{R}$ assigns a real-valued score to the fact $r(s, o)$. As some models have real-valued parameters and some other models have complex-valued parameters, we define the space of the parameters $\mathbb{C}^{|\Theta|}$ directly over the complex space.

Let us also define the trilinear product of three vectors over the complex space:

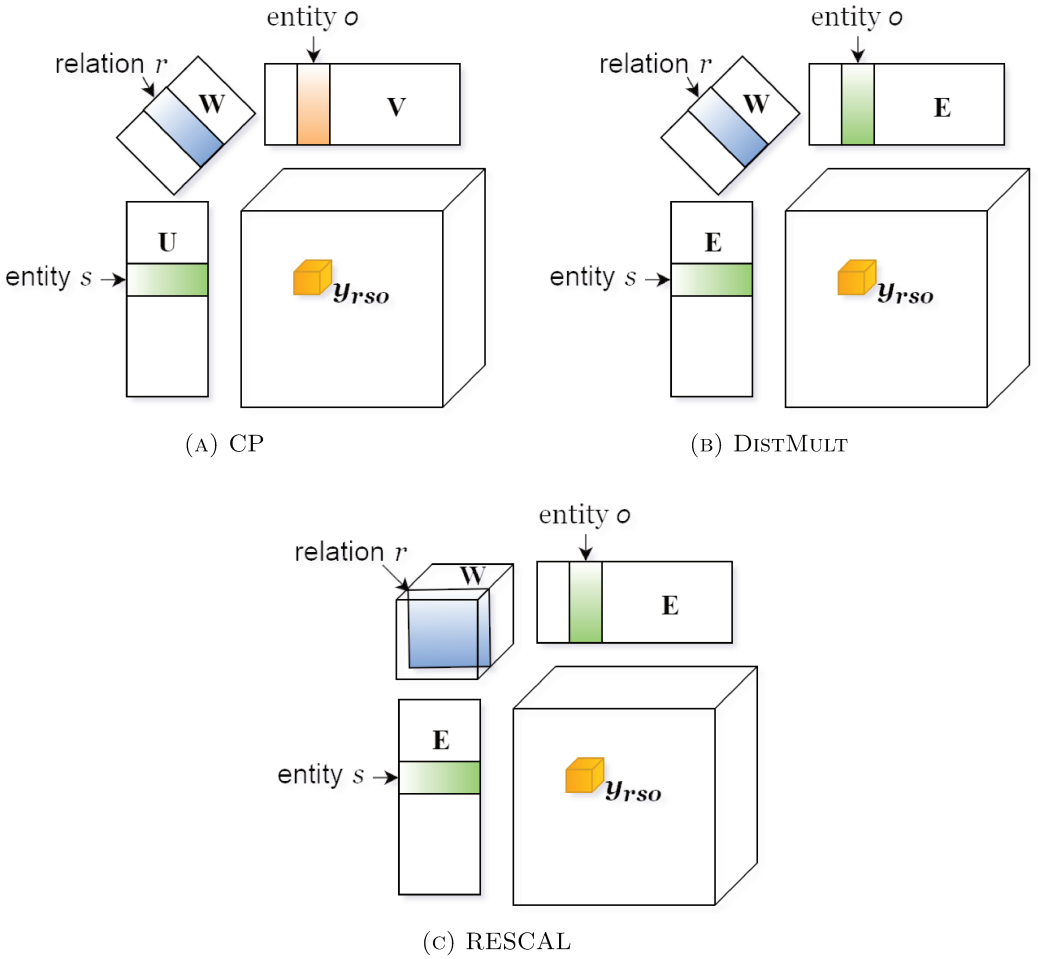
$$\begin{aligned} \langle a, b, c \rangle &= \sum_{j=1}^{K} a_j b_j c_j \\ &= a^\top (b \odot c) \end{aligned} \tag{2.1}$$

where $a, b, c \in \mathbb{C}^K$, and $\odot$ is the Hadamard product, that is the element-wise product between two vectors of same length.

2.2.1.1 Models Compared in this Work

In the following we present in detail the model scoring functions and parameters that we experimentally compare in this work. Those models are among the most popular and best-performing link-prediction models. The models' scoring functions and parameters are summarized in Table 2.2.

Each of the following models use latent representations of variable length, controlled by the hyper-parameter $K \in \mathbb{Z}_{++}$, the rank of the decomposition. We start by introducing the most natural model, a general decomposition for tensors: the Canonical-Polyadic (CP) decomposition [Hitchcock, 1927], also know as CANDECOMP [Caroll and Chang, 1970], and PARAFAC [Harshman, 1970].



(A) CP (B) DistMult

(C) RESCAL

FIGURE 2.3: Graphic representation of the CP, DistMult and RESCAL models as tensor factorization models, with their latent parameters.

Canonical-Polyadic Decomposition (CP) The Canonical-Polyadic decomposition involves one latent matrix for each dimension of the decomposed tensor, so in our case we have three latent matrices as $\mathbf{Y}$ is a 3$^{\text{rd}}$-order tensor. Its scoring function is

$$\phi(r, s, o; \Theta) = \langle w_r, u_s, v_o \rangle \tag{2.2}$$

where $U, V \in \mathbb{R}^{N_e \times K}$ are the embedding matrices of entities depending on whether they appear as subject (U) of the triple or as object (V), and $W \in \mathbb{R}^{N_r \times K}$ is the embedding matrix of the relations.

This model is a very general tensor decomposition, though it is not really tailored to our problem, since our tensor is a stack of N_r square matrices where rows and columns represent the same underlying objects: the entities. Indeed, its completely decorrelated representations u_i and v_i of the same entity $i \in \mathcal{E}$ make it harder for this model to generalize, as we will see in Chapter 4.

RESCAL RESCAL [Nickel et al., 2011] differs from the CP decomposition in two points: there is only one embedding per entity instead of having one embedding for entities as subject and another one for entities as objects; and each relation is represented by a matrix embedding instead of a vector. Its scoring function is

$$\phi(r, s, o; \Theta) = e_s^\top W_r e_o \tag{2.3}$$

where $E \in \mathbb{R}^{N_e \times K}$ is the embedding matrix of the entities, and $\mathbf{W} \in \mathbb{R}^{N_r \times K \times K}$ the embedding tensor of the relations. Thus $W_r \in \mathbb{R}^{K \times K}$ is the embedding matrix of the relation r.

RESCAL was the first model to propose unique embeddings for entities—simultaneously with Bordes et al. [2011]—which yielded significant performance improvement, and since then unique entity embeddings have been adopted by most of the subsequent models. However, its matrix representations of relations makes its scoring function time and space complexity quadratic in the rank K of the decomposition. This also leads to potential overfitting.

F model This model proposed by Riedel et al. [2013] maps all possible subject and object entity pairs $p = (s, o) \in \mathcal{E} \times \mathcal{E}$ to a single dimension. Each row in the entity embedding matrix corresponds to one pair of entities. The scoring function computes the dot product of the embedding of the pair p with the embedding of the relation r:

$$\phi(r, s, o; \Theta) = e_p^\top w_r \tag{2.4}$$

where $E \in \mathbb{R}^{N_e^2 \times K}$ is the embedding matrix of the pairs of entities, and $W \in \mathbb{R}^{N_r \times K}$ the embedding matrix of the relations. It is actually a decomposition of the matrix that results from a specific unfolding of the $\mathbf{Y}$ tensor.

Its pairwise nature gives this model an advantage over non-compositional pairs of entities. However, its memory complexity is quadratic in the number of entities N_e. In practice, unobserved pairs of entities are not stored in memory as they are useless. Though this is also the weak point of this model: it cannot predict scores for unobserved pairs of entities since it only learns pairwise representations.

TransE The TransE model [Bordes et al., 2013b] imposes a geometrical structural bias on the model: the subject entity vector should be close to the object entity vector once translated by the relation vector. For a given q-norm (generally $q = 1$ or $q = 2$)

over the embedding space,

$$\phi(r, s, o; \Theta) = -||(e_s + w_r) - e_o||_q \tag{2.5}$$

where $E \in \mathbb{R}^{N_e \times K}$ is the embedding matrix of the entities, and $W \in \mathbb{R}^{N_r \times K}$ is the embedding matrix of the relations. Deriving the norm in the scoring function exposes another perspective on the model and unravels its factorial nature, as it gives a sum of bilinear terms as explored by García-Durán et al. [2014]:

$$\phi(r, s, o; \Theta) \approx e_s^\top e_o + e_o^\top w_r - e_s^\top w_r \tag{2.6}$$

where constant multipliers and norms of the embeddings have been ignored here. These bigram terms will help in some specific situations as shown in Section 5.3.

It is difficult to capture symmetric relations with this model. Indeed, having $\phi(r, s, o; \Theta) = \phi(r, o, s; \Theta)$ implies either $e_s = e_o$, or $w_r^\top (e_o - e_s) = 0$. Since $e_s \neq e_o$ in general for $s \neq o$, and w_r is in general not the zero vector—in order to share latent dimensions' information with the other relation embeddings—modeling symmetric relations such as `similar`, `cousin`, or `related` implies a strong geometrical constraint on entity embeddings: their difference must be orthogonal to the relation embedding w_r. The model thus has to make a trade-off between (i) correctly modelling the symmetry of the relation r, (ii) not zeroing its relation embedding w_r, and (iii) not altering too much the entity embeddings to meet the orthogonality requirement between w_r and $(e_o - e_s)$ for all $e, o \in \mathcal{E}$.

DistMult The DistMult model [Yang et al., 2015] can be seen as a simplification of the RESCAL model, where the unique representation of entities is kept, while the representation of the relations is brought back to vectors instead of matrices:

$$\phi(r, s, o; \Theta) = \langle e_s, w_r, e_o \rangle \tag{2.7}$$

where $E \in \mathbb{R}^{N_e \times K}$ is the embedding matrix of the entities, and $W \in \mathbb{R}^{N_r \times K}$ the embedding matrix of the relations.

The major drawback of this model is its symmetry over the subject and object entity roles. Indeed we have $\phi(r, s, o; \Theta) = \phi(r, o, s; \Theta)$, for all $s, o \in \mathcal{E}$. But many antisymmetric relations appear in knowledge graphs such as `older`, `partOf`, `hypernym`. One does not want to assign the same score to `older(a,b)` as to `older(b,a)`!

2.2.1.2 Other Latent Factor Models

Akin to the CP model, there exist various classical tensor decomposition, such as the Tucker decomposition [Tucker, 1963], also known as higher-order singular value decomposition (HOSVD) [De Lathauwer et al., 2000], from which many of the presented latent factor models are adaptations. Tensor decompositions and their applications are surveyed in Kolda and Bader [2009]; Comon et al. [2009].

The first ones to propose to use factorization methods, already popular in the neighbor field of collaborative filtering (see Section 2.3.1), to tackle link prediction in knowledge graphs were Franz et al. [2009] and Sutskever et al. [2009], who respectively used the CP model, and proposed the Bayesian clustered tensor factorization model (BCTF). The BCTF scoring function can be seen as an intermediate between the CP and the RESCAL models, as relations are modeled with matrices and entities with two separate vectors depending on whether they appear as subject or as object of the triple:

$$\phi(r, s, o; \Theta) = u_s^\top W_r v_o \tag{2.8}$$

where $U, V \in \mathbb{R}^{N_e \times K}$ are the embedding matrices of entities depending on whether they appear as subject (U) of the triple or as object (V), and $W \in \mathbb{R}^{N_r \times K \times K}$ the embedding tensor of the relations. The model is learned in a Bayesian setting with a Chinese restaurant process prior over the embeddings.

Jenatton et al. [2012] proposed a similar model, with a non-probabilistic clustering over the relations matrices, by expressing them as a low rank, L^1-constrained decomposition:

$$W_r = \sum_{d=1}^{D} \alpha_d^r (a_d b_d^T) \tag{2.9}$$

where D is the rank of the decomposition of the relations parameters, $A, B \in \mathbb{R}^{D \times D}$, and $\alpha^r \in \mathbb{R}^D$ is constrained by a hyperparameter: $||\alpha^r||_1 \leq \lambda_\alpha$. The scoring function is itself also slightly different as they add bias terms to the subject and object-entity embeddings:

$$\phi(r, s, o; \Theta) = (u_s + z)^\top W_r (v_o + z') \tag{2.10}$$

where $z, z' \in \mathbb{R}^K$.

Various models built on TransE have been proposed, including the TransH model [Wang et al., 2014] that models relations as translating hyperplanes instead of vectors, and the TransR model [Lin et al., 2015b] that learns relation and entity embeddings in different spaces and maps entity embeddings to relation space using linear operators,

before performing a translation. He et al. [2015] propose to learn these models with Gaussian embeddings, replacing the vector norm with the KL-divergence or the expected likelihood. The TATEC model [Garcia-Duran et al., 2016] combines TransE bigram terms and RESCAL scoring function in a single model, but trained separately. Welbl et al. [2016] extended the F model to learn pairwise embeddings not only of entities, but of subject/relation and object/relation pairs too. Verga et al. [2017] address the squared complexity in the number of the entities of the F model, by expressing the pairwise embeddings of entities as combinations of the relation embeddings in which the pair appear. For each observed triple $(r, s, o) \in \mathcal{T}_\Omega$, the corresponding pair embedding e_p for $p = (s, o) \in \mathcal{E} \times \mathcal{E}$ is

$$e_p = f(\{w_{r'} \mid (r', s, o) \in \mathcal{T}_\Omega\}) \tag{2.11}$$

where f is the composition function of the relation embedding. This model only learns relation embeddings, yet it performs just as well as the original F model. Moreover, it gives it the ability to naturally generalize to unseen entities.

The holographic embeddings model [Nickel et al., 2016b] proposes to combine vectorial entity embeddings using discrete circular convolution between the subject and object embeddings. This model and its link to the model proposed in this manuscript are discussed in detail in Section 3.4.

Sometimes, entities and relations come with additional domain knowledge. A common feature of entities is their type, such as *person* or *place*, that defines incompatibilities for some relations in which such typed entities can appear. For example, a place cannot be the president of a person. Chang et al. [2014]; Krompaß et al. [2015]; Sedghi and Sabharwal [2016] enhance predictions of existing factorization models by not using incompatible triples during training, whereas they are usually considered as false triples.

2.2.1.3 Losses and Negative Sampling

Commonly used matrix and tensor decompositions such as SVD and CP natively minimize the squared error. Classical decomposition algorithms for these models, based on iterative methods or alternating minimization, cannot efficiently handle missing triples as missing, and consider them as negatives instead. This corresponds to minimizing:

$$\mathcal{L}(\mathbf{Y}; \Theta) = \sum_{r \in \mathcal{R}} \sum_{s \in \mathcal{E}} \sum_{o \in \mathcal{E}} ||\phi(r, s, o; \Theta) - y_{rso}||_2^2 \tag{2.12}$$

where $y_{rso} = -1$ if $(r, s, o) \notin \mathcal{T}_\Omega$—though in this dense case the value zero is more often used for negatives. However, as the tensor $\mathbf{Y}$ has binary values ± 1, using a binary loss is indeed more appropriate. We discuss theoretical motivation for doing so in Section 3.1.2.2.

Jenatton et al. [2012] used instead the negative log-likelihood of the logistic model:

$$\mathcal{L}(\mathbf{Y};\Theta) \quad = \quad \sum_{r\in\mathcal{R}}\sum_{s\in\mathcal{E}}\sum_{o\in\mathcal{E}} \log(1+\exp(-y_{rso}\phi(r,s,o;\Theta)))\,, \tag{2.13}$$

and Nickel and Tresp [2013]; London et al. [2013] showed it worked better than the squared loss, in all cases on dense datasets (under the closed-world assumption) with observed negatives. Acar et al. [2010] and London et al. [2013] proposed a weighted version of respectively CP and RESCAL to avoid imputing test triples when learning the decomposition, and improved performances in the closed-world case.

Drumond et al. [2012] first acknowledged the importance of the open-world assumption. By treating missing triples as missing, they exposed a large gap between the predictive performances of dense and sparse versions of the CP model. In general under the open-world assumption, only positive triples are observed. One thus has to generate negatives to learn a supervised model. To do so, they make the assumption that an observed triple $(r,s,o) \in \mathcal{T}_\Omega$ should be ranked higher than unobserved triples with a different object entity $(r,s,o') \notin \mathcal{T}_\Omega$. They implement this constraint through the Bayesian Personalized Ranking optimization criterion [Rendle and Schmidt-Thieme, 2010], by uniformly sampling object entities o':

$$\mathcal{L}(\Omega;\Theta) = \sum_{((r,s,o),y_{rso})\in\Omega} \log(\sigma(y_{rso} - y_{rso'})) \tag{2.14}$$

where σ is the logistic function $\sigma(x) = \frac{1}{1+\mathrm{e}^{-x}}$.

Bordes et al. [2013b] extended this assumption to subject entities: for each positive triple $(r,s,o) \in \mathcal{T}_\Omega$ they corrupt either the subject or the object of the triple at random, and optimize a slightly different pairwise loss, with a max-margin criterion:

$$\mathcal{L}(\Omega;\Theta) = \sum_{(r,s,o)\in\mathcal{T}_\Omega} \max(0, \gamma + \sigma(\phi(r,s',o';\Theta)) - \sigma(\phi(r,s,o;\Theta)) \tag{2.15}$$

where $\gamma \in [0,1]$ is the margin hyperparameter. This loss has been often used in subsequent works [Yang et al., 2015; Nickel et al., 2016b]. In this work, we instead consider all such corrupted triples explicitly as negatives, which is also known as the *local closed-world assumption* [Dong et al., 2014]: all $(r,s',o),(r,s,o') \notin \mathcal{T}_\Omega$ for each $(r,s,o) \in \mathcal{T}_\Omega$ are considered as negatives: $y_{rs'o} = y_{rso'} = -1$. We optimize a classical log-likelihood loss, and show that it can bring a large improvement over the max-margin loss (see Section 4.3.6), and that sampling more than one negative per positive triple also sensibly improves prediction scores (see Section 4.3.4).

2.2.2 Other Link-Prediction Approaches

Early relational learning approaches for relational databases that follows a schema used probabilistic models. The general idea is to map a probabilistic graphical model to the database schema architecture, and use observed entries to learn the corresponding probability distribution [Friedman et al., 1999; Taskar et al., 2001; Heckerman et al., 2007; Getoor and Taskar, 2007; Raedt et al., 2016].

Logic-based link prediction consists in using both observed facts and logical rules to infer the truth of unobserved facts. In this case one must either handcraft rules, or learn them through inductive logic programming (ILP) for example [Muggleton and De Raedt, 1994; Dzeroski and Lavrac, 1994]. Many contributions have been made using inductive logic programming for relational data during the last decades [Muggleton, 1995; Lisi, 2010; Galárraga et al., 2015]. Inference can be achieved deterministically by logical deduction, or probabilistically to cope with uncertainty of the data. Different probabilistic logic-based inference models have been proposed [Ngo and Haddawy, 1997; Wellman et al., 1992; Kersting and De Raedt, 2001; Frasconi et al., 2014; Kok and Domingos, 2007]. The main contribution along this line of research is probably Markov Logic Networks (MLNs) [Richardson and Domingos, 2006]. MLNs take as input a set of first-order rules and facts, build a Markov random field between facts co-occuring in possible groundings of the formulae, from which they learn a weight over each of these rules that represents their likeliness of being applied at inference time. Different improvements over this model have been proposed [Riedel, 2008; Noessner et al., 2013]. Among them, Pujara et al. [2013] used probabilistic soft logic [Brocheler et al., 2010] to assign continuous truth values to atoms instead of boolean ones, which resulted in increased prediction accuracy and scalability.

In the neural tensor network (NTN) model, Socher et al. [2013] combined linear transformations and multiple bilinear forms of subject and object embeddings to jointly feed them into a nonlinear neural layer:

$$\phi(r, s, o; \Theta) = u_r^T f(e_s^T W_r^{[1:D]} e_o + V_r [e_s \;\; e_o]^\top + b_r)\,, \tag{2.16}$$

where $D \in \mathbb{Z}_{++}$ is an additional hyperparameter, $e_s, e_o \in \mathbb{R}^K$ are learned entity embeddings; $W_r \in \mathbb{R}^{K \times K \times D}$, $V_r \in \mathbb{R}^{D \times 2K}$, b_r,$u_r \in \mathbb{R}^D$ are the learned relation parameters, and f is a non-linear activation function. Its non-linearity and multiple ways of including interactions between embeddings gives it an advantage in expressiveness over simpler latent factor models. As a downside, its very large number of parameters can make the NTN model harder to train and make it overfit more easily. Authors also propose to learn the entity embeddings as a composition of the word embeddings of their labels.

Doing so can significantly improve results, depending on the model and the dataset. Bordes et al. [2011] proposed the Structured Embeddings (SE) model, a generalization of Siamese networks:

$$\phi(r, s, o; \Theta) = ||W_r e_s - W'_r e_o||_q \,, \tag{2.17}$$

where $W_r, W'_r \in \mathbb{R}^{K \times K}$ are the relation embeddings. Though it looks like TransE, deriving the norm shows that the two matrix embeddings of relations play the role of two fully connected layers. Subsequently, Bordes et al. [2014a] proposed the Semantic Matching Energy (SME) model, an explicit two-layer network where subject and object embeddings are similarly combined with a right and left relation embeddings first, then intermediate left and right representation are merged into the final score. Nguyen et al. [2016] proposed STransE, a combination of the SE and TransE models. Dong et al. [2014] use a two-layer perceptron:

$$\phi(r, s, o; \Theta) = u^T f(A[w_r \;\; e_s \;\; e_o]^\top) \,, \tag{2.18}$$

where f is a non-linear activation function, $e_s, e_o, w_r \in \mathbb{R}^K$, $A \in \mathbb{R}^{D \times 3K}$, $u \in \mathbb{R}^D$ where $D \in \mathbb{Z}_{++}$ is an hyperparameter controlling the size of the second layer.

Aforementioned latent models of knowledge graphs consider triples separately from each other, and capture dependencies between conjunctions of relations such as $\texttt{livesInCity}(\texttt{a}, \texttt{b}) \wedge \texttt{isInCountry}(\texttt{b}, \texttt{c}) \Rightarrow \texttt{livesInCountry}(\texttt{a}, \texttt{c})$ from redundancy in the data. From a graph perspective, such multi-relation inferences correspond to *paths* in the knowledge graph. Different models propose to take into account these path patterns explicitly. The path ranking algorithm [Lao et al., 2011] predicts missing triples by combining the results of different random walks accross the knowledge graph. Lin et al. [2015a]; Das et al. [2016]; Neelakantan et al. [2015] proposed to consider all possible paths between each pair of observed entities (s, o) for $(r, s, o) \in \mathcal{T}_\Omega$, using a recurrent neural network to model paths of arbitrary length. Conversely, Guu et al. [2015] introduced the task of answering path-based queries instead of simply predicting triples. A path query consist of a source entity s and a sequence of relations $(r_1, \ldots, r_n)$. The answer is the set of entities o that can be reached from s by that sequence of relations such that all intermediate triples $(s, r_1, e_1), \ldots, (e_n, r_n, o)$ are true. They propose a general framework to train and predict on such paths by recursively composing scoring functions that provide intermediate representations of subject/relation pairs—the condition that scoring function must fulfil to be *composable*.

Trilinear models for example are composable in this sense as a trilinear product can be factorized in the object-entity embedding. For example with the DistMult model: $\langle e_s, w_r, e_o \rangle = (e_s \odot w_r)^\top e_o$, where the intermediate representation is the Hadamard

product between the subject entity and the relation embeddings $e_s \odot w_r$. Other graph-related approaches include the additive relational effect model [Nickel et al., 2014] that learns a linear combination over metrics computed on the knowledge graph such as common neighbors or Katz centrality, and combines it with RESCAL's scoring function; and Gaifman models that learn neighborhood embeddings of local structures in the knowledge graph [Niepert, 2016].

The factorization machines model, proposed by Rendle [2010], enhances supervised linear models by learning vectorial representations of the features of the samples, combined d-linearly, where d is an hyperparameter setting the degree of the model. Given a feature vector $x \in \mathbb{R}^n$ and its corresponding label $y \in \mathbb{R}$, a factorization machine of degree $d = 3$ gives:

$$\hat{y}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} (u_i^\top u_j) x_i x_j + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=j+1}^{n} \langle v_i, v_j, v_k \rangle \, x_i x_j x_k \quad (2.19)$$

where $w_0, \ldots, w_n \in \mathbb{R}$, $U, V \in \mathbb{R}^{n \times K}$. This model generalizes the CP decomposition: by encoding in $x \in \mathbb{R}^{N_r + 2N_e}$ the concatenation of the one-hot representation of the (r, s, o) triple indexes $x = [e_r^1, e_s^1, e_o^1]$, with $y = y_{rso}$, where e_i^1 has a 1 at index i and zeros everywhere else, $e_r^1 \in \mathbb{R}^{N_r}$, $e_s^1, e_o^1 \in \mathbb{R}^{N_e}$. This adds bigram terms—as in TransE—unigram terms and biases to the trilinear term of the CP decomposition. With different encoding of the feature of x, the author shows generalization of diverse matrix and tensor factorization models. This model has also been adapted to scale to classical schema-based relational databases (such as SQL) [Rendle, 2013].

Many authors have proposed to use text as distant supervision to enhance knowledge graphs, by extracting triples from raw text, which increased predictive performances for link prediction [Riedel et al., 2013; Toutanova et al., 2015; Surdeanu and Tibshirani, 2012; Yao et al., 2011; Mintz et al., 2009].

2.2.3 Learning Logic within Latent Space Models

In this thesis, we evaluate latent space models on their ability to learn logical reasoning from observed data only (see Chapter 5). Similarly to our approach, Bowman et al. [2015b] learned some *natural logic* operations directly from data with recurrent neural tensor networks, to tackle natural language processing tasks such as entailment or equivalence. Natural logic is a theoretical framework for natural language inference that uses natural language strings as the logical symbols. Singh et al. [2015] investigated learning from a few synthetic examples for relational learning on different latent factor models.

Following a different goal, other approaches formalised the encoding of logical operations as tensor operations. Smolensky et al. [2016] applied it to the bAbI data set reasoning tasks, and Grefenstette [2013] to general Boolean operations.

Advances in bringing both worlds together include the work of Rocktaschel et al. [2015]; Rocktäschel et al. [2014] and Demeester et al. [2016], where a latent factor model is used, as well as a set of logical rules. An error-term over the rules is added to the classical latent factor objective function. In Rocktäschel and Riedel [2016], a fully differentiable neural theorem prover is used to handle both facts and rules, whereas Minervini et al. [2017] use adversarial training to do so. Wang and Cohen [2016] learned first-order logic embeddings from formulae learned by ILP. Similar proposals for integrating logical knowledge in distributional representations of words include the work of Lewis and Steedman [2013]. Conversely, Yang et al. [2015] learn a latent factor model over the facts only, and then try to extract rules from the learned embeddings. [Yoon et al., 2016] proposed to use projections of the subject and object-entity embeddings that conserve transitivity and symmetry.

2.3 Related Factorization Problems and Methods

We here survey related work concerning the vast field of matrix and tensor decompositions, and the use of complex numbers therein.

2.3.1 Matrix and Tensor Completion

The application of factorization methods in relational learning comes from their large success in a preceding, closely-related problem: *collaborative filtering*. Collaborative filtering is a special case of link prediction in knowledge graphs: a matrix $X \in \mathbb{R}^{n \times m}$ is partially observed—and not a tensor—however it is real-valued and it is not necessarily square. Rows represent users, columns represent items, and entries $x_{ij} \in \mathbb{R}$ for observed (i, j) pairs are implicit or explicit feedback, such as ratings. Typical factorization models are of the form $X \approx UV^\top$, where each row u_i corresponds to a user $i \in \mathcal{U}$ and each column v_j corresponds to an item $j \in \mathcal{I}$. In this problem, $\mathcal{U} \cap \mathcal{I} = \emptyset$ conversely to knowledge graphs, where entities can be either the subject or the object of a relation. Despite this, this parametrization with different left (U) and right (V) matrices persisted in knowledge-graph factorization models up to the RESCAL and SE models [Nickel et al., 2011; Bordes et al., 2011]. Interestingly, these rectangular matrix factorization models for collaborative filtering can be seen as a special case of knowledge graph factorization models *with* single entity embeddings $E \in \mathbb{R}^{N_e \times K}$. By writing the set of entities $\mathcal{E} = \mathcal{U} \cup \mathcal{I}$

and having a single relation r, embeddings of users and items of observed triples (r, i, j) are indeed disjoint: $e_i \neq e_j$ for any $i \in \mathcal{U}$, $j \in \mathcal{I}$, and correspond to the U and V matrices since users are always subjects and items are always objects. Hence classical factorization models for link prediction subsume rectangular matrix factorization with disjoint set of entities as rows and columns, provided that the open-world assumption is enforced (unobserved user-user, item-item and item-user triples are ignored) and that a non-binary loss is used to handle the real-valued entries $x_{ij} \in \mathbb{R}$.

Completing the missing entries of such feedback matrices has direct applications in recommender systems, and factorization approaches became popular with the famous Netflix prize [Koren, 2008; Koren et al., 2009]. In most partially-observed matrix and tensor-factorization models, optimizing directly over the low-rank factor matrices is a non-convex problem. A well-known relaxation of the matrix completion problem consists in minimizing its trace-norm, which is the sum of its singular values:

$$\begin{aligned} \min \quad & ||\hat{X}||_* \\ \text{subject to} \quad & \hat{x}_{ij} = x_{ij}, \;\; (i,j) \in \Omega \,, \end{aligned} \tag{2.20}$$

where $X \in \mathbb{R}^{n \times m}$ and Ω is the set of the observed values in X. This approach has strong guarantees to recover the minimal rank of the partially observed matrix X and can be cast as a semi-definite program to solve it [Candès and Tao, 2010; Candes and Recht, 2012]. Convex extensions to collective matrix factorization have also been proposed [Singh and Gordon, 2008; Bouchard et al., 2013], and the classical tensor-factorization models [Comon et al., 2009; Kolda and Bader, 2009] also had their convex relaxations for completion [Tomioka et al., 2010; Romera-Paredes and Pontil, 2013]. In most convex factorization methods, the reconstructed matrix/tensor must be instantiated in memory, which is a serious space bottleneck. More scalable approaches have been proposed for matrix completion, based on iterative sparse singular value decompositions (SVD) [Cai et al., 2010], allowing for not storing the whole reconstructed matrix $\hat{X}$ in memory. Though the cost of computing numerous SVDs iteratively is prohibitive for very large scale matrices. Convex tensor factorization models have similar scalability issues.

Though, non-convex approaches that optimize over the low-rank latent factors actually work very well in practice [Koren et al., 2009; Nickel et al., 2016a]. Some first theoretical results start explaining this, and showed that on some non-convex matrix and tensor factorization problems and under certain conditions, all local minima are global [Ge et al., 2016; Bhojanapalli et al., 2016; Haeffele and Vidal, 2015]. Specifically, Ge et al. [2016] showed that this is the case for positive semi-definite matrix completion. This being said, given the size of the problems we tackle in this work, we cannot afford convex relaxation. Optimization is conducted over the non-convex low-rank parametrization

(see Section 3.3), as in all state-of-the-art factorization models for knowledge-graph completion [Nickel et al., 2016a].

Among other contributions are the pairwise interaction tensor factorization (PITF) [Rendle and Schmidt-Thieme, 2010], that handles feedback between users', item and tags through tensor factorization. Abernethy et al. [2009] integrate users' and items' attributes in their factorization model. Ermis and Bouchard [2014] use quadratic approximation of the logistic loss to speed-up the decomposition, and Zhang et al. [2007] propose to learn binary latent factors instead of real-valued factors. Lee et al. [2016] make low-rank decomposition of local sub-blocks of the matrix separately before summing them together, and show it improves prediction accuracy for collaborative filtering. Many proposals have been made to distribute stochastic gradient descent for matrix and tensor factorizations [Yu et al., 2012; Yun et al., 2014; Gemulla et al., 2011; Niu et al., 2011], which allow to scale to always bigger problems.

2.3.2 Complex Numbers in Factorization Methods

When factorization methods are applied, the representation of the decomposition is generally chosen in accordance with the data, despite the fact that most real square matrices only have eigenvalues in the complex domain. Indeed in the machine learning community, the data is usually real-valued, and thus eigendecomposition is used for symmetric matrices, or other decompositions such as (real-valued) singular value decomposition [Beltrami, 1873], non-negative matrix factorization [Paatero and Tapper, 1994], or canonical polyadic decomposition when it comes to tensors [Hitchcock, 1927]. Conversely, in signal processing, data is often complex-valued [Stoica and Moses, 2005] and the complex-valued counterparts of these decompositions are then used. Joint diagonalization is also a much more common tool than in machine learning for decomposing sets of (complex) dense square matrices [Belouchrani et al., 1997; De Lathauwer et al., 2001]. Classic complex matrix decompositions and their properties are clearly exposed in Horn and Johnson [2012].

Some little-known work in analysis of dense square matrices relates to our contribution, as they consider complex-valued spectral models for asymmetric real-valued square matrices [Chino, 2002]. In particular, Escoufier and Grorud [1980] proposed to encode real-valued square matrices as complex-valued Hermitian matrices, where the real-part corresponds to the symmetric part of the real-valued matrix, and the imaginary part corresponds to the the antisymmetric part of the real-valued matrix.

Some works on recommender systems use complex numbers as an encoding facility, to merge two real-valued relations, similarity and liking, into one single complex-valued

matrix which is then decomposed with complex embeddings [Kunegis et al., 2012; Xie et al., 2015]. Still, unlike our work, it is not real data that is decomposed in the complex domain. In deep learning, Danihelka et al. [2016] proposed an long short-term memory network extension with an associative memory based on complex-valued vectors for memorization tasks, and Hu et al. [2016] a complex-valued neural network for speech synthesis. In both cases again, the data is first encoded in complex vectors that are then fed into the network.

Conversely to these contributions, this work suggests that processing real-valued data with complex-valued representations, through a projection onto the real-valued subspace, can be a very simple way of increasing the expressiveness of the model considered.

Chapter 3

Complex-Valued Tensor Factorization and Completion

In this chapter we describe a new tensor factorization and completion model, based on complex-valued factor matrices. Each row in these matrices represents one entity or one relations, these vectors are called *embeddings*. In the previous chapter, we have seen that recent proposals resorts to more and more complicated scoring function to increase their expressiveness. Here we argue that the standard dot product between embeddings can be a very effective scoring function, provided that one uses the right *representation*: instead of using embeddings containing real numbers, we discuss and demonstrate the capabilities of complex embeddings. When using complex vectors, that is vectors with entries in $\mathbb{C}$, the dot product is often called the *Hermitian* (or sesquilinear) dot product, as it involves the conjugate-transpose of one of the two vectors. As a consequence, the dot product is not symmetric any more, and facts about one relation can receive different scores depending on the ordering of the entities involved in the fact. In summary, complex embeddings naturally represent arbitrary relations while retaining the efficiency of a dot product, that is linearity in both space and time complexity.

We first provide justification and intuition for using complex embeddings in the square matrix case, where there is only a single type of relation between entities, and show the existence of the proposed decomposition for all possible relations. The formulation is then extended to a stacked set of square matrices in a third-order tensor to represent multiple relations. We then describe a stochastic gradient descent algorithm to learn the model on partially-observed tensors, where we present an equivalent reformulation of the proposed model that involves only real embeddings. This should help practitioners when implementing our method, without requiring the use of complex numbers in their

software implementation. Finally, we study the theoretical links with a simultaneously and independently proposed model, HOLE [Nickel et al., 2016b].

3.1 Relations as the Real Parts of Low-Rank Normal Matrices

We consider in this section a simplified link prediction task with a single relation, and introduce complex embeddings for low-rank matrix factorization.

We will first discuss the desired properties of embedding models, show how this problem relates to the spectral theorems, and discuss the classes of matrices these theorems encompass in the real and in the complex case. We then propose a new matrix decomposition—to the best of our knowledge—and a proof of its existence for all real square matrices. Finally we discuss the rank of the proposed decomposition.

3.1.1 Modeling Relations

Let $\mathcal{E}$ be a set of entities, with $n := N_e = |\mathcal{E}|$ to have lighter notations in this chapter. The truth of the single relation holding between two entities is represented by a sign value $y_{so} \in \{-1, 1\}$, where 1 represents true facts and -1 false facts, $s \in \mathcal{E}$ is the subject entity and $o \in \mathcal{E}$ is the object entity. The probability for the relation holding true is given by

$$P(y_{so} = 1) = \sigma(x_{so}) \tag{3.1}$$

where $X \in \mathbb{R}^{n \times n}$ is a latent matrix of scores indexed by the subject (rows) and object entities (columns), Y is a partially-observed sign matrix indexed in identical fashion, and σ is a suitable sigmoid function. Throughout this manuscript we use the logistic inverse link function $\sigma(x) = \frac{1}{1+\mathrm{e}^{-x}}$.

3.1.1.1 Handling Both Asymmetry and Unique Entity Embeddings

In this work we pursue three objectives: finding a generic structure for X that leads to (*i*) a computationally efficient model, (*ii*) an expressive enough approximation of common relations in real world knowledge graphs, and (*iii*) good generalization performances in practice. Standard matrix factorization approximates X by a matrix product $UV^\top$, where U and V are two functionally-independent $n \times K$ matrices, K being the rank of the matrix. Within this formulation it is assumed that entities appearing as subjects are different from entities appearing as objects. This extensively studied type of model

is closely related to the singular value decomposition (SVD) and fits well with the case where the matrix X is rectangular, as explained in Section 2.3.1.

