Reasoning on Graph-Structured Data with Deep-Learning, Path-Based Methods, and Tensor Factorization

Marcel Hildebrandt

München, 2020

Reasoning on Graph-Structured Data with Deep-Learning, Path-Based Methods, and Tensor Factorization

Marcel Hildebrandt

Dissertation
an der Fakultät für Mathematik, Informatik und Statistk der
Ludwig-Maximilians-Universität München

vorgelegt von
Marcel Hildebrandt
aus Berlin

München, den 03.11.2020

Erstgutachter:	Prof. Dr. Volker Tresp
Zweitgutachter	Prof. Dr. Abraham Bernstein
Drittgutachter:	Prof. Dr. Tim Rocktäschel
Tag der mündlichen Prüfung:	19.02.2021

Eidesstattliche Versicherung

(Siehe Promotionsordnung vom 12.07.11, § 8, Abs. 2 Pkt. .5.)

Hiermit erkläre ich an Eidesstatt, dass die Dissertation von mir selbstständig, ohne unerlaubte Beihilfe angefertigt ist.

Hildebrandt, Marcel

Name, Vorname

14.03.2021	Marcel Hildebrandt
Ort, Datum	Unterschrift Doktorand/in

Formular 3.2

Contents

Acknowledgements

This dissertation is the results of the last three years that I spent as a PhD student at LMU and Siemens Technology. Throughout this time, I received great support and assistance from my supervisor, colleagues, friends, and my family.

First, I want to express my deepest gratitude to my supervisor Prof. Dr. Volker Tresp for the continuous support during my PhD studies. While giving me the freedom and trust to pursue various ideas, Volker was always available, gave insightful comments, and steered me in the right direction when I needed guidance. Without his invaluable help and the unique environment that Volker has created at Siemens and LMU, this dissertation would not have been possible.

I am also very grateful to my colleagues at Siemens Dr. Martin Ringsquandl, Dr. Mitchell Joblin, and Serghei Mogoreanu for their commitment, interest, and passion. I knew that I could always ask them for advice and get honest feedback. In particular, I want to thank them for the endless discussions, the times when they encouraged me, or challenged my ideas. Furthermore, I am indebted to my fellow PhD Students Rajat Koner and Yunpu Ma, who helped me develop some of the central ideas of this thesis. Special thanks go to my former students Hang Li, Jorge Andres Quintero Serna, and Swathi Shyam Sunder. They exceeded all expectations and made an invaluable contribution to this thesis.

I would also like to extend my gratitude to Dr. Steffen Lamparter. He gave me the opportunity to join his research group at Siemens and provided me with the freedom and resources to follow my interest and realize my ideas. I also want to thank Dr. Thomas Hubauer, who helped me get started at Siemens and was always supportive of my research goals.

A big thank you to Natasha for proofreading all of my papers and being a great friend. And of course, special thanks go to Laura for her love, patience, and tolerating states of approximate order and cleanliness during stressful periods.

Last but not least, I am grateful to my family for their continuous support and understanding. Nobody was more important for the completion of this thesis than you. Thank you!

Abstract

During the course of their lives, people continuously gather information and use this knowledge when making predictions, deriving at a decision, or, generally speaking, when they reason and interact with their environment. Most machine learning models still lack the ability to leverage explicit, factual background information which could be a great benefit for any of those tasks. In this thesis, we propose that knowledge graphs might be the basis for both representing factual knowledge and integrating it into prediction and decision making. Over the last decades, they have become a widely used resource to structure factual knowledge in a machine-readable format. The nodes in the graph correspond to entities of the real-world. Typed edges between pairs of nodes indicate their relationships and encode factual statements.

Modern, large-scale knowledge graphs store massive amounts of information, often up to many billions of facts. However, most knowledge graphs suffer from an inherent incompleteness in the sense that they do not include all true facts in the scope of the actual entities and relations. Moreover, knowledge graphs may also contain triples that correspond to false facts. Thus, two canonical machine learning tasks are concerned with first, automatic knowledge curation and refinement with the goal to infer missing facts based on observed triples, second, to classify the truth value of facts. The first part of this thesis is concerned with path-based reasoning on knowledge graphs for fact classification. In contrast to pure embedding-based techniques, path-based reasoning methods attempt to construct predictive paths on knowledge graphs, which serve as explicit features for prediction tasks. We employ reinforcement learning for goal-directed path extraction in two different settings. First, we develop the idea of debate dynamics, where the fact-checking task in a knowledge graph is framed as a debate game between two competing agents. The arguments of the debate correspond to paths on the knowledge graph. In contrast to existing path-based methods that only mine supportive features for the truthfulness of a fact, the goal of the debate setup is to examine a query fact from two different angles. This is realized by training the agents to extract two sets of opposing arguments that provide evidence for the fact being true or the fact being false, respectively. Hence, the agents can be regarded as sparse, adversarial, and interpretable feature generators. Our method allows for interactive reasoning on knowledge graphs where the users can raise additional arguments or evaluate the debate taking common sense reasoning and external information into account. We show via experiments on popular benchmark datasets and a user study that our method outperforms several baseline methods while being more interpretable.