However, in many knowledge graph completion problems, the same entity i can appear as both subject *or* object and will have two different embedding vectors, u_i and v_i, depending on whether it appears as subject or object of a relation. It seems natural to learn unique embeddings of entities, as initially proposed by Nickel et al. [2011] and Bordes et al. [2011] and since then used systematically in other prominent approaches [Bordes et al., 2013b; Yang et al., 2015; Socher et al., 2013]. In the factorization setting, using the same embeddings for left- and right-side factors boils down to a specific case of eigenvalue decomposition: *orthogonal diagonalization*.

Definition 1. A real square matrix $X \in \mathbb{R}^{n \times n}$ is orthogonally diagonalizable if it can be written as $X = EWE^\top$, where $E, W \in \mathbb{R}^{n \times n}$, W is diagonal, and E orthogonal so that $EE^\top = E^\top E = I$ where I is the identity matrix.

The spectral theorem for symmetric matrices tells us that a matrix is orthogonally diagonalizable if and only if it is symmetric [Cauchy, 1829]. It is therefore often used to approximate covariance matrices, kernel functions and distance or similarity matrices.

However as previously stated, this paper is explicitly interested in problems where matrices—and thus the relation patterns they represent—can also be antisymmetric, or even not have any particular symmetry pattern at all (asymmetry). In order to both use a unique embedding for entities and extend the expressiveness to asymmetric relations, researchers have generalised the notion of dot products to *scoring functions*, also known as *composition functions*, that allow more general combinations of embeddings. We recall several examples of scoring functions in Table 2.2.

These models propose different trade-offs between the three essential points:

- Expressiveness, which is the ability to represent symmetric, antisymmetric and more generally asymmetric relations.
- Scalability, which means keeping linear time and space complexity scoring function.
- Generalization, for which having unique entity embeddings is critical.

RESCAL [Nickel et al., 2011] and NTN [Socher et al., 2013] are very expressive, but their scoring functions have quadratic complexity in the rank of the factorization. DistMult [Yang et al., 2015] can be seen as a joint orthogonal diagonalization with real embeddings, hence handling only symmetric relations. Conversely, TransE [Bordes et al., 2013b] handles symmetric relations to the price of strong constraints on its entity embeddings, as

explained in the previous chapter. The canonical-polyadic decomposition (CP) [Hitchcock, 1927] generalizes poorly with its different embeddings for entities as subject and as object.

We reconcile expressiveness, scalability and generalization by going back to the realm of well-studied matrix factorizations, and making use of complex linear algebra, a scarcely used tool in the machine learning community.

3.1.1.2 Decomposition in the Complex Domain

We introduce a new decomposition of real square matrices using unitary diagonalization, the generalization of orthogonal diagonalization to complex matrices. This allows decomposition of *arbitrary* real square matrices with unique representations of rows and columns.

Let us first recall some notions of complex linear algebra as well as specific cases of diagonalization of real square matrices, before building our proposition upon these results.

A complex-valued vector $x \in \mathbb{C}^K$, with $x = \mathrm{Re}(x) + i\mathrm{Im}(x)$ is composed of a real part $\mathrm{Re}(x) \in \mathbb{R}^K$ and an imaginary part $\mathrm{Im}(x) \in \mathbb{R}^K$, where i denotes the square root of -1. The conjugate $\overline{x}$ of a complex vector inverts the sign of its imaginary part: $\overline{x} = \mathrm{Re}(x) - i\mathrm{Im}(x)$.

Conjugation appears in the usual dot product for complex numbers, called the *Hermitian* product, or *sesquilinear* form, which is defined as:

$$\begin{aligned} \langle u, v \rangle &:= \bar{u}^\top v \\ &= \mathrm{Re}(u)^\top \mathrm{Re}(v) + \mathrm{Im}(u)^\top \mathrm{Im}(v) \\ &\quad + i(\mathrm{Re}(u)^\top \mathrm{Im}(v) - \mathrm{Im}(u)^\top \mathrm{Re}(v)) \,. \end{aligned}$$

A simple way to justify the Hermitian product for composing complex vectors is that it provides a valid topological norm in the induced vector space. For example, $\bar{x}^\top x = 0$ implies $x = 0$ while this is not the case for the bilinear form $x^\top x$ as there are many complex vectors x for which $x^\top x = 0$.

This yields an interesting property of the Hermitian product concerning the order of the involved vectors: $\langle u, v \rangle = \overline{\langle v, u \rangle}$, meaning that the real part of the product is symmetric, while the imaginary part is antisymmetric.

For matrices, we shall write $X^* \in \mathbb{C}^{n \times m}$ for the conjugate-transpose $X^* = (\overline{X})^\top = \overline{X^\top}$. The conjugate transpose is also often written $X^\dagger$ or X^{H}.

Definition 2. A complex square matrix $X \in \mathbb{C}^{n \times n}$ is unitarily diagonalizable if it can be written as $X = EWE^*$, where $E, W \in \mathbb{C}^{n \times n}$, W is diagonal, and E is unitary such that $EE^* = E^*E = I$.

Definition 3. A complex square matrix X is normal if it commutes with its conjugate-transpose so that $XX^* = X^*X$.

We can now state the spectral theorem for normal matrices.

Theorem 1 (Spectral theorem for normal matrices, von Neumann [1929]). Let X be a complex square matrix. Then X is unitarily diagonalizable if and only if X is normal.

It is easy to check that all real symmetric matrices are normal, and have pure real eigenvectors and eigenvalues. But the set of purely real normal matrices also includes all real antisymmetric matrices (useful to model hierarchical relations such as `IsOlder`), as well as all real orthogonal matrices (including permutation matrices), and many other matrices that are useful to represent binary relations, such as assignment matrices which represent bipartite graphs. However, far from all matrices expressed as $X = EWE^*$ are purely real, and Equation (3.1) requires the scores X to be purely real.

As we only focus on *real* square matrices in this work, let us summarize all the cases where X is real square and $X = EWE^*$ if X is unitarily diagonalizable, where $E, W \in \mathbb{C}^{n \times n}$, W is diagonal and E is unitary:

- X is symmetric if and only if X is orthogonally diagonalizable and E and W are purely real.
- X is normal and non-symmetric if and only if X is unitarily diagonalizable and E and W are *not* both purely real.
- X is not normal if and only if X is not unitarily diagonalizable.

We generalize all three cases by showing that, for any $X \in \mathbb{R}^{n \times n}$, there exists a unitary diagonalization in the complex domain, of which the real part equals X:

$$X = \mathrm{Re}(EWE^*)\,. \tag{3.2}$$

In other words, the unitary diagonalization is projected onto the real subspace.

Theorem 2. Suppose $X \in \mathbb{R}^{n \times n}$ is a real square matrix. Then there exists a normal matrix $Z \in \mathbb{C}^{n \times n}$ such that $\mathrm{Re}(Z) = X$.

Proof. Let $Z := X + iX^\top$. Then

$$Z^* = X^\top - iX = -i(iX^\top + X) = -iZ\,,$$

so that

$$ZZ^* = Z(-iZ) = (-iZ)Z = Z^*Z\,.$$

Therefore Z is normal. □

Note that there also exists a normal matrix $Z = X^\top + iX$ such that $\mathrm{Im}(Z) = X$.

Following Theorem 1 and Theorem 2, any real square matrix can be written as the real part of a complex diagonal matrix through a unitary change of basis.

Corollary 1. Suppose $X \in \mathbb{R}^{n\times n}$ is a real square matrix. Then there exist $E, W \in \mathbb{C}^{n\times n}$, where E is unitary, and W is diagonal, such that $X = \mathrm{Re}(EWE^*)$.

Proof. From Theorem 2, we can write $X = \mathrm{Re}(Z)$, where Z is a normal matrix, and from Theorem 1, Z is unitarily diagonalizable. □

Applied to the knowledge graph completion setting, the rows of E here are vectorial representations of the entities corresponding to rows and columns of the relation score matrix X. The score for the relation holding true between entities s and o is hence

$$x_{so} = \mathrm{Re}(e_s^\top W \bar{e}_o) \tag{3.3}$$

where $e_s, e_o \in \mathbb{C}^n$ and $W \in \mathbb{C}^{n\times n}$ is diagonal. For a given entity, its subject embedding vector is the complex conjugate of its object embedding vector.

To illustrate this difference of expressiveness with respect to real-valued embeddings, let us consider two complex embeddings $e_s, e_o \in \mathbb{C}$ of dimension 1, with arbitrary values: $e_s = 1 - 2i$, and $e_o = -3 + i$; as well as their real-valued, twice-bigger counterparts: $e'_s = \left(\begin{smallmatrix}1\\-2\end{smallmatrix}\right) \in \mathbb{R}^2$ and $e'_o = \left(\begin{smallmatrix}-3\\1\end{smallmatrix}\right) \in \mathbb{R}^2$. In the real-valued case, that corresponds to the DistMult model [Yang et al., 2015], the score is $x_{so} = e'^\top_s W' e'_o$. Figure 3.1 represents the heatmaps of the scores x_{so} and x_{os}, as a function of $W \in \mathbb{C}$ in the complex-valued case, and as a function of $W' \in \mathbb{R}^2$ diagonal in the real-valued case. In the real-valued case, that is symmetric in the subject and object entities, the scores x_{so} and x_{os} are equal for any value of $W' \in \mathbb{R}^2$ diagonal. Whereas in the complex-valued case, the variation of $W \in \mathbb{C}$ allows to score x_{so} and x_{os} with any desired pair of values.

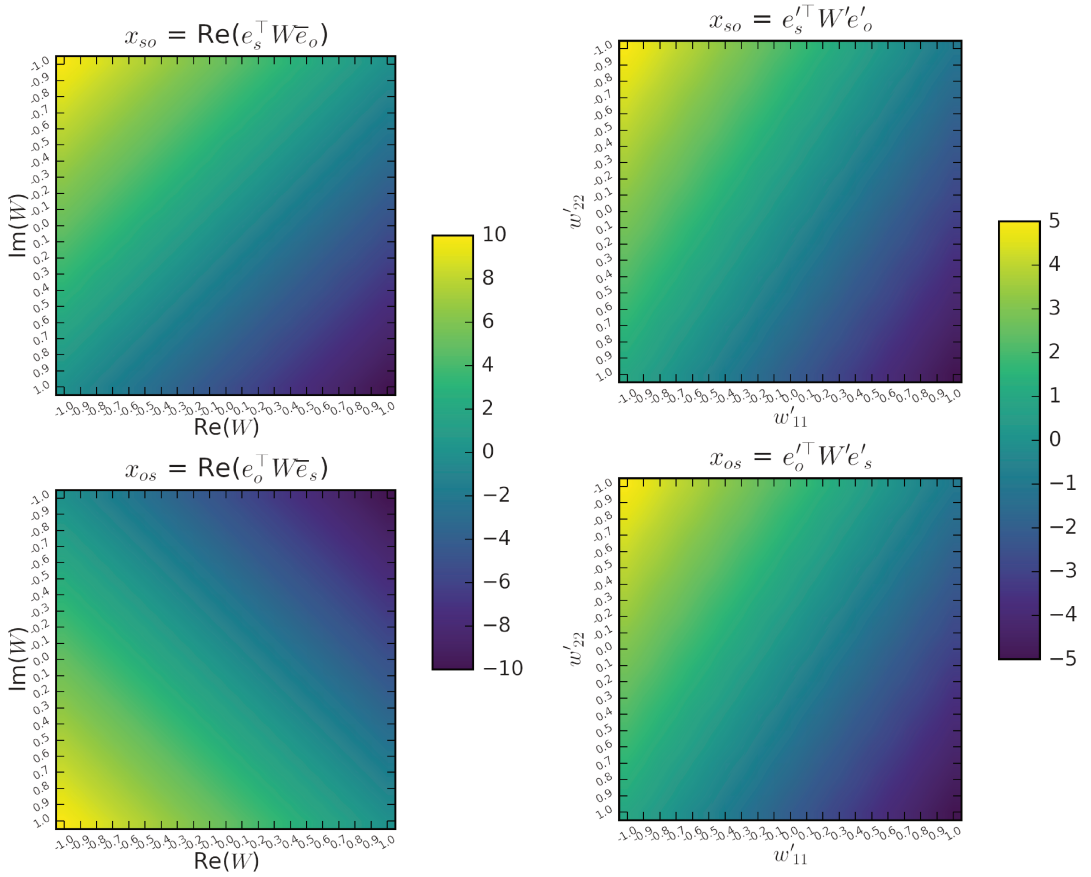


FIGURE 3.1: Left: Scores $x_{so} = \mathrm{Re}(e_s^\top W \overline{e}_o)$ (top) and $x_{os} = \mathrm{Re}(e_o^\top W \overline{e}_s)$ (bottom) for the proposed complex-valued decomposition, plotted as a function of $W \in \mathbb{C}$, for fixed entity embeddings $e_s = 1 - 2i$, and $e_o = -3 + i$. Right: Scores $x_{so} = e_s'^\top W' e_o'$ (top) and $x_{os} = e_o'^\top W' e_s'$ (bottom) for the corresponding real-valued decomposition with the same number of free real-valued parameters (*i.e.* in twice the dimension), plotted as a function of $W' \in \mathbb{R}^2$ diagonal, for fixed entity embeddings $e_s' = \left(\begin{smallmatrix} 1 \\ -2 \end{smallmatrix}\right)$ and $e_o' = \left(\begin{smallmatrix} -3 \\ 1 \end{smallmatrix}\right)$. By varying $W \in \mathbb{C}$, the proposed complex-valued decomposition can attribute any pair of scores to x_{so} and x_{os}, whereas $x_{so} = x_{os}$ for all $W' \in \mathbb{R}^2$ with the real-valued decomposition.

This decomposition however is non-unique, a simple example of this non-uniqueness is obtained by adding a purely imaginary constant to the eigenvalues. Let $X \in \mathbb{R}^{n \times n}$, and $X = \mathrm{Re}(EWE^*)$ where E is unitary, W is diagonal. Then for any real constant $c \in \mathbb{R}$ we have:

$$\begin{aligned} X &= \mathrm{Re}(E(W + icI)E^*) \\ &= \mathrm{Re}(EWE^* + icEIE^*) \\ &= \mathrm{Re}(EWE^* + icI) \\ &= \mathrm{Re}(EWE^*)\,. \end{aligned}$$

In general, there are many other possible couples of matrices E and W that preserve the real part of the decomposition. In practice however this is no synonym of low generalization abilities, as many effective matrix and tensor decomposition methods used in machine learning lead to non-unique solutions [Paatero and Tapper, 1994; Nickel et al., 2011]. In this case also, the learned representations prove useful as shown in the experimental section.

3.1.2 Low-Rank Decomposition

Addressing knowledge graph completion with data-driven approaches assumes that there is a sufficient regularity in the observed data to generalize to unobserved facts. When formulated as a matrix completion problem, as it is the case in this section, one way of implementing this hypothesis is to make the assumption that the matrix has low rank or approximately low rank. We first discuss the rank of the proposed decomposition, and then introduce the sign-rank and extend the bound developed on the rank to the sign-rank.

3.1.2.1 Rank Upper Bound

First, we recall one definition of the rank of a matrix [Horn and Johnson, 2012].

Definition 4. The rank of an m-by-n complex matrix $\text{rank}(X) = \text{rank}(X^\top) = k$, if X has exactly k linearly independent columns.

Also note that if X is diagonalizable so that $X = EWE^{-1}$ with $\text{rank}(X) = k$, then W has k non-zero diagonal entries for some diagonal W and some invertible matrix E. From this it is easy to derive a known additive property of the rank:

$$\text{rank}(B + C) \leq \text{rank}(B) + \text{rank}(C) \tag{3.4}$$

where $B, C \in \mathbb{C}^{m \times n}$.

We now show that any rank k real square matrix can be reconstructed from a $2k$-dimensional unitary diagonalization.

Corollary 2. Suppose $X \in \mathbb{R}^{n \times n}$ and $rank(X) = k$. Then there exist $E \in \mathbb{C}^{n \times 2k}$ such that the columns of E form an orthonormal basis of $\mathbb{C}^{2k}$, $W \in \mathbb{C}^{2k \times 2k}$ is diagonal, and $X = \text{Re}(EWE^*)$.

Proof. Consider the complex square matrix $Z := X + iX^\top$. We have $\text{rank}(iX^\top) = \text{rank}(X^\top) = \text{rank}(X) = k$.

From Equation (3.4), $\text{rank}(Z) \leq \text{rank}(X) + \text{rank}(iX^\top) = 2k$.

The proof of Theorem 2 shows that Z is normal. Thus $Z = EWE^*$ with $E \in \mathbb{C}^{n \times 2k}$, $W \in \mathbb{C}^{2k \times 2k}$ where the columns of E form an orthonormal basis of $\mathbb{C}^{2k}$, and W is diagonal. □

Since E is not necessarily square, we replace the unitary requirement of Corollary 1 by the requirement that its columns form an orthonormal basis of its smallest dimension, $2k$.

Also, given that such decomposition always exists in dimension n (Theorem 2), this upper bound is not relevant when $\text{rank}(X) \geq \frac{n}{2}$.

3.1.2.2 Sign-Rank Upper Bound

Since we encode the truth values of each fact with ± 1, we deal with square *sign matrices*: $Y \in \{-1, 1\}^{n \times n}$. Sign matrices have an alternative rank definition, the *sign-rank*.

Definition 5. The sign-rank $\text{rank}_\pm(Y)$ of an m-by-n sign matrix Y, is the rank of the m-by-n real matrix of least rank that has the same sign-pattern as Y, so that

$$\text{rank}_\pm(Y) := \min_{X \in \mathbb{R}^{m \times n}} \{\text{rank}(X) \,|\, \text{sign}(X) = Y\},$$

where $\text{sign}(X)_{ij} = \text{sign}(x_{ij})$.

We define the *sign function* of $c \in \mathbb{R}$ as

$$\text{sign}(c) = \begin{cases} 1 & \text{if } c \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

where the value $c = 0$ is here arbitrarily assigned to 1 to allow zero entries in X, conversely to the stricter usual definition of the sign-rank.

To make generalization possible, we hypothesize that the true matrix Y has a low sign-rank, and thus can be reconstructed by the sign of a low-rank score matrix X. The low sign-rank assumption is theoretically justified by the fact that the sign-rank is a natural complexity measure of sign matrices [Linial et al., 2007a] and is linked to learnability [Alon et al., 2016] and empirically confirmed by the wide success of factorization models [Nickel et al., 2016a].

Using Corollary 2, we can now show that any square sign matrix of sign-rank k can be reconstructed from a rank $2k$ unitary diagonalization.

Corollary 3. Suppose $Y \in \{-1, 1\}^{n \times n}$, $\text{rank}_{\pm}(Y) = k$. Then there exists $E \in \mathbb{C}^{n \times 2k}$, $W \in \mathbb{C}^{2k \times 2k}$ where the columns of E form an orthonormal basis of $\mathbb{C}^{2k}$, and W is diagonal, such that $Y = \text{sign}(\text{Re}(EWE^*))$.

Proof. By definition, if $\text{rank}_{\pm}(Y) = k$, there exists a real square matrix X such that $\text{rank}(X) = k$ and $\text{sign}(X) = Y$. From Corollary 2, $X = \text{Re}(EWE^*)$ where $E \in \mathbb{C}^{n \times 2k}$, $W \in \mathbb{C}^{2k \times 2k}$ where the columns of E form an orthonormal basis of $\mathbb{C}^{2k}$, and W is diagonal. □

Previous attempts to approximate the sign-rank in relational learning did not use complex numbers. Previous work showed the existence of compact factorizations under conditions on the sign matrix [Nickel et al., 2014]. Our results show that if a square sign matrix has sign-rank k, then it can be exactly decomposed through a $2k$-dimensional unitary diagonalization.

Although we can only show the existence of a complex decomposition of rank $2k$ for a matrix with sign-rank k, the sign rank of Y is often *much* lower than the rank of Y, as we do not know any matrix $Y \in \{-1, 1\}^{n \times n}$ for which $\text{rank}_{\pm}(Y) > \sqrt{n}$ [Alon et al., 2016]. For example, the $n \times n$ identity matrix has rank n, but its sign-rank is only 3! By swapping the columns $2j$ and $2j - 1$ for j in $1, \ldots, \frac{n}{2}$, the identity matrix corresponds to the relation `marriedTo`, a relation known to be hard to factorize over the reals [Nickel et al., 2014], since the rank is invariant by row/column permutations. Yet our model can express it at most in rank 6, for any n.

Hence, by enforcing a low-rank $K \ll n$ on EWE^*, individual relation scores $x_{so} = \text{Re}(e_s^\top W \bar{e}_o)$ between entities s and o can be efficiently predicted, as $e_s, e_o \in \mathbb{C}^K$ and $W \in \mathbb{C}^{K \times K}$ is diagonal.

Finding the K that matches the sign-rank of Y corresponds to finding the smallest K that brings the 0–1 loss on X to 0, as link prediction can be seen as binary classification of the facts. In practice, and as classically done in machine learning to avoid this NP-hard problem, we use a continuous surrogate of the 0–1 loss, in this case the logistic loss as described in Section 3.3, and validate models on different values of K, as described in Chapter 4.

3.1.2.3 Rank Bound Discussion

Corollaries 2 and 3 use the aforementioned subadditive property of the rank to derive the $2k$ upper bound. Let us give an example for which this bound is strictly greater than k.

Consider the following 2-by-2 sign matrix:

$$Y = \begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix}.$$

Not only is this matrix not normal, but one can also easily check that there is no *real* normal 2-by-2 matrix that has the same sign-pattern as Y. Clearly, Y is a rank 1 matrix since its columns are linearly dependent, hence its sign-rank is also 1. From Corollary 3, we know that there is a normal matrix whose real part has the same sign-pattern as Y, and whose rank is at most 2.

However, there is no rank 1 unitary diagonalization of which the real part equals Y. Otherwise we could find a 2-by-2 complex matrix Z such that $\mathrm{Re}(z_{11}) < 0$ and $\mathrm{Re}(z_{22}) > 0$, where $z_{11} = e_1 w \bar{e}_1 = w|e_1|^2$, $z_{22} = e_2 w \bar{e}_2 = w|e_2|^2$, $e \in \mathbb{C}^2, w \in \mathbb{C}$. This is obviously unsatisfiable. This example generalizes to any n-by-n square sign matrix that only has -1 on its first row and is hence rank 1, the same argument holds considering $\mathrm{Re}(z_{11}) < 0$ and $\mathrm{Re}(z_{nn}) > 0$.

This example shows that the upper bound on the rank of the unitary diagonalization showed in Corollaries 2 and 3 can be strictly greater than k, the rank or sign-rank, of the decomposed matrix. However, there might be other examples for which the addition of an imaginary part could—additionally to making the matrix normal—create some linear dependence between the rows/columns and thus decrease the rank of the matrix, up to a factor of 2.

We summarize this section in three points:

1. The proposed factorization encompasses all possible score matrices X for a single binary relation.
2. By construction, the factorization is well suited to represent both symmetric and antisymmetric relations.
3. Relation patterns can be efficiently approximated with a low-rank factorization using complex-valued embeddings.

3.2 Extension to Multi-Relational Data

Let us now extend the previous discussion to models with multiple relations. Let $\mathcal{R}$ be the set of relations, with $m := N_r = |\mathcal{R}|$. We shall now write $\mathbf{X} \in \mathbb{R}^{m \times n \times n}$ for the score

tensor, $X_r \in \mathbb{R}^{n \times n}$ for the score matrix of the relation $r \in \mathcal{R}$, and $\mathbf{Y} \in \{-1, 1\}^{m \times n \times n}$ for the partially-observed sign tensor.

Given one relation $r \in \mathcal{R}$ and two entities $s, o \in \mathcal{E}$, the probability that the fact *r(s,o)* is true given by

$$P(y_{rso} = 1) = \sigma(x_{rso}) = \sigma(\phi(r, s, o; \Theta)) \tag{3.5}$$

where ϕ is the scoring function of the model considered and Θ denotes the model parameters. We recall that we denote the set of all possible facts (or triples) for a knowledge graph by $\mathcal{T} = \mathcal{R} \times \mathcal{E} \times \mathcal{E}$. While the tensor $\mathbf{X}$ as a whole is unknown, we assume that we observe a set of true and false triples $\Omega = \{((r, s, o), y_{rso}) \,|\, (r, s, o) \in \mathcal{T}_\Omega\}$ where $y_{rso} \in \{-1, 1\}$ and $\mathcal{T}_\Omega \subset \mathcal{T}$ is the set of observed triples. The goal is to find the probabilities of entries $y_{r's'o'}$ for a set of targeted unobserved triples $\{(r', s', o') \in \mathcal{T} \setminus \mathcal{T}_\Omega\}$.

Depending on the scoring function $\phi(r, s, o; \Theta)$ used to model the score tensor $\mathbf{X}$, we obtain different models. Examples of scoring functions are given in Table 2.2.

3.2.1 Complex Factorization Extension to Tensors

The single-relation model is extended by jointly factorizing all the square matrices of scores into a 3$^{\text{rd}}$-order tensor $\mathbf{X} \in \mathbb{R}^{m \times n \times n}$, with a different diagonal matrix $W_r \in \mathbb{C}^{K \times K}$ for each relation r, and by sharing the entity embeddings $E \in \mathbb{C}^{n \times K}$ across all relations:

$$\begin{aligned} \phi(r, s, o; \Theta) &= \text{Re}(e_s^\top W_r \bar{e}_o) \\ &= \text{Re}(\sum_{k=1}^{K} w_{rk} e_{sk} \bar{e}_{ok}) \\ &= \text{Re}(\langle w_r, e_s, \bar{e}_o \rangle) \end{aligned} \tag{3.6}$$

where K is the rank hyperparameter, $e_s, e_o \in \mathbb{C}^K$ are the rows in E corresponding to the entities s and o, $w_r = \text{diag}(W_r) \in \mathbb{C}^K$ is a complex vector, and $\langle a, b, c \rangle := \sum_k a_k b_k c_k$ is the component-wise multilinear dot product[1]. For this scoring function, the set of parameters Θ is $\{e_i, w_r \in \mathbb{C}^K, i \in \mathcal{E}, r \in \mathcal{R}\}$. This resembles the real part of a complex matrix decomposition as in the single-relation case discussed above. However, we now have a different vector of eigenvalues for every relation. Expanding the real part of this

[1] This is not the Hermitian extension of the multilinear dot product as there appears to be no standard definition of the Hermitian multilinear product in the linear algebra litterature.

product gives:

$$\begin{aligned}\mathrm{Re}(\langle w_r, e_s, \bar{e}_o\rangle) \;=\;& \langle \mathrm{Re}(w_r), \mathrm{Re}(e_s), \mathrm{Re}(e_o)\rangle \\ &+ \langle \mathrm{Re}(w_r), \mathrm{Im}(e_s), \mathrm{Im}(e_o)\rangle \\ &+ \langle \mathrm{Im}(w_r), \mathrm{Re}(e_s), \mathrm{Im}(e_o)\rangle \\ &- \langle \mathrm{Im}(w_r), \mathrm{Im}(e_s), \mathrm{Re}(e_o)\rangle \;. \end{aligned} \tag{3.7}$$

These equations provide two interesting views of the model:

- *Changing the representation*: Equation (3.6) would correspond to DistMult with real embeddings (see Table 2.2), but handles asymmetry thanks to the complex conjugate of the object-entity embedding.
- *Changing the scoring function*: Equation (3.7) only involves real vectors corresponding to the real and imaginary parts of the embeddings and relations.

By separating the real and imaginary parts of the relation embedding w_r as shown in Equation (3.7), it is apparent that these parts naturally act as weights on each latent dimension: $\mathrm{Re}(w_r)$ over the real part of $\langle e_o, e_s\rangle$ which is symmetric, and $\mathrm{Im}(w)$ over the imaginary part of $\langle e_o, e_s\rangle$ which is antisymmetric.

Indeed, the decomposition of each score matrix X_r for each $r \in \mathcal{R}$ can be written as the sum of a symmetric matrix and an antisymmetric matrix. To see this, let us rewrite the decomposition of each score matrix X_r in matrix notation. We write the real part of matrices with primes $E' = \mathrm{Re}(E)$ and imaginary parts with double primes $E'' = \mathrm{Im}(E)$:

$$\begin{aligned} X_r &= \mathrm{Re}(EW_rE^*) \\ &= \mathrm{Re}((E' + iE'')(W'_r + iW''_r)(E' - iE'')^\top) \\ &= (E'W'_rE'^\top + E''W'_rE''^\top) + (E'W''_rE''^\top - E''W''_rE'^\top)\,. \end{aligned} \tag{3.8}$$

It is trivial to check that the matrix $E'W'_rE'^\top + E''W'_rE''^\top$ is symmetric and that the matrix $E'W''_rE''^\top - E''W''_rE'^\top$ is antisymmetric. Hence this model is well suited to model jointly symmetric and antisymmetric relations between pairs of entities, while still using the same entity representations for subjects and objects. When learning, it simply needs to collapse $W''_r = \mathrm{Im}(W_r)$ to zero for symmetric relations $r \in \mathcal{R}$, and $W'_r = \mathrm{Re}(W_r)$ to zero for antisymmetric relations $r \in \mathcal{R}$, as X_r is indeed symmetric when W_r is purely real, and antisymmetric when W_r is purely imaginary.

From a geometrical point of view, each relation embedding w_r is an anisotropic scaling of the basis defined by the entity embeddings E, followed by a projection onto the real subspace.

3.2.2 Existence of the Tensor Factorization

Let us first discuss the existence of the multi-relational model where the rank of the decomposition $K \leq n$, which relates to simultaneous unitary decomposition.

Definition 6. A family of matrices $X_1, \ldots, X_m \in \mathbb{C}^{n \times n}$ is simultaneously unitarily diagonalizable, if there is a single unitary matrix $E \in \mathbb{C}^{n \times n}$, such that $X_i = EW_iE^*$ for all i in $1, \ldots, m$, where $W_i \in \mathbb{C}^{n \times n}$ are diagonal.

Definition 7. A family of normal matrices $X_1, \ldots, X_m \in \mathbb{C}^{n \times n}$ is a commuting family of normal matrices, if $X_iX_j^* = X_i^*X_j$, for all i, j in $1, \ldots, m$.

Theorem 3 (see Horn and Johnson [2012]). Suppose $\mathcal{F}$ is the family of matrices $X_1, \ldots, X_m \in \mathbb{C}^{n \times n}$. Then $\mathcal{F}$ is a commuting family of normal matrices if and only if $\mathcal{F}$ is simultaneously unitarily diagonalizable.

To apply Theorem 3 to the proposed factorization, we would have to make the hypothesis that the relation score matrices X_r are a commuting family, which is too strong a hypothesis. Actually, the model is slightly different since we take only the real part of the tensor factorization. In the single-relation case, taking only the real part of the decomposition rids us of the normality requirement of Theorem 1 for the decomposition to exist, as shown in Theorem 2.

In the multiple-relation case, it is an open question whether taking the real part of the simultaneous unitary diagonalization will enable us to decompose families of arbitrary real square matrices—that is with a single unitary matrix E that has *at most* n columns. Though it seems unlikely, we could not find a counter-example yet.

However, by letting the rank of the tensor factorization K to be greater than n, we can show that the proposed tensor decomposition exists for families of arbitrary real square matrices, by simply concatenating the decomposition of Theorem 2 of each real square matrix X_i.

Theorem 4. Suppose $X_1, \ldots, X_m \in \mathbb{R}^{n \times n}$. Then there exists $E \in \mathbb{C}^{n \times nm}$ and $W_i \in \mathbb{C}^{nm \times nm}$ are diagonal, such that $X_i = \mathrm{Re}(EW_iE^*)$ for all i in $1, \ldots, m$.

Proof. From Theorem 2 we have $X_i = \mathrm{Re}(E_iW_iE_i^*)$, where $W_i \in \mathbb{C}^{n \times n}$ is diagonal, and each $E_i \in \mathbb{C}^{n \times n}$ is unitary for all i in $1, \ldots, m$.

Let $E = [E_1 \dots E_m]$, and

$$\Lambda_i = \begin{bmatrix} \mathbf{0}^{((i-1)n)\times((i-1)n)} & & \\ & W_i & \\ & & \mathbf{0}^{((m-i)n)\times((m-i)n)} \end{bmatrix}$$

where $\mathbf{0}^{l\times l}$ the zero $l \times l$ matrix. Therefore $X_i = \mathrm{Re}(E\Lambda_i E^*)$ for all i in $1, \dots, m$. □

By construction, the rank of the decomposition is at most nm. When $m \leq n$, this bound actually matches the general upper bound on the rank of the canonical polyadic (CP) decomposition [Hitchcock, 1927; Kruskal, 1989]. Since m corresponds to the number of relations and n to the number of entities, m is always smaller than n in real world knowledge graphs, hence the bound holds in practice.

Though when it comes to relational learning, we might expect the actual rank to be much lower than nm for two reasons. The first one, as discussed above, is that we are dealing with sign tensors, hence the rank of the matrices X_r need only match the sign-rank of the partially-observed matrices Y_r. The second one is that the matrices are related to each other, as they all represent the same entities in different relations, and thus benefit from sharing latent dimensions. As opposed to the construction exposed in the proof of Theorem 4, where other relations dimensions are canceled out. In practice, the rank needed to generalize well is indeed much lower than nm as we show experimentally in Figure 4.7.

Also, note that with the construction of the proof of Theorem 4, the matrix $E = [E_1 \dots E_m]$ is not unitary any more. However the unitary constraints in the matrix case serve only the proof of existence, which is just one solution among the infinite ones of same rank. In practice, imposing orthonormality is essentially a numerical commodity for the decomposition of dense matrices, through iterative methods for example [Saad, 1992]. When it comes to matrix and tensor completion, and thus generalisation, imposing such constraints is more of a numerical hassle than anything else, especially for gradient methods. As there is no apparent link between orthonormality and generalisation properties, we did not impose these constraints when learning the model.

3.3 Algorithm

Algorithm 1 describes stochastic gradient descent (SGD) to learn the proposed multi-relational model with the AdaGrad learning-rate updates [Duchi et al., 2011]. Stochastic gradient descent is a natural and scalable way of respecting the open-world assumption,

that is treating missing triples as missing instead of negatives. We refer to the proposed model as ComplEx, for Complex Embeddings. We expose a version of the algorithm that uses only real-valued vectors, in order to facilitate its implementation. To do so, we use separate real-valued representations of the real and imaginary parts of the embeddings.

These real and imaginary part vectors are initialized with vectors having a zero-mean normal distribution with unit variance. If the training set Ω contains only positive triples, negatives are generated for each batch using the *local closed-world assumption* as in Bordes et al. [2013b]. That is, for each triple, we randomly change either the subject or the object, to form a negative example. In this case the parameter $\eta > 0$ sets the number of negative triples to generate for each positive triple. Collision with positive triples in Ω is not checked, as it occurs rarely in real world knowledge graphs as they are largely sparse, and may also be computationally expensive.

Squared gradients are accumulated to compute AdaGrad learning rates, then gradients are updated. Every s iterations, the parameters Θ are evaluated over the evaluation set Ω_v (*evaluate_AP_or_MRR*$(\Omega_v; \Theta)$ function in Algorithm 1). If the data set contains both positive and negative examples, average precision (AP) is used to evaluate the model. If the data set contains only positives, then mean reciprocal rank (MRR) is used as average precision cannot be computed without true negatives. The ranking of each validation triple $r(s, o)$ is computed among all possible subject and object substitutions : $r(s', o)$ and $r(s, o')$, for each s', o' in $\mathcal{E}$, as used in previous studies [Bordes et al., 2013b; Nickel et al., 2016b]. Substituted triples that are in the train set are removed for computing the rankings, which is known as *filtered* MRR. The optimization process is stopped when the measure considered decreases compared to the last evaluation (early stopping).

Bern(p) is the Bernoulli distribution, the *one_random_sample*$(\mathcal{E})$ function sample uniformly one entity in the set of all entities $\mathcal{E}$, and the *sample_batch_of_size_b*(Ω, b) function sample b true and false triples uniformly at random from the training set Ω.

For a given embedding size K, let us rewrite Equation (3.7), by denoting the real part of embeddings with primes and the imaginary part with double primes: $e'_i = \mathrm{Re}(e_i)$, $e''_i = \mathrm{Im}(e_i)$, $w'_r = \mathrm{Re}(w_r)$, $w''_r = \mathrm{Im}(w_r)$. The set of parameters is $\Theta = \{e'_i, e''_i, w'_r, w''_r \in \mathbb{R}^K, i \in \mathcal{E}, r \in \mathcal{R}\}$, and the scoring function involves only real vectors:

$$\begin{aligned} \phi(r, s, o; \Theta) = \quad & \langle w'_r, e'_s, e'_o \rangle + \langle w'_r, e''_s, e''_o \rangle \\ & + \langle w''_r, e'_s, e''_o \rangle - \langle w''_r, e''_s, e'_o \rangle \end{aligned} \tag{3.9}$$

where each entity and each relation has two real embeddings.

Gradients are now easy to write:

$$\begin{aligned}
\nabla_{e'_s}\phi(r,s,o;\Theta) &= (w'_r \odot e'_o) + (w''_r \odot e''_o),\\
\nabla_{e''_s}\phi(r,s,o;\Theta) &= (w'_r \odot e''_o) - (w''_r \odot e'_o),\\
\nabla_{e'_o}\phi(r,s,o;\Theta) &= (w'_r \odot e'_s) - (w''_r \odot e''_s),\\
\nabla_{e''_o}\phi(r,s,o;\Theta) &= (w'_r \odot e''_s) + (w''_r \odot e'_s),\\
\nabla_{w'_r}\phi(r,s,o;\Theta) &= (e'_s \odot e'_o) + (e''_s \odot e''_o),\\
\nabla_{w''_r}\phi(r,s,o;\Theta) &= (e'_s \odot e''_o) - (e''_s \odot e'_o),
\end{aligned}$$

where $\odot$ is the element-wise (Hadamard) product.

We optimized the negative log-likelihood of the logistic model described in Equation (3.5) with L^2 regularization on the entity and relation embeddings in Θ:

$$\mathcal{L}(\Omega;\Theta) = \sum_{((r,s,o),y)\in\Omega} \log(1+\exp(-y\phi(r,s,o;\Theta))) + \lambda||\Theta_{\{r,s,o\}}||_2^2 \qquad (3.10)$$

where $\lambda \in \mathbb{R}_+$ is the regularization parameter.

To handle regularization, note that using separate representations for the real and imaginary parts does not change anything as the squared L^2-norm of a complex vector $v = v' + iv''$ is the sum of the squared modulus of each entry:

$$\begin{aligned}
||v||_2^2 &= \sum_j \sqrt{v'^2_j + v''^2_j}^2\\
&= \sum_j v'^2_j + \sum_j v''^2_j\\
&= ||v'||_2^2 + ||v''||_2^2\,,
\end{aligned}$$

which is actually the sum of the L^2-norms of the vectors of the real and imaginary parts.

We can finally write the gradient of $\mathcal{L}$ with respect to a *real* embedding v for one triple (r,s,o) and its truth value y:

$$\nabla_v \mathcal{L}(\{((r,s,o),y)\};\Theta) = -y\sigma(-y\phi(r,s,o;\Theta))\nabla_v\phi(r,s,o;\Theta) + 2\lambda v\,. \qquad (3.11)$$

Algorithm 1 Stochastic gradient descent with AdaGrad for the ComplEx model

Input Training set Ω, validation set Ω_v, learning rate $\alpha \in \mathbb{R}_{++}$, rank $K \in \mathbb{Z}_{++}$, L^2 regularization factor $\lambda \in \mathbb{R}_{+}$, negative ratio $\eta \in \mathbb{Z}_{++}$, batch size $b \in \mathbb{Z}_{++}$, maximum iteration $m \in \mathbb{Z}_{++}$, validate every $s \in \mathbb{Z}_{++}$ iterations, AdaGrad regularizer $\epsilon = 10^{-8}$.

Output Embeddings e', e'', w', w''.

$e'_i \sim \mathcal{N}(\mathbf{0}^k, I^{k\times k})$, $e''_i \sim \mathcal{N}(\mathbf{0}^k, I^{k\times k})$ for each $i \in \mathcal{E}$
$w'_i \sim \mathcal{N}(\mathbf{0}^k, I^{k\times k})$, $w''_i \sim \mathcal{N}(\mathbf{0}^k, I^{k\times k})$ for each $i \in \mathcal{R}$
$g_{e'_i} \leftarrow \mathbf{0}^k$, $g_{e''_i} \leftarrow \mathbf{0}^k$ for each $i \in \mathcal{E}$
$g_{w'_i} \leftarrow \mathbf{0}^k$, $g_{w''_i} \leftarrow \mathbf{0}^k$ for each $i \in \mathcal{R}$
previous_score $\leftarrow 0$
for $i = 1, \ldots, m$ **do**
  for $j = 1, \ldots, |\Omega|/b$ **do**
    $\Omega_b \leftarrow$ *sample_batch_of_size_b*(Ω, b)
    // Negative sampling:
    $\Omega_n \leftarrow \{\emptyset\}$
    for $((r, s, o), y)$ in Ω_b **do**
      for $l = 1, \ldots, \eta$ **do**
        $e \leftarrow$ *one_random_sample*$(\mathcal{E})$
        if Bern$(0.5) > 0.5$ **then**
          $\Omega_n \leftarrow \Omega_n \cup \{((r, e, o), -1)\}$
        else
          $\Omega_n \leftarrow \Omega_n \cup \{((r, s, e), -1)\}$
        end if
      end for
    end for
    $\Omega_b \leftarrow \Omega_b \cup \Omega_n$
    for $((r, s, o), y)$ in Ω_b **do**
      for v in Θ **do**
        // AdaGrad updates:
        $g_v \leftarrow g_v + (\nabla_v \mathcal{L}(\{((r, s, o), y)\}; \Theta))^2$
        // Gradient updates:
        $v \leftarrow v - \frac{\alpha}{g_v + \epsilon} \nabla_v \mathcal{L}(\{((r, s, o), y)\}; \Theta)$
      end for
    end for
  end for
  // Early stopping
  if $i \mod s = 0$ **then**
    current_score $\leftarrow$ *evaluate_AP_or_MRR*$(\Omega_v; \Theta)$
    if *current_score* $\leq$ *previous_score* **then**
      break
    end if
    previous_score $\leftarrow$ *current_score*
  end if
end for
return Θ

3.4 Link with Holographic Embeddings

In this section we investigate the link between the proposed ComplEx model, and a simultaneously and independently proposed model, the holographic embeddings (HolE) [Nickel et al., 2016b]. We show that they have equivalent scoring functions, up to a constant factor, but that ComplEx's formulation of the scoring function has a lower time complexity. A similar proof as independently been proposed by Hayashi and Shimbo [2017].

We will consider discrete Fourier transform (DFT) of purely real vectors only : $\mathcal{F} : \mathbb{R}^K \to \mathbb{C}^K$. For $j \in \{0, \ldots, K-1\}$:

$$\mathcal{F}(x)_j = \sum_{k=0}^{K-1} x_k e^{-2i\pi j \frac{k}{K}} \tag{3.12}$$

where $\mathcal{F}(x)_j \in \mathbb{C}$ is the j^{th} value in the resulting complex vector $\mathcal{F}(x) \in \mathbb{C}^K$. Note that the vector components in Equation (3.12) are indexed from 0 to $K-1$.

The holographic embeddings model (HolE) represents relations and entities with real-valued embeddings $E \in \mathbb{R}^{N_e \times K}$, $W \in \mathbb{R}^{N_r \times K}$, and scores a triple (r, s, o) with the dot product between the embedding of the relation p and the circular correlation $\star : \mathbb{R}^K \times \mathbb{R}^K \to \mathbb{R}^K$ of the embeddings of entities s and o:

$$\phi^h(r, s, o) = w_r^\top (e_s \star e_o)\,. \tag{3.13}$$

The circular correlation can be written with the discrete Fourier transform (DFT),

$$e_s \star e_o = \mathcal{F}^{-1}(\overline{\mathcal{F}(e_s)} \odot \mathcal{F}(e_o)) \tag{3.14}$$

where $\mathcal{F}^{-1} : \mathbb{C}^K \to \mathbb{C}^K$ is the inverse DFT. In this case, the embedding vectors are real $e_s, e_o, w_r \in \mathbb{R}^K$, and so is the result of the inverse DFT, since the circular correlation of real-valued vectors results in a real-valued vector.

We recall the scoring function of the proposed model (ComplEx), that represents relations and entities with complex-valued embeddings $E \in \mathbb{C}^{N_e \times K}$, $W \in \mathbb{C}^{N_r \times K}$, and scores a triple (r, s, o) with the real part of the trilinear product of the corresponding embeddings:

$$\phi^c(r, s, o) = \text{Re}\left(\langle w_r, e_s, \overline{e}_o \rangle\right) \tag{3.15}$$

where $e_s, e_o, w_r \in \mathbb{C}^K$ are complex vectors, and $\overline{e}_o$ is the complex conjugate of the vector e_o.

First, recall Parseval's Theorem:

Theorem 5. Suppose $x, y \in \mathbb{R}^K$ are real vectors. Then $x^\top y = \frac{1}{K}\mathcal{F}(x)^\top \overline{\mathcal{F}(y)}$.

Using Theorem 5 as well as Equations (3.13) and (3.14), we can then rewrite the scoring function of HoLE as:

$$\begin{aligned}
\phi^h(r,s,o) &= w_r^\top (e_s \star e_o) \\
&= w_r^\top (\mathcal{F}^{-1}(\overline{\mathcal{F}(e_s)} \odot \mathcal{F}(e_o))) \\
&= \frac{1}{K}\mathcal{F}(w_r)^\top \overline{\mathcal{F}(\mathcal{F}^{-1}(\overline{\mathcal{F}(e_s)} \odot \mathcal{F}(e_o)))} \\
&= \frac{1}{K}\mathcal{F}(w_r)^\top (\mathcal{F}(e_s) \odot \overline{\mathcal{F}(e_o)}) \\
&= \frac{1}{K}\left\langle \mathcal{F}(w_r), \mathcal{F}(e_s), \overline{\mathcal{F}(e_o)} \right\rangle . \qquad (3.16)
\end{aligned}$$

We now derive a property of the DFT on real vectors x, showing that the resulting complex vector $\mathcal{F}(x)$ has a partially symmetric structure, for $j \in \{1, \ldots, K-1\}$:

$$\begin{aligned}
\mathcal{F}(x)_{(K-j)} &= \sum_{k=0}^{K-1} x_k e^{-2i\pi(K-j)\frac{k}{K}} \\
&= \sum_{k=0}^{K-1} x_k e^{-2i\pi k} e^{2i\pi j \frac{k}{K}}
\end{aligned}$$

and given that k is an integer: $e^{-2i\pi k} = 1$,

$$\begin{aligned}
&= \sum_{k=0}^{K-1} x_k e^{2i\pi j \frac{k}{K}} \\
&= \sum_{k=0}^{K-1} x_k \overline{e^{-2i\pi j \frac{k}{K}}}
\end{aligned}$$

and since $x_k \in \mathbb{R}$,

$$\begin{aligned}
&= \sum_{k=0}^{K-1} \overline{x_k e^{-2i\pi j \frac{k}{K}}} \\
&= \overline{\mathcal{F}(x)_j} . \qquad (3.17)
\end{aligned}$$

Two special cases arise, the first one is $F(x)_0$, which is not concerned by the above symmetry property:

$$\begin{aligned}\mathcal{F}(x)_0 &= \sum_{k=0}^{K-1} x_k e^{-2i\pi 0 \frac{k}{K}} \\ &= \sum_{k=0}^{K-1} x_k \\ &=: s(x) \in \mathbb{R}\,. \end{aligned} \tag{3.18}$$

And the second one is $F(x)_{\frac{K}{2}}$ when K is even:

$$\begin{aligned}\mathcal{F}(x)_{(K-\frac{K}{2})} = \overline{\mathcal{F}(x)_{\frac{K}{2}}} &= \mathcal{F}(x)_{\frac{K}{2}} \\ &= \sum_{k=0}^{K-1} x_k e^{-2i\pi \frac{Kk}{2K}} \\ &= \sum_{k=0}^{K-1} x_k e^{-i\pi k} \\ &= \sum_{k=0}^{\frac{K}{2}-1} x_{2k} - x_{2k+1} \\ &=: t(x) \in \mathbb{R}\,. \end{aligned} \tag{3.19}$$

From Equations (3.17) to (3.19), we write the general form of the Fourier transform $\mathcal{F}(x) \in \mathbb{C}^K$ of a real vector $x \in \mathbb{R}^K$:

$$\mathcal{F}(x) = \begin{cases} [s(x)\ \ x'\ \ t(x)\ \ \overline{x'}_{\leftarrow}], & \text{if } K \text{ is even,} \\ [s(x)\ \ x'\ \ \overline{x'}_{\leftarrow}], & \text{if } K \text{ is odd.} \end{cases} \tag{3.20}$$

where $x', x'_{\leftarrow} \in \mathbb{C}^{\lceil \frac{K}{2} \rceil - 1}$, with $x' = [\mathcal{F}(x)_1, \ldots, \mathcal{F}(x)_{\lceil \frac{K}{2} \rceil - 1}]$, and $x'_{\leftarrow}$ is x' in reversed order: $x'_{\leftarrow} = [\mathcal{F}(x)_{\lceil \frac{K}{2} \rceil - 1}, \ldots, \mathcal{F}(x)_1]$.

We can then derive Equation (3.16) for $w_r, e_s, e_o \in \mathbb{R}^K$, first with K being odd:

$$\begin{aligned}
\phi^h(r,s,o) &= \frac{1}{K}\left\langle \mathcal{F}(w_r), \mathcal{F}(e_s), \overline{\mathcal{F}(e_o)} \right\rangle \\
&= \frac{1}{K}\left\langle [s(w_r)\ \ w'_r\ \ \overline{w}'_{r\leftarrow}], [s(e_s)\ \ e'_s\ \ \overline{e}'_{s\leftarrow}], \overline{[s(e_o)\ \ e'_o\ \ \overline{e}'_{o\leftarrow}]} \right\rangle \\
&= \frac{1}{K}\left\langle [s(w_r)\ \ w'_r\ \ \overline{w}'_r], [s(e_s)\ \ e'_s\ \ \overline{e}'_s], [s(e_o)\ \ \overline{e}'_o\ \ e'_o] \right\rangle \\
&= \frac{1}{K}\left(s(w_r)s(e_s)s(e_o) + \left\langle w'_r, e'_s, \overline{e}'_o \right\rangle + \left\langle \overline{w}'_r, \overline{e}'_s, e'_o \right\rangle \right) \\
&= \frac{1}{K}\left(s(w_r)s(e_s)s(e_o) + \left\langle w'_r, e'_s, \overline{e}'_o \right\rangle + \overline{\left\langle w'_r, e'_s, \overline{e}'_o \right\rangle} \right) \\
&= \frac{1}{K}\left(s(w_r)s(e_s)s(e_o) + 2\,\mathrm{Re}\left(\left\langle w'_r, e'_s, \overline{e}'_o \right\rangle \right) \right) \\
&= \frac{2}{K}\mathrm{Re}\left(\left\langle [\sqrt[3]{\tfrac{1}{2}}s(w_r)\ \ w'_r], [\sqrt[3]{\tfrac{1}{2}}s(e_s)\ \ e'_s], [\sqrt[3]{\tfrac{1}{2}}s(e_o)\ \ \overline{e}'_o] \right\rangle \right) \\
&= \frac{2}{K}\mathrm{Re}\left(\left\langle w''_r, e''_s, \overline{e}''_o \right\rangle \right) \\
&= \frac{2}{K}\phi^c(r,s,o)
\end{aligned} \tag{3.21}$$

where $w''_r, e''_s, e''_o \in \mathbb{C}^{\lceil \frac{K}{2} \rceil}$. The derivation is similar when K is even, with double prime vectors being $x'' = [\sqrt[3]{\frac{1}{2}}s(x)\ \ \sqrt[3]{\frac{1}{2}}t(x)\ \ x'] \in \mathbb{C}^{\frac{K}{2}+1}$.

The two scoring functions are thus directly proportional. Both models have an equal memory complexity, as the complex vectors $w''_r, e''_s, e''_o \in \mathbb{C}^{\lceil \frac{K}{2} \rceil}$ take twice as much memory as real-valued ones of same size—for a given floating-point precision. Though the complex formulation of the scoring function brings time complexity from $\mathcal{O}(K\log(K))$ down to $\mathcal{O}(K)$.

We investigate in the next chapter the discrepancy of results between our proposal and HolE results reported in [Nickel et al., 2016b], and postulate that they are due to the use of two different loss functions. Experiments in Section 4.3.6 correlate with originally reported results for HolE, and confirm this hypothesis.

3.5 Discussion and Future Directions

Though the proposed decomposition is clearly not unique, we will see in the next chapter that it is able to learn meaningful representations of entities and relations. Still, characterizing all possible unitary diagonalizations that preserve the real part is an interesting open question. Especially in an approximation setting with a constrained rank, in order to characterize the decompositions that minimize a given reconstruction error. That might allow the creation of an iterative algorithm similar to eigendecomposition

iterative methods [Saad, 1992] for computing such a decomposition for any given real square matrix.

The proposed decomposition could also find applications in many asymmetric square matrices decompositions applications, such as spectral graph theory for directed graphs [Cvetković et al., 1997], but also factorization of asymmetric measures matrices such as asymmetric distance matrices [Mao and Saul, 2004] and asymmetric similarity matrices [Pirasteh et al., 2015].

From an optimization point of view, the objective function (Equation (3.10)) is clearly non-convex, and we could indeed not be reaching a globally optimal decomposition using stochastic gradient descent. Recent results show that there are no spurious local minima in the completion problem of positive semi-definite matrix [Ge et al., 2016; Bhojanapalli et al., 2016]. Studying the extensibility of these results to our decomposition is another possible line of future work. The first step would be generalizing these results to symmetric real-valued matrix completion, then generalization to normal matrices should be straightforward. The two last steps would be extending to matrices that are expressed as real part of normal matrices, and finally to the joint decomposition of such matrices as a tensor.

Practically, an obvious extension is to merge our approach with known extensions to tensor factorization models in order to further improve predictive performance. For example, the use of pairwise embeddings [Riedel et al., 2013; Welbl et al., 2016] together with complex numbers might lead to improved results in situations that involve non-compositionality. Adding bigram embeddings to the objective could also improve the results as shown on other models [Garcia-Duran et al., 2016].

Chapter Summary

We proposed a new matrix and tensor decomposition with complex-valued latent factors called ComplEx. The decomposition exists for all real square matrices, expressed as the real part of normal matrices. The result extends to sets of real square matrices—tensors—and answers to the requirements of the knowledge graph completion task : handling a large variety of different relations including antisymmetric and asymmetric ones, while being scalable. We described a stochastic gradient descent algorithm to learn from partially-observed knowledge graphs, that either contain both positive and negative triples or only positive ones. Finally we discussed the theoretical links with an independently proposed model, HolE.

Chapter 4

Experiments and Applications

To evaluate our proposal, we used both synthetic experiments to assess our claims, and classical link-prediction benchmarks. First, we justify empirically that using the logistic-loss yields much better generalization with low-ranks than the squared loss on some typical synthetic relations. In another synthetic experiment, we demonstrate the ability of the ComplEx model to jointly learn a symmetric and an antisymmetric relations. Then we evaluate it on classical closed-world datasets: Kinships and UMLS; as well as classical open-world benchmarks: WN18 and FB15K which are respectively subsets of WordNet [Fellbaum, 1998] and Freebase [Bollacker et al., 2008]. We also experimentally explore the discussed theoretical links between HolE and ComplEx. Finally, we propose a different application of our model for enriching distributed representations of words.

We compared ComplEx to state-of-the-art models, namely TransE [Bordes et al., 2013b], DistMult [Yang et al., 2015], RESCAL [Nickel et al., 2011] and also to the canonical polyadic decomposition (CP) [Hitchcock, 1927], to emphasize empirically the importance of learning unique embeddings for entities. For experimental fairness, we reimplemented these models within the same framework as the ComplEx model, using a Theano-based SGD implementation[1] [Bergstra et al., 2010].

For the TransE model, results were obtained with its original max-margin loss, as it turned out to yield better results for this model only. To use this max-margin loss on data sets with observed negatives (Sections 4.1.2 and 4.2), positive triples were replicated when necessary to match the number of negative triples, as described in Garcia-Duran et al. [2016]. We also trained it with L^1 and L^2 norms, results are reported for the best performing one in each experiment. As in the original paper, we did not use regularization over the parameters but instead we enforced entity embeddings to have unit norm $||e_i||_2 = 1$ for all $i \in \mathcal{E}$ [Bordes et al., 2013b].

[1] https://github.com/lmjohns3/downhill

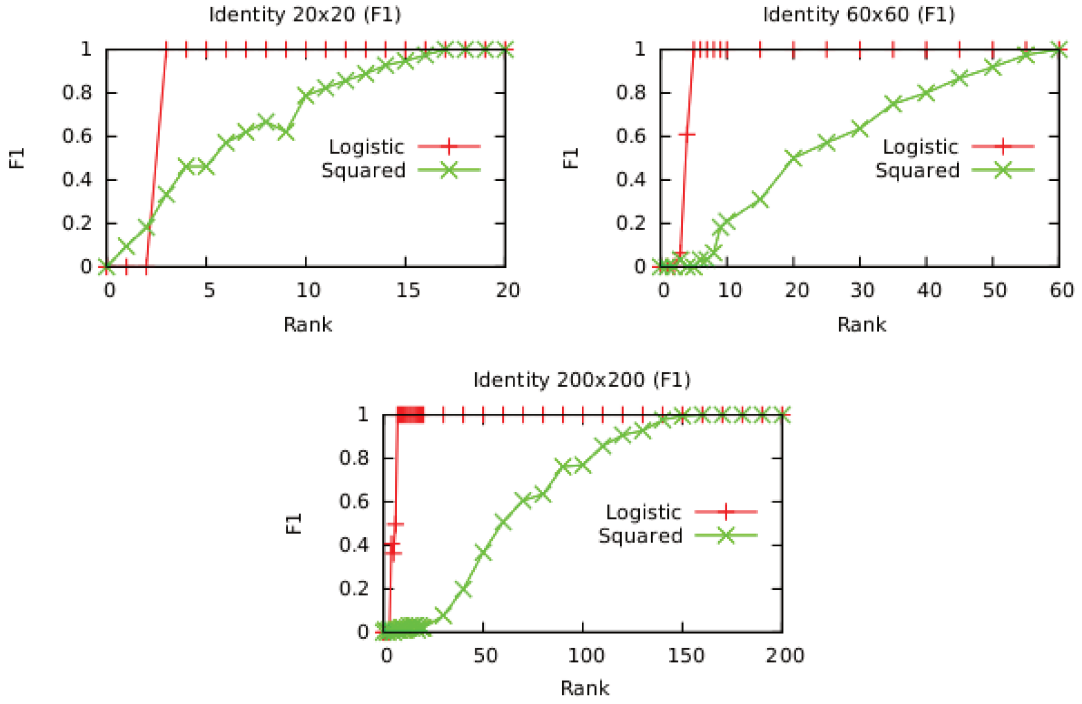


FIGURE 4.1: Identity matrix: F1 measure on the training data corresponding to a fully observed identity matrix, as function of the embedding size, for various matrix sizes. The two curves correspond to the minimization of the squared loss and the logistic loss.

All other models are trained with the negative log-likelihood of the logistic model (Equation (3.10)). In all the following experiments we used a maximum number of iterations $m = 1000$, a batch size $b = \frac{|\Omega|}{100}$, and validated the models for early stopping every $s = 50$ iterations. The regularization parameter λ is validated in $\{0.1, 0.03, 0.01, 0.003, 0.001, 0.0003, 0.00001, 0.0\}$ and the learning rate α is initialized to 0.5.

4.1 Synthetic Validation Experiments

First we compare the logistic and squared loss on synthetic matrix completion representing simple relations, and then we assess our model ability to learn symmetric and antisymmetric patterns.

4.1.1 Comparing Logistic and Squared Losses

We validate the intuition of using the logistic loss over the squared loss when decomposing sign matrices and tensors, that is matrices with -1 and 1 only. We are here interested in evaluating the two losses on matrix completion problems, independently of the actual expressiveness of link-prediction models. We hence propose to compare them on the

most general decomposition model, the CP model, which relates to SVD in the matrix case (single relation). We here minimize the negative log-likelihood of both losses on observed entries.

Finding the decomposition that matches the sign-pattern of a given sign matrix amounts to bringing the 0–1 loss to 0, which is theoretically possible if the rank of decomposition is greater or equal to the sign-rank of the decomposed sign matrix. However to avoid this combinatorial problem, the logistic loss is classically used as a surrogate. Sign-identity $n \times n$ matrices—where 0 are replaced with -1—are known to have a rank of n, but to have a constant sign-rank of 3 [Alon et al., 2016], as discussed in Section 3.1.2.2. As the rank (and sign-rank) are invariant by column permutation, identity-permuted matrices can be used as a permutation relation in knowledge graph that assign each entity to another one, such as `isMarriedTo`. To assess the quality of the logistic loss as a surrogate of the sign-rank, we decompose fully observed identity matrices, and compare reconstruction error between the squared and logistic losses. We report the F1-measure in Figure 4.1. On the smallest matrix (20×20), the logistic loss actually matches the sign-rank as it reaches perfect reconstruction with an embedding size of $K = 3$. On bigger matrices (60×60 and 200×200), the actual rank required to decompose an identity matrix with the logistic loss seems to scale logarithmically with the size of the matrix; whereas it scales linearly with the squared loss. Using the logistic loss allows for decomposing permutation matrices with a rank much closer to the true sign-rank than using the squared loss.

We further conduct our experiments on $n \times n$ matrix completion problems, first on an upper tri-diagonal synthetic relation, which can be seen as a sequential relation (Figure 4.2). And second on block upper-diagonal patterns, which can be seen as transitive groups of entities, such as `olderBrotherOf` (Figure 4.3). In the observed matrices (left), white denotes -1, black 1, and grey unobserved entries. In the reconstructed matrices for the squared (middle) and logistic (right) losses, values are represented in grey-scale. The logistic loss reaches perfect reconstruction with $K = 3$ in the sequential case and with $K = 4$ in the transitive case, whereas the squared loss reconstruction is largely corrupted for these ranks.

These experiments show us that if the logistic loss is minimized, many common relations such as permutation matrices, sequential relations, and transitive relations can be represented with surprisingly small embeddings. In the following experiments, we used the logistic loss with all models.

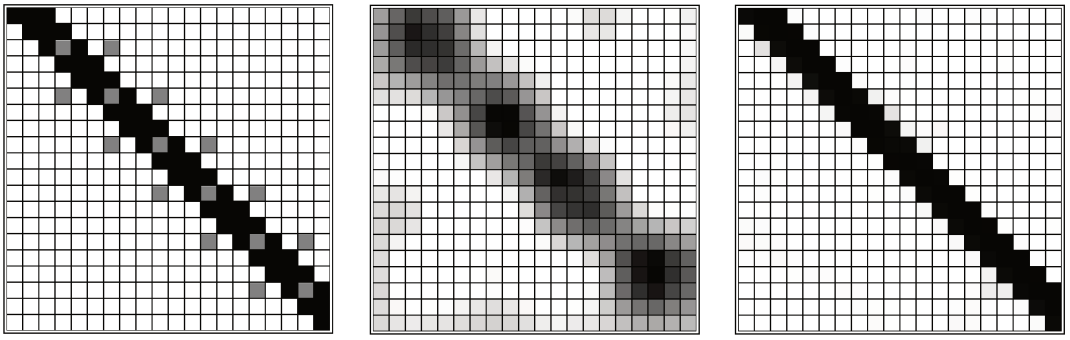


FIGURE 4.2: Sequential relation learning with rank-3 embeddings. One can see that for a fixed embedding size, the predictive accuracy of the logistic-loss model is much higher.

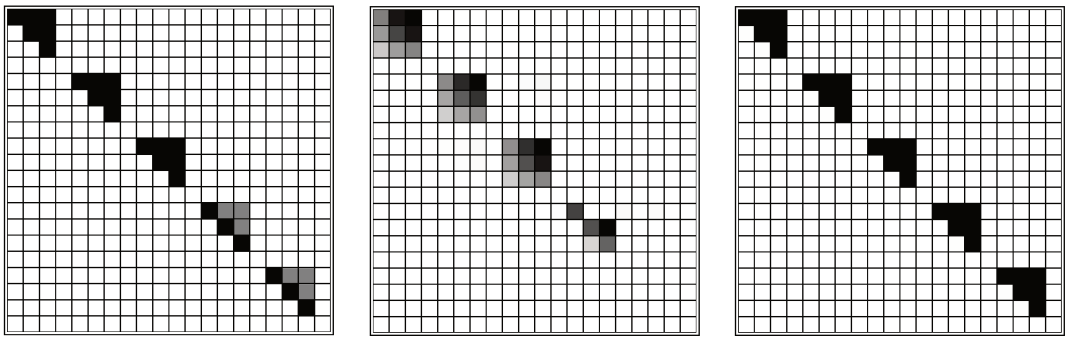


FIGURE 4.3: Transitive relation learning with logistic loss and rank-4 embeddings. Gray cells represent missing values in the training set (left). We see that they are perfectly recovered by predicting relation probabilities (right).

4.1.2 Symmetry and Antisymmetry

To assess our claim that COMPLEX can accurately model jointly symmetry and antisymmetry, we randomly generated a knowledge graph of two relations and 30 entities. One relation is entirely symmetric, while the other is completely antisymmetric. This data set corresponds to a $2 \times 30 \times 30$ tensor. Figure 4.4 shows a part of this randomly generated tensor, with a symmetric slice and an antisymmetric slice, decomposed into training, validation and test sets. To ensure that all test values are predictable, the upper triangular parts of the matrices are always kept in the training set, and the diagonals are unobserved. We conducted a 5-fold cross-validation on the lower-triangular matrices, using the upper-triangular parts plus 3 folds for training, one fold for validation and one fold for testing. Each training set contains 1392 observed triples, whereas validation and test sets contain 174 triples each. Figure 4.5 shows the best cross-validated average precision (area under the precision-recall curve) for different factorization models of ranks ranging up to 50, and error bars show the standard deviation over the 5 runs.

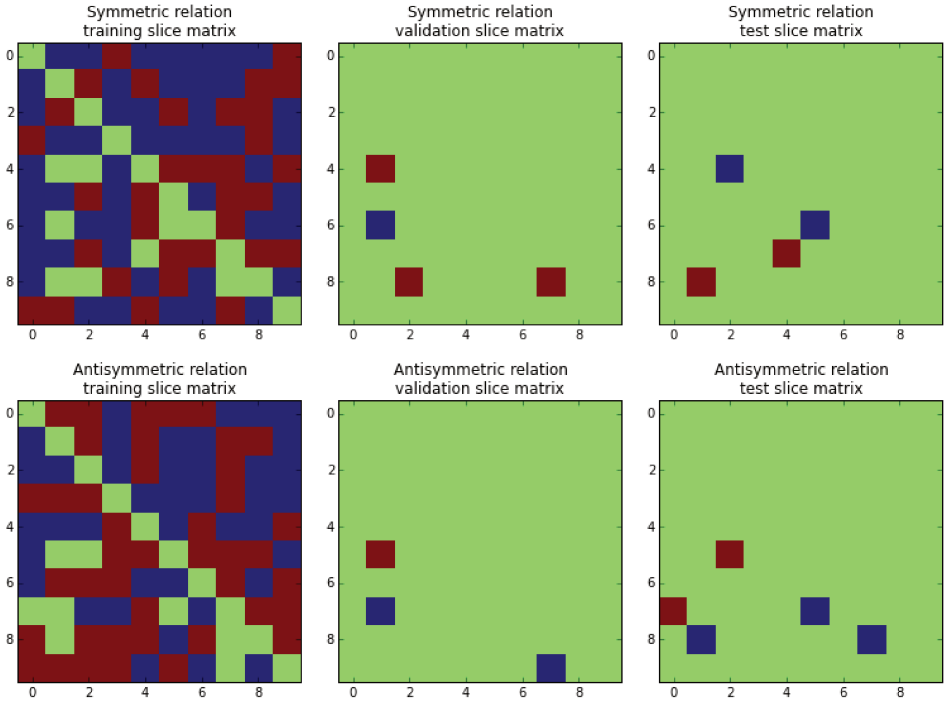


FIGURE 4.4: Parts of the training, validation and test sets of the generated experiment with one symmetric and one antisymmetric relation. Red pixels are positive triples, blue are negatives, and green missing ones. Top: Plots of the symmetric slice (relation) for the 10 first entities. Bottom: Plots of the antisymmetric slice for the 10 first entities.

As expected, DISTMULT [Yang et al., 2015] is not able to model antisymmetry and only predicts the symmetric relations correctly. Although TRANSE [Bordes et al., 2013b] is not a symmetric model, it performs poorly in practice, particularly on the antisymmetric relation. RESCAL [Nickel et al., 2011], with its large number of parameters, quickly overfits as the rank grows. Canonical Polyadic (CP) decomposition [Hitchcock, 1927] fails on both relations as it has to push symmetric and antisymmetric patterns through the entity embeddings. Surprisingly, only COMPLEX succeeds even on such simple data.

4.2 Real Fully-Observed Data Sets: Kinships and UMLS

We then compare all models on two fully observed data sets, that contain both positive and negative triples, also called the *closed-world assumption*. The Kinships data set [Denham, 1973] describes the 26 different kinship relations of the Alyawarra tribe in Australia, among 104 individuals. The unified medical language system (UMLS) data set [McCray, 2003] represents 135 medical concepts and diseases, linked by 49 relations describing their interactions. Metadata for the two data sets is summarized in Table 4.1.

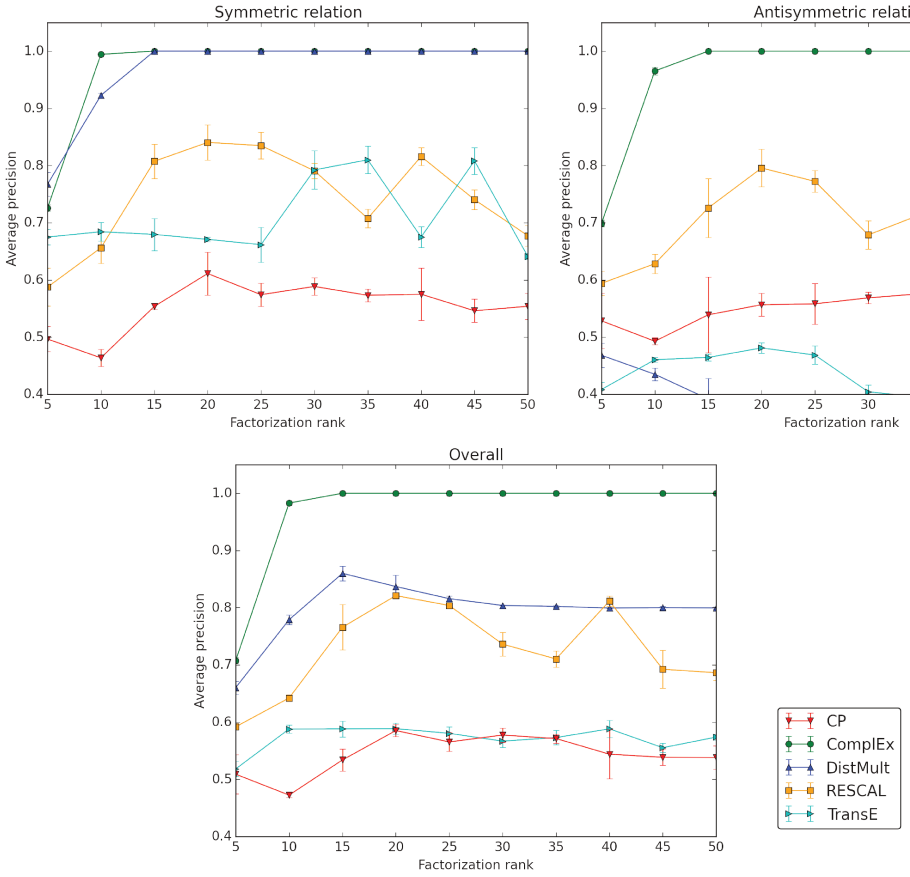


FIGURE 4.5: Average precision (AP) for each factorization rank ranging from 5 to 50 for different state-of-the-art models on the synthetic task. Learning is performed jointly on the symmetric relation and on the antisymmetric relation. Top-left: AP over the symmetric relation only. Top-right: AP over the antisymmetric relation only. Bottom: Overall AP.

Data set	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	Total number of triples
Kinships	104	26	281,216
UMLS	135	49	893,025

TABLE 4.1: Number of entities $|\mathcal{E}|$, relations $|\mathcal{R}|$, and observed triples (all are observed) for the Kinships and UMLS data sets.

We performed a 10-fold cross-validation, keeping 8 for training, one for validation and one for testing. Figure 4.6 shows the best cross-validated average precision for ranks ranging up to 50, and error bars show the standard deviation over the 10 runs.

On both data sets COMPLEX, RESCAL and CP are very close, with a slight advantage for COMPLEX on Kinships, and for RESCAL on UMLS. DISTMULT performs poorly here as many relations are antisymmetric both in UMLS (causal links, anatomical hierarchies) and Kinships (being father, uncle or grand-father).