Second, we consider the visual question answering task, which is concerned with answering free form questions about an image. Our approach is based on deriving a graphical description of a scene where the nodes correspond to the detected objects and typed edges indicate spatial and semantic relationships between the objects. This representation is called a scene graph and its formalism resembles the knowledge graph framework that we adopt in this thesis. Given the scene graph of an image, we propose a novel reasoning module where an agent navigates over the graph until a conclusive reasoning path is obtained that allows to answer the question. We investigate the performance of our scene graph reasoning module on a challenging visual question answering dataset based on manually curated scene graphs. We find that our method reaches human-like performance, which is currently the upper bound for visual questions answering datasets.

Knowledge graphs are not only mere containers of facts. They can convey knowledge and act as a backbone in various artificial intelligence applications. Modern representation learning techniques condense structural information of entities and relations in low-dimensional vector spaces. Based on these embeddings, relational information can be ingested by other machine learning modules that perform various downstream tasks. Similarly, many artificial intelligence tasks can be phrased as a link prediction problem in a knowledge graph. The second part of this thesis makes use of this idea and tackles the real-world task of recommending components for industrial control systems. Concretely, we formulate the recommendation task as a link prediction problem in a knowledge graph that combines two data sources: technical information about available components and historical configuration data. In this context, we develop two novel recommendation engines employing a tensor factorization and an autoencoder coupled with a tensor factorization, respectively. Based on a real-world, industrial dataset, we find that our methods significantly outperform several baseline methods, while satisfying important real-world demands. These requirements include the ability to compute recommendations efficiently in real-time in a partially inductive setting and the ability to cope with the cold start problem.

Zusammenfassung

Im Laufe ihres Lebens sammeln Menschen stetig Informationen über ihre Umwelt und beziehen diese Information mit ein, wenn sie Vorhersagen machen, Entscheidungen treffen oder, allgemein gesprochen, über ihre Umwelt Schlüsse ziehen und mit ihr interagieren. Die meisten Modelle des maschinellen Lernens können nicht auf explizite, faktische Hintergrundinformationen zurückgreifen. In dieser Arbeit schlagen wir Wissensgraphen als Grundlage für die Repräsentation von faktischen Wissen und dessen Integration in Vorhersage- und Entscheidungsmodelle vor. In den letzten Jahrzehnten haben sie sich zu einem weit verbreiteten Werkzeug entwickelt, um Wissen in einem maschinenlesbaren Format zu strukturieren. Die Knoten im Graphen entsprechen dabei Entitäten der realen Welt. Die typisierten Kanten zwischen Knotenpaaren geben ihre Beziehungen an und entsprechen faktischen Aussagen.

Moderne, groß angelegte Wissensgraphen speichern riesige Mengen an Informationen, oftmals bis zu vielen Milliarden Fakten. Trotzdem sind viele Wissensgraphen unvollständig. Damit ist gemeint, dass wahre Fakten, die durch die vorliegenden Entitäten und Relationen ausgedrückt werden können, nicht im Wissensgraphen enthalten sind. Außerdem sind oft auch falsche Fakten in Wissengsraphen kodiert. Gängige Aufgaben des maschinellen Lernens beschäftigen sich daher mit der Vorhersage von neuen Kanten sowie mit der Klassifikation von Fakten. Der erste Teil dieser Arbeit befasst sich mit pfadbasierten Methoden auf Wissensgraphen zur Klassifikation des Wahrheitsgehaltes von Fakten. Im Gegensatz zu Techniken, die nur auf Repräsentationslernen basieren, zielen pfadbasierte Methoden darauf ab, prädiktive Pfade auf Wissensgraphen zu konstruieren, die als strukturierte Erklärung für Vorhersageaufgaben dienen. Wir setzen bestärkendes Lernen für die zielgerichtete Pfadextraktion ein. Unser Forschungsbeitrag liegt hierbei in zwei Bereichen. Zum einem entwickeln wir die Idee der Debattendynamik, bei der die Aufgabe der Faktenklassifikation als Debattierspiel zwischen zwei konkurrierenden Agenten formuliert wird. Dabei entsprechen die Argumente Pfaden im Wissensgraphen. Die meisten pfadbasierten Methoden zielen darauf ab, lediglich Merkmale zu extrahieren, die den Wahrheitsgehalt eines Faktes unterstützen. Das Ziel des Debattendynamik besteht hingegen darin, einen Fakt aus zwei unterschiedlichen Blickwinkeln zu untersuchen. Wir realisieren dies dadurch, dass wir zwei konkurrierende Agenten trainieren, Argumente jeweils für und gegen die Validität eines Faktes zu extrahieren. In dem Sinne fungieren die Argumente als adverseriale und interpretierbare Merkmale für die Vorhersage des Wahrheitswertes eines Faktes. Die resultierende Methode erlaubt die Entwicklung von interaktiven Anwendungen, in denen der Benutzer zusätzliche Argumente einbringen kann oder das Ergebnis einer Debatte mithilfe von Kontextinformationen eigenständig evaluieren kann. Wir zeigen auf der Basis von Experimenten mit verschiedenen Benchmarkdatensätzen sowie einer Benutzerumfrage, dass unsere Methode bessere Ergebnisse als existierende Methoden erzielt und darüber hinaus interpretierbare Erklärungen liefert.

Des Weiteren betrachten wir die Aufgabe der automatischen Beantwortung von Fragen mit visuellem Bezug. Dabei sind die Fragen in natürlicher Sprache formuliert und beziehen sich auf den Inhalt eines Bildes. Unser Ansatz basiert darauf, eine Netzwerkdarstellung der dargestellten Szene herzuleiten, in der die Knoten den Objekten entsprechen und die typisierten Kanten räumliche und semantische Beziehungen zwischen diesen Objekten angeben. Diese Darstellung wird als Szenengraph bezeichnet und ähnelt dem Formalismus für Wissensgraphen, den wir in dieser Arbeit verwenden. Basierend auf dem Szenengraph entwickeln wir ein neuartiges Argumentationsmodul, bei dem ein Agent über dem Graphen navigiert, bis ein schlüssiger Argumentationspfad extrahiert wird, der die Beantwortung der Frage ermöglicht. Wir zeigen mithilfe von manuell erstellten Szenengraphen auf einem anspruchsvollen Datensatz, dass unsere Methode in etwa menschliches Leistungsniveau erreicht, welches aktuell eine obere Schranke für das Leistungsvermögen maschineller Lernmethoden darstellt.

Wissensgraphen sind nicht nur als Container von Fakten. Sie können das Wissen aufbereiten, vermitteln und fungieren dadurch als Rückgrat in verschiedenen Anwendungen der künstlichen Intelligenz. Insbesondere ermöglichen moderne Methoden des Repräsentationslernens die Verdichtung von strukturellen Informationen über Entitäten und Relationen in niedrigdimensionalen Vektorräumen. Diese Einbettungen ermöglichen, relationale Informationen in andere maschinelle Lernmodule miteinfließen zu lassen. Außerdem können viele Aufgaben der künstlichen Intelligenz als Kantenvorhersageproblem in

Wissensgraphen formuliert werden. Im zweiten Teil der vorliegenden Arbeit nutzen wir diese Idee und befassen uns mit der Aufgabe, Komponenten für industrielle Steuerungssysteme zu empfehlen. Dabei formulieren wir die Generierung von Empfehlungen als Kantenvorhersageproblem in einem Wissensgraphen, welcher zwei Datenquellen kombiniert: technische Informationen über verfügbare Komponenten und historische Konfigurationsdaten. In diesem Zusammenhang entwickeln wir zwei neuartige Empfehlungssysteme auf der Basis jeweils einer Tensorfaktorisierung und eines Autoencoder gekoppelt mit einer Tensorfaktorisierung. Wir vergleichen die Ergebnisse mit existierend Methoden auf der Basis eines industriellen Datensatzes. Unsere experimentellen Ergebnisse zeigen, dass unseren Methoden signifikant bessere Resultate als bereits existierende Empfehlungssysteme erzielen. Außerdem analysieren wir unsere Methoden in Bezug auf ihre Anwendbarkeit in der Praxis und zeigen, dass unsere Methoden mit dem Kaltstartproblem umgehen und unter partiell induktiven Bedingungen effizient Empfehlungen berechnen können.

List of Publications and Declaration of Authorship

- Marcel Hildebrandt, Jorge Andres Quintero Serna, Yunpu Ma, Martin Ringsquandl, Mitchell Joblin, and Volker Tresp. *Reasoning on Knowledge Graphs with Debate Dynamics.* In Proceedings of the AAAI Conference on Artificial Intelligence, 2020.