The fact that CP, RESCAL and COMPLEX work so well on these data sets illustrates the importance of having an expressive enough model, as DISTMULT fails because of

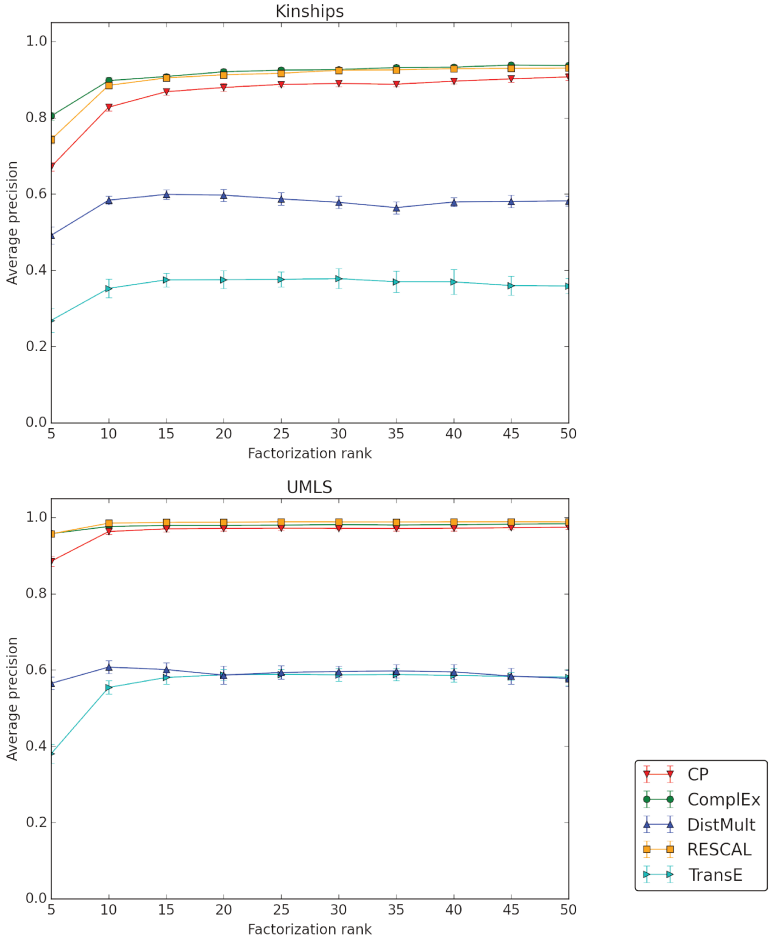


FIGURE 4.6: Average precision (AP) for each factorization rank ranging from 5 to 50 for different state-of-the-art models on the Kinships data set (top) and on the UMLS data set (bottom).

antisymmetry; the power of the multilinear product—that is the tensor factorization approach—as TRANSE can be seen as a sum of bilinear products [Garcia-Duran et al., 2016]; but not yet the importance of having unique entity embeddings, as CP works well. We believe having separate subject and object-entity embeddings works well under the closed world assumption, because of the amount of training data compared to the number of embeddings to learn. Though when only a fractions of the positive training examples are observed (as it is most often the case), we will see in the next experiments that enforcing unique entity embeddings is key to good generalization.

4.3 Real Sparse Data Sets: FB15K and WN18

Finally, we evaluated COMPLEX on the FB15K and WN18 data sets, as they are well established benchmarks for the link-prediction task. FB15K is a subset of Freebase [Bollacker et al., 2008], a curated knowledge graph of general facts, whereas WN18 is a

			Number of triples in sets:		
Data set	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	Training	Validation	Test
WN18	40,943	18	141,442	5,000	5,000
FB15K	14,951	1,345	483,142	50,000	59,071

TABLE 4.2: Number of entities $|\mathcal{E}|$, relations $|\mathcal{R}|$, and observed triples in each split for the FB15K and WN18 data sets.

subset of WordNet [Fellbaum, 1998], a database featuring lexical relations between words. We used the same training, validation and test set splits as in Bordes et al. [2013b]. Table 4.2 summarizes the metadata of the two data sets.

4.3.1 Experimental Setup

As both data sets contain only positive triples, we generated negative samples using the local closed-world assumption, and use the mean reciprocal rank (MRR) for evaluation, where ranking of each test triple $r(s, o)$ is computed among all possible subject and object substitutions—as described in Section 3.3. The MRR and Hits at N are standard evaluation measures for these data sets and come in two flavours: raw and filtered. The filtered metrics are computed *after* removing all the other positive observed triples that appear in either training, validation or test set from the ranking, whereas the raw metrics do not remove these.

Since ranking measures are used, previous studies generally preferred a max-margin ranking loss for the task [Bordes et al., 2013b; Nickel et al., 2016b]. We chose to use the negative log-likelihood of the logistic model. We tried both losses in preliminary work, and training the models with the log-likelihood yielded better results than with the max-margin ranking loss, especially on FB15K—except with TransE.

We report both filtered and raw MRR, and filtered Hits at 1, 3 and 10 in Table 4.3 for the evaluated models. We have shown in Section 3.4 that the scoring function of the HolE model is equivalent to ComplEx —which has also been independently shown by Hayashi and Shimbo [2017]. We record the original results for HolE as reported in Nickel et al. [2016b] and briefly discuss the discrepancy of results obtained with ComplEx.

Reported results are given for the best set of hyper-parameters evaluated on the validation set for each model, after a distributed grid-search on the following values: $K \in \{10, 20, 50, 100, 150, 200\}$, $\lambda \in \{0.1, 0.03, 0.01, 0.003, 0.001, 0.0003, 0.0\}$, $\alpha \in \{1.0, 0.5, 0.2, 0.1, 0.05, 0.02, 0.01\}$, $\eta \in \{1, 2, 5, 10\}$ with λ the L^2 regularization parameter, α the initial learning rate, and η the number of negatives generated per positive training triple. We also tried varying the batch size but this had no impact and we settled with 100 batches per epoch. With the best hyper-parameters, training the ComplEx model on a single

	WN18					FB15K				
	MRR		Hits at			MRR		Hits at		
Model	Filtered	Raw	1	3	10	Filtered	Raw	1	3	10
CP	0.075	0.058	0.049	0.080	0.125	0.326	0.152	0.219	0.376	0.532
TransE	0.454	0.335	0.089	0.823	0.934	0.380	0.221	0.231	0.472	0.641
RESCAL	0.894	0.583	0.867	0.918	0.935	0.461	0.226	0.324	0.536	0.720
DistMult	0.822	0.532	0.728	0.914	0.936	0.654	**0.242**	0.546	0.733	0.824
HolE*	0.938	**0.616**	0.930	**0.945**	**0.949**	0.524	0.232	0.402	0.613	0.739
ComplEx	**0.941**	0.587	**0.936**	**0.945**	0.947	**0.692**	**0.242**	**0.599**	**0.759**	**0.840**

Table 4.3: Filtered and raw mean reciprocal rank (MRR) for the models tested on the FB15K and WN18 data sets. Hits@N metrics are filtered. *Results reported from Nickel et al. [2016b] for HolE model.

GPU (NVIDIA Tesla P40) takes 45 minutes on WN18 ($K = 150, \eta = 1$), and three hours on FB15K ($K = 200, \eta = 10$).

4.3.2 Results

WN18 describes lexical and semantic hierarchies between concepts and contains many antisymmetric relations such as hypernymy, hyponymy, and being part of. Indeed, the DistMult and TransE models are outperformed here by ComplEx and HolE, which are on a par with respective filtered MRR scores of 0.941 and 0.938, which is expected as both models are equivalent.

Table 4.4 shows the filtered MRR for the reimplemented models and each relation of WN18, confirming the advantage of ComplEx on antisymmetric relations while losing nothing on the others. 2D projections of the relation embeddings (Figures 4.10 & 4.11) visually corroborate the results.

On FB15K, the gap is much more pronounced and the ComplEx model largely outperforms HolE, with a filtered MRR of 0.692 and 59.9% of Hits at 1, compared to 0.524 and 40.2% for HolE. This difference of scores between the two models, though their scoring functions are equivalent, is due to the use of the aforementioned max-margin loss in the original HolE publication [Nickel et al., 2016b] that performs worse than the log-likelihood on this dataset, and to the generation of more than one negative sample per positive in these experiments. We will further explore this interpretation in Section 4.3.6. The fact that DistMult yields fairly high scores (0.654 filtered MRR) is also due to the task itself and the evaluation measures used. As the dataset only involves true facts, the test set never includes the opposite facts $r(o, s)$ of each test fact $r(s, o)$ for *antisymmetric* relations—as the opposite fact is always false. Thus highly scoring the opposite fact barely impacts the rankings for antisymmetric relations. This is not the case in the fully observed experiments (Section 4.2), as the opposite fact is known to be false—for

Relation name	ComplEx	RESCAL	DistMult	TransE	CP
hypernym	**0.953**	0.935	0.791	0.446	0.109
hyponym	**0.946**	0.932	0.710	0.361	0.009
member_meronym	**0.921**	0.851	0.704	0.418	0.019
member_holonym	**0.946**	0.861	0.740	0.465	0.134
instance_hypernym	**0.965**	0.833	0.943	0.961	0.233
instance_hyponym	**0.945**	0.849	0.940	0.745	0.040
has_part	**0.933**	0.879	0.753	0.426	0.035
part_of	**0.940**	0.888	0.867	0.455	0.094
member_of_domain_topic	**0.924**	0.865	0.914	0.861	0.007
synset_domain_topic_of	**0.930**	0.855	0.919	0.917	0.153
member_of_domain_usage	**0.917**	0.629	**0.917**	0.875	0.001
synset_domain_usage_of	**1.000**	0.541	**1.000**	**1.000**	0.134
member_of_domain_region	**0.865**	0.632	0.635	**0.865**	0.001
synset_domain_region_of	0.919	0.655	0.888	**0.986**	0.149
derivationally_related_form	**0.946**	0.928	0.940	0.384	0.100
similar_to	**1.000**	0.001	**1.000**	0.244	0.000
verb_group	**0.936**	0.857	0.897	0.323	0.035
also_see	0.603	0.302	**0.607**	0.279	0.020

Table 4.4: Filtered Mean Reciprocal Rank (MRR) for the models tested on each relation of the WordNet data set (WN18).

antisymmetric relations—and largely impacts the average precision of the DistMult model (Figure 4.6).

RESCAL, that represents each relation with a $K \times K$ matrix, performs well on WN18 as there are few relations and hence not so many parameters. On FB15K though, it probably overfits due to the large number of relations and thus the large number of parameters to learn, and performs worse than a less expressive model like DistMult. On both data sets, TransE and CP are largely left behind. This illustrates again the power of the multilinear product in the first case, and the importance of learning unique entity embeddings in the second. CP performs especially poorly on WN18 due to the small number of relations, which magnifies this subject/object difference.

Figure 4.7 shows that the filtered MRR of the ComplEx model quickly converges on both data sets, showing that the low-rank hypothesis is reasonable in practice. The little gain of performances for ranks comprised between 50 and 200 also shows that ComplEx does not perform better because it has twice as many parameters for the same rank—the real and imaginary parts—compared to other linear space complexity models, but indeed thanks to its better expressiveness.

Best ranks were generally 150 or 200, in both cases scores were always very close for all models, suggesting there was no need to grid-search on higher ranks. The number of negative samples per positive sample also had a large influence on the filtered MRR on FB15K (up to +0.08 improvement from 1 to 10 negatives), but not much on WN18.

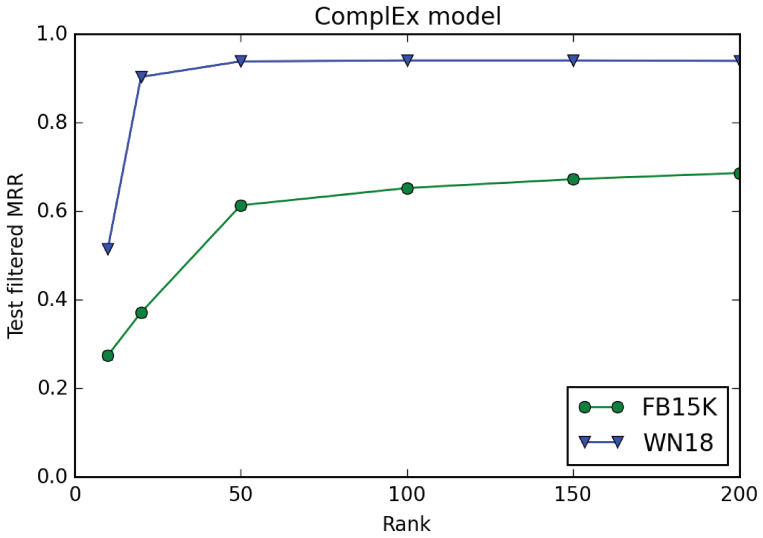


FIGURE 4.7: Best filtered MRR for COMPLEX on the FB15K and WN18 data sets for different ranks. Increasing the rank gives little performance gain for ranks of $50 - 200$.

On both data sets regularization was important (up to $+0.05$ on filtered MRR between $\lambda = 0$ and optimal one). We found the initial learning rate to be very important on FB15K, while not so much on WN18. We think this may also explain the large gap of improvement COMPLEX provides on this data set compared to previously published results—as DISTMULT results are also better than those previously reported [Yang et al., 2015]—along with the use of the log-likelihood objective. It seems that in general AdaGrad is relatively insensitive to the initial learning rate, perhaps causing some overconfidence in its ability to tune the step size online and consequently leading to less efforts when selecting the initial step size.

4.3.3 Training time

As defended in Section 3.1, having a linear time and space complexity becomes critical when the dataset grows. To illustrate this, we report in Figure 4.8 the evolution of the filtered MRR on the validation set as a function of time, for the best set of validated hyper-parameters for each model. The convergence criterion used is the decrease of the validation filtered MRR (computed every 50 iterations), with a maximum number of iterations of 1000 (see Algorithm 1). All models have a linear complexity except for RESCAL that has a quadratic one in the rank of the decomposition, as it learns one matrix embedding for each relation $r \in \mathcal{R}$. Timings are measured on a single NVIDIA Tesla P40 GPU.

On WN18, all models reach convergence in a reasonable time, between 15 minutes and 1 hour and 20 minutes. The difference between RESCAL and the other models is not sharp

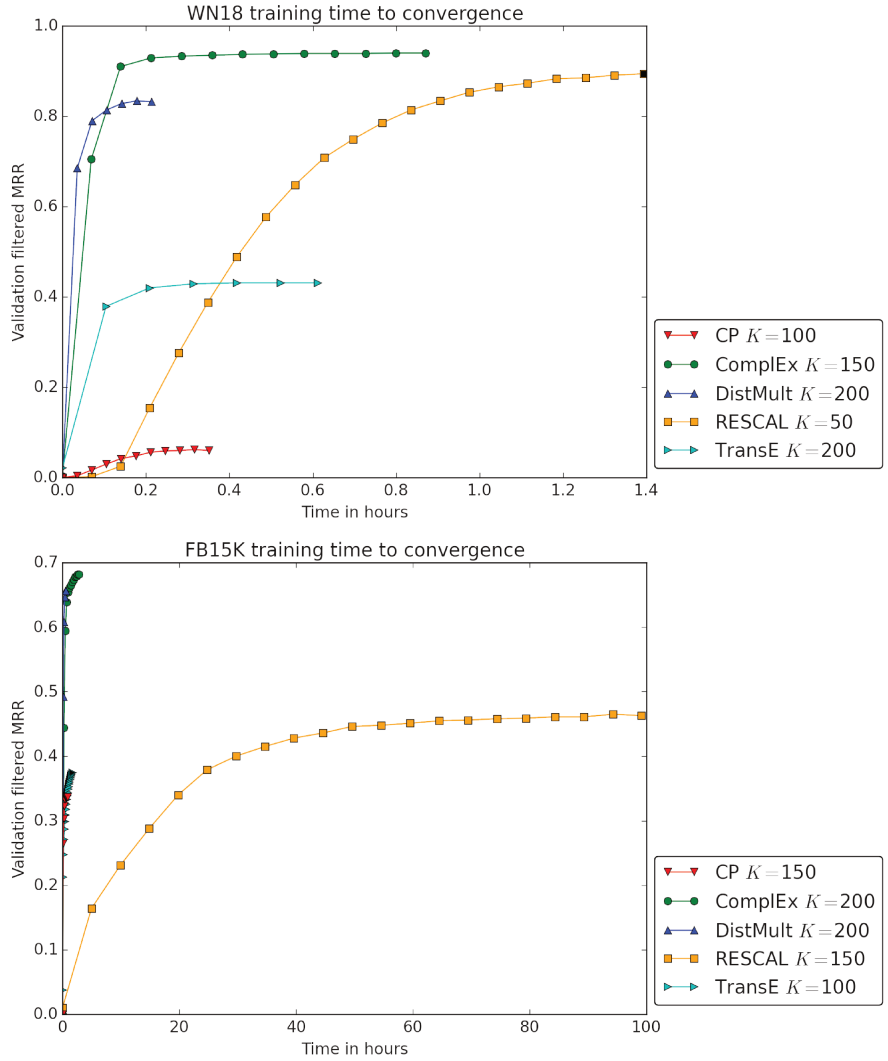


FIGURE 4.8: Evolution of the filtered MRR on the validation set as a function of time, on WN18 (top) and FB15K (bottom) for each model for its best set of hyper-parameters. The best rank K is reported in legend. Final black marker indicates that the maximum number of iterations (1000) has been reached (RESCAL on WN18, TRANSE on FB15K).

there, first because its optimal embedding size ($K = 50$) is lower compared to the other models. Secondly, there are only $|\mathcal{R}| = 18$ relations in WN18, hence the memory footprint of RESCAL is pretty similar to the other models—because it represents only relations with matrices and not entities. On FB15K, the difference is much more pronounced, as RESCAL optimal rank is similar to the other models; and with $|\mathcal{R}| = 1345$ relations, RESCAL has a much higher memory footprint, which implies more processor cache misses due to the uniformly-random nature of the SGD sampling.

RESCAL took more than four days to train, whereas other models took between 40 minutes and 3 hours. While a few days might seem manageable, this could not be the case on larger data sets, as FB15K is but a small subset of Freebase that contains

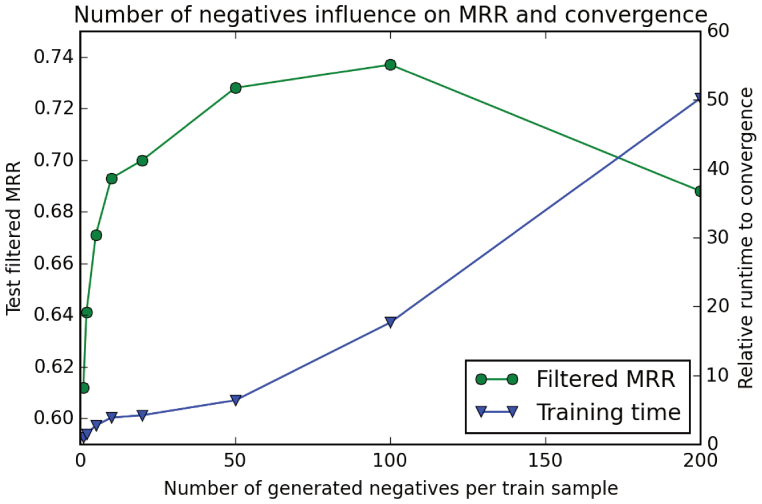


FIGURE 4.9: Influence of the number of negative triples generated per positive training example on the filtered test MRR and on training time to convergence on FB15K for the COMPLEX model with $K = 200$, $\lambda = 0.01$ and $\alpha = 0.5$. Times are given relative to the training time with one negative triple generated per positive training sample (= 1 on time scale).

$|\mathcal{R}| = 35000$ relations [Bollacker et al., 2008]. This experimentally supports our claim that linear complexity is required for scalability.

4.3.4 Influence of Negative Samples

We further investigated the influence of the number of negatives generated per positive training sample. In the previous experiment, due to computational limitations, the number of negatives per training sample, η, was validated over the set $\{1, 2, 5, 10\}$. On WN18 it proved to be of no help to have more than one generated negative per positive. Here we explore in which proportions increasing the number of generated negatives leads to better results on FB15K. To do so, we fixed the best validated λ, K, α obtained from the previous experiment. We then let η vary in $\{1, 2, 5, 10, 20, 50, 100, 200\}$.

Figure 4.9 shows the influence of the number of generated negatives per positive training triple on the performance of COMPLEX on FB15K. Generating more negatives clearly improves the results up to 100 negative triples, with a filtered MRR of 0.737 and 64.8% of Hits@1, before decreasing again with 200 negatives, probably due to the too large class imbalance. The model also converges with fewer epochs, which compensates partially for the additional training time per epoch, up to 50 negatives. It then grows linearly as the number of negatives increases.

This calls to develop a more intelligent negative sampling procedure, to generate more informative negatives with respect to the positive sample from which they have been

sampled. This would reduce the number of negatives required to reach good performance, thus accelerating training time. When the knowledge graph comes with a schema that defines entity types (person, place or song for example) this information can be used to sample negatives by corrupting positive triples with entities of the same type, as shown by [Sedghi and Sabharwal, 2016].

4.3.5 WN18 Embeddings Visualization

We used principal component analysis (PCA) to visualize embeddings of the relations of the WordNet data set (WN18). We plotted the four first components of the best DistMult and ComplEx model's embeddings in Figures 4.10 & 4.11. For the ComplEx model, we simply concatenated the real and imaginary parts of each embedding.

Most of WN18 relations describe hierarchies, and are thus antisymmetric. Each of these hierarchic relations has its inverse relation in the data set. For example: `hypernym` / `hyponym`, `part_of` / `has_part`, `synset_domain_topic_of` / `member_of_domain_topic`. Since DistMult is unable to model antisymmetry, it will correctly represent the nature of each pair of opposite relations, but not the direction of the relations. Loosely speaking, in the `hypernym` / `hyponym` pair the nature is sharing semantics, and the direction is that one entity generalizes the semantics of the other. This makes DistMult representing the opposite relations with very close embeddings. It is especially striking for the third and fourth principal component (Figure 4.11). Conversely, ComplEx manages to oppose spatially the opposite relations.

4.3.6 Comparing ComplEx and HolE

Following the equivalence discussion with the scoring function of the HolE model in Section 3.4, we now experimentally compare the differences between the two models.

In Table 4.5, results for the ComplEx and HolE models agreed on the WN18 data set, but diverged on FB15K. Since both models are equivalent, we assumed that this is due to the different loss functions that were used. To assess this hypothesis, we reimplemented both losses over the ComplEx model scoring function within the same framework, and compared them on the WN18 and FB15K data sets.

In the original HolE publication [Nickel et al., 2016b], a pairwise max-margin loss is optimized over each positive and its corrupted negative (r, s', o'):

$$\mathcal{L}(\Omega;\Theta) \quad = \quad \sum_{((r,s,o),y)\in\Omega} \max(0, \gamma + \sigma(\phi(r,s',o';\Theta)) - \sigma(\phi(r,s,o;\Theta))) \qquad (4.1)$$

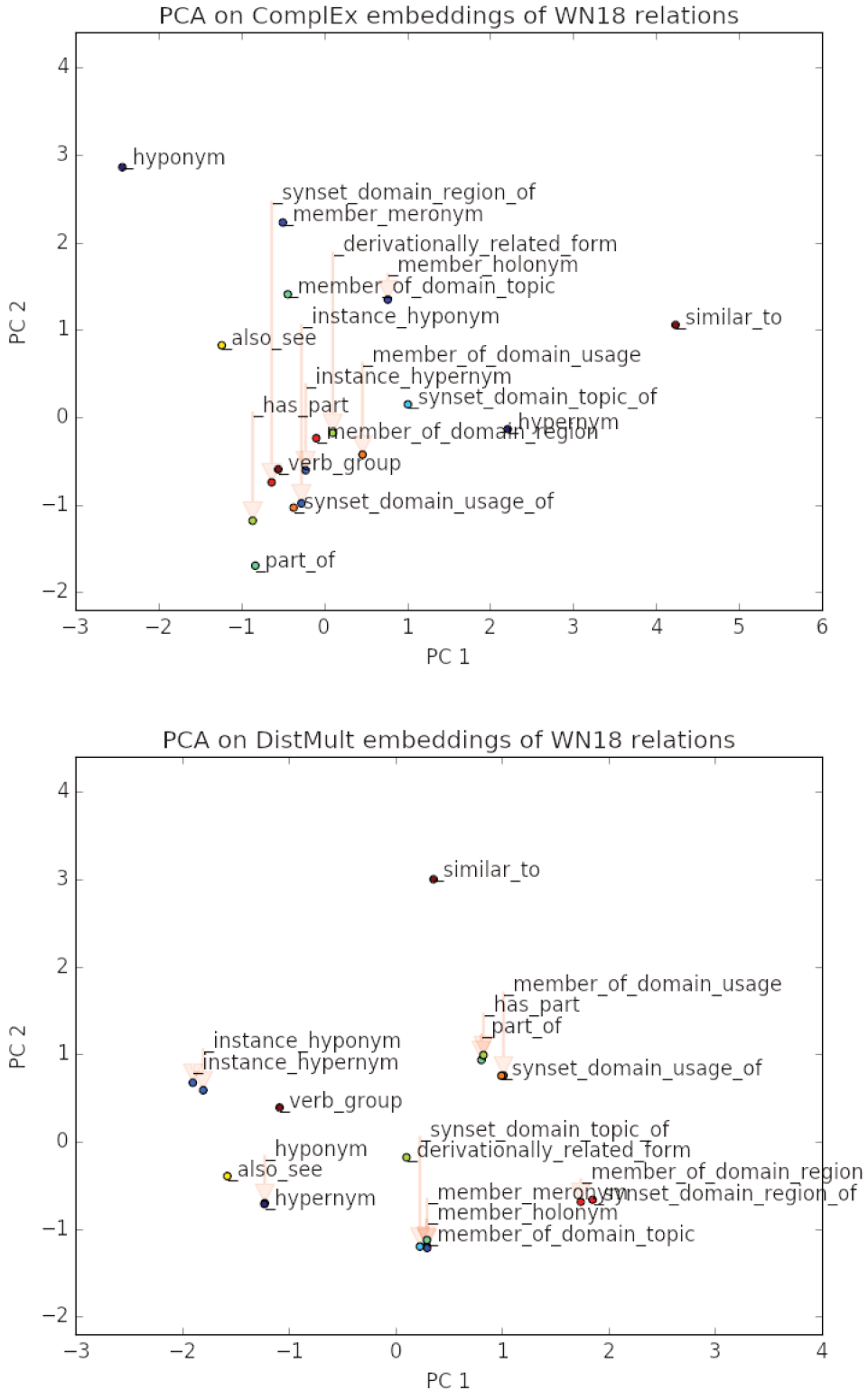


FIGURE 4.10: Plots of the first and second components of the WN18 relations embeddings using principal component analysis. Red arrows link the labels to their point. Top: COMPLEX embeddings. Bottom: DISTMULT embeddings. Opposite relations are clustered together by DISTMULT while correctly separated by COMPLEX.

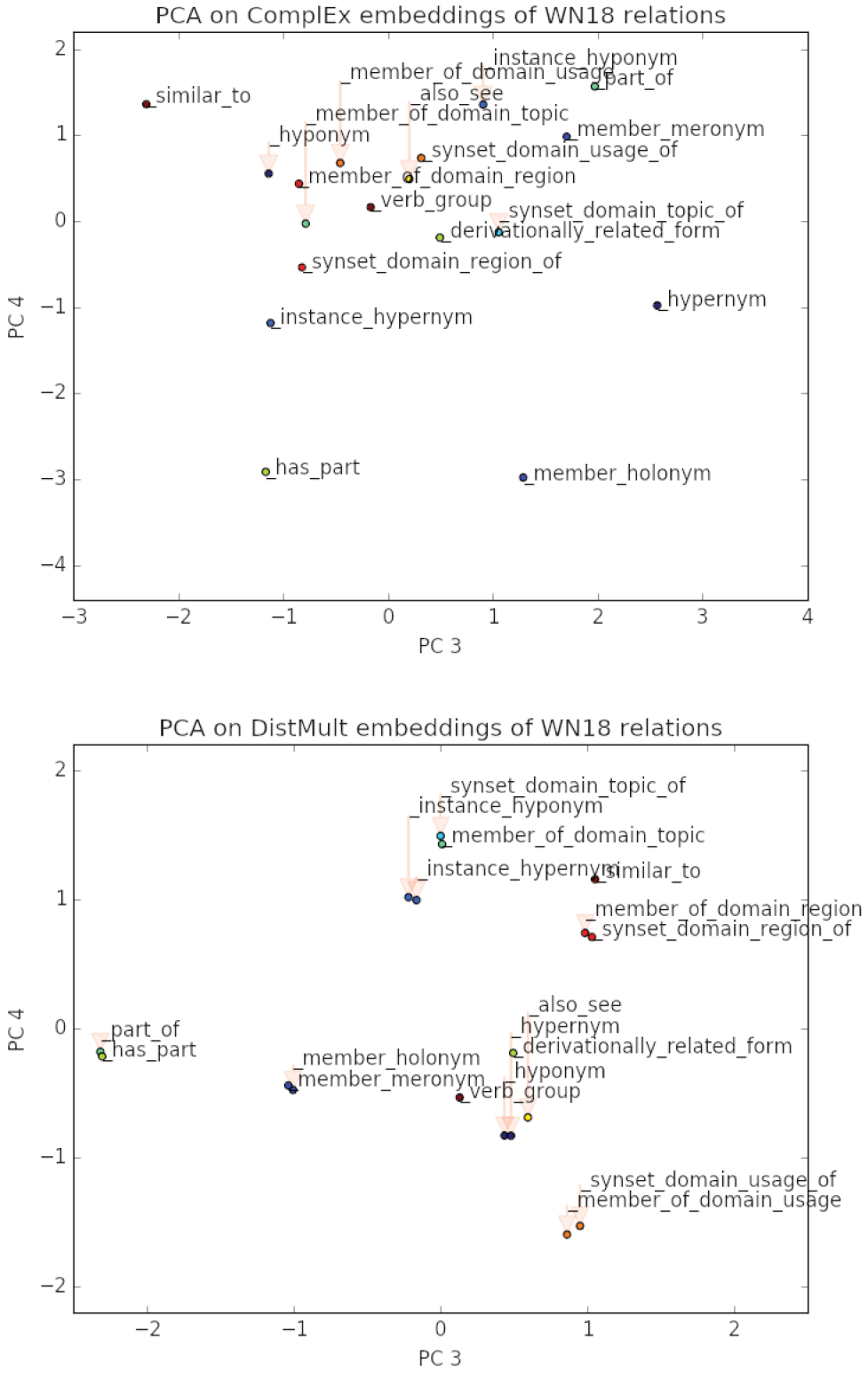


FIGURE 4.11: Plots of the third and fourth components of the WN18 relations embeddings using principal component analysis. Red arrows link the labels to their point. Top: COMPLEX embeddings. Bottom: DISTMULT embeddings. Opposite relations are clustered together by DISTMULT while correctly separated by COMPLEX.

	WN18					FB15K				
	MRR		Hits at			MRR		Hits at		
Loss	Filtered	Raw	1	3	10	Filtered	Raw	1	3	10
Max-margin	0.938	**0.605**	0.932	0.942	**0.949**	0.541	**0.298**	0.411	0.627	0.757
Neg-LL	**0.941**	0.587	**0.936**	**0.945**	0.947	**0.639**	0.250	**0.523**	**0.725**	**0.825**

TABLE 4.5: Filtered and raw mean reciprocal rank (MRR), Hits@N metrics are filtered, for the COMPLEX model with the pairwise max-margin loss and the negative log-likelihood on WN18 and FB15K data sets.

where γ is the margin hyperparameter. The entity embeddings are also constrained to unit norm : $||e_i||_2 = 1$, for all $i \in \mathcal{E}$. Whereas we optimized the log-likelihood loss as explained in the previous chapter.

The results are reported for the best validated models after a distributed grid-search on the following values: $K \in \{10, 20, 50, 100, 150, 200\}$, $\lambda \in \{0.1, 0.03, 0.01, 0.003, 0.001, 0.0003, 0.0\}$ for the log-likelihood loss, and $\gamma \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$ for the max-margin loss. The raw and filtered mean reciprocal ranks (MRR), as well as the filtered hits at 1, 3 and 10 are reported in Table 4.5.

The max-margin loss results are consistent with the HOLE ones originally reported in Nickel et al. [2016b], confirming the equivalence of the scoring functions, and our hypothesis that the loss was responsible for the difference in previously reported results. The log-likelihood results are also coherent, as one must note that the higher scores reported on FB15K in Table 4.5 are due to the use of more than one generated negative sample for each positive training triple. Here, we generated a single negative sample for each positive one in order to keep the comparison fair between the two losses.

The choice of the loss is of little consequence on the WN18 dataset, whereas the log-likelihood loss performs much better on FB15K. While much research attention has been given to scoring functions in link prediction, little has been said about the losses, and the max-margin loss has been used in most of the existing work [Bordes et al., 2013b; Yang et al., 2015; Riedel et al., 2013]. Properties of both losses should be studied to understand this discrepancy of results on some datasets, as well as a more extensive empirical comparison of both losses to assess whether or not the log-likelihood should be systematically chosen over the max-margin loss.

4.4 Learning Complex Word Embeddings

In the next experiments, we explore the applicability of the multi-relational factorization framework for enriching distributed representation of words.

With the release of the `word2vec` [Mikolov et al., 2013] and gloVe [Pennington et al., 2014] trained word embeddings, machine learning researchers started to widely reuse pre-trained models on different tasks. This breakthrough let us glimpse the possible future existence of a fully modular library of pre-trained representations. However, this potential has not yet been fully grasped and exploited. In many use cases of word embeddings, those are just used for initialization, and then fine-tuned for the task. Allow us here a loose comparison with software development. The current use of word embeddings would be as if, after the release of the first ever written library package, every one went forking its source code for his own purpose, and *no one* was building—and releasing—another package built on top of it. In this work, we argue for embracing the full modularity potential that is now offered to our community through the incremental building of trained representations.

Among various different applications, pre-trained word embeddings have been used for recognizing textual entailment (RTE) [Marelli et al., 2014; Bowman et al., 2015a]. On one hand, integrating external resources such as WordNet in combination with distributional representations of words proved to be very useful for this task [Marelli et al., 2014]. This is intuitively understandable, as distributional representations are trained on a *symmetrical* information, co-occurrence, yet entailment is an *antisymmetric* property, and resources as WordNet [Fellbaum, 1998] contains antisymmetric information between words such as hypernymy or meronymy. Levy et al. [2015]; Bowman et al. [2015b] also discussed the limits of distributional representations for entailment prediction. On the other hand, using only fixed word embeddings during the optimization process largely reduces the number of parameters and allows for using larger and better performing models [Rocktäschel et al., 2016; Liu et al., 2016]. Here we reconcile both aspects by first giving the word representations these asymmetric properties by enriching them with external knowledge, in the form of knowledge graphs. However, conversely to previous works that either learn embeddings jointly on a corpus and on external resources [Xu et al., 2014; Liu et al., 2015], or refine pre-trained embeddings with external resources [Faruqui et al., 2015]; we propose to *extend* the pre-trained vectorial representations of words, to encode this new knowledge in a modular fashion.

In the WordNet knowledge graph, words are the entities. The ComplEx model ability to model antisymmetric relations between pairs of entities comes from the complex conjugation of the object-entity embedding, that is the change of sign of its imaginary part. As we are interested here into both (i) encoding antisymmetric information about words, and (ii) keeping the approach modular, we train the ComplEx model on WordNet while keeping the real part of the word embeddings constant and initialized from pre-trained embeddings, and only learn their imaginary part to fit WordNet antisymmetric relations between words.

Embeddings	Embeddings size	Layers size	Train	Dev	Test
Word2vec	$K = K^r = 300$	$L = 2K^r = 600$	0.7903	0.7706	0.7792
Word2vec	$K = K^r = 300$	$L = 2K^c = 1200$	0.8332	0.7835	0.7771
Word2vec+ComplEx	$K = K^c = 600$	$L = 2K^r = 600$	0.8054	**0.7840**	**0.7875**
Word2vec+ComplEx	$K = K^c = 600$	$L = 2K^c = 1200$	0.8245	0.7805	0.7850

Table 4.6: Accuracies on the SNLI corpus with the word2vec embeddings, and the embeddings enhanced with the ComplEx model on WordNet, for different sizes of the intermediate layers.

To sum up:

- We propose to extend vectorial representations of words with knowledge graphs, instead of refining these vectors.
- To encode antisymmetric information about words into vectors, we leverage on the asymmetry of the Hermitian dot product.
- Only the imaginary part is learned, keeping the approach modular and incremental.

4.4.1 Imaginary Part Only Learning

To train our word embeddings with the ComplEx model, we reused the WN18 subset of WordNet [Bordes et al., 2013b], that mainly contains antisymmetric relations. It is initially composed of $|\mathcal{E}| = 40{,}943$ words. For each entity $i \in \mathcal{E}$, we initialized the real parts of their embeddings $\mathrm{Re}(e_i) \in \mathbb{R}^K$ with pre-trained `word2vec` vectors[2] of dimension $K = 300$. To do so, we dropped POS tag information as well as the different meanings of each words (that were represented as different entities) in WN18, and merged them together as a single entity. This resulted into an intersection of $|\mathcal{E}| = 16{,}561$ words with the `word2vec` embeddings, and $|\Omega| = 63{,}251$ observed positive triples.

The training is performed as described in the previous WN18 experiment, except for the size of the embeddings that is not validated as it is fixed to $K = 300$, the dimension of the pre-trained embeddings. This time, only the imaginary part of the entity embeddings is learned, while the real part is kept constant to the pre-trained initialization value. We next assess our extended word embeddings on a classical entailment classification data set, SNLI [Bowman et al., 2015a].