  I conceived of the original research contributions. My co-author Jorge Andres Quintero Serna and I performed most of the implementations. Jorge Andres Quintero Serna and I designed the experimental protocol and ran the experiments. Jorge Andres Quintero Serna conducted the evaluation. I wrote the initial draft of the manuscript. All co-authors contributed to the subsequent corrections. I regularly discussed this work with Yunpu Ma, Martin Ringsquandl, Mitchell Joblin, and my supervisor Volker Tresp.

  The publication serves as Chapter 3 of this thesis.

- Marcel Hildebrandt, Hang Li, Rajat Koner, Volker Tresp, and Stephan Günnemann. *Scene Graph Reasoning for Visual Question Answering.* In International Conference on Machine Learning: Workshop GRL+, 2020.

  Rajat Koner and I conceived the original research contributions. Hang Li and I designed the scene graph reasoning module and Hang Li performed most of the implementations related to scene graph reasoning. Rajat Koner worked on the scene graph generator focusing on finding appropriate methods and on the implementation. Hang Li, Rajat Koner, and I designed the experimental protocol. Hang Li ran the experiments and conducted the evaluation. I wrote the initial draft of the manuscript. All co-authors contributed to the subsequent corrections. I regularly discussed this work with Stephan Günnemann and my supervisor Volker Tresp.

The publication serves as Chapter 4 of this thesis.

- Marcel Hildebrandt, Swathi Shyam Sunder, Serghei Mogoreanu, Ingo Thon, Volker Tresp, and Thomas Runkler. *Configuration of industrial automation solutions using multi-relational recommender systems.* In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 2018.

  I conceived the original research contributions. My co-author Swathi Shyam Sunder and I performed most of the implementations. Serghei Mogoreanu managed the data and conducted the preprocessing. Swathi Shyam Sunder, Serghei Mogoreanu and I designed the experimental protocol and ran the experiments. Swathi Shyam Sunder conducted most of the evaluation. Serghei Mogoreanu and Ingo Thon gave guidance and shaped the project with respect to its real-world applicability. Swathi Shyam Sunder and I wrote the initial draft of the manuscript. All co-authors contributed to the subsequent corrections. I regularly discussed this work with Thomas Runkler and my supervisor Volker Tresp.

  The publication serves as Chapter 5 of this thesis.

- Marcel Hildebrandt, Swathi Shyam Sunder, Serghei Mogoreanu, Mitchell Joblin, Akhil Mehta, Ingo Thon, and Volker Tresp (2019, June). *A recommender system for complex real-world applications with nonlinear dependencies and knowledge graph context.* In Proceedings of the European Semantic Web Conference, 2019.

  I conceived the original research contributions. My co-author Swathi Shyam Sunder and I performed most of the implementations. Serghei Mogoreanu managed the data and conducted the preprocessing. Swathi Shyam Sunder, Serghei Mogoreanu, Akhil Mehta, and I designed the experimental protocol and ran the experiments. Swathi Shyam Sunder and Akhil Mehta conducted most of the evaluation. I wrote the initial draft of the manuscript. Serghei Mogoreanu and Ingo Thon gave guidance and shaped the paper with respect to the real-world use case underlying this work. All co-authors contributed to the subsequent corrections. I regularly discussed this work with Mitchell Joblin and my supervisor Volker Tresp.

The publication serves as Chapter 6 of this thesis.

Other Publications

- Sebastian Matuszewski, Marcel Hildebrandt, Ana-Hermina Ghenu, Jeffrey Jensen, and Claudia Bank. *A statistical guide to the design of deep mutational scanning experiments.* In Genetics, 2016.
- Sebastian Matuszewski, Marcel Hildebrandt, Guillaume Achaz, and Jeffrey Jensen. *Coalescent processes with skewed offspring distributions and nonequilibrium demography.* In Genetics, 2018.
- Martin Ringsquandl, Evgeny Kharlamov, Daria Stepanova, Marcel Hildebrandt, Steffen Lamparter, Raffaello Lepratti, Ian Horrocks, and Peer Kröger. *Event-enhanced learning for KG completion.* In Proceedings of the European Semantic Web Conference, 2018.
- Yunpu Ma, Marcel Hildebrandt, Stephan Baier, and Volker Tresp. *Holistic Representations for Memorization and Inference.* In Conference on Uncertainty in Artificial Intelligence, 2018.
- Marcel Hildebrandt, Mohamed Khalil, Christoph Bergs, Volker Tresp, Roland Wüchner, Kai-Uwe Bletzinger, and Michael Heizmann. *Remaining Useful Life Estimation for Unknown Motors Using a Hybrid Modeling Approach.* In IEEE International Conference on Industrial Informatics, 2019.
- Christoph Bergs, Mohamed Khalil, Marcel Hildebrandt, Michael Heizmann, Roland Wüchner, Kai-Uwe Bletzinger, and Volker Tresp. *Health indication of electric motors using a hybrid modeling approach.* In tm-Technisches Messen, 2019.
- Marcel Hildebrandt, Jorge Andres Quintero Serna, Yunpu Ma, Martin Ringsquandl, Mitchell Joblin, and Volker Tresp. *Debate Dynamics for Human-comprehensible Fact-checking on Knowledge Graphs.* In AAAI Fall Symposium: Human-Centered AI, 2019.

- Yushan Liu, Marcel Hildebrandt, Mitchell Joblin, Martin Ringsquandl, and Volker Tresp. *Integrating Logical Rules Into Neural Multi-Hop Reasoning for Drug Repurposing.* In International Conference on Machine Learning: Workshop GRL+, 2020.

Chapter 1

Introduction

1.1 Motivation

Our society produces massive amounts of data that describe complex systems consisting of various interacting components. Such data often exhibit rich structures and interdependencies that cannot be captured by features aligned on a regular grid (e.g., structured in a design matrix) that do not consider the relations between the systems' components. For instance, every day billions of people interact on social media, interconnected sensors collect data that allows to monitor and optimize the operation of electrical grids, or magnetic resonance imaging techniques can trace the activity of neural circuits formed by neurons and synapses. Graphs provide an intuitive and concise abstraction to model discrete, complex systems. Thereby the components of the system are represented by nodes, their interactions as edges. Depending on the studied phenomenon, the interactions between nodes can result in diverse patterns that differ in topology and scale. This irregular, inherently non-Euclidean nature imposes a challenge for machine learning methods since basic algebraic operations that are the foundation of most machine learning techniques cannot be directly applied to graph-structured data. Thus, canonical learning task such as link prediction, graph or node classification, and graph generation, that often correspond to highly relevant real-world tasks such as computing recommendations or synthesizing chemical molecules require specialized methods that operate on graph-structures.

This thesis is mainly concerned with machine learning on graphs that store structured representations of knowledge about the world via collections of factual statements. Such graphs are often referred to as a knowledge graphs (KGs). The goal of a KG is to accumulate and convey knowledge in a machine-readable format. Thereby, nodes represent entities of the real world and edges that come with different labels indicate the relations between them. Originating from the Semantic Web [5] and the Linked Open Data vision [4], KGs often integrate data from numerous sources resulting in diverse structures and granularities of knowledge. Thereby, the absence of a fixed schema allows the knowledge to evolve in a flexible manner [28]. Even though modern, large-scale KGs contain a massive amount of facts, they still suffer from an inherent incompleteness in the sense that true facts are not contained the KG. In addition, KGs may also contain false facts. Employing a graph-structured abstraction of knowledge enables the usage of a range of graph reasoning techniques that aim to infer new knowledge or detect false facts. Early research on KGs was geared towards advancing the logical reasoning capabilities of KGs. These approaches are based on handcrafted or automatically mined rules and come with the advantage that reasoning is relatively accurate, explicit, and, therefore, explainable. However, due to scalability issues of logical reasoning approaches and driven by the success of machine learning methods on graphs and other data modalities, many of the dominant approaches in KG reasoning follow the representation learning paradigm. Like word embeddings that capture semantic meanings of words in natural language, KG embeddings are vector space representations that preserve semantic structures of entities and relations and allow the efficient execution of downstream tasks. Compared to logical reasoning methods, learning techniques on KGs often come with fewer requirements on the modeling side, thus, alleviating the knowledge acquisition bottleneck. This leads to a reduced formalism, which is sufficient to solve many practical learning tasks. As a consequence, knowledge graphs are now not only containers of facts. They also act as the backbone for many machine learning applications. In particular, KG embeddings allow relational knowledge to enhance different learning tasks such as named entity disambiguation in natural language processing [21] or visual relation detection [3] in computer vision.

This thesis investigates machine learning on KGs in different settings. For example, we develop a novel method for the KG completion task (i.e., deriving missing facts) on generic, cross-domain KGs. This setting is typically used to benchmark KG reasoning techniques. In addition, we also consider real-world industrial tasks where the underlying KG focuses on a specific domain. The vision of the Industry 4.0 enables KGs to be applied to manufacturing processes as a practical tool to organize data from all stages of the production pipeline. We study KGs in a setting where complex engineering equipment consists of multiple sub-systems. By storing existing system configurations in a KG and computing corresponding embeddings, we demonstrate how one can leverage the underlying relational structure and adjust learning technologies to build recommender systems that meet crucial real-world demands such as efficiency requirements or the ability to overcome the cold start problem. Finally, we also work in a setting where the graph structure is not readily available but needs to be extracted from the raw data. Concretely, we consider images and derive semantic and spatial relationships among the objects in the depicted scene leading to scene graph representations [29]. These scene graphs resemble KGs in the sense that the relations between the detected objects are encoded via typed edges. Given this graph representation of an image, advances in KG reasoning and deep graph learning are combined to tackle a challenging computer vision task.