4.4.2 Results on Entailment: SNLI

The SNLI dataset contains 570,000 human-written English sentences pairs, labeled with three classes: entailment, contradiction, and neutral. To compare our embeddings extended with an imaginary part against the `word2vec` ones, we reused an existing neural network architecture available online[3], which is a simple but yet strong baseline. The model is very similar to the one originally proposed by Bowman et al. [2015a], except it uses ReLU layers instead of hyperbolic tangent ones. For each pair of sentences, the corresponding word embeddings of size K are passed through a first K-ReLU translation layer. For each sentence, the translated word embeddings are summed together, both sentence sums are then concatenated into a layer of size $L = 2K$ and fed through three L-ReLU layers, before a final 3-way softmax. Formally, each word in the vocabulary $w \in \mathcal{V}$ has an input word embedding $e_w \in \mathbb{R}^K$, that is not updated during training. Let $f_n^i(x) = \max(0, W^i x)$ be a ReLU layer, where $x \in \mathbb{R}^K$ and $W^i \in \mathbb{R}^{n \times K}$. For each pair of sentences (s_1, s_2) with its label y, the model holds in a single line:

$$\hat{y} = \text{softmax}\Big(f_L^4\Big(f_L^3\Big(f_L^2\Big(\Big[\sum_{w \in s_1} f_K^1(e_w), \sum_{w \in s_2} f_K^1(e_w)\Big]\Big)\Big)\Big)\Big)\,. \tag{4.2}$$

To use our complex embeddings in this real-valued network, we concatenated the learnt imaginary parts to their original `word2vec` real parts, resulting in word embedding vectors of size $K^c = 600$ for each word. `word2vec` words that were not in the WN18 subset were assigned a zero vector for their imaginary part. Comparatively, the original `word2vec` vectors are of size $K^r = 300$, and correspond to the real part—which is the first half—of our complex vectors. As the last layers are of size $L = 2K$, the resulting network has more parameters with the complex embeddings as $K^c = 2K^r$. To compare the two sets of embeddings fairly, we trained the model twice with $L = 2K^c$ and $L = K^r$ for each set of embeddings. The hyper-parameters are left as provided: optimization is conducted with RMSProp [Tieleman and Hinton, 2012], L_2-regularization strength of 4×10^{-6}, dropout of 0.2 and early-stopping. Out-of-vocabulary embeddings are zeroed. The resulting accuracies are reported in Table 4.6. The proposed complex word embeddings brings an improvement of almost one point of accuracy on the test set.

These results are promising as WN18 is but a small subset of WordNet, and only $16,561$ word embeddings were extended with an imaginary part. We expect to yield a better improvement when using a larger dump of WordNet. In the future, it would be interesting to feed these complex embeddings into a complex-valued neural network, such as used by Danihelka et al. [2016]. Though in principle, using a twice-larger real-valued network

[2] `https://code.google.com/archive/p/word2vec/`
[3] `https://github.com/Smerity/keras_snli`

over the concatenation of the real and imaginary parts of the input vectors (as done here) is more general. Indeed, the complex-valued vector-matrix product is but a specific linear combination of their real and imaginary parts, that hence can be learnt by the a twice-larger real-valued fully-connected layer over the concatenation of the real and imaginary part of the input vectors.

This practice of extending vectorial representations could be generalized without all the complex algebra apparatus. When training an embedding model, one could, instead of fine tuning word embeddings, fix them and add to them new free dimensions to optimize, and then publish these newly trained dimensions. An online platform that aggregates all the uploaded pre-trained embeddings on various tasks, and allow for an easy download of a selected concatenation of them could sensibly accelerate the progress of artificial intelligence, as open-source did for the progress of computer science.

Chapter Summary

We first showed the importance of using a binary loss for decomposing matrices of binary relations. Then we confirmed the ability of the ComplEx model to efficiently learn symmetric and antisymmetric relations. Experiments on real data sets confirm its theoretical versatility, as it substantially improves over the state-of-the-art. It shows that real world relations can be efficiently approximated as the real part of low-rank normal matrices. We underlined the importance of some hyper-parameters, especially the number of negatives generated, as well as the choice of the loss. Finally, we proposed a novel way of enriching distributional word embeddings with knowledge graphs, by extending vectorial representations, which proved to enhance entailment recognition.

Chapter 5

Inductive Abilities of Latent Factor Models

Artificial intelligence is becoming more driven by its empirical successes than by the quest for a principled formalisation of reasoning, making it more of an empirical science than a theoretical one. Experimental design is a key skill of empirical scientists, and a well-designed experiment should expose model limitations to enable improving on them. Indeed, seeking falsification is up to now the best definition of science [Popper, 1934]. In machine learning, it is extremely simple to come up with an experiment that will fail. However it is less easy to think of one that brings an informative failure—when one thinks of a failing experiment at all. The bAbI data set [Weston et al., 2015], proposing a set of 20 prerequisite tasks for reasoning over natural language, is an example of an informative experiment, by the specific reasoning type that each task targets. Inspired by the idea of this work, we designed simple tasks for relational learning that assess basic properties of relations, as well as simple reasonings such as kinship relations.

In many machine learning fields, research is drifting away from first-order logic methods. Most of the time, this drift is justified by better predictive performances and scalability of the new methods. It is especially true with link prediction, where latent factor models became more popular than logic-based models [Nickel et al., 2011; Bordes et al., 2013b; Trouillon et al., 2016b]. Logic-based link prediction consists in using both observed facts and logical rules to infer the truth of unobserved facts. For example, given the entities `Alice`, `Eve` and `Bob` and the relations `mother` and `grandmother`, if `mother(Alice,Eve)` and `mother(Eve,Bob)` are true facts, then `grandmother(Alice,Bob)` is also true. Inferring this last fact from the first two however, requires knowing that the mother of one's mother is one's grandmother, which can be expressed by the first-order formula: $\forall \mathtt{x} \forall \mathtt{y} \forall \mathtt{z}\, \mathtt{mother}(\mathtt{x}, \mathtt{y}) \wedge \mathtt{mother}(\mathtt{y}, \mathtt{z}) \Rightarrow \mathtt{grandmother}(\mathtt{x}, \mathtt{z})$. Logical deduction can

be conducted deterministically, or probabilistically to cope with uncertainty of the data [Richardson and Domingos, 2006; Kersting and De Raedt, 2001]. Beyond known problems such as complexity or brittleness, an obvious limitation arises in this setup: logical rules over the knowledge graph relations are required for inference, and many knowledge graphs only provide observed facts [Dong et al., 2014; Auer et al., 2007]. In this case rules can be handcrafted, or learnt, generally through inductive logic programming (ILP) methods [Muggleton and De Raedt, 1994; Dzeroski and Lavrac, 1994].

Latent factor models do not suffer this limitation, as the learned model is never represented explicitly in a symbolic way, but rather as vectorial embeddings of the entities and relations. Such representations can make the model difficult to interpret, and although they show better predictive abilities, it has not yet been explored how well those models are able to overcome this absence of logical rules, and how their inference abilities differ from logic-based models.

To do so, we evaluate state-of-the-art latent factor models for relational learning on synthetic tasks, each designed to target a specific inference ability, and see how well they discover structure in the data. As we are only interested in evaluating inductive abilities of these models, and not their ability to cope with uncertainty, we design synthetic experiments with noise-free deterministic data. The choice of this very favorable setup for deterministic logical inference clarifies the approach followed here and its very purpose: we *do not* evaluate latent factor models as an *end*, but as a *means* to point out their weaknesses and stimulate research towards models that do not suffer from combinatorial complexity—as advocated by Bottou [2014]. Computational complexity, and namely polynomiality, could turn out to be the very criterion for machine intelligence [Aaronson, 2011]. Beyond complexity, one could also argue against explicitly learning logical expressions to tackle knowledge graph completion that, "when solving a given problem, try to avoid solving a more general problem as an intermediate step" [Vapnik, 1995].

In the previous chapter, we started to investigate synthetic symmetric and antisymmetric relations and special cases of transitivity, with specific training/testing splits, targeted at specific abilities. Here, we first extend these experiments with randomly generated combinations of all of the three main properties of binary relations: reflexivity, symmetry and transitivity. The splits between training, validation and test are random, as we want to assess models' ability to learn from realistically distributed data, and with more and more missing triples. Conversely, the symmetry experiments described in Section 4.1.2 are much easier as the upper-triangular matrix was always in the training set, as the goal was to see if the models can learn with perfect information. Then we set up tasks

that represent real reasoning over family genealogies. On this data, we explore different types of training/testing splits that map to different types of inference.

5.1 Experimental Setup

To assess whether latent factor models are able to generalize from data without any first-order logic rules, we generate synthetic data from such rules, and assess the models' ability to learn these patterns in a classical training, validation and test splitting of the data. The proportion of positives and negatives is respected across the sets. We evaluate the state-of-the-art latent factor models described in Section 2.2.1.1. Those are RESCAL, CP, DistMult, TransE, the F model and our proposal, ComplEx. Algorithm 1 describes the training algorithm, that is stochastic gradient descent with mini-batches (10 batches for the relation properties experiment, and 100 for the families experiment), AdaGrad [Duchi et al., 2011] with an initial learning rate of $\alpha = 0.1$, and early stopping when average precision decreased on the validation set calculated every 50 epochs. The λ regularization parameter was validated over the values $\{0.1, 0.03, 0.01, 0.003, 0.001, 0.0003, 0.00001, 0.0\}$ for each model for each factorization rank K. Parameters are initialized from a centered unit-variance Gaussian distribution.

Results are evaluated with average precision, as we also generate negative triples in these synthetic experiments. For each factorization rank, the models with best validated λ are kept. Average precisions are macro-averaged over 10 runs, and error bars show the standard deviation over these 10 runs. We also computed the average precision of a deterministic logic inference engine, by applying the corresponding rules that were used to generate each data set. For each fact $r(s, o)$ in the test set, its probability $P(y_{rso} = 1)$ is set to 1 if the fact can be logically deduced true from the facts of the training and validation sets, 0 if it can be deduced to be false, and 0.5 otherwise. This simulate test metrics of what perfect induction would yield, and gives an upper-bound on the performance of any method.

All data sets are made available[1].

5.2 Learning Relation Properties

In this section we define the three main properties of binary relations, and devise different experimental setups for learning them individually or jointly, and with more or less observed data.

[1] `https://github.com/ttrouill/induction_experiments`

5.2.1 Experimental Design

Relations in knowledge graphs have different names in the different areas of mathematics. Logicians call them binary *predicates*, as they are Boolean-valued functions of two variables. For set theorists, they are binary *endorelations*, as they operate on two elements of a single set, in our case the set of entities $\mathcal{E}$. In set theory, relations are characterized by three main properties: reflexivity/irreflexivity, symmetry/antisymmetry and transitivity. The definitions of these properties are given in first-order logic in Table 5.1.

Different combinations of these properties define basic building blocks of set theory such as equivalence relations that are reflexive, symmetric and transitive relations, or partial orders that are reflexive, antisymmetric and transitive relations [Halmos, 1998]. Examples are given in Table 5.2.

Property	Definition
Reflexivity	$\forall a \;\; r(a,a)$
Irreflexivity	$\forall a \;\; \neg r(a,a)$
Symmetry	$\forall a \forall b \;\; r(a,b) \Rightarrow r(b,a)$
Antisymmetry	$\forall a \forall b \;\; r(a,b) \wedge r(b,a) \Rightarrow a = b$
Transitivity	$\forall a \forall b \forall c \;\; r(a,b) \wedge r(b,c) \Rightarrow r(a,c)$

TABLE 5.1: Definitions of the main properties of binary relations.

Binary endorelations by property

	reflexivity	symmetry	transitivity	symbol	example
undirected graph	irreflexive	symmetric			
tournament	irreflexive	antisymmetric			pecking order
dependency	reflexive	symmetric			
strict weak order	irreflexive	antisymmetric	yes	$<$	
total preorder	reflexive		yes	$\leq$	
preorder	reflexive		yes	$\leq$	preference
partial order	reflexive	antisymmetric	yes	$\leq$	subset
partial equivalence		symmetric	yes		
equivalence relation	reflexive	symmetric	yes	$\sim, \cong, \approx, \equiv$	equality
strict partial order	irreflexive	antisymmetric	yes	$<$	proper subset

TABLE 5.2: Different types of binary relations in set theory. From Wikipedia page on binary relations [Wikipedia, 2004].

There are many such common examples of these combinations in knowledge graphs, as there are many hierarchical and similarity relations. For example, the relations `older` and `father` are both strict hierarchies, thus antisymmetric and irreflexive. But one is transitive (`older`) whereas the other is not, and that makes all the difference at inference time. Similarly for symmetric relations, such as `has-the-same-parents-as`

and `friend`, your sibling's parents are also yours which makes the first relation transitive, whereas your friend's friends are not necessarily yours. Note that this makes the `has-the-same-parents-as` relation reflexive—it is thus an equivalence relation.

Relational learning models must be able to handle relations that exhibit each of the possible combinations of these properties, since they are all very common, but imply different types of reasoning, as already acknowledged by Bordes et al. [2013a]. Given that a relation can be reflexive, irreflexive, or neither; symmetric, antisymmetric, or neither; and transitive or not, we end up with 18 possible combinations. However we will not address the cases of little interest where (i) none of these properties are true, (ii) only reflexivity or irreflexivity is true, (iii) the irreflexive, symmetric and transitive case as the only consistent possibility is that all facts are false, and (iv) the irreflexive transitive case that again must be either all false, or antisymmetric—and thus corresponds to an already existing case—to be consistent. Indeed, if one observes two true facts $r(s, o)$ and $r(o, s)$, by application of the transitivity rule, $r(s, s)$ and $r(o, o)$ must be true, which explains the inconsistency of cases (iii) and (iv), as they are irreflexive. This leaves us with 13 cases of interest. To evaluate the ability of models to learn these properties, we generate random 50×50 matrices that exhibit each combination.

To do so, we sample random square sign matrices $Y \in \{-1, 1\}^{N_e \times N_e}$. First we fill the diagonal with 1, -1 or missing depending on reflexivity/irreflexivity or none. Then we make successive passes over the data to make it [anti-]symmetric and/or transitive, until all of the properties are true over the whole matrix. A pass to make a matrix symmetric consists in assigning $y_{ji} \leftarrow y_{ij}$ for all $i, j \in 1, \ldots, N_e$ where $i < j$, and $y_{ji} \leftarrow -y_{ij}$ to make it antisymmetric. A pass to make a matrix transitive consists in assigning $y_{ij} \leftarrow 1$ if there exists a $k \in 1, \ldots, N_e$ such that $y_{ik} = y_{kj} = 1$, for all $i, j \in 1, \ldots, N_e$. When no more assignment is made during the passes it means the desired properties are true, and the relation generation is finished.

We also sample each matrix under the constraint of having a balanced number of positives and negatives up to $\pm 1\%$. Though there are many more negatives than positives in real knowledge graphs, in practice negatives are generally subsampled or generated to match the number of positive facts [Bordes et al., 2013b; Nickel et al., 2016b].

We first learn each relation individually as in a single relation knowledge graph, and then jointly. In the joint case, note that since each relation is generated independently, there is no signal shared across the relations that would help predicting facts of one relation from facts of another relation, thus only the ability to learn each relation patterns is tested. The proportion of observed facts is generally very small in real knowledge graphs. To assess models robustness to missing data, we also reduce the proportion of the training set when learning the different relations jointly.

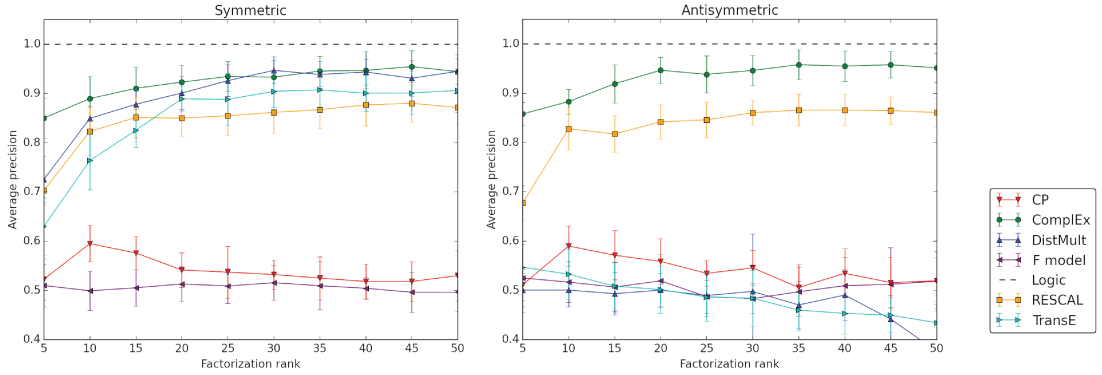


FIGURE 5.1: Generated symmetric (left) and antisymmetric (right) relations with 50 entities. Average precision for each rank ranging from 5 to 50 for each model.

5.2.2 Results

Results are first reported on each relation, then jointly and with decreasing proportion of training data. Reported average-precision scores are macro-averaged over a 10-fold cross-validation, with 80% training, 10% validation and 10% test in the individual learning case. In the joint learning case, the training percentage varies between 80% and 10%, the validation set size is kept constant at 10%, and the test set contains the remaining samples—between 10% and 80%.

5.2.2.1 Individual Relation Learning

First of all, results were identical for all models whether the relations were reflexive, irreflexive, or neither (unobserved). This tells us that reflexivity and irreflexivity are not so important in practice as they do not add or remove any quality in the prediction of latent factor models. We report only results for different combinations of symmetry/antisymmetry and transitivity in the main text. Results of combinations including reflexivity and irreflexivity are reported in Appendix B.

Figure 5.1 shows the average precision for each model over the generated symmetric and antisymmetric relations. Surprisingly, on such simple relations with 80% of observed data, only COMPLEX and RESCAL manage to learn from the symmetric and antisymmetric patterns, with a non-negligible advantage for the COMPLEX model. Moreover, we can see that with higher ranks, RESCAL starts overfitting as its average precision decreases presumably due to its quadratic number of parameters with respect to the rank, since each relation r is represented by a matrix $W_r \in \mathbb{R}^{K \times K}$, as showed in Section 2.2.1.1.

The CP model probably fails due to its uncorrelated representations of entities as subject and as objects, which makes it unable to model symmetry and antisymmetry. DISTMULT

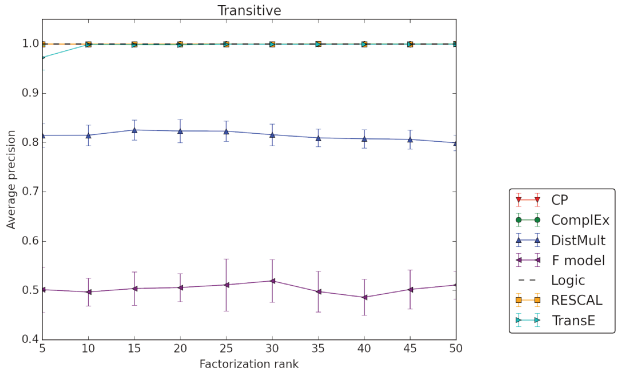


FIGURE 5.2: Generated transitive relation with 50 entities. Average precision for each rank ranging from 5 to 50 for each model.

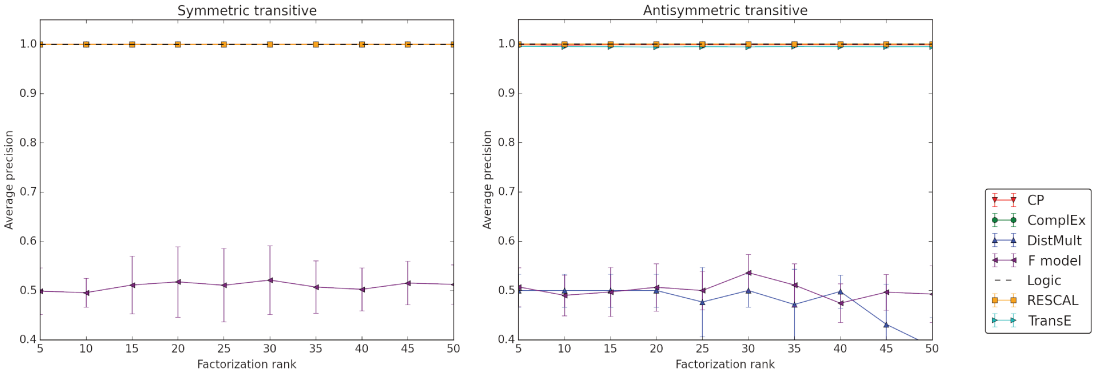


FIGURE 5.3: Generated symmetric and transitive (left) and antisymmetric and transitive (right) relations with 50 entities. Average precision for each rank ranging from 5 to 50 for each model.

unsurprisingly fails in the non-symmetric cases, due to the symmetric nature of its scoring function, and thus succeeds in the symmetric case. More unexpectedly, the TRANSE model has a hard time on antisymmetry, but performs well on the symmetric relation, by zeroing its relation embedding, as explained in Section 2.2.1. The F model, cannot actually generalize in a single relation case, as it has one single embedding for each (ordered) entity pair. For any fact $r(s, o)$ in the test set, the entity pair (s, o) has never been seen in the training set, and thus has a random embedding vector.

Figure 5.3 shows results for the symmetric transitive and antisymmetric transitive relations, and Figure 5.2 for the transitive only relations. Almost all models, except the F model and DISTMULT in the non-symmetric cases, perfectly generalize with very low-rank. This tells us that transitivity actually brings so much structure in the data that it makes the problem very easy for latent factor models when most of the data is observed.

Most state-of-the-art latent factor models are surprisingly unable to model all the basic properties of binary relations. Though most of the time those relations are learnt together,

but also with much less observed facts. We next assess the models ability to learn these five relations together, and their robustness to sparse observations by gradually decreasing the size of the training set.

5.2.2.2 Joint Learning

Figure 5.4 shows the results when all five above relations are jointly learned, for different proportions of the training set: 80%, 40%, 20%, 10%. As expected the scores drop, and the gap between the—deterministic logic—upper-bound and latent factor models widen with the decrease of training data. ComplEx proves to be the most robust to missing data down to 20%, but match logical inference only with 80% of training data.

RESCAL again overfits with the rank increasing, but is the best performing model with 10% of the training set, up to rank $K = 30$. This suggests that having richer relation representations than entity representations, that is with more parameters, can be profitable for learning relation properties from little data. However the reason why the variance of RESCAL's average precision decreases again for $K \geq 40$ remains mysterious.

The CP and TransE models seem to be more sensitive to missing data as their curves progressively get away from RESCAL's one with the percentage of observed data decreasing. DistMult, being a symmetric model, is below the other models in the four settings as some of the relations are not symmetric.

Since each relation is generated independently, having observed the entity pair (s, o) in the other relations does not help the F model, and it thus fails here too. At 10%, we see that the latent factor models cannot match logical inference, suggesting that the number of examples is not sufficient to learn these properties.

Finally, in the last setting with 10% of the training set, the best models are still 10 points below the best achievable average precision, showing that they need a large amount of training data to correctly learn these basic properties of binary relations.

These results should be taken cautiously as this experiment does *not* state that in general at least 80% of the facts should be observed in order to learn these properties correctly. Indeed, here the 5 relations are completely uncorrelated, while in real knowledge graphs they generally are correlated and thus share information. Also, as often in machine learning, the ratio between the number of parameters and the number of data points is more informative about generalization than the number of data points alone.

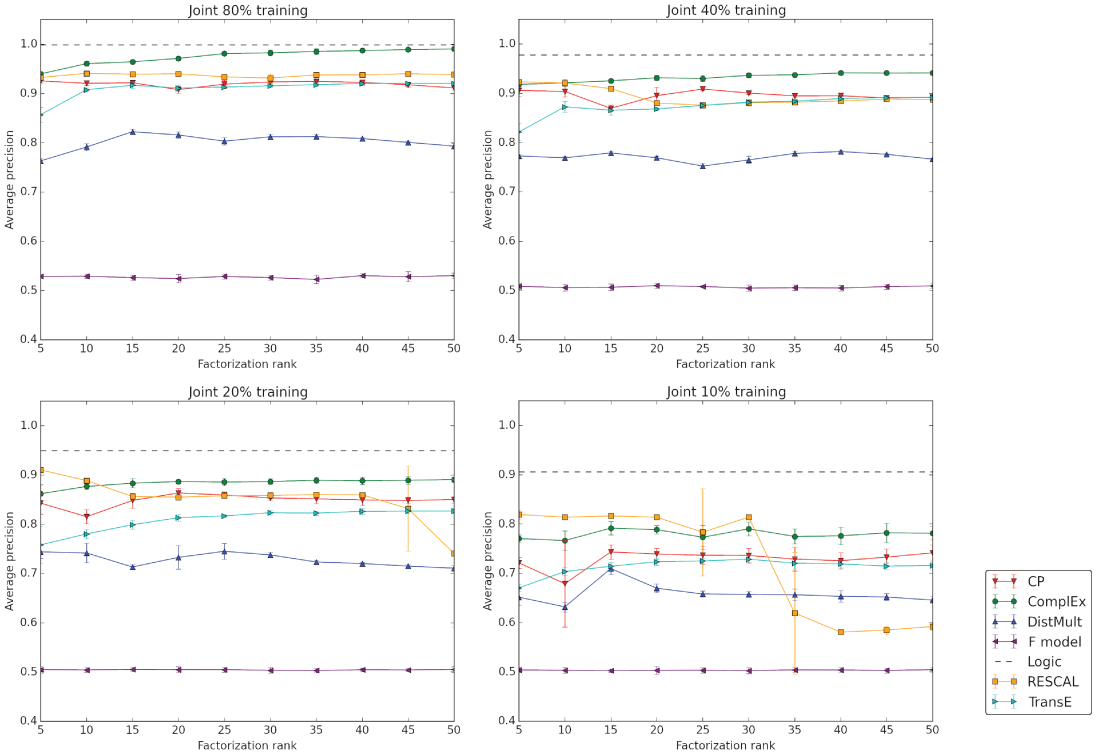


FIGURE 5.4: Joint learning of the 5 relations with 50 entities: one symmetric, one antisymmetric, one transitive, one symmetric and transitive, and one antisymmetric and transitive. Average Precision for each factorization rank ranging from 5 to 50 of each model. Top-Left: 80% train set, Top-Right: 40% train set, Bottom-Left: 20% train set, Bottom-Right: 10% train set.

Relation Properties Experiments Summary:

- Only COMPLEX manages to learn each combination near perfectly.
- RESCAL is the most robust to missing data with low ranks.
- There is room for improvement on all models when a lot of data is missing.

5.3 Learning Inter-Relational Patterns: Family Relationships

We generated family trees and their corresponding kinship relations and facts, and designed three different splits of the data. The three splits try to assess different inductive properties of the latent models, by giving more or less supporting facts in the training set.

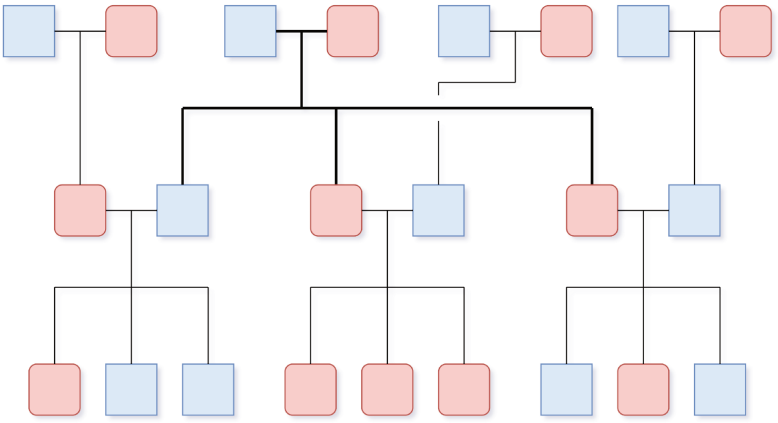


FIGURE 5.5: Example of a generated family tree.

5.3.1 Experimental Design

Predicting family relationships is an old task in AI, popularised by Hinton's kinship data set [Hinton, 1986]. Generated synthetic families for testing link-prediction models have also been recently proposed [García-Durán et al., 2014]. In this public dataset, generated families are all intertwined with each other in it. We here want each family to be disjoint from the other ones, such that there is no true fact between entities of two different families, and we will see why below.

We propose here to generate family relations from synthetic family trees, namely: `mother`, `father`, `husband`, `wife`, `son`, `daughter`, `brother`, `sister`, `uncle`, `aunt`, `nephew`, `niece`, `cousin`, `grandfather`, `grandson`, `grandmother` and `granddaughter`.

We sample five families independently that span over three generations from a unique couple, with three children per couple of random sex, where husbands, wives and their parents were added to wed the middle generation. Figure 5.5 shows an example of such a family tree. The whole data set totals 115 entities—23 persons per family—and the 17 relations mentioned above. Each family thus consists in 8993 true and false facts.

Within these traditional families that feature only married heterosexual couples that do not divorce and have children, one can figure out that the relations `mother`, `father`, `son` and `daughter` are sufficient to deduce the 13 remaining ones. Examples of rules that allow deducing these 13 relations from the 4 main ones are shown in Table 5.3. From this fact, we devise three different splits of the data.

Let us first introduce notations for each subset of the observed facts set Ω. As each family is independent from the four others, we will denote each entity set of each family from 1 to 5: $\mathcal{E}^1, \ldots, \mathcal{E}^5$, where $\mathcal{E}^i \cap \mathcal{E}^j = \emptyset$ with $i \neq j$. Accordingly, we will write the observed facts of each family $\Omega^1, \ldots, \Omega^5$, where for all $((r, s, o), y_{rso}) \in \Omega^i$ we have $s, o \in \mathcal{E}^i$. Observed

$\forall a \forall b \forall c$ father$(a, c) \wedge$ mother$(b, c) \Rightarrow$ husband(a, b)
$\forall a \forall b \forall c$ father$(a, c) \wedge$ mother$(b, c) \Rightarrow$ wife(b, a)
$\forall a \forall b \forall c$ daughter$(a, c) \wedge$ son$(b, c) \Rightarrow$ sister(a, b)
$\forall a \forall b \forall c$ daughter$(a, c) \wedge$ son$(b, c) \Rightarrow$ brother(b, a)
$\forall a \forall b \forall c$ father$(a, b) \wedge$ father$(b, c) \Rightarrow$ grandfather(a, c)
$\forall a \forall b \forall c$ son$(a, b) \wedge$ son$(b, c) \Rightarrow$ grandson(a, c)
$\forall a \forall b \forall c$ mother$(a, b) \wedge$ mother$(b, c) \Rightarrow$ grandmother(a, c)
$\forall a \forall b \forall c$ daughter$(a, b) \wedge$ daughter$(b, c) \Rightarrow$ granddaughter(a, c)
$\forall a \forall b \forall c \forall d$ son$(a, b) \wedge$ daughter$(b, c) \wedge$ son$(d, c) \Rightarrow$ uncle(d, a)
$\forall a \forall b \forall c \forall d$ daughter$(a, b) \wedge$ son$(b, c) \wedge$ daughter$(d, c) \Rightarrow$ aunt(d, a)
$\forall a \forall b \forall c \forall d$ son$(a, b) \wedge$ daughter$(b, c) \wedge$ son$(d, c) \Rightarrow$ nephew(a, d)
$\forall a \forall b \forall c \forall d$ daughter$(a, b) \wedge$ son$(b, c) \wedge$ daughter$(d, c) \Rightarrow$ niece(a, d)
$\forall a \forall b \forall c \forall d \forall e$ son$(a, b) \wedge$ daughter$(b, c) \wedge$ son$(d, c) \wedge$ daughter$(e, d) \Rightarrow$ cousin(a, e)

TABLE 5.3: Examples of rules to deduce all relations from the four relations: `mother`, `father`, `son` and `daughter`.

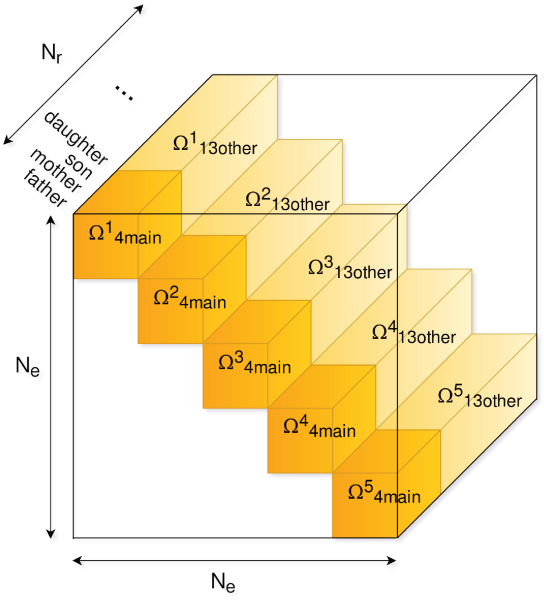
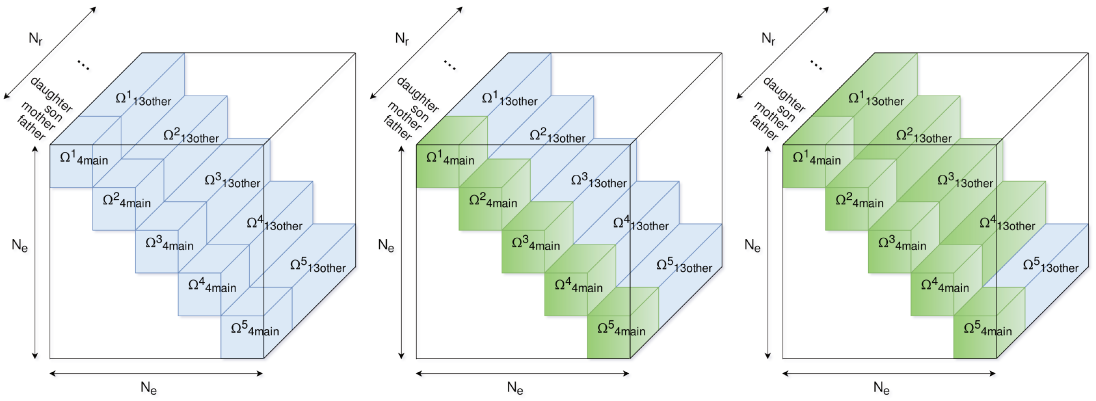


FIGURE 5.6: Tensor representation of the observed subsets for the family experiments. The part in dark orange represents the sets containing the four relations `mother`, `father`, `son` and `daughter`, while the part in light orange represents the 13 other relations.

fact sets that contain only the 4 main relations `mother`, `father`, `son` and `daughter` are denoted by $\Omega_{4\text{main}}$, and the facts involving the 13 other relations by $\Omega_{13\text{other}}$. We thus have for each family i: $\Omega^i = \Omega^i_{4\text{main}} \cup \Omega^i_{13\text{other}}$. Figure 5.6 draws the corresponding tensor with each subset of the observed facts. Finally, let the sampling function $\mathcal{S}_p(\Omega)$ be a uniformly random subset of Ω of size $|\mathcal{S}_p(\Omega)| = \lceil p|\Omega| \rceil$, with $0 \leq p \leq 1$, p being the proportion of the set that is randomly sampled.

We propose to split the data in three different ways to explore inductive abilities of the models. The first split is the classical *random split* between training, validation



(A) Random split (B) Evidence split (C) Family split

FIGURE 5.7: Tensor representation of the three different splits. Green sets are always contained in the training set Ω_{train}, whereas blue sets are split among training, validation and test sets.

and test sets, it will mainly serve as a control experiment for the other splits. In the second split, we aim at evaluating whether latent factor models are able to leverage this information. To do so, we ensure that all the relations `mother`, `father`, `son` and `daughter` of the five families are in the training set, and we split the 13 remaining ones between training, validation and test set. Formally: $\Omega_{\text{train}} = \Omega_{\text{4main}} \cup \mathcal{S}_p(\Omega_{\text{13other}})$. We will call this splitting scheme the *evidence split*, as the training set always contains the sufficient evidence to deduce the 13 other relations—that is the four main ones.

Thirdly, we assess the ability of latent factor models to transfer knowledge learnt from a family to another, that is between disjoint set of entities. In this split, the training set always contains all the relations for four out of the five families plus all the `mother`, `father`, `son` and `daughter` of the fifth family, while the 13 other relations of this fifth family are split between training, validation and test set. Formally: $\Omega_{\text{train}} = \Omega^{1-4} \cup \Omega^5_{\text{4main}} \cup \mathcal{S}_p(\Omega^5_{\text{13other}})$. We will call it the *family split*. Figure 5.7 shows tensor drawings of the three splits.

For each split we explore different values of $p \in \{0.8, 0.4, 0.2, 0.1\}$. We also run with $p = 0$ in the last (family) split, which corresponds to $\Omega_{\text{train}} = \Omega^{1-4} \cup \Omega^5_{\text{4main}}$, that is 4 entirely observed family, plus the 4 main relations of the fifth one. Observe that it only makes sense to have $p = 0$ in this last split. If latent factor models have expected inductive abilities, they would be able to understand genealogical reasoning from the four first families, and use this learned information to correctly predict the 13 other relations of the fifth family from its four main ones. Note that in the last two splits, a deterministic logic inference system makes perfect predictions—given rules such as the ones shown in Table 5.3—for any value of p. The number of facts in the training, validation and test sets of each split are summarized in Table 5.4.