1.2 Contributions of this Thesis

The research focus of this thesis is threefold: In the first part we address the black-box problem that is inherent to most embedding-based KG reasoning methods. More specifically, when using KG embedding for inferring new triples or fact-checking, it usually remains hidden from the user what contributed to the prediction of a model. Using a transparent method may lead to more trustworthiness, allows to keep humans in the loop, and facilitates debugging especially in a situations where the data is biased or has some other flaws. As a remedy to the non-transparent nature of KG embedding methods, we propose the idea of debate dynamics for reasoning on KGs to tackle the black box problem. Concretely, we develop the novel method R2D2 for triple classification (basically, fact-checking) on KGs. The goal is to train two opposing reinforcement learning agents that engage in a debate game. Thereby both agents extract arguments – paths in the knowledge graph – that serve as evidence for and against the truthfulness of the fact. We empirically find that R2D2 achieves state-of-the-art performance with respect to the triple classification accuracy while being more interpretable than existing embedding-based methods.

The second part considers the visual question answering task (VQA), which is concerned with answering image-related free-form questions. Given an image and a free-form question, the task is to find the correct answer based on the presented scene. Heuristically speaking, successful VQA systems are required to possess the faculty of sight (processing the image), the faculty of hearing (processing the question), and the faculty of reason (derive the correct answer). Therefore, VQA is often described as an AI-complete task and the performance of state-of-the-art methods still falls short of humans on challenging VQA datasets. Many existing VQA approaches are agnostic towards the explicit relational structure of the objects in the presented scene. They rely solely on processing regional features of the image via complex neural network architectures. We propose a novel reasoning module that constructs explicit reasoning chains on a derived network representation of the depicted scene. As a testament to the reasoning and language-understanding capabilities, we show that our method achieves human-like performance based on manually curated scene graphs.

Finally, the last part of this thesis is application-driven and deals with automation systems as an example of complex industrial equipment. In this context, we leverage an existing tensor factorization technique to develop the novel recommendation system RESCOM that incorporates both historical user behavior and technical information merged in KG. We demonstrate that RESCOM can handle real-world requirements such as computing recommendations in linear time. Moreover, in contrast to most other embedding-based methods, RESCOM can operate in a partially inductive setting where embeddings for new equipment can be computed in real-time without retraining the model. Our experimental findings show that incorporating technical context information allows to tackle the cold start problem. At the time of writing, RESCOM is employed in one of the major engineering configurators at Siemens AG with around 60,000 active users and around 170,000 configured automation solutions corresponding to a potential revenue of more than 132M€ per month. In the same context, we propose the recommender engine NECTR as an extension to RESCOM. NECTR consists of a tensor factorization model coupled with a graph autoencoder. While the tensor factorization encodes the items' technical information, the neural network captures the non-linear interactions among configured items. Thereby NECTR is more expressive than RESCOM, since it is able to capture higher-order interactions among the configured items that typically arise in the context of complex equipment where the functionality results from the interplay of its components. We show experimentally that this leads to significant performance gains with respect to standard performance measures such as the mean rank, the mean reciprocal rank, or Hits@K for $K = 1, 3, 5, 10$. NECTR is scheduled to substitute RESCOM as a recommender system in the aforementioned engineering configurator in the upcoming release version.

1.3 Overview

The remainder of this thesis is organized as follows. Chapter 2 contains background information on KGs and graph learning. That chapter is not meant to be a survey. Its purpose is rather to introduce important concepts and lay the foundations for the techniques that are used in the remainder of this thesis. Concretely, we detail the notations used in this work in Section 2.1. In Section 2.2 we review KGs and canonical machine learning tasks on KGs. We proceed with a short introduction to representation learning on graphs in Section 2.3. In Section 2.4 we review path-based reasoning methods on KGs that bridge the gap between rule-based and embedding methods on KGs. The remaining chapters contain our published works: Chapter 3 contains our publication on debate dynamics for explainable KG reasoning [26]. In Chapter 4 we present our publication on a novel method for scene graph reasoning [25]. Chapter 5 and 6 contain our published work on recommender systems for industrial automation solutions [23, 24]. We conclude in Chapter 7 and sketch directions for future works.

Chapter 2

Background

2.1 Notation

Before proceeding, we first define the mathematical notation that we use throughout this thesis.

Sets are either indicated by their canonical symbols (e.g., $\mathbb{N}$ denotes the set of natural numbers) or by calligraphic letters (e.g., the vertex set of a graph is given by $\mathcal{V}$, the edge set by $\mathcal{E}$). Scalars ($x \in \mathbb{R}$) and elements of sets (e.g., $v \in \mathcal{V}$) are given by lower case letters. In the context of this thesis, unless it is clear from the context or stated otherwise, sets are assumed to be finite and indexed (i.e., for any set $\mathcal{X}$, there exists a function $\mathcal{I} : \mathcal{X} \to \mathbb{N}$, that maps each element of $\mathcal{X}$ to an index). Column vectors are indicated by bold lower case letters ($\mathbf{x} \in \mathbb{R}^n$), and matrices by upper case letters ($X \in \mathbb{R}^{n_1 \times n_2}$). Further, $X_{i,:} \in \mathbb{R}^{n_2}$ and $X_{:,j} \in \mathbb{R}^{n_1}$ denote the i-th row and j-th column of X, respectively. Moreover, $X^\intercal$ denotes the transpose of X. To ease the notation, we identify elements of a set with their indices (i.e., $\{x_1, x_2, \ldots, x_n\} \cong \{1, 2, \ldots, n\}$). That means if X is a matrix that contains embeddings for all vertices in a graph in its rows and v corresponds to the i-th node (i.e., $\mathcal{I}(v) = i$), then we use $X_{v,:}$ or $X_i, :$ interchangeably to denote the embedding for v_i. If there is no ambiguity, we denote with $\mathbf{v}$ the vector space embedding of v. Furthermore, $\operatorname{diag}(x) \in \mathbb{R}^{n \times n}$ turns $x \in \mathbb{R}^n$ into the diagonal matrix with the entries of x on the main diagonal. We denote with $\mathbb{1}_i \in \mathbb{R}^n$ the one-hot vector that has zero entries everywhere except for a one at the i-th position.

For the purpose of this thesis a tensor is a high-dimensional generalization of a matrix that is used to represent collections of multi-relational, pairwise interactions. Third-order tensors are given by bold upper case letters ($\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$). Slices of a third-order tensor (i.e., two-dimensional sections obtained by fixing one index) are denoted by $\mathbf{X}_{i,:,:} \in \mathbb{R}^{n_2 \times n_3}$, $\mathbf{X}_{:,j,:} \in \mathbb{R}^{n_1 \times n_3}$, and $\mathbf{X}_{:,:,k} \in \mathbb{R}^{n_1 \times n_2}$, respectively.

Moreover, $\|\cdot\|_p : \mathbb{R}^n \to R_{\geq 0}$ denotes the p-norm given by $\|x\|_p = (\sum_{i=1}^{n} |x_i|^p)^{\frac{1}{p}}$. In particular, $\|\cdot\|_2$ denotes the Euclidean norm. The Frobenius norm $\|\cdot\|_F : \mathbb{R}^{n_1 \times n_2} \to R_{\geq 0}$ is given by $\|X\|_F = \sqrt{\sum_{i=1}^{n_1} \sum_{j=1}^{n_2} X_{i,j}^2}$.

Let $\mathcal{X}, \mathcal{Y}$, and $\mathcal{Z}$ denote generic sets and consider the functions $f : \mathcal{X} \to \mathcal{Y}$ and $g : \mathcal{Y} \to \mathcal{Z}$. Then the function $f \circ g : \mathcal{X} \to \mathcal{Z}$ denotes the composition of f and g (i.e., $f \circ g\,(x) = f(g(x)), \forall x \in \mathcal{X}$). In addition, $f \otimes g$ denotes the mapping given by $f \otimes g\,(x, y) = (f(x), g(y))$ for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$.