Split	Set	Size with $p =$ 0.8	0.4	0.2	0.1	0
Random	$\Omega_{\text{train}} = \mathcal{S}_p(\Omega)$	35973	17987	8994	4496	-
	$\Omega_{\text{valid}} = \mathcal{S}_{0.1}(\Omega)$	4496	4496	4496	4496	-
	$\Omega_{\text{test}} = \mathcal{S}_{(0.9-p)}(\Omega)$	4496	22482	31475	35973	-
Evidence	$\Omega_{\text{train}} = \Omega_{\text{4main}} \cup \mathcal{S}_p(\Omega_{\text{13other}})$	38089	24334	17457	14019	-
	$\Omega_{\text{valid}} = \mathcal{S}_{0.1}(\Omega_{\text{13other}})$	3438	3438	3438	3438	-
	$\Omega_{\text{test}} = \mathcal{S}_{(0.9-p)}(\Omega_{\text{13other}})$	3438	17193	24070	27508	-
Family	$\Omega_{\text{train}} = \Omega^{1-4} \cup \Omega^5_{\text{4main}} \cup \mathcal{S}_p(\Omega^5_{\text{13other}})$	43589	40839	39463	38776	38088
	$\Omega_{\text{valid}} = \mathcal{S}_{0.1}(\Omega^5_{\text{13other}})$	688	688	688	688	688
	$\Omega_{\text{test}} = \mathcal{S}_{(0.9-p)}(\Omega^5_{\text{13other}})$	688	3438	4814	5501	6189

TABLE 5.4: Training, validation and test set numbers for each split for each value of p.

Similar splits of data have already been proposed to evaluate rule-based inference models (for example the UW-CSE dataset [Richardson and Domingos, 2006]), which are able of such transfer of reasoning between disjoint sets of entities. Interestingly, such data sets have rarely been reused in the subsequent latent factor model literature. Results reported next might give us a hint why this is the case.

5.3.2 Results

Results are reported for each split separately. In each of them we again decrease progressively the amount of training data, and report average precision macro-averaged over 10 runs for each configuration.

5.3.2.1 Random Split

In the first random split, we try to evaluate the quantity of training data needed to learn to reason in genealogies. Figure 5.8 shows the average precision of each model for ranks ranging from 5 to 50, for each value of p. Only COMPLEX and RESCAL are able to generalize almost perfectly with 80% of observed data, which first tells us that these models are indeed capable to learn such genealogical reasonings. As many relations are antisymmetric, it is no surprise that DISTMULT and TRANSE cannot reach perfect predictions, as they already failed in the antisymmetric synthetic relation.

The COMPLEX model generalizes quickly with small ranks, but is outrun by RESCAL—with small ranks—and TRANSE when the percentage of observed data decreases below $p = 0.2$. We conjecture that TRANSE's robustness is due to its bilinear terms, and especially the one that involves the subject and the object embeddings—$e_s^\top e_o$—as shown in Section 2.2.1.1, that can give high scores to pairs of entities belonging to the same

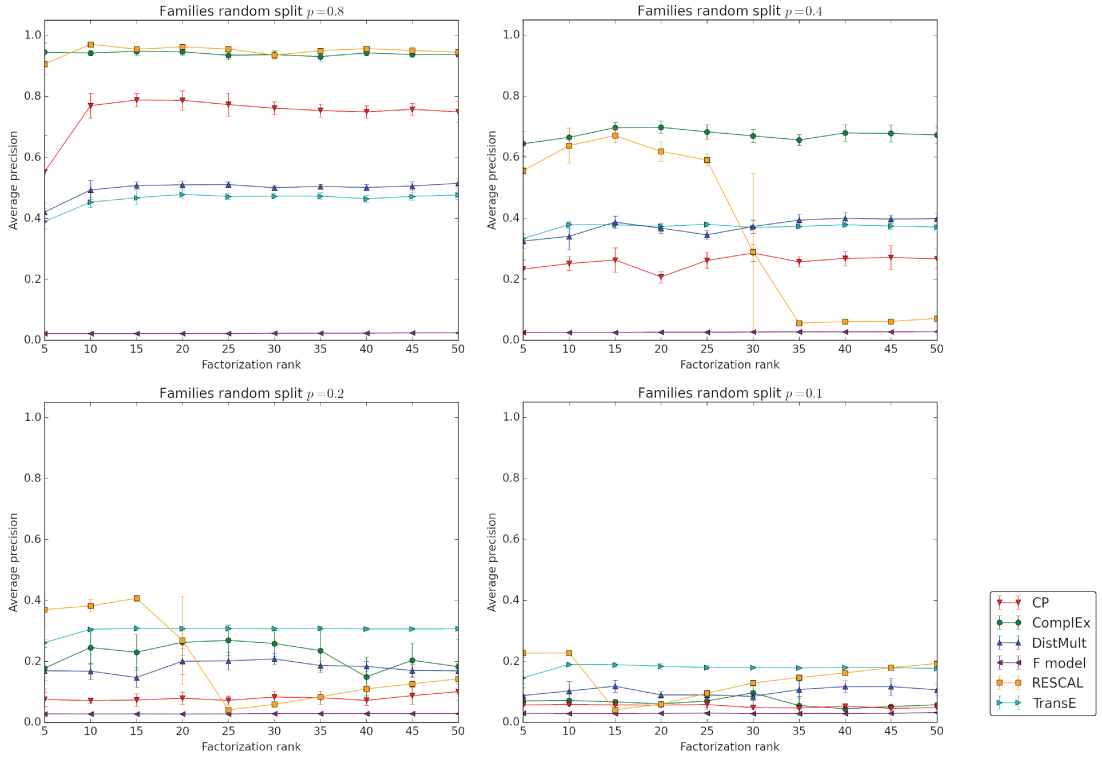


FIGURE 5.8: Average Precision for each factorization rank ranging from 5 to 50 of each model on the families experiment with the random split. Top-left: $p = 0.8$, top-right: $p = 0.4$, bottom-left: $p = 0.2$, bottom-right: $p = 0.1$.

family. For RESCAL, its richer representations of relations by matrices probably help here, as long as the rank is not too big which clearly causes overfitting.

The CP decomposition scores drop quickly with the proportion of observed data, because of its unrelated subject and object representations. The F model here fails again, for a simple reason: these relations are exclusive between themselves for a given pair of entities (s, o). Indeed, if `father`(s, o) is true for example, then none of the other relations between s and o will be true—at least not in these families. Hence if the F model has to predict the score of the fact $r(s, o)$, it has no other true triple involving (s, o) to support its decision. It will also have troubles on the two next splits for the same reason. Note that in this split, the logic upper-bound is not given as one would need to know all possible rules to deduce the 17 relations from each of them—and not only from the four main ones—to compute this upper-bound.

5.3.2.2 Evidence Split

In this split, we recall that all the `mother`, `father`, `son` and `daughter` relations are always in the train set for the 5 families. The value of p ranging from 0.8 to 0.1 corresponds

here to the proportion of the 13 other relations that are also put into the training set. The test and validation sets are only composed of these 13 relations.

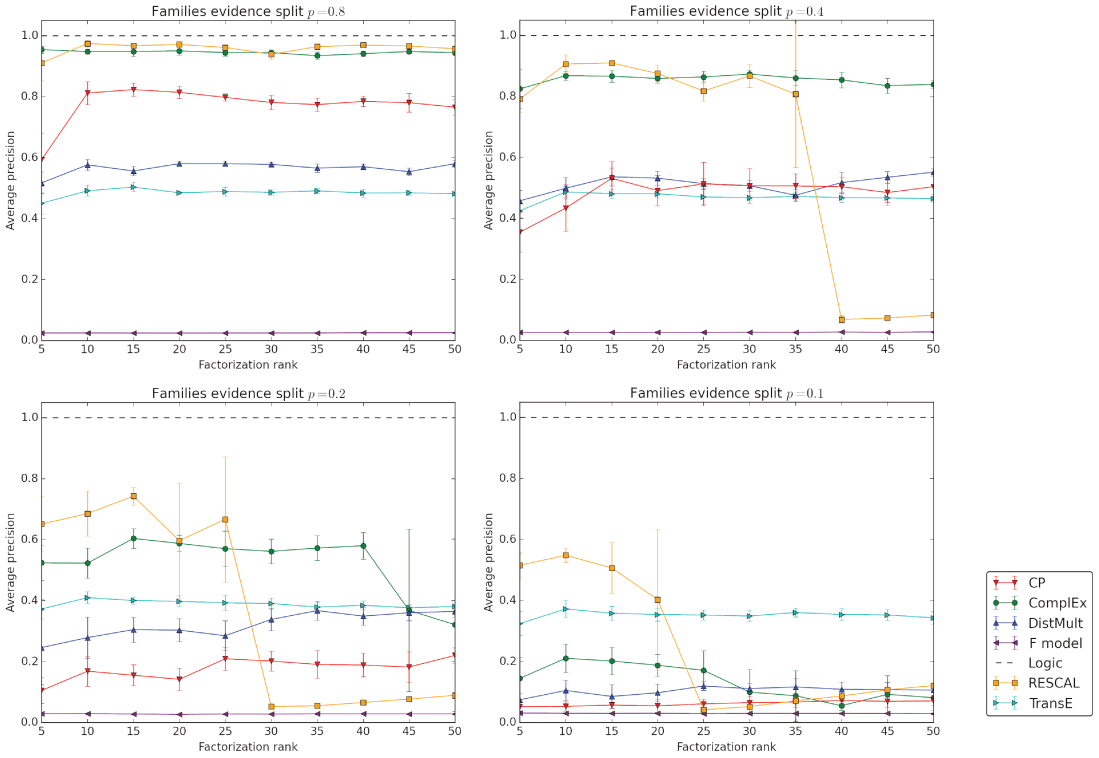


FIGURE 5.9: Average Precision for each factorization rank ranging from 5 to 50 of each model on the families experiment with the evidence split. Top-left: $p = 0.8$, top-right: $p = 0.4$, bottom-left: $p = 0.2$, bottom-right: $p = 0.1$.

Compared to the random split setting, we can see in Figure 5.9 that the performances of the models decrease more slowly with the percentage of observed data. This shows that latent factor models are able to use the information provided by these four relations from which all of the others can be deduced.

RESCAL is here clearly the best model for all values of p, as long as K is not too big. It exhibits again a behavior that seems to have two equilibria distributed around a pivotal K at which average precision suddenly drops, with high variance of the predictions around that K. COMPLEX also seems to show a lighter overfitting with high values of K when $p \leq 0.2$. TRANSE confirms an advantage with $p = 0.1$ with a notable rise of average precision compared to the random split. CP, DISTMULT and the F model fail again for the same reasons as in the random split.

However, given the rules to deduce the 13 other relations from the four main ones, recall that a logical inference engine is able to reach an average precision of one. Though improvement compared to the random split setup is large, the gap with logical inference is still wide with $p = 0.1$ and $p = 0.2$, showing that latent factor models have troubles making the link between the four main relations and the 13 other ones when limited training

data is available. This could be due to the imbalance in the number of each relation in the training set that this split introduces, biasing the entity embeddings towards a better reconstruction of the 4 main relations, to the detriment of the generalization over the 13 remaining ones. Weighting the facts in accordance with the preponderance of each relation in the dataset could improve performances here.

5.3.2.3 Family Split

In this last split, all the `mother`, `father`, `son` and `daughter` are in the train set for all families, but also all the 13 other relations of four out of the five families. The value of p corresponds here to the amount of the 13 other relations *of the fifth family only* that are in the training set too.

The curves in Figure 5.10 show a clear improvement over the previous ones in Figure 5.9. RESCAL is again the best model as it reaches average precisions ≥ 0.9 even down to $p = 0.1$—with small ranks again. ComplEx is in these cases the best with high ranks, though much below RESCAL's best scores when $p = 0.1$.

Does that mean these models were able to exploit the additional information? Yes and no. We conjecture that the better results for p ranging from 0.8 to 0.1 are partly due to the relation imbalance problem—explained in the previous split—being much smaller here, as all the relations of four families are given in in the training set.

To ensure that models indeed did not generalized from the four perfectly informed families, we reduced the proportion p of the 13 other relations of the fifth family that are in the training set to zero—which thus constitute the whole validation and test sets. And though the models are provided with four perfectly informed families, and all the needed facts to predict the missing ones in the fifth family, they fail in this last setting as shown in the bottom plot of Figure 5.9. RESCAL and TransE resist better than the other models again in this last setting with $p = 0$.

This is easily explained, as disconnected sets of entities, here families, correspond to different blocks in the tensor $\mathbf{Y}$, as shown in Figure 5.6. Entities that are in different families $s, o \in \Omega^i$, $s', o' \in \Omega^j$, $i \neq j$, are never involved together in an observed fact: $((s, r, o'), y_{sro'}), ((s', r, o), y_{s'ro}) \notin \Omega$, for any relation $r \in \mathcal{R}$. Thus when learning their embeddings e_s, e_o and $e_{s'}, e_{o'}$, the only link they share is the embedding of the relation r that is involved in the scoring functions $\phi(r, s, o)$ and $\phi(r, s', o')$. This interpretation is also supported by RESCAL scores, which benefits from its higher number of parameters of its relation representations $W_r \in \mathbb{R}^{K \times K}$, which increases the amount of information shared across the families.

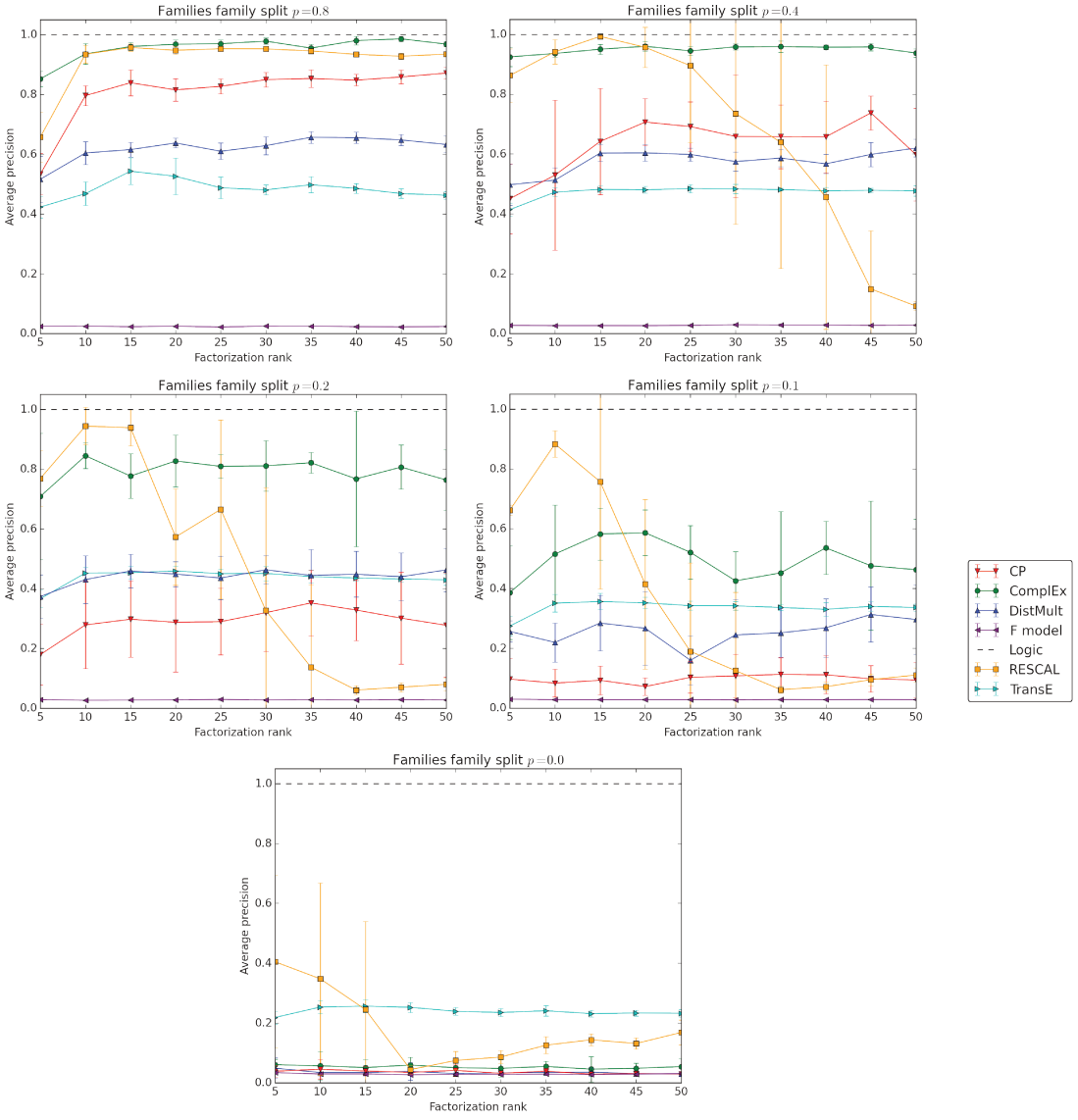


FIGURE 5.10: Average Precision for each factorization rank ranging from 5 to 50 of each model on the families experiment with the family split. Top-left: $p = 0.8$, top-right: $p = 0.4$, middle-left: $p = 0.2$, middle-right: $p = 0.1$, bottom: $p = 0.0$.

To formally bring this problem to light, let us re-frame the tensor approximation problem as a system of inequalities. As $P(y_{rso} = 1) = \sigma(\phi(r, s, o; \Theta))$, true triples should have a score $\phi(r, s, o; \Theta) > 0$, and false triples $\phi(r, s, o; \Theta) < 0$. For the sake of the example, we will consider factorizing the two relations `sister` and `grandfather`, with two families of 2 persons each, using the DISTMULT model. Observing the true fact `sister(a,b)` or the true fact `sister(b,a)` where $a, b \in \mathcal{E}$, allows us to deduce that a and b are not the grandfather of each other:

- $\texttt{sister}(\texttt{a}, \texttt{b}) \Rightarrow \neg\texttt{grandfather}(\texttt{a}, \texttt{b})$
- $\texttt{sister}(\texttt{a}, \texttt{b}) \Rightarrow \neg\texttt{grandfather}(\texttt{b}, \texttt{a})$

- $\texttt{sister}(\texttt{b},\texttt{a}) \Rightarrow \neg\texttt{grandfather}(\texttt{a},\texttt{b})$
- $\texttt{sister}(\texttt{b},\texttt{a}) \Rightarrow \neg\texttt{grandfather}(\texttt{b},\texttt{a})$

Similarly to the family split with $p = 0$, let us have both relations fully observed for a first family that contains entities $a, b \in \mathcal{E}^1$, and only the facts of the relation `sister` observed for entities of a second family $c, d \in \mathcal{E}^2$. The resulting $2 \times 4 \times 4$ partially-observed binary tensor is:

$$\texttt{sister}: \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{pmatrix} -1 & 1 & & \\ 1 & -1 & & \\ & & -1 & 1 \\ & & 1 & -1 \end{pmatrix} \end{array}, \qquad \texttt{grandfather}: \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{pmatrix} -1 & -1 & & \\ -1 & -1 & & \\ & & \cdot & \cdot \\ & & \cdot & \cdot \end{pmatrix} \end{array} \tag{5.1}$$

where $\cdot$ and empty spaces are unobserved facts. From the first, fully observed family we wish to learn the above rules and the irreflexivity of the `grandfather` relation, to correctly complete the `grandfather` facts between entities c and d.

As the observed blocks—and the block we wish to recover—are symmetric here, there is no expressiveness issue with using DistMult. Decomposing this tensor with the DistMult model with $K = 2$ such that true facts have probability $P(y_{rso} = 1) > 0.5$ and false facts have probability $P(y_{rso} = 1) < 0.5$, amounts to solving the following system of inequalities:

$$\begin{cases} w_{s1}e_{a1}^2 + w_{s2}e_{a2}^2 < 0 \\ w_{s1}e_{b1}^2 + w_{s2}e_{b2}^2 < 0 \\ w_{s1}e_{a1}e_{b1} + w_{s2}e_{a2}e_{b2} > 0 \\ w_{g1}e_{a1}^2 + w_{g2}e_{a2}^2 < 0 \\ w_{g1}e_{b1}^2 + w_{g2}e_{b2}^2 < 0 \\ w_{g1}e_{a1}e_{b1} + w_{g2}e_{a2}e_{b2} < 0 \\ \\ w_{s1}e_{c1}^2 + w_{s2}e_{c2}^2 < 0 \\ w_{s1}e_{d1}^2 + w_{s2}e_{d2}^2 < 0 \\ w_{s1}e_{c1}e_{d1} + w_{s2}e_{c2}e_{d2} > 0 \end{cases} \tag{5.2}$$

where $e_i \in \mathbb{R}^2$ is the embedding of entity $i \in \mathcal{E}$, $w_s \in \mathbb{R}^2$ is the embedding of the relation `sister`, and $w_g \in \mathbb{R}^2$ is the embedding of the relation `grandfather`. The six first inequalities involve the entities a and b, and the three lower ones involve the entities c and d.

Correctly reconstructing the `grandfather` facts between c and d would thus require their embeddings to satisfy the same three additional inequalities:

$$\begin{cases} w_{g1}e_{c1}^2 + w_{g2}e_{c2}^2 < 0 \\ w_{g1}e_{d1}^2 + w_{g2}e_{d2}^2 < 0 \\ w_{g1}e_{c1}e_{d1} + w_{g2}e_{c2}e_{d2} < 0 \end{cases} . \tag{5.3}$$

However, it is easy to check that arbitrary solutions to the system (5.2) for e_c and e_d does not necessarily satisfy the system (5.3), and hence does not necessarily predict the `grandfather` facts between c and d correctly. Also, this would be true even if we added more families like a and b with both relations fully observed, as this would not add more constraints on e_c and e_d.

This explains why all models fail in the family split with $p = 0$: nothing encourages less constrained entities to have embeddings that resemble the ones of similar, more constrained entities; and adding more examples of more constrained entities does not help.

Family Experiments Summary:

- RESCAL is the best model in all different splits, but overfits with a too big K.
- RESCAL and TransE are the most robust to missing data.
- ComplEx behaves well with more data and hardly overfits.
- Relation imbalance in the training set can be a problem when the test set is distributed differently, and could be easily fixed by weighting the facts accordingly.
- The absence of explicit parameter sharing between entity representations prevents knowledge transfer between disjoint sets of entities.

5.4 Future Research Directions

Overall, the ComplEx model proved to have the more stable generalization abilities across all the synthetic experiments. Most models showed a good ability to learn basic relation properties, except on antisymmetry where only ComplEx succeeded. This said, when decreasing the size of the training set down to 10% on joint learning of the relation properties, the best models were 10 points of average precision behind the best possible score. Improving models towards learning basic binary relation properties from less data thus seems a promising direction.

Some models showed their advantages in some specific settings. RESCAL and TransE showed a good robustness when a lot of data is missing in the family experiments, thanks to the bilinear terms for TransE, and the rich matrix relation representations of RESCAL. The F model was not fit for these experiments, but its pairwise terms are known to give it an advantage for non-compositional pairs of entities [Welbl et al., 2016].

Different possible combinations seem promising. The behaviour of RESCAL and ComplEx on symmetric and antisymmetric experiments suggests that encoding these patterns through complex conjugation is more stable than using the non-commutative matrix product. But RESCAL's matrix representations of relations helped a lot in the family experiments, as long as the rank was not too high, suggesting that there might be a middle ground between K and K^2 to be found for the parametric representation of the relations. Using tridiagonal or pentadiagonal (or more) symmetric matrices for relation representations within the ComplEx model could be an answer to these problems.

Combining the scoring functions of the TransE and F models with ComplEx could also lead to a more robust model. The combination of bilinear and trilinear terms has already been explored within real-valued models [García-Durán et al., 2014], also with vectorial weights over each term [Jenatton et al., 2012], as well as combining different pairwise terms [Welbl et al., 2016; Singh et al., 2015], which yielded better performance in all cases.

The main defect of latent factor models that this experimental survey points to is their low ability to transfer knowledge between disjoint set of entities, as shown in the last family split with $p = 0$. Real knowledge graphs might not have fully disjoint subsets, but rather some less-connected sub-graphs, between which this effect is likely to appear too. We believe improving this ability of latent factor models is key.

One already-pursued way to harness this problem is to enable latent factor models to make use of logic rules [Rocktaschel et al., 2015; Demeester et al., 2016]. As already said, those rules are not always available, and thus latent factor models should be improved in order to have this ability to learn from disjoint subsets, while still operating without rules.

Intuitively, sharing parameters across all entity representations could also solve this issue, as used in Bayesian clustered factorization models [Sutskever et al., 2009]. Though those models have known scalability issues. A possible, more scalable way to implement a shared parametrization between the entity embeddings $E \in \mathbb{C}^{N_e \times K}$ is through a nested factorization, where the matrix E is itself expressed as a low-rank factorization, as it has already been proposed for the relation embeddings [Jenatton et al., 2012]. Another

one could be a suited regularization over the whole matrix E: in most proposals E is regularized row-wise with $||e_i||_2^2$ for all $i \in \mathcal{E}$—as shown in Section 3.3.

Another linked limitation of latent factor models—that does not require experiments to be shown—is their inability to generalize to new entities without retraining. Indeed for new facts involving a new entity i, its embedding $e_i \in \mathbb{C}^K$ is unknown. But in a logic-based setting, only the new facts involving the new entity are necessary to infer other facts from known rules. Some recent works started tackling this problem: Verga et al. [2017] proposed a solution for the F model, by expressing entity pair embeddings as combinations of the relation embeddings in which they appear. Hamaguchi et al. [2017] used graph neural networks to handle unseen entities at test time.

The evidence split in the family experiments also pointed out a potential problem of imbalance in the distribution of the relations across the facts when the train and test sets are distributed differently. Correcting this imbalance via down-weighting the facts involving the most frequent relations could be a solution, as well as sharing the parametrization between the relations.

A non-mentioned aspect of the problem in this paper is the theoretical learnability of such logic formulas, a field that has been extensively covered [Valiant, 1984; Kearns and Valiant, 1994; Muggleton and De Raedt, 1994; Dzeroski and Lavrac, 1994]. However logic learnability by latent factor models has not yet been specifically studied. Recently established links between sign matrices complexity—specifically the sign-rank [Linial et al., 2007b]—and VC-dimension open the door to such theoretical study [Alon et al., 2016], and possible extensions to the tensor case. This being said, theoretical guarantees generally come under the condition that the training and test sets are drawn from the same distribution, which is not the case in the last two splits of the family experiments: a theoretical analysis of the learnability of such cases might require a new theoretical framework for statistical learning.

Chapter Summary

We experimentally surveyed state-of-the-art latent factor models for link prediction in knowledge graphs, in order to assess their ability to learn (i) binary relation properties, (ii) genealogical relations, directly from observed facts, as well as their robustness to missing data. Latent factor models yield good performances in the first case, while having more difficulties in the second one. Specifically, we show that such models do not reason as it is generally meant for logical inference engines, as they are unable to transfer their predictive abilities between disjoint subsets of entities. The different behaviors of the

models in each experimental setup suggest possible enhancements and research directions, including combining them, as well as it exposes each model's advantages and limitations.

Chapter 6

Conclusion

Knowledge-based systems, such as automated personal agents or recommender systems, require robust link-prediction abilities to become viable, as the knowledge graphs they rely on are often largely incomplete. This work aimed at improving factorization models for link prediction in knowledge graphs. We followed an empirical approach to spot weaknesses of existing models, starting with the very basics: properties of binary relations. From the evidence that the correct modeling of all these properties, especially antisymmetry, was not already covered by existing models, we designed a new tensor factorization model named ComplEx. We turned ourselves to the large legacy of matrix theory for inspiration, and leveraged on complex linear algebra to create this new model. The ComplEx model fulfilled the task of modeling all basic properties of binary relations, and provided new state-of-the-art results on classic benchmarks for link prediction, while being scalable. We finished our study as we started it, with experiments on which all current factorization models—including ComplEx—fail, thereby opening the path to future improvements.

This last chapter summarizes the contributions of this thesis, and proposes future research directions.

6.1 Contributions

We proposed a novel, non-unique decomposition for arbitrary square matrices, based on the projection onto the real sub-space of a unitary diagonalization (Section 3.1). This decomposition always exists (Theorem 2) with a number of dimensions that is at most twice as large as the rank of the decomposed matrix. These properties are also true for sign matrices, and their corresponding complexity measure, the sign-rank. We extended

the analysis to the 3$^{\text{rd}}$-order tensor case when jointly decomposing a set of arbitrary square matrices (Section 3.2), and showed that the decomposition also exists with a rank upper-bound matching the canonical polyadic decomposition's (CP) rank upper-bound, despite having only two factor matrices—for the relations and the entities—instead of three—one per dimension for CP. As knowledge graphs correspond to *partially-observed* sign tensors, we proposed a stochastic gradient descent algorithm to learn the proposed decomposition model while naturally ignoring the missing values. Not imputing the missing values is essential for generalization [Drumond et al., 2012], but also for scalability given the size and sparsity of knowledge graphs.

When we started this work, our goal was to create a model that would be expressive enough to model all possible relations, yet ensure a linear time and space complexity to be scalable, and that generalizes well on real data. The use of complex-valued embeddings allowed us to achieve this goal, by keeping unique representations of entities which is essential to ensure good generalization, vectorial representations of relations for scalability, and correctly modeling asymmetry through the use of the complex conjugation. Experiments confirmed its abilities in practice, as ComplEx yields state-of-the-art results on all classic link-prediction data sets, but can also successfully learn all combinations of the basic binary-relation properties. The assumption that knowledge graphs tend to have low sign-rank relations that can be efficiently approximated with a binary surrogate such as the logistic function, combined with our model, was confirmed in practice as prediction scores converged with low embedding sizes. The ComplEx model especially confirmed its ability to model antisymmetric relations on WordNet data. But also that it could be used for enriching vectorial representations of words, which proved useful in the natural language processing task of entailment recognition. Finally, the ComplEx model is among the first works to bring complex linear algebra in the machine learning community.

We conducted an experimental survey on state-of-the-art latent factor models for link prediction, to better understand the effect of different parametrization choices on the ability to learn patterns from observed data. Specifically, experiments tested the models' ability to learn combinations of basic relation properties, to learn genealogical relations given different evidence about the families, and the models' robustness to missing data. These last experiments on families exposed the inability of latent factor models to transfer knowledge between disjoint sub-graphs in knowledge graphs. The matrix representations of relations in the RESCAL model yielded a better robustness to this issue, and in general to missing data, though it also caused it to overfit when the rank of the decomposition becomes too large. Bigram terms of the TransE model also shown a good robustness to this effect, by highly scoring pairs of entities belonging to the same family. The DistMult model has expected problems with asymmetric relations, and the F model with knowledge

graphs featuring exclusive relations. The CP model unrelated representations of entities as subject and object make it very sensitive to missing data. The ComplEx model proves to be a safe choice as it performs well in most cases. These findings point where to improve existing models, as well as which choices to make and to avoid for practitioners, depending on the distribution of their data.

6.2 Future Work

We divide future research directions into theoretical and practical directions.

Theoretical Directions

Among the theoretical properties of the proposed decomposition that have been discussed, several improvements are possible. The tightness of the $2K$ upper-bound on the existence of the decomposition in the matrix case, discussed in Section 3.1.2.3, could be investigated. We showed that the decomposition was not unique, characterizing the ensemble of existing decompositions and their generalization properties at a given cut-off rank K could help design more efficient algorithms to compute it. In the tensor case, we showed that the decomposition always exist provided K is big enough, however we could not prove or disprove its existence with embeddings of size inferior or equal to the dimension of the square matrices $K \leq N_e$ (see Section 3.2.2).

We briefly mentioned the extension of the sign-rank to the tensor case, however its properties has not yet been studied. Exploring sign-rank for tensors and its properties, especially in the case of a set of square matrices, is a yet unexplored field. In practice, we demonstrated that the logistic loss is a good surrogate for matching the sign-pattern of sign matrices and tensors. How good is that surrogate for bilinear and trilinear models could be quantified. This leads to addressing the non-convexity of these models, and more precisely quantifying the spuriousness of local minima. In recent studies, Ge et al. [2016] showed that in the bilinear semi-definite matrix completion problem, all local minima were in fact global. The stability of prediction scores from different random initializations that we observed with the ComplEx model could be the result of such a property. This does not exclude also studying convex relaxations for sign matrices: the trace-norm is well-known to be the convex hull of the classical rank [Candes and Recht, 2012], but the convex hull of the sign-rank is as yet unknown. Finally, the links between the VC-dimension and the sign-rank [Alon et al., 2016] open a path to study the learnability of first-order logic rules from ground predicates encoded as sign matrices by decomposition models.

Practical Directions

In this work, we chose to consider sampling negative triples using the local closed-world assumptions as real negatives, and to optimize the classical log-likelihood loss instead of the more often used max-margin pairwise loss. On the FB15K data set of Freebase, this yielded a large improvement in predictive abilities (see Section 4.3.6), which highlights the importance of the loss in the link-prediction problem, an aspect of the problem that has yet been barely studied, and thus should be explored. In Section 4.3.4, we showed that sampling more than one negative for each positive triple can also bring a large performance improvement. However the procedure is costly as it adds as many samples to optimize over, and thus calls for a more intelligent sampling of negatives, that contrast more with the positives from which they have been sampled.

Chapter 5's experimental survey of existing models pointed out many possible enhancements of existing models, including combining parts of their scoring functions. Our study of the models' robustness to missing data could be extended to assess their capacity to cope with corrupted data. Solving the learning problem between disjoint sets of entities require a scalable way of binding the parametrization of entity embeddings together that is yet to be found. Furthermore, most existing latent factor models are unable to generalize to new triples involving unseen entities and relations without a retraining step.

There are also more general future directions for knowledge graph models. Integrating time is one, as some facts are only true for a given period, such as the living place of a person or the president of a country. But also a proper handling of entities that represent algebraic values or dates, such as `hasAge(John,42)`, for which it makes little sense to learn an embedding for each different value. Extension to relations between more than two entities, n-tuples, is not straightforward, as ComplEx's expressiveness comes from the complex conjugation of the object-entity, that breaks the symmetry between the subject and object embeddings in the scoring function. This stems from the Hermitian product, which seems to have no standard multilinear extension in the linear algebra literature, this question hence remains largely open.

The ComplEx model could also be used in other problems than link prediction, actually for any problem that can be formulated as the completion of one or more square matrices. Decomposing knowledge graphs itself could also serve other applications by learning or enhancing vectorial representations of entities, which are then used for some downstream task, as we showed with word embeddings for entailment recognition (see Section 4.4).

As a final word, it is by building experiments that target specific inference abilities, starting with the basics, that we were put on the track of weak spots to improve on.

We believe this is a good experimental design practice, and humbly hope to inspire the reader.

Appendix A

Accelerating Stochastic Gradient Descent via Online Learning to Sample

Another contribution of this thesis, which is only partially related to knowledge graph completion—by its application on matrix factorization—is reported in this Appendix.

Stochastic Gradient Descent (SGD) is one of the most widely used techniques for online optimization in machine learning. In this work, we accelerate SGD by adaptively learning how to sample the most useful training examples at each time step. First, we show that SGD can be used to learn the best possible sampling distribution of an importance sampling estimator. Second, we show that the sampling distribution of an SGD algorithm can be estimated online by incrementally minimizing the variance of the gradient. The resulting algorithm—called Adaptive Weighted SGD (AW-SGD)—maintains a set of parameters to optimize, as well as a set of parameters to sample learning examples. We show that AW-SGD yields faster convergence on matrix factorization, where rows and columns are not sampled uniformly.

We first introduce the idea of this work in Appendix A.3, before reviewing the related work in Appendix A.2. We show that SGD can be used to find the optimal sampling distribution of an importance sampling estimator (Appendix A.3). This variance reduction technique is then used during the iterations of a SGD algorithm by learning how to reduce the variance of the gradient (Appendix A.4). We then illustrate this algorithm—called Adaptive Weighted SGD (AW-SGD)—on matrix factorization (Appendix A.5). Other application domains such as image classification and reinforcement learning are reported in [Bouchard et al., 2015b], but were not part of this thesis, and thus are not reported here.

A.1 Introduction

In many real-world problems, one has to face intractable integrals, such as averaging on combinatorial spaces or non-Gaussian integrals. Stochastic approximation is a class of methods introduced in 1951 by Herbert Robbins and Sutton Monro [Robbins and Monro, 1951] to solve intractable equations by using a sequence of approximate and random evaluations. Stochastic gradient descent [Bottou, 1998] is a special type of stochastic approximation method that is widely used in large scale learning tasks thanks to its scalability and good generalization properties [Bottou and Bousquet, 2011].

We are interested in using SGD to minimize functions of the form:

$$\gamma(w) \quad := \quad \mathbb{E}_{x\sim P}\left[f(x;w)\right] = \int_{\mathcal{X}} f(x;w)dP(x) \tag{A.1}$$

where P is a known fixed distribution and f is a function that maps $\mathcal{X} \times \mathcal{W}$ into $\mathbb{R}$, i.e. a family of functions on the metric space $\mathcal{X}$ and parametrized by $w \in \mathcal{W}$. SGD is a stochastic approximation method that consists in using approximate gradients computed on subspaces of $\mathcal{X}$ that are equal on average to the true gradient $\nabla_w \gamma(w)$ [Bottou, 1998]. In many applications, including supervised learning techniques, the function f is a log-likelihood and P is an empirical distribution with density $\frac{1}{n}\sum_{i=1}^{n}\delta(x, x_i)$ where $\{x_1, \cdots, x_n\}$ is a set of i.i.d. data sampled from an unknown distribution.

At a given step t, SGD can be viewed as a two-step procedure: (i) sampling $x_t \in \mathcal{X}$ according to the distribution P; (ii) doing an approximate gradient step with step-size ρ_t:

$$w_{t+1} = w_t - \rho_t \nabla_w f(x_t; w_t)\,. \tag{A.2}$$

The convergence properties of SGD are directly linked to the variance of the gradient estimate [Bach and Moulines, 2011]. Consequently, some improvements to this basic algorithm focus on the use of (i) parameter averaging [Polyak and Juditsky, 1992] to reduce the variance of the final estimator, (ii) the sampling of mini-batches [Friedlander and Schmidt, 2012] when multiple points are sampled at the same time to reduce the variance of the gradient, and (iii) the use of adaptive step sizes to have per-dimension learning rates, e.g., AdaGrad [Duchi et al., 2011].

We propose another general technique, which can be used in conjunction with the aforementioned ones, which is to *reduce the gradient variance by learning how to sample training points*. Rather than learning the fixed optimal sampling distribution and then optimizing the gradient, we propose to *dynamically learn an optimal sampling distribution at the same time as the original SGD algorithm*. Our formulation uses a stochastic process

that focuses on the minimization of the gradient variance, which amounts to doing an additional SGD step (to minimize gradient variance) along each SGD step (to minimize the learning objective). There is a constant extra cost to pay at each iteration, but it is the same for each iteration, and when simulations are expensive or the data access is slow, this extra computational cost is compensated for by the increase in convergence speed, as quantified in our experiments.

A.2 Related Work

The idea of speeding up learning by modifying the importance sampling distribution in SGD has been recently analyzed by [Hazan et al., 2011] who showed that a particular choice of the sampling distribution could lead to sub-linear performance guarantees for support vector machines. We can see our approach as a generalization of this idea to other models, by including the learning of the sampling distribution as part of the optimization. The work of [Mineiro and Karampatziakis, 2013] shows that using a simple model to choose which data to resample from is a useful thing to do, but they do not learn the sampling model while optimizing. The two approaches mentioned above can be viewed as the extreme case of adaptive sampling, where there is one step to learn the sampling distribution, and then a second step to learn the model using this sampling distribution. Training language models has been shown to be faster with adaptive importance sampling [Senecal and Bengio, 2003; Bengio and Senecal, 2008], but the authors did not directly minimize the variance of the estimator.

Regarding variance-reduction techniques, in addition to the aforementioned ones (Polyak-Ruppert Averaging [Polyak and Juditsky, 1992], batching [Friedlander and Schmidt, 2012], and adaptive learning rates like AdaGrad [Duchi et al., 2011]), an additional technique is to use *control variates* (see for instance [Ross, 1997]). It has been recently used by Paisley et al. [2012] to estimate non-conjugate potentials in a variational stochastic gradient algorithm. The techniques described here can also be straightforwardly extended to the optimization of a control variate. In the neural net community, adapting the order in which the training samples are used is called *curriculum learning* [Bengio et al., 2009], and our approach can be seen under this framework, although our algorithm is more general as it can speed-up learning for arbitrary integrals, not only sums of losses over the training data.

A.3 Adaptive Importance Sampling

We first show in this section that SGD is a powerful tool for optimizing the sampling distribution of Monte Carlo estimators. This will motivate our Adaptive Weighted SGD algorithm in which the sampling distribution is not kept constant, but learned during the optimization process.

We consider a family $\{Q_\tau\}$ of sampling distributions on $\mathcal{X}$, such that P is absolutely continuous with respect to Q_τ for any τ in the parametric set $\mathcal{T}$. By Radon-Nikodym theorem, the density $q(\cdot;\tau) = \frac{dQ_\tau}{dP}$ exists since P and Q_τ are probability measures, hence σ-finite. Importance sampling is a common method to estimate the integral in Equation A.1. It corresponds to a Monte Carlo estimator of the form (we omit the dependency on w for clarity):

$$\hat{\gamma} \quad = \quad \frac{1}{T}\sum_{t=1}^{T}\frac{f(x)}{q(x_t;\tau)}, \quad x \sim Q_\tau \tag{A.3}$$

where we refer to Q_τ as the *importance distribution*. It is an *unbiased* estimator of γ, i.e. the expectation of $\hat{\gamma}$ is exactly the desired quantity γ.

To compare estimators, we can use a variance criterion. The variance of this estimator depends on τ:

$$\sigma^2(\tau) \quad = \quad \mathrm{Var}_\tau\left[\hat{\gamma}\right] = \frac{1}{T}\mathbb{E}_\tau\left[\left(\frac{f(x)}{q(x;\tau)}\right)^2\right] - \frac{\gamma^2}{T} \tag{A.4}$$

where $\mathbb{E}_\tau\left[.\right]$ and $\mathrm{Var}_\tau\left[.\right]$ denote the expectation and variance with respect to distribution Q_τ.

To find the best possible sampling distribution in the sampling family $\{Q_\tau\}$, one can minimize the variance $\sigma^2(\tau)$ with respect to τ. The optimal parameter $\tau^* \in \mathcal{T}$ is such that $q(.,\tau^*) \propto |f|$. In such a case, the variance $\sigma^2(\tau^*)$ of the estimator is null: one can estimate the integral with a single sample. In general, however, the parametric family does not contain a normalized version of $|f|$. In addition, the minimization of the variance $\sigma^2(\tau)$ has often no closed-form solution. This motivates the use of approximate variance-reduction methods.

Algorithm 2 Minimal Variance Importance Sampling

Require: Initial sampling parameter vector $\tau_0 \in \mathcal{T}$
Require: Learning rates $\{\eta_t\}_{t\geq 0}$
  for $t = 0, 1, 2, \cdots, T-1$ **do**
    $x_t \sim Q_{\tau_t}$
    $\tau_{t+1} \leftarrow \tau_t + \eta_t \left(\frac{f(x_t)}{q(x_t;\tau_t)}\right)^2 \nabla_\tau \log q(x_t;\tau_t)$
  end for
  Output $\hat{\gamma} \leftarrow \frac{1}{T}\sum_t \frac{f(x_t)}{q(x_t;\tau_t)}$

A possible approach is to minimize $\sigma^2(\tau)$ with respect to the importance parameter τ. The gradient is:

$$\begin{aligned} \nabla_\tau \sigma^2(\tau) &= \nabla_\tau \mathbb{E}_\tau \left[\left(\frac{f(x)}{q(x;\tau)}\right)^2\right] \\ &= -2\mathbb{E}_\tau \left[\frac{f(x)^2 \nabla_\tau q(x;\tau)}{q(x;\tau)^3}\right] \\ &= -2\mathbb{E}_\tau \left[\left(\frac{f(x)}{q(x;\tau)}\right)^2 \nabla_\tau \log q(x;\tau)\right]. \end{aligned} \tag{A.5}$$

This quantity has no closed form solution in general, but we can use a SGD algorithm with a gradient step equal on average to this quantity. To obtain an estimator g of the gradient with expectation given by Equation A.5, it is enough to sample a point x_t according to Q_τ and then set $g := -(f(x_t)/q(x_t;\tau))^2 \nabla_\tau \log q(x_t;\tau)$. This is then repeated until convergence. The full iterative procedure is summarized in Algorithm 2.

In the experiments below, we show that learning the importance weight of an importance sampling estimator using SGD can lead to a significant speed-up in several machine learning applications, including the estimation of empirical loss functions and the evaluation of a policy in a reinforcement learning scenario. In the following, we show that this idea can also be used in a sequential setting (the function f can change over time), and when f has multivariate outputs, so that we can control the variance of the gradient of a standard SGD algorithm and, ultimately, speedup the convergence.

A.4 Biased Sampling in Stochastic Optimization

In this section, we first analyze a weighted version of the SGD algorithm where points are sampled non-uniformly, as in importance sampling, and then derive an adaptive version of this algorithm, where the sampling distribution evolves with the iterations.

A.4.1 Weighted Stochastic Gradient Descent

As introduced previously, our goal is to minimize the expectation of a parametric function f (cf. Equation A.1). As in importance sampling, we do not need to sample according to the base distribution P at each iteration of SGD. Instead, we can use any distribution Q_τ defined on $\mathcal{X}$ such that P is absolutely continuous with respect to Q_tau, if each gradient step is properly re-weighted by the density $q(\cdot;\tau) = dQ_\tau/dP$. Each iteration t of the algorithm consists in two steps: (i) sample $x_t \in \mathcal{X}$ according to distribution Q_τ; (ii) do an approximate gradient step:

$$w_{t+1} = w_t - \rho_t \frac{\nabla_w f(x_t; w_t)}{q(x_t, \tau_t)}\,. \tag{A.6}$$

Depending on the importance distribution Q_τ, this algorithm can have different convergence properties from the original SGD algorithm. As mentioned previously, the best sampling distribution would be the one that gives a small variance to the weighted gradient in Equation A.6. The main issue is that it depends on the parameters w_t, which are different at each iteration.

Our main observation is that we can *minimize the variance of the gradient using the previous iterates*, under the assumption that this variance does not change too quickly when w_t is updated. We argue that this is reasonable in practice as learning rate policies for ρ_t usually assume a small constant learning rate, or a decreasing schedule [Bottou, 1998]. In the next section, we build on that observation to build a new algorithm that learns the best sampling distribution Q in an online fashion.

A.4.2 Adaptive Weighted Stochastic Gradient Descent

As in Appendix A.3, we consider a family $\{Q_\tau\}$ of sampling distributions parametrized by τ in the parametric set $\mathcal{T}$. Using the sampling distribution Q_τ with probability density function $q(x;\tau) = \frac{dQ_\tau(x)}{dP(x)}$, we can now evaluate the efficiency of the sampling distributions Q_τ based on the covariance $\Sigma(w,\tau)$:

$$\Sigma(w,\tau) := \mathrm{Var}_\tau\left[\nabla_w f(x;w)/q(x,\tau)\right] \tag{A.7}$$

$$= \mathbb{E}_\tau\left[\frac{\nabla_w f(x;w)\nabla_w^\top f(x;w)}{q(x;\tau)^2}\right] - \nabla_w\gamma(w)\nabla_w^\top\gamma(w)\,. \tag{A.8}$$

Algorithm 3 Adaptive Weighted SGD (AW-SGD)

Require: Initial target and sampling parameter vectors $w_0 \in \mathcal{W}$ and $\tau_0 \in \mathcal{T}$
Require: Learning rates $\{\rho_t\}_{t\geq 0}$ and $\{\eta_t\}_{t\geq 0}$
  for $t = 0, 1, \cdots, T-1$ **do**
    $x_t \sim Q_{\tau_t}$
    $d_t \leftarrow \frac{\nabla_w f(x_t; w_t)}{q(x_t;\tau_t)}$
    $w_{t+1} \leftarrow w_t - \rho_t d_t$
    $\tau_{t+1} \leftarrow \tau_t + \eta_t \|d_t\|^2 \nabla_\tau \log q(x_t; \tau_t)$
  end for

For a given function $f(\cdot; w)$ we would like to find the parameter $\tau^*(w)$ of the sampling distribution that minimizes the trace of the covariance $\Sigma(w; \tau)$, i.e.:

$$\tau^*(w) \in \arg\min_\tau \mathbb{E}_\tau \left[\left\| \frac{\nabla_w f(x; w)}{q(x; \tau)} \right\|^2 \right] . \tag{A.9}$$

Consequently, a simple SGD algorithm with gradient steps having small variance consists in the following two steps at each iteration t:

1. Perform a weighted stochastic gradient step using distribution Q_{τ_t} to obtain w_{t+1};
2. Compute $\tau_t = \tau^*(w_t)$ by solving Equation A.9, i.e. find the parameter τ_t minimizing the variance of the gradient at point w_t. This can be done approximately by applying M steps of stochastic gradient descent.

The inner-loop SGD algorithm involved in the second step can be based on the current sample, and the stochastic gradient direction is

$$\begin{aligned} \nabla_\tau tr(\Sigma(w_t, \tau)) &= \nabla_\tau \mathbb{E}_\tau \left[\left\| \frac{\nabla_{w_t} f(x; w_t)}{q(x; \tau)} \right\|^2 \right] \\ &= -2\mathbb{E}_\tau \left[\left\| \frac{\nabla_{w_t} f(x; w_t)}{q(x; \tau)} \right\|^2 \nabla_\tau \log q(x; \tau) \right] . \end{aligned} \tag{A.10}$$

In our experiments, we observed that it is enough to do a single step of the inner loop, i.e. $M = 1$. We call this simplified algorithm the *Adaptive Weighted SGD Algorithm* and its pseudo-code is given in Algorithm 3. We see that AW-SGD is a slight modification of the standard SGD—or any variant of it, such as AdaGrad [Duchi et al., 2011], AdaDelta [Zeiler, 2012] or RMSProp [Tieleman and Hinton, 2012]—but where the sampling distribution evolves during the algorithm, thanks to the update of τ_t. This algorithm is useful when the approximate gradient has a variance that can be significantly reduced by choosing better samples.

The benefits of this algorithm have been illustrated in three different applications: image classification, matrix factorization and reinforcement learning [Bouchard et al., 2015b]. We here report only the work that is part of this thesis: the matrix factorization applications.

A.5 Application to Matrix Factorization

We applied AW-SGD to learn how to sample the rows and columns in a SGD-based low-rank matrix decomposition algorithm. Let $Y \in \mathbb{R}^{n\times m}$ be a matrix that we want to approximate with a rank-K decomposition $UV^\top$, where $U \in \mathbb{R}^{n\times K}$ and $V \in \mathbb{R}^{m\times K}$. We consider a differentiable loss function $\ell(z; y)$ where $z \in \mathbb{R}$ and y is observed. With the squared loss, each entry of Y is a real scalar and $\ell(z, y) = (z - y)^2$. The full loss function is

$$\gamma(U, V) \quad = \quad \sum_{i=1}^{n}\sum_{j=1}^{m} \ell(u_i v_j^\top, y_{ij})\,. \tag{A.11}$$

We consider the sampling distributions $\{Q_\tau\}$ over the set $\mathcal{X} := \{1, \cdots, n\} \times \{1, \cdots, m\}$, where we independently sample a row i and a column j according to the discrete distributions $\varsigma(\tau')$ and $\varsigma(\tau'')$ respectively, with $\tau' \in \mathbb{R}^n$, $\tau'' \in \mathbb{R}^m$, $\tau = (\tau', \tau'') \in \mathbb{R}^{m+n}$, and $x = (i, j)$. We define:

$$\varsigma(z) = (e^{z_1}, e^{z_2}, \cdots, e^{z_p}) / \left(\sum_{i=1}^{p} e^{z_i}\right)\,, \tag{A.12}$$

$$q(x, \tau) = \varsigma(\tau')\,\varsigma(\tau'') \tag{A.13}$$

where $z \in \mathbb{R}^p$ and $\varsigma : \mathbb{R}^p \to \mathbb{R}^p$ is the softmax function. Using the squared loss, as in the experiments below, the update equations in AW-SGD (Algorithm 3) are:

$$f(x_t; u_t, v_t) = \ell(u_{i_t} v_{j_t}^\top, y_{i_t j_t}) = (u_{i_t} v_{j_t}^\top - y_{i_t j_t})^2\,, \tag{A.14}$$

$$\nabla_{u_{i_t}} f(x_t; u_t, v_t) = 2 v_{j_t} (u_{i_t} v_{j_t}^\top - y_{i_t j_t})\,, \tag{A.15}$$

$$\nabla_{v_{j_t}} f(x_t; u_t, v_t) = 2 u_{i_t} (u_{i_t} v_{j_t}^\top - y_{i_t j_t})\,, \tag{A.16}$$

$$\nabla_{\tau'} \log q(x_t; \tau_t) = e_i - \varsigma(\tau')\,, \tag{A.17}$$

$$\nabla_{\tau''} \log q(x_t; \tau_t) = e_j - \varsigma(\tau'') \tag{A.18}$$

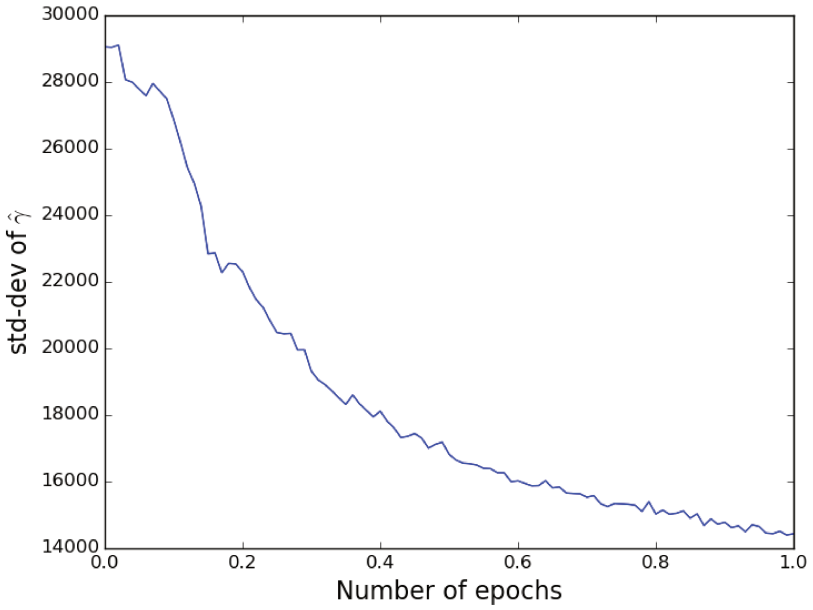


FIGURE A.1: Results of the minimal variance importance Sampling algorithm (Algorithm 1.). The curve shows the standard deviation of the estimator of the loss $\hat{\gamma}$ as a function of the number of matrix entries that have been observed.

where $e_i \in \mathbb{R}^n$ and $e_j \in \mathbb{R}^m$, vectors with 1 at index i and j respectively, and all other components are 0.

In the matrix factorization experiments, we used the mini-batch technique with batches of size 100, ρ_0 and η_0 were tuned to yield the minimum γ at convergence, separately with each algorithm. All results are averaged over 10 runs. τ' and τ'' were initialized with zeros to get an initial uniform sampling distribution over the rows and columns. The model learning rate decrease was set to $\rho_t = \rho_0/((nm/2) + t)$, $\eta_t = \eta_0$ was kept constant.

In Figure A.1, we generated a rank-K matrix $Y \in \mathbb{R}^{n \times m}$, for $n = m = 100$ and $K = 10$, by sampling U and V using independent centered Gaussian variables with unit variance. The $x-$axis is expressed in epochs, where one epoch corresponds to sampling nm values in the matrix. To illustrate the benefit of adaptive sampling, we create a high-variance block by multiplying by 100 the entries a randomly-drawn 20×20 square block in Y, to experimentally assess the benefit of a non-uniform sampling strategy. The results of the minimal variance important sampling scheme (Algorithm 2) are shown on the left. After having sampled $\frac{nm}{2}$ matrix entries, the standard deviation of the importance sampling estimator is divided by two. tFigure A.2 shows the loss decrease of SGD and AW-SGD on the same matrix for multiple learning rates. AW-SGD converges significantly faster than the best uniformly-sampled SGD, even after 1 epoch through the data. On average, AW-SGD requires half of the number of iterations to converge to the same value.

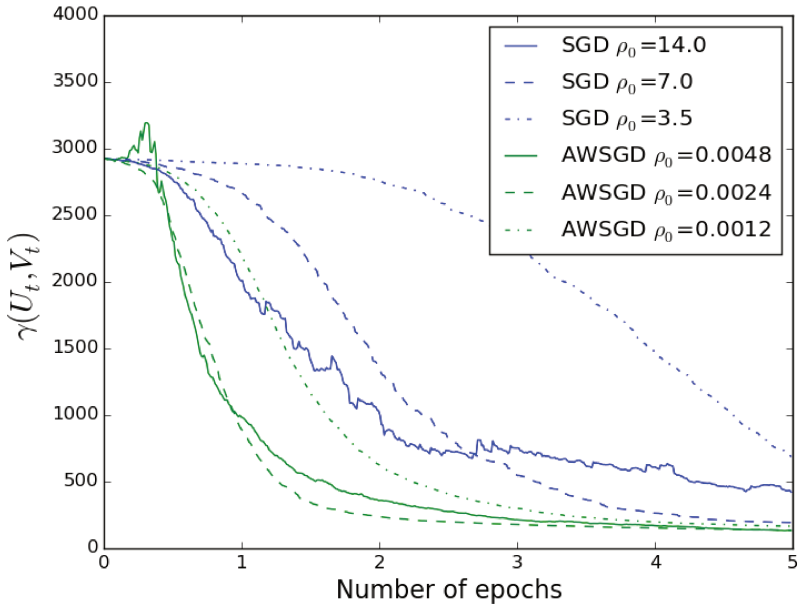


FIGURE A.2: Comparison of the convergence speed of the AW-SGD algorithm compared to the standard SGD algorithm (uniform sampling of rows and columns) on the matrix factorization experiment.

In-painting experiment We compared both algorithms on the MNIST dataset [LeCun et al., 1998], on which low-rank decomposition techniques have been successfully applied [Das Gupta and Xiao, 2011]. We factorized with $K = 50$ the training set for the zero digit, a 5923×784 matrix, where each line is a 28×28 image of a handwritten zero, and each column one pixel. Figure A.3 shows the loss decrease for both algorithms on the first iteration. AW-SGD requires significantly less samples to reach the same error. At convergence, AW-SGD showed an average 2.52× speedup in execution time compared to SGD, showing that its sampling choices compensate for its parametrization overhead.

Non-stationary data On Figure A.4 we progressively substituted images of handwritten zeros by images of handwritten ones. It shows, every 2000 samples (i.e. 0.0005 epoch), the heatmap of the sampling probability of each pixel, $\varsigma(\tau'')$, reorganized as 28×28 grids. Substitution from zeros to ones was made between 10000 and 20000 samples (on 2^{nd} line). One can distinctly recognize the zero digit first, that progressively fades out for the one digit. This transitions shows that AW-SGD learns to sample the digits that are likely to have a high impact on the loss. The algorithm adapts online to changes in the underlying distribution (transitions from one digit to another).

Combined with adaptive step size algorithms such as AdaGrad, we noticed that Adagrad did not improve the convergence speed of AW-SGD in our matrix factorization experiments. A possible explanation is that adaptive sampling favors some rows and columns,

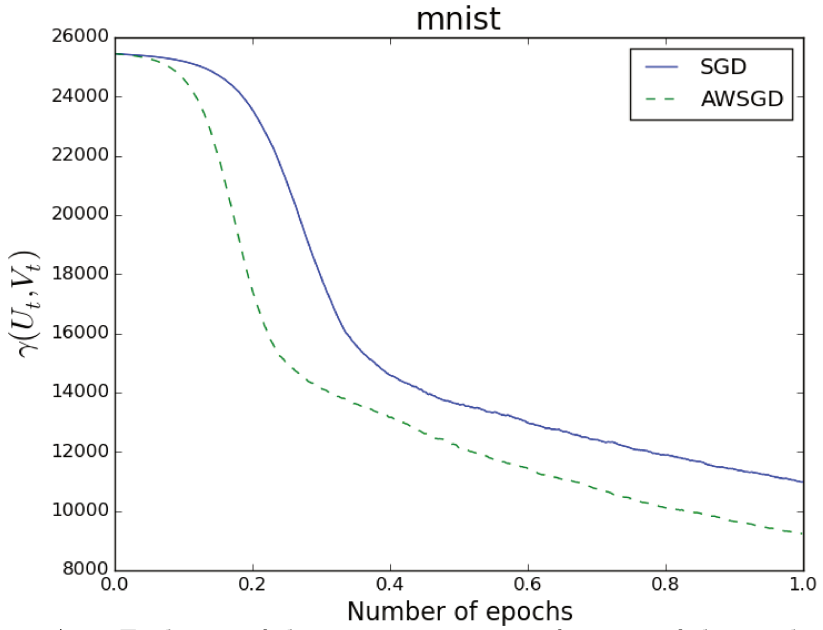


FIGURE A.3: Evolution of the training error as a function of the number of epochs for the uniformly-sampled SGD and AW-SGD for the matrix factorization application applied to MNIST data.

and AdaGrad compensates the non-uniform sampling, such that using AW-SGD and AdaGrad simultaneously converges only slightly faster than AdaGrad alone. It should behave similarly on other parametrizations of τ where τ indices are linked to parameters indices. In preliminary work, we observed that AdaGrad performances were not matching the best validated learning rates.

A.6 Adapting to Non-Uniform Architectures

In many large-scale infrastructures, such as computation servers shared by many people, data access or gradient computation are unknown in advance. These hardware systems are sometimes called Non-Uniform Memory Access [Nieplocha et al., 1996]. It can be the case with large-scale matrix factorization, when some parts of the matrix are stored locally, and others downloaded through the network. How can we inform the algorithm that it should sample more often data stored locally?

A simple modification of AW-SGD can enable the algorithm to adapt to non-uniform computation time. The key idea is to learn dynamically to minimize the expected loss decrease per unit time. To make AW-SGD take into account this access time, we simply weighted the update of τ in Algorithm 3 by dividing it by the simulated access time Δ_x to the sample x. This is summarized in Algorithm 4.

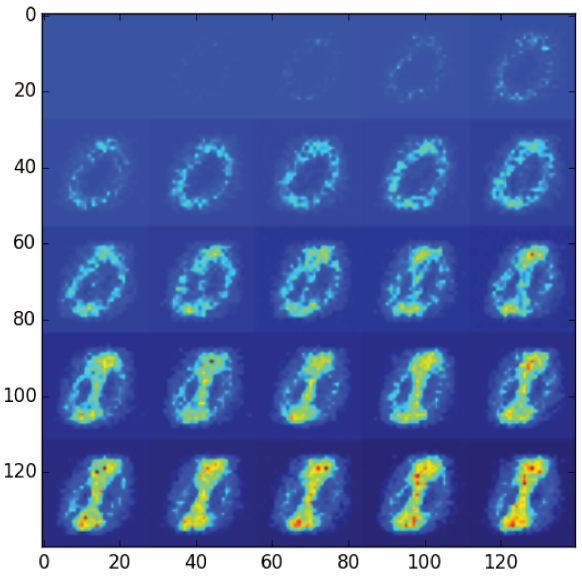


FIGURE A.4: Illustration of the evolution of the sampling distribution when data are not i.i.d. Each heatmap contains the sampling probability of each pixel in the MNIST matrix factorization experiment.

Algorithm 4 Time-Aware AW-SGD

Require: Initial values for $w_0 \in \mathcal{W}$ and $\tau_0 \in \mathcal{T}$
Require: Learning rates $\{\rho_t\}_{t\geq 0}$ and $\{\eta_t\}_{t\geq 0}$
  for $t = 0, 1, \cdots, T-1$ **do**
    $s_t \leftarrow$ getCurrentTime()
    $x_t \sim Q_{\tau_t}$
    $d_t \leftarrow \frac{\nabla_w f(x_t; w_t)}{q(x_t; \tau_t)}$
    $w_{t+1} \leftarrow w_t - \rho_t d_t$
    $e_t \leftarrow$ getCurrentTime()
    $\tau_{t+1} \leftarrow \tau_t + \frac{\eta_t}{e_t - s_t} \|d_t\|^2 \nabla_\tau \log q(x_t; \tau_t)$
  end for

As an experiment, we show in the matrix factorization case that the time-aware AW-SGD is able to learn and exploit the underlying hardware when the data does not fit entirely in memory, and one part of the data has an extra access cost. To do so, we generate a 1000×1000 rank-10 matrix, but without a high variance block, so that variance is uniform across rows and columns. For the first half of the rows of the matrix, i.e. $i < \frac{n}{2}$, we consider the data as being in main memory, and simulate an access cost of 100ns for each sampling in those rows, inspired by Jeff Dean's typical access numbers [Dean, 2007]. For the other half of the rows, $i \geq \frac{n}{2}$, we multiply that access cost by a factor c, we will call the slow-block access factor. The simulated access time to the sample (i, j), $\Delta_{(i,j)}$ is thus given by: $\Delta_{(i,j)} = 10^{-7} \times c$ if $i \geq \frac{n}{2}$, and $\Delta_{(i,j)} = 10^{-7}$ if $i \geq \frac{n}{2}$. We allowed c to range from 2 to 2^{20}.

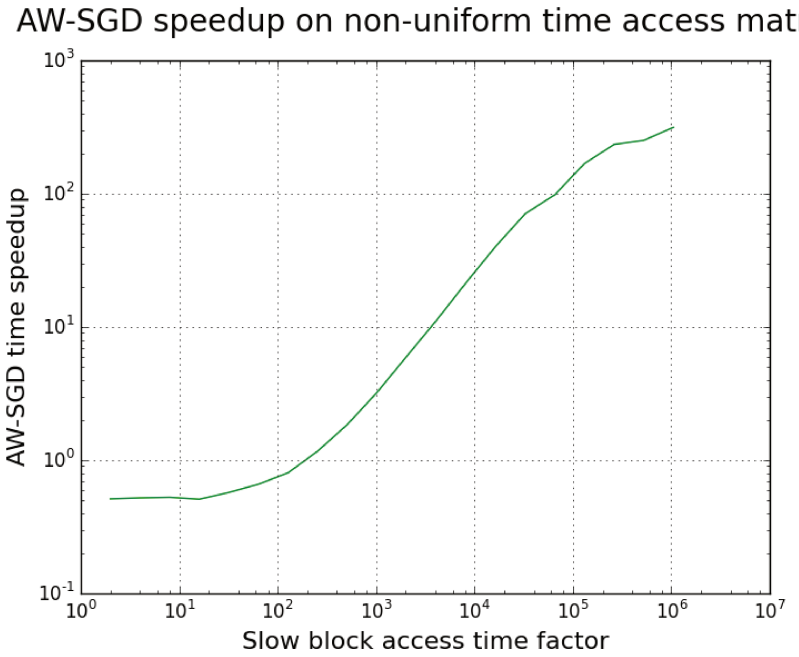


FIGURE A.5: Results of the matrix factorization application on the simulated matrix with two blocks with different access times. The AW-SGD speedup over SGD is plotted against the multiplying factor c.

The speedup achieved by the time-aware AW-SGD over SGD is plotted against the evolution of the factor c in Figure A.5. For each algorithm, we summed the real execution time and the simulated access times in order to take into account the time-aware AW-SGD sampling overhead. The speedup is computed after one epoch, by dividing SGD total time by AW-SGD total time. AW-SGD is faster for all $c \geq 200$, where $c = 200$ corresponds to a random read on a solid-state drive (SSD). For smaller c AW-SGD is slower, since the data is homogeneous, and time access difference is not yet big enough to compensate its overhead. At $c = 5000$, corresponding to a read from another computer's memory on local network, the speedup reaches $10\times$. At $c = 50000$, a hard drive seek, AW-SGD is $100\times$ faster. This shows that the time-aware AW-SGD overhead is compensated for by its sampling choices.

However, one could think that the AW-SGD algorithm would simply completely avoid sampling in the slow part of the matrix, and thus not learning to reconstruct it at all. Figure A.6 shows that this is not the case as the loss of both algorithms decreases almost at the same rate per epoch. The time speed-up per epoch coming from the non-uniform sampling of AW-SGD hence does not hurt its convergence, compared to standard SGD.

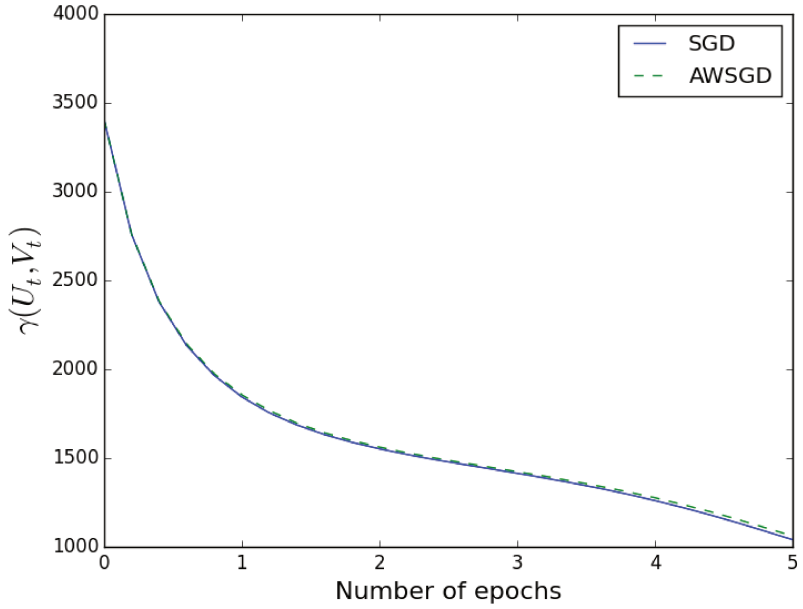


FIGURE A.6: Evolution of the training error as a function of the number of epochs on the simulated matrix with different access costs, with $c = 5000$, for the uniformly-sampled SGD and AW-SGD using best ρ_0 for each algorithm.

A.7 Conclusion

In this work, we argue that SGD and importance sampling can strongly benefit from each other. SGD algorithms can be used to learn the minimal variance sampling distribution, while importance sampling techniques can be used to improve the gradient estimation of SGD algorithm. We have introduced a simple yet efficient Adaptive Weighted SGD algorithm that can optimize a function while optimizing the way it samples the examples. We showed that this framework can be used in a large variety of problems, and experimented with it on matrix factorization showing a significant speed-up by optimizing the way the samples are generated.

There are many more applications in which these variance reduction techniques have a strong potential. For example, in variational inference, the objective function is an integral and SGD algorithms are often used to increase convergence [Hoffman et al., 2013; Paisley et al., 2012]. Computing these integrals stochastically could be made more efficient by sampling non-uniformly in the integration space. Also, the estimation of intractable log-partition function, as required by Boltzmann machines, are potential candidate models in which importance sampling has already been proposed, but without variance reduction technique [Salakhutdinov and Murray, 2008].

This work also shows that we can learn about the algorithm while optimizing, as shown by the time-aware AW-SGD. This idea can be extended to design new types of meta-algorithms that *learn to optimize* or *learn to coach other algorithms*.

Appendix B

Results with Reflexivity and Irreflexivity

In this appendix we report results of the individual learning of combinations of relation properties including reflexivity and irreflexivity. Those results are included for completeness as they are similar to the cases that are neither reflexive nor irreflexive, reported in Section 5.2.2.1. Figure B.1 shows results for the 5 combinations with reflexivity, and Figure B.2 for the 3 combinations with irreflexivity. The irreflexive transitive case, and the irreflexive symmetric transitive case are not reported as they are not consistent, as explained in Section 5.2.1. The single noticeable difference is in the symmetric irreflexive case, where all models perform slightly worse compared to the symmetric and symmetric reflexive cases, especially TransE.

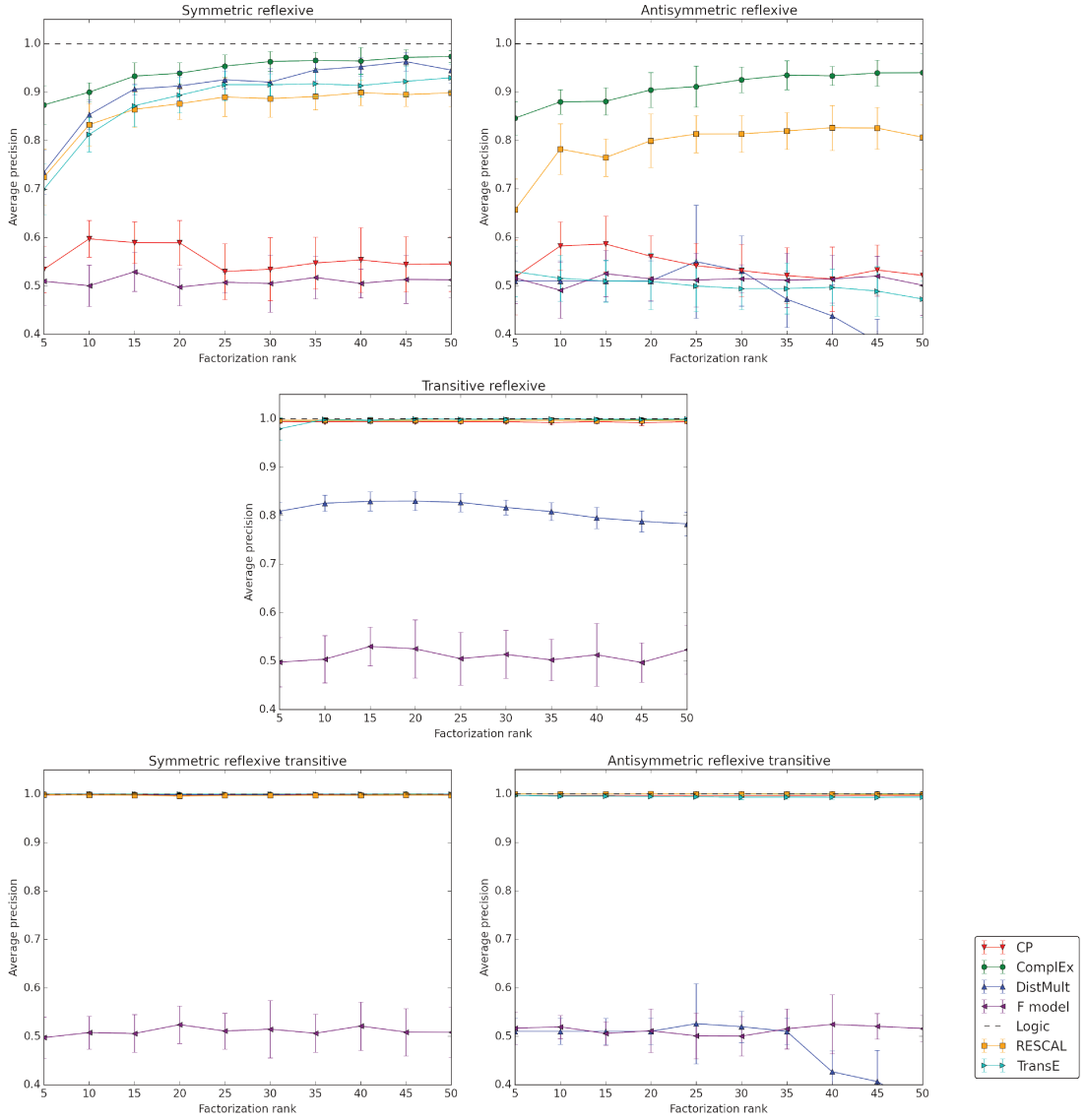


FIGURE B.1: Generated reflexive relations with 50 entities, combined with symmetry (top-left), antisymmetry (top-right), transitivity (middle), symmetry and transitivity (bottom-left) and antisymmetry and transitivity (bottom-right). Average precision for each rank ranging from 5 to 50 for each model.