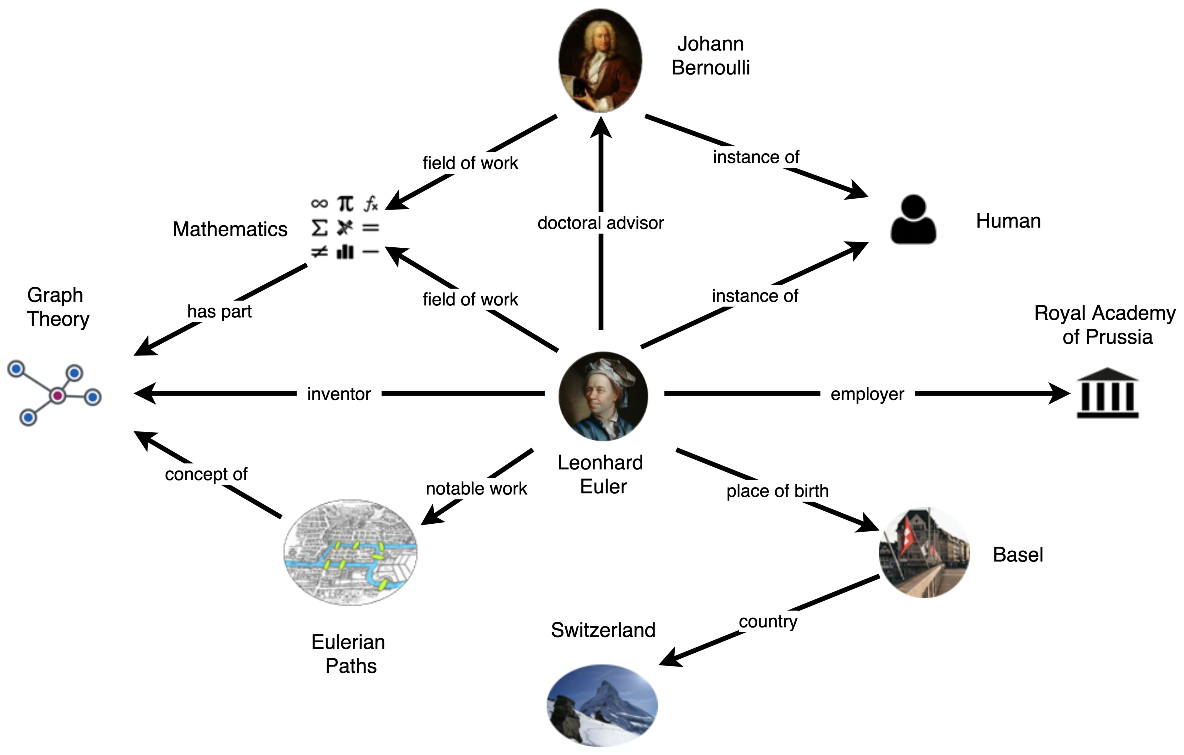

Figure 2.1: Excerpt of the graph neighborhood of Leonhard Euler in Wikidata [51].

2.2 Knowledge Graphs

The idea to store and convey knowledge in a graphical format holds a long tradition in artificial intelligence research. Richens [41] proposed a machine-readable semantic network in the 1950s where a graph serves as an auxiliary, intermediate concept language for machine translation. The idea was first to construct a graph representation of the sentence in the source language, where nodes correspond to the entities in the sentence and the edges to their relations. Then, based on this graph, the sentence in the target language can be built. The term knowledge graph itself was later introduced in the 1970s to describe communication flow in systems analysis [44, 28]. However, only after the presentation of the Google Knowledge Graph in 2012 [45], when Gooogle announced an initiative to store and semantically annotate a significant amount of human knowledge in a graphical database, the term KG has gained tremendous attention. The goal of the Google Knowledge Graph was to build a structured model of the world that can enrich the results of Google's search engine and dialogue systems in smart assistants such as

Google Home [17].

After Google's announcement, numerous companies and researchers referred to the Google Knowledge Graph despite the absence of official documentation. The modern usage of the phrase KG is often coupled with the vision of the Semantic Web [5] and Linked Open Data [4], which inspired the compilation of large scale, cross-domain knowledge bases that were rebranded as KGs. However, up to this day, there is still no precise, widely accepted definition of the term. While many authors stress the necessity of a formal ontology and well-defined semantics, other authors refer to any graph-based knowledge representation as a KG (see [40] for a discussion of the term). We adopt a pragmatic and inclusive definition popular in the machine learning community (see [30, 9]). In particular, for our purpose, a KG stores factual information as a list of triples. This leads to the following definition that we use throughout this thesis.

Definition 1 (Knowledge Graph).
Let $\mathcal{E}$ and $\mathcal{R}$ denote the set of entities and the set of binary relations defined on $\mathcal{E}$, respectively. Thereby, an entity in $\mathcal{E}$ may correspond to an instances or a class (i.e., a group of instances). A KG is given by $\mathcal{KG} \subset \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, hence, a collection of facts stored as triples of the form (s, p, o) – where s denotes the subject entity, p the predicate relation, and o the object entity.

As specified in the definition above, a list of triples has a natural multi-relational, directed graph representation: The entities correspond to nodes and directed edges represent the relations. The direction of an edge indicates the position of the entities in a triple. Concretely, the subject entity becomes the source node and the object entity the target node. The edge type is given by the predicate relation between the source and the target node. This data model is sometimes also referred to as a heterogeneous information network. A small KG is shown in Figure 2.1. For example, consider the entities *Leonhard Euler*, *Basel* $