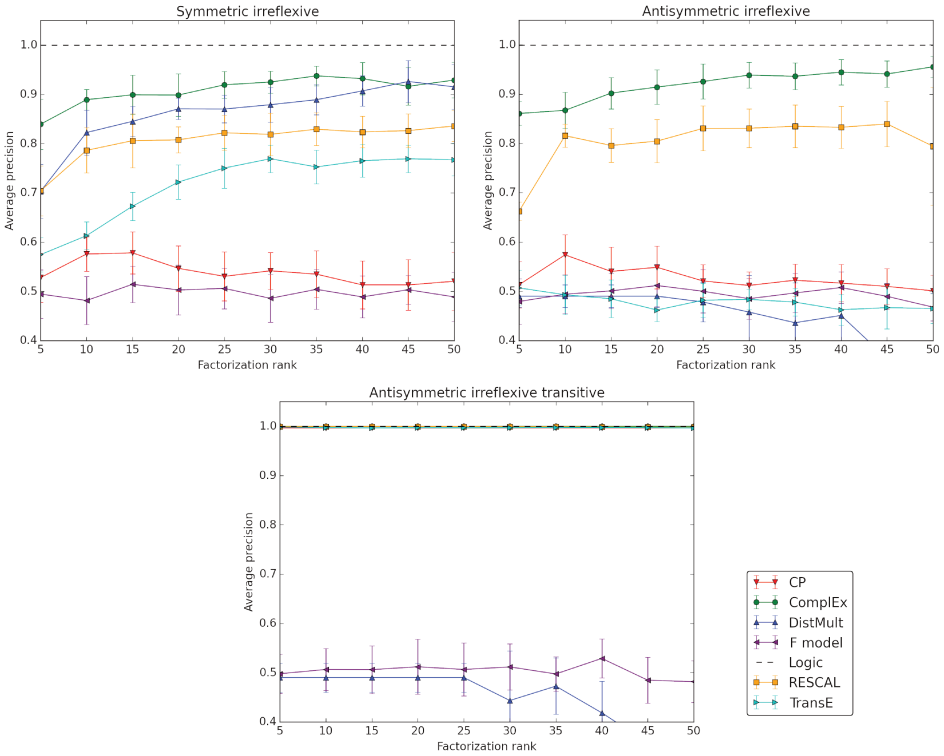


FIGURE B.2: Generated irreflexive relations with 50 entities, combined with symmetry (top-left), antisymmetry (top-right) and antisymmetry and transitivity (bottom). Average precision for each rank ranging from 5 to 50 for each model.

Bibliography

Aaronson, S. (2011). Why philosophers should care about computational complexity. *arXiv preprint arXiv:1108.1791*.

Abernethy, J., Bach, F., Evgeniou, T., and Vert, J.-P. (2009). A new approach to collaborative filtering: Operator estimation with spectral regularization. *Journal of Machine Learning Research*, 10:803–826.

Acar, E., Dunlavy, D. M., Kolda, T. G., and Mørup, M. (2010). Scalable tensor factorizations with missing data. In *Proceedings of the 2010 SIAM international conference on data mining*, pages 701–712.

Alon, N., Moran, S., and Yehudayoff, A. (2016). Sign rank versus vc dimension. In *Conference on Learning Theory*, pages 47–80.

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., and Ives, Z. (2007). DBpedia: A nucleus for a web of open data. In *International Semantic Web Conference, Busan, Korea*, pages 11–15. Springer.

Bach, F. and Moulines, E. (2011). Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems*, pages 451–459.

Belleau, F., Nolin, M.-A., Tourigny, N., Rigault, P., and Morissette, J. (2008). Bio2rdf: towards a mashup to build bioinformatics knowledge systems. *Journal of biomedical informatics*, 41(5):706–716.

Belouchrani, A., Abed-Meraim, K., Cardoso, J.-F., and Moulines, E. (1997). A blind source separation technique using second-order statistics. *IEEE Transactions on Signal Processing*, 45(2):434–444.

Beltrami, E. (1873). Sulle funzioni bilineari. *Giornale di Matematiche ad Uso degli Studenti Delle Universita*, 11(2):98–106.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *International Conference on Machine Learning*, pages 41–48.

Bengio, Y. and Senecal, J.-S. (2008). Adaptive importance sampling to accelerate training of a neural probabilistic language model. *Neural Networks, IEEE Transactions on*, 19(4):713–722.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010). Theano: a CPU and GPU math expression compiler. In *Python for Scientific Computing Conference (SciPy)*.

Bhattacharya, I. and Getoor, L. (2007). Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data*, 1(1):5.

Bhojanapalli, S., Neyshabur, B., and Srebro, N. (2016). Global optimality of local search for low rank matrix recovery. *arXiv preprint arXiv:1605.07221*.

Bilgic, M., Licamele, L., Getoor, L., and Shneiderman, B. (2006). D-dupe: An interactive tool for entity resolution in social networks. In *IEEE Symposium On Visual Analytics Science And Technology*, pages 43–50.

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10).

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD 08 Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Bordes, A., Glorot, X., Weston, J., and Bengio, Y. (2014a). A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. (2013a). Irreflexive and hierarchical relations as translations. *arXiv preprint arXiv:1304.7158*.

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. (2013b). Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.

Bordes, A., Weston, J., Collobert, R., and Bengio, Y. (2011). Learning structured embeddings of knowledge bases. In *AAAI Conference on Artificial Intelligence*.

Bordes, A., Weston, J., and Usunier, N. (2014b). Open question answering with weakly supervised embedding models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 165–180.

Bottou, L. (1998). Online algorithms and stochastic approximations. In Saad, D., editor, *Online Learning and Neural Networks*. Cambridge University Press, Cambridge, UK.

Bottou, L. (2014). From machine learning to machine reasoning. *Machine learning*, 94(2):133–149.

Bottou, L. and Bousquet, O. (2011). The tradeoffs of large scale learning. In Sra, S., Nowozin, S., and Wright, S. J., editors, *Optimization for Machine Learning*, pages 351–368. MIT Press.

Bouchard, G., Singh, S., and Trouillon, T. (2015a). On approximate reasoning capabilities of low-rank vector spaces. *AAAI Spring Symposium on Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches*.

Bouchard, G., Trouillon, T., Perez, J., and Gaidon, A. (2015b). Online learning to sample. *arXiv preprint arXiv:1506.09016*.

Bouchard, G., Yin, D., and Guo, S. (2013). Convex collective matrix factorization. In *International Conference on Artificial Intelligence and Statistics*, volume 13, pages 144–152.

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015a). A large annotated corpus for learning natural language inference. In *Empirical Methods in Natural Language Processing*, pages 632–642.

Bowman, S. R., Potts, C., and Manning, C. D. (2015b). Recursive neural networks can learn logical semantics. In *ACL Workshop on Continuous Vector Space Models and their Compositionality*.

Brocheler, M., Mihalkova, L., and Getoor, L. (2010). Probabilistic similarity logic. In *Conference on Uncertainty in Artificial Intelligence*.

Cai, J.-F., Candès, E. J., and Shen, Z. (2010). A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982.

Candes, E. and Recht, B. (2012). Exact matrix completion via convex optimization. *Communications of the ACM*, 55(6):111–119.

Candès, E. J. and Tao, T. (2010). The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080.

Cardano, G. (1545). *Artis Magnæ, Sive de Regulis Algebraicis Liber Unus*.

Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr, E. R., and Mitchell, T. M. (2010). Toward an architecture for never-ending language learning. In *AAAI Conference on Artificial Intelligence*.

Caroll, J. D. and Chang, J. J. (1970). Analysis of individual differences in multidimensional scaling via n-way generalization of Eckart–Young decomposition. *Psychometrika*, 35:283–319.

Cauchy, A.-L. (1829). Sur l'équation à l'aide de laquelle on détermine les inégalités séculaires des mouvements des planètes. *Œuvres complètes, II^e série*, 9:174–195.

Chakrabarti, S., Dom, B., and Indyk, P. (1998). Enhanced hypertext categorization using hyperlinks. In *ACM Special Interest Group on Management of Data Record*, volume 27, pages 307–318.

Chang, K. W., Yih, W. T., Yang, B., and Meek, C. (2014). Typed tensor decomposition of knowledge bases for relation extraction. In *EMNLP*.

Chino, N. (2002). Complex space models for the analysis of asymmetry. In *Measurement and Multivariate Analysis*, pages 107–114. Springer.

Christen, P. (2012). *Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection*. Springer Science & Business Media.

Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387.

Comon, P., Luciani, X., and De Almeida, A. L. (2009). Tensor decompositions, alternating least squares and other tales. *Journal of chemometrics*, 23(7-8):393–405.

Cvetković, D. M., Rowlinson, P., and Simic, S. (1997). *Eigenspaces of graphs*. Number 66. Cambridge University Press.

Cyganiak, R., Wood, D., and Lanthaler, M. (2014). Rdf 1.1 concepts and abstract syntax. *W3C Recommendation*.

Danihelka, I., Wayne, G., Uria, B., Kalchbrenner, N., and Graves, A. (2016). Associative long short-term memory. *arXiv preprint arXiv:1602.03032*.

Das, R., Neelakantan, A., Belanger, D., and McCallum, A. (2016). Chains of reasoning over entities, relations, and text using recurrent neural networks. *arXiv preprint arXiv:1607.01426*.

Das Gupta, M. and Xiao, J. (2011). Non-negative matrix factorization as a feature selection tool for maximum margin classifiers. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2841–2848. IEEE.

De Lathauwer, L., De Moor, B., and Vandewalle, J. (2000). A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278.

De Lathauwer, L., De Moor, B., and Vandewalle, J. (2001). Independent component analysis and (simultaneous) third-order tensor diagonalization. *IEEE Transactions on Signal Processing*, 49(10):2262–2271.

De Raedt, L. (2008). *Logical and relational learning*. Springer Science & Business Media.

Dean, J. (2007). Software engineering advice from building large-scale distributed systems. `http://research.google.com/people/jeff/stanford-295-talk.pdf#13`.

Demeester, T., Rocktäschel, T., and Riedel, S. (2016). Lifted rule injection for relation embeddings. In *Empirical Methods in Natural Language Processing*, pages 1389–1399.

Denham, W. W. (1973). *The detection of patterns in Alyawara nonverbal behavior*. PhD thesis, University of Washington, Seattle.

Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., and Zhang, W. (2014). Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 601–610.

Dredze, M., McNamee, P., Rao, D., Gerber, A., and Finin, T. (2010). Entity disambiguation for knowledge base population. In *Annual Meeting of the Association for Computational Linguistics*, pages 277–285.

Drumond, L., Rendle, S., and Schmidt-Thieme, L. (2012). Predicting RDF triples in incomplete knowledge bases with tensor factorization. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing - SAC '12*, page 326, New York, New York, USA. ACM Press.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Dzeroski, S. and Lavrac, N. (1994). *Inductive logic programming: t echniques and applications*. Ellis Horwood, New York.

Ermis, B. and Bouchard, G. (2014). Iterative splits of quadratic bounds for scalable binary tensor factorization. In *Conference on Uncertainty in Artificial Intelligence*, pages 192–199.

Escoufier, Y. and Grorud, A. (1980). Analyse factorielle des matrices carrees non symetriques. *Data analysis and informatics*, 1:263–276.

Faruqui, M., Dodge, J., Jauhar, S. K., Dyer, C., Hovy, E. H., and Smith, N. A. (2015). Retrofitting word vectors to semantic lexicons. In *North American Chapter of the*

Association of Computational Linguistics: Human Language Technologies, pages 1606–1615.

Fellbaum, C. (1998). *WordNet*. Wiley Online Library.

Fortunato, S. (2010). Community detection in graphs. *Physics reports*, 486(3):75–174.

Franz, T., Schultz, A., Sizov, S., and Staab, S. (2009). Triplerank: Ranking semantic web data by tensor decomposition. In *International Semantic Web Conference*, pages 213–228.

Frasconi, P., Costa, F., De Raedt, L., and De Grave, K. (2014). klog: A language for logical and relational learning with kernels. *Artificial Intelligence*, 217:117–143.

Friedlander, M. and Schmidt, M. (2012). Hybrid deterministic-stochastic methods for data fitting. *SIAM Journal on Scientific Computing*.

Friedman, N., Getoor, L., Koller, D., and Pfeffer, A. (1999). Learning probabilistic relational models. In *International Joint Conference on Artificial Intelligence*, volume 99, pages 1300–1309.

Galárraga, L., Teflioudi, C., Hose, K., and Suchanek, F. M. (2015). Fast rule mining in ontological knowledge bases with amie+. *The VLDB Journal*, 24(6):707–730.

García-Durán, A., Bordes, A., and Usunier, N. (2014). Effective blending of two and three-way interactions for modeling multi-relational data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 434–449.

Garcia-Duran, A., Bordes, A., Usunier, N., and Grandvalet, Y. (2016). Combining two and three-way embedding models for link prediction in knowledge bases. *Journal of Artificial Intelligence Research*, 55:715–742.

Ge, R., Lee, J. D., and Ma, T. (2016). Matrix completion has no spurious local minimum. *arXiv preprint arXiv:1605.07272*.

Gemulla, R., Nijkamp, E., Haas, P. J., and Sismanis, Y. (2011). Large-scale matrix factorization with distributed stochastic gradient descent. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 69–77.

Getoor, L. and Taskar, B. (2007). *Introduction to Statistical Relational Learning*. The MIT Press.

Giannandrea, J. (2011). Cikm 2011 industry event: John giannandrea on freebase a rosetta stone for entities. `http://thenoisychannel.com/2011/11/15/cikm-2011-industry-event-john-giannandrea-on-freebase-a-rosetta-stone-for-entities`.

Google Blog (2012). Introducing the knowledge graph: things, not strings. `https://googleblog.blogspot.fr/2012/05/introducing-knowledge-graph-things-not.html`.

Grefenstette, E. (2013). Towards a formal distributional semantics: Simulating logical calculi with tensors. In *Joint Conference on Lexical and Computational Semantics*.

Guu, K., Miller, J., and Liang, P. (2015). Traversing knowledge graphs in vector space. *Empirical Methods in Natural Language Processing*.

Haeffele, B. D. and Vidal, R. (2015). Global optimality in tensor factorization, deep learning, and beyond. *arXiv preprint arXiv:1506.07540*.

Halmos, P. R. (1998). *Naive set theory*. Springer Science & Business Media.

Hamaguchi, T., Oiwa, H., Shimbo, M., and Matsumoto, Y. (2017). Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach. *arXiv preprint arXiv:1706.05674*.

Harshman, R. A. (1970). Foundations of the PARAFAC procedure: models and conditions for an explanatory multimodal factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84.

Hayashi, K. and Shimbo, M. (2017). On the equivalence of holographic and complex embeddings for link prediction. *arXiv preprint arXiv:1702.05563*.

Hazan, E., Koren, T., and Srebro, N. (2011). Beating sgd: Learning svms in sublinear time. In *Advances in Neural Information Processing Systems*, pages 1233–1241.

He, S., Liu, K., Ji, G., and Zhao, J. (2015). Learning to represent knowledge graphs with gaussian embedding. In *ACM International on Conference on Information and Knowledge Management*, pages 623–632.

Heckerman, D., Meek, C., and Koller, D. (2007). Probabilistic entity-relationship models, prms, and plate models. *Introduction to statistical relational learning*, pages 201–238.

Hinton, G. E. (1986). Learning distributed representation of concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*.

Hitchcock, F. L. (1927). The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematical Physics*, 6(1):164–189.

Hoffart, J., Suchanek, F. M., Berberich, K., and Weikum, G. (2013). Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61.

Hoffman, M., Blei, D., Wang, C., and Paisley, J. (2013). Stochastic variational inference. *Journal of Machine Learning Research*, 14(1):1303–1347.

Horn, R. A. and Johnson, C. R. (2012). *Matrix analysis*. Cambridge University Press.

Hu, Q., Yamagishi, J., Richmond, K., Subramanian, K., and Stylianou, Y. (2016). Initial investigation of speech synthesis based on complex-valued neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5630–5634.

Huang, H. and Liu, C. (2009). Query evaluation on probabilistic rdf databases. In *International Conference on Web Information Systems Engineering*, pages 307–320. Springer.

Jenatton, R., Bordes, A., Roux, N. L., and Obozinski, G. (2012). A Latent Factor Model for Highly Multi-relational Data. In *Advances in Neural Information Processing Systems*, pages 3167–3175.

Kearns, M. and Valiant, L. (1994). Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM*, 41(1):67–95.

Kerrien, S., Aranda, B., Breuza, L., Bridge, A., Broackes-Carter, F., Chen, C., Duesbury, M., Dumousseau, M., Feuermann, M., Hinz, U., et al. (2011). The IntAct molecular interaction database in 2012. *Nucleic acids research*.

Kersting, K. and De Raedt, L. (2001). Towards combining inductive logic programming with bayesian networks. In *International Conference on Inductive Logic Programming*, pages 118–131.

Kok, S. and Domingos, P. (2007). Statistical predicate invention. In *International Conference on Machine Learning*, pages 433–440.

Kolda, T. G. and Bader, B. W. (2009). Tensor decompositions and applications. *SIAM Review*, 51(3):455–500.

Köpcke, H. and Rahm, E. (2010). Frameworks for entity matching: A comparison. *Data & Knowledge Engineering*, 69(2):197–210.

Koren, Y. (2008). Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 426–434.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

Krompaß, D., Baier, S., and Tresp, V. (2015). Type-constrained representation learning in knowledge graphs. In *International Semantic Web Conference*, pages 640–655.

Krompaß, D., Nickel, M., and Tresp, V. (2014). Querying factorized probabilistic triple databases. In *International Semantic Web Conference*, pages 114–129. Springer.

Kruskal, J. B. (1989). Rank, decomposition, and uniqueness for 3-way and n-way arrays. In *Multiway data analysis*, pages 7–18. North-Holland Publishing Co.

Kunegis, J., Gröner, G., and Gottron, T. (2012). Online dating recommender systems: The split-complex number approach. In *ACM RecSys Workshop on Recommender Systems and the Social Web*, pages 37–44. ACM.

Lafferty, J., McCallum, A., Pereira, F., et al. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning*, volume 1, pages 282–289.

Lao, N., Mitchell, T., and Cohen, W. W. (2011). Random walk inference and learning in a large scale knowledge base. In *Empirical Methods in Natural Language Processing*, pages 529–539.

Lawrence, S., Giles, C. L., and Bollacker, K. D. (1999). Autonomous citation matching. In *ACM Conference on Autonomous Agents*, pages 392–393.

Leacock, C. and Chodorow, M. (1998). Combining local context and wordnet similarity for word sense identification. *WordNet: An electronic lexical database*, 49(2):265–283.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, J., Kim, S., Lebanon, G., Singer, Y., and Bengio, S. (2016). Llorma: Local low-rank matrix approximation. *Journal of Machine Learning Research*, 17:1–24.

Lenat, D. B. (1995). Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38.

Levy, O., Remus, S., Biemann, C., Dagan, I., and Ramat-Gan, I. (2015). Do supervised distributional methods really learn lexical inference relations? In *North American Chapter of the Association of Computational Linguistics: Human Language Technologies*, pages 970–976.

Lewis, M. and Steedman, M. (2013). Combining distributional and logical semantics. *Annual Meeting of the Association for Computational Linguistics*, pages 179–192.

Lin, Y., Liu, Z., Luan, H., Sun, M., Rao, S., and Liu, S. (2015a). Modeling relation paths for representation learning of knowledge bases. *Empirical Methods in Natural Language Processing*.

Lin, Y., Liu, Z., Sun, M., Liu, Y., and Zhu, X. (2015b). Learning entity and relation embeddings for knowledge graph completion. In *AAAI Conference on Artificial Intelligence*, pages 2181–2187.

Linial, N., Mendelson, S., Schechtman, G., and Shraibman, A. (2007a). Complexity measures of sign matrices. *Combinatorica*, 27(4):439–463.

Linial, N., Mendelson, S., Schechtman, G., and Shraibman, A. (2007b). Complexity measures of sign matrices. *Combinatorica*, 27(4):439–463.

Lisi, F. A. (2010). Inductive logic programming in databases: From datalog to. *Theory and Practice of Logic Programming*, 10(3):331–359.

Liu, Q., Jiang, H., Wei, S., Ling, Z.-H., and Hu, Y. (2015). Learning semantic word embeddings based on ordinal knowledge constraints. In *Annual Meeting of the Association for Computational Linguistics*, pages 1501–1511.

Liu, Y., Sun, C., Lin, L., and Wang, X. (2016). Learning natural language inference using bidirectional lstm model and inner-attention. *arXiv preprint arXiv:1605.09090*.

Lohr, S. (2010). Aiming to learn as we do, a machine teaches itself. *New York Times, The (NY)*, 5.

London, B., Rekatsinas, T., Huang, B., and Getoor, L. (2013). Multi-relational learning using weighted tensor decomposition with modular loss. *arXiv preprint arXiv:1303.1733*.

Ma, Y., Crook, P. A., Sarikaya, R., and Fosler-Lussier, E. (2015). Knowledge graph inference for spoken dialog systems. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5346–5350.

Macskassy, S. and Provost, F. (2005). Suspicion scoring based on guilt-by-association, collective inference, and focused. In *International Conference on Intelligence Analysis*.

Mao, Y. and Saul, L. K. (2004). Modeling distances in large-scale networks by matrix factorization. In *ACM SIGCOMM conference on Internet Measurement*, pages 278–287.

Marelli, M., Bentivogli, L., Baroni, M., Bernardi, R., Menini, S., and Zamparelli, R. (2014). Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *SemEval*.

McCallum, A., Nigam, K., and Ungar, L. H. (2000). Efficient clustering of high-dimensional data sets with application to reference matching. In *ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 169–178.

McCray, A. T. (2003). An upper-level ontology for the biomedical domain. *Comparative and Functional Genomics*, 4(1):80–84.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mineiro, P. and Karampatziakis, N. (2013). Loss-proportional subsampling for subsequent erm. In *International Conference on Machine Learning*, pages 522–530.

Minervini, P., Demeester, T., Rocktäschel, T., and Riedel, S. (2017). Adversarial sets for regularising neural link predictors. In *Conference on Uncertainty in Artificial Intelligence*.

Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *Annual Meeting of the Association for Computational Linguistics*, pages 1003–1011.

Momtchev, V., Peychev, D., Primov, T., and Georgiev, G. (2009). Expanding the pathway and interaction knowledge in linked life data. *International Semantic Web Challenge*.

Muggleton, S. (1995). Inverse entailment and progol. *New generation computing*, 13(3-4):245–286.

Muggleton, S. and De Raedt, L. (1994). Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19:629–679.

Neelakantan, A., Roth, B., and McCallum, A. (2015). Compositional vector space models for knowledge base completion. In *Annual Meeting of the Association for Computational Linguistics*, pages 156–166.

Neville, J. and Jensen, D. (2003). Collective classification with relational dependency networks. In *International Workshop on Multi-Relational Data Mining*, pages 77–91.

Ngo, L. and Haddawy, P. (1997). Answering queries from context-sensitive probabilistic knowledge bases. *Theoretical Computer Science*, 171(1):147–177.

Nguyen, D. Q., Sirts, K., Qu, L., and Johnson, M. (2016). Stranse: a novel embedding model of entities and relationships in knowledge bases. In *North American Chapter of the Association of Computational Linguistics: Human Language Technologies*, pages 460–466.

Nickel, M. (2013). *Tensor factorization for relational learning*. PhD thesis, Ludwig-Maximilians-Universität.

Nickel, M., Jiang, X., and Tresp, V. (2014). Reducing the rank in relational factorization models by including observable patterns. In *Advances in Neural Information Processing Systems*, pages 1179–1187.

Nickel, M., Murphy, K., Tresp, V., and Gabrilovich, E. (2016a). A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33.

Nickel, M., Rosasco, L., and Poggio, T. A. (2016b). Holographic embeddings of knowledge graphs. In *AAAI Conference on Artificial Intelligence*, pages 1955–1961.

Nickel, M. and Tresp, V. (2013). Logistic tensor factorization for multi-relational data. *arXiv preprint arXiv:1306.2084*.

Nickel, M., Tresp, V., and Kriegel, H.-P. (2011). A three-way model for collective learning on multi-relational data. In *International Conference on Machine Learning*, pages 809–816.

Niepert, M. (2016). Discriminative gaifman models. In *Advances in Neural Information Processing Systems*, pages 3405–3413.

Nieplocha, J., Harrison, R. J., and Littlefield, R. J. (1996). Global arrays: A nonuniform memory access programming model for high-performance computers. *The Journal of Supercomputing*, 10(2):169–189.

Niu, F., Recht, B., and Wright, S. J. (2011). Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 1–22.

Noessner, J., Niepert, M., and Stuckenschmidt, H. (2013). RockIt: Exploiting Parallelism and Symmetry for MAP Inference in Statistical Relational Models. In *CoRR*.

Paatero, P. and Tapper, U. (1994). Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126.

Paisley, J., Blei, D., and Jordan, M. (2012). Variational bayesian inference with stochastic search. *International Conference on Machine Learning*.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Pirasteh, P., Hwang, D., and Jung, J. J. (2015). Exploiting matrix factorization to asymmetric user similarities in recommendation systems. *Knowledge-Based Systems*, 83:51–57.

Polyak, B. T. and Juditsky, A. B. (1992). Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*.

Popper, K. (1934). *Logik der Forschung*. Mohr Siebeck.

Pujara, J., Miao, H., Getoor, L., and Cohen, W. W. (2013). Knowledge graph identification. In *International Semantic Web Conference*, pages 542–557.

Raedt, L. D., Kersting, K., Natarajan, S., and Poole, D. (2016). Statistical relational artificial intelligence: Logic, probability, and computation. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 10(2):1–189.

Rendle, S. (2010). Factorization machines. In *IEEE International Conference on Data Mining*, pages 995–1000.

Rendle, S. (2013). Scaling factorization machines to relational data. In *Very Large Data Bases*, volume 6, pages 337–348.

Rendle, S. and Schmidt-Thieme, L. (2010). Pairwise interaction tensor factorization for personalized tag recommendation. In *ACM international conference on Web search and data mining*, pages 81–90.

Richardson, M. and Domingos, P. (2006). Markov logic networks. *Machine Learning*, 62(1-2):107–136.

Riedel, S. (2008). Improving the accuracy and efficiency of MAP inference for markov logic. In *Conference on Uncertainty in Artificial Intelligence*, pages 468–475.

Riedel, S., Yao, L., McCallum, A., and Marlin, B. M. (2013). Relation extraction with matrix factorization and universal schemas. In *North American Chapter of the Association of Computational Linguistics: Human Language Technologies*, pages 74–84.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *Annals of Mathematical Statistics*, 22(3).

Rocktäschel, T., Bosnjak, M., Singh, S., and Riedel, S. (2014). Low-dimensional embeddings of logic. In *Workshop on semantic parsing at ACL*.

Rocktäschel, T., Grefenstette, E., Hermann, K. M., Kočiskỳ, T., and Blunsom, P. (2016). Reasoning about entailment with neural attention. In *International Conference on Learning Representations*.

Rocktäschel, T. and Riedel, S. (2016). Learning knowledge base inference with neural theorem provers. *Workshop on Automated Knowledge Base Construction at NAACL-HLT*, pages 45–50.

Rocktaschel, T., Singh, S., and Riedel, S. (2015). Injecting Logical Background Knowledge into Embeddings for Relation Extraction. In *North American Chapter of the Association of Computational Linguistics: Human Language Technologies*, pages 1119–1129.

Romera-Paredes, B. and Pontil, M. (2013). A new convex relaxation for tensor completion. In *Advances in Neural Information Processing Systems*, pages 2967–2975.

Ross, S. (1997). Simulation academic press. *San Diego*.

Rummel, R. J. (1976). *The dimensionality of nations project: attributes of nations and behavior of nations dyads, 1950-1965*. Number 5409. Inter-University Consortium for Political Research.

Saad, Y. (1992). *Numerical methods for large eigenvalue problems*, volume 158. SIAM.

Sahoo, S. S., Halb, W., Hellmann, S., Idehen, K., Thibodeau Jr, T., Auer, S., Sequeda, J., and Ezzat, A. (2009). A survey of current approaches for mapping of relational databases to rdf. *W3C RDB2RDF Incubator Group Report*, pages 113–130.

Salakhutdinov, R. and Murray, I. (2008). On the quantitative analysis of deep belief networks. In *International Conference on Machine Learning*, pages 872–879.

Scott, S. and Matwin, S. (1998). Text classification using wordnet hypernyms. In *Use of WordNet in natural language processing systems*, pages 38–44.

Sedghi, H. and Sabharwal, A. (2016). Knowledge completion for generics using guided tensor factorization. *arXiv preprint arXiv:1612.03871*.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. (2008). Collective classification in network data. *AI magazine*, 29(3):93.

Senecal, J.-S. and Bengio, Y. (2003). Adaptive importance sampling to accelerate training of a neural probabilistic language model. Technical report, Idiap.

Singh, A. P. and Gordon, G. J. (2008). Relational learning via collective matrix factorization. In *ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 650–658. ACM.

Singh, S., Rocktäschel, T., and Riedel, S. (2015). Towards combined matrix and tensor factorization for universal schema relation extraction. In *Workshop on Vector Space Modeling for Natural Language Processing at NAACL-HLT*, pages 135–142.

Smolensky, P., Lee, M., He, X., Yih, W.-t., Gao, J., and Deng, L. (2016). Basic reasoning with tensor product representations. *arXiv preprint arXiv:1601.02745*.

Socher, R., Chen, D., Manning, C. D., and Ng, A. (2013). Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934.

Stoica, P. and Moses, R. L. (2005). *Spectral analysis of signals*, volume 452. Pearson Prentice Hall Upper Saddle River, NJ.

Suchanek, F. M., Kasneci, G., and Weikum, G. (2007). Yago: a core of semantic knowledge. In *International Conference on World Wide Web*, pages 697–706.

Surdeanu, M. and Tibshirani, J. (2012). Multi-instance multi-label learning for relation extraction. In *Empirical Methods in Natural Language Processing*.

Sutskever, I., Tenenbaum, J. B., and Salakhutdinov, R. R. (2009). Modelling relational data using bayesian clustered tensor factorization. In *Advances in neural information processing systems*, pages 1821–1828.

Taskar, B., Segal, E., and Koller, D. (2001). Probabilistic classification and clustering in relational data. In *International Joint Conference on Artificial Intelligence*, volume 17, pages 870–878.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2).

Tomioka, R., Hayashi, K., and Kashima, H. (2010). Estimation of low-rank tensors via convex optimization. *arXiv preprint arXiv:1010.0789*.

Toutanova, K., Chen, D., Pantel, P., Poon, H., Choudhury, P., and Gamon, M. (2015). Representing text for joint embedding of text and knowledge bases. In *Empirical Methods in Natural Language Processing*, volume 15, pages 1499–1509.

Traud, A., Kelsic, E., Mucha, P., and Porter, M. (2009). Community structure in online collegiate social networks. *Bulletin of the American Physical Society*, 54.

Trouillon, T., Dance, C. R., Gaussier, É., and Bouchard, G. (2016a). Decomposing real square matrices via unitary diagonalization. *arXiv:1605.07103*.

Trouillon, T., Dance, C. R., Welbl, J., Riedel, S., Gaussier, É., and Bouchard, G. (2017a). Knowledge graph completion via complex tensor factorization. *arXiv preprint arXiv:1702.06879*. To appear in the Journal of Machine Learning Research.

Trouillon, T., Gaussier, É., Dance, C. R., and Bouchard, G. (2017b). On inductive abilities of latent factor models. Submitted to the Journal of Artificial Intelligence Research.

Trouillon, T. and Nickel, M. (2017). Complex and holographic embeddings of knowledge graphs: a comparison. *International Workshop on Statistical Relational AI*.

Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., and Bouchard, G. (2016b). Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, volume 48, pages 2071–2080.

Tucker, L. R. (1963). Implications of factor analysis of three-way matrices for measurement of change. *Problems in measuring change*, pages 122–137.

Tyler, J. R., Wilkinson, D. M., and Huberman, B. A. (2005). E-mail as spectroscopy: Automated discovery of community structure within organizations. *The Information Society*, 21(2):143–153.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142.

Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc.

Varelas, G., Voutsakis, E., Raftopoulou, P., Petrakis, E. G., and Milios, E. E. (2005). Semantic similarity methods in wordnet and their application to information retrieval on the web. In *ACM International Workshop on Web Information and Data Management*, pages 10–16.

Verga, P., Neelakantan, A., and McCallum, A. (2017). Generalizing to unseen entities and entity pairs with row-less universal schema. In *European Chapter of the Association of Computational Linguistics*.

von Neumann, J. (1929). Zur algebra der funktionaloperationen und der theorie der normalen operatoren. *Mathematische Annalen*, 102:370–427.

Wang, W. Y. and Cohen, W. W. (2016). Learning first-order logic embeddings via matrix factorization. In *International Joint Conference on Artificial Intelligence*, pages 2132–2138.

Wang, Z., Zhang, J., Feng, J., and Chen, Z. (2014). Knowledge graph embedding by translating on hyperplanes. In *AAAI Conference on Artificial Intelligence*, pages 1112–1119.

Welbl, J., Bouchard, G., and Riedel, S. (2016). A factorization machine framework for testing bigram embeddings in knowledge base completion. In *Workshop on Automated Knowledge Base Construction at NAACL-HLT*, pages 103–107.

Wellman, M. P., Breese, J. S., and Goldman, R. P. (1992). From knowledge bases to decision models. *The Knowledge Engineering Review*, 7(01):35–53.

Weston, J., Bordes, A., Chopra, S., Rush, A. M., van Merriënboer, B., Joulin, A., and Mikolov, T. (2015). Towards AI-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Wikipedia (2004). Binary relation — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Binary_relation`.

Xie, F., Chen, Z., Shang, J., Feng, X., and Li, J. (2015). A link prediction approach for item recommendation with complex numbers. *Knowledge-Based Systems*, 81:148–158.

Xu, C., Bai, Y., Bian, J., Gao, B., Wang, G., Liu, X., and Liu, T.-Y. (2014). Rc-net: A general framework for incorporating knowledge into word representations. In *Conference on Information and Knowledge Management*, pages 1219–1228.

Yampolskiy, R. V. (2012). Ai-complete, ai-hard, or ai-easy-classification of problems in ai. In *Midwest Artificial Intelligence and Cognitive Science Conference*, pages 94–101.

Yang, B., Yih, W.-t., He, X., Gao, J., and Deng, L. (2015). Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations*.

Yao, L., Riedel, S., and Mccallum, A. (2011). Structured relation discovery using generative models. In *Empirical Methods in Natural Language Processing*, pages 1456–1466.

Yoon, H.-G., Song, H.-J., Park, S.-B., and Park, S.-Y. (2016). A translation-based knowledge graph embedding preserving logical property of relations. In *North American Chapter of the Association of Computational Linguistics: Human Language Technologies*, pages 907–916.

Yu, H.-F., Hsieh, C.-J., Si, S., and Dhillon, I. (2012). Scalable coordinate descent approaches to parallel matrix factorization for recommender systems. In *International Conference on Data Mining*, pages 765–774.

Yun, H., Yu, H.-F., Hsieh, C.-J., Vishwanathan, S., and Dhillon, I. (2014). Nomad: Non-locking, stochastic multi-machine algorithm for asynchronous and decentralized matrix completion. *Very Large Data Bases*, 7(11):975–986.

Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zhang, Z., Li, T., Ding, C., and Zhang, X. (2007). Binary matrix factorization with applications. In *International Conference on Data Mining*, pages 391–400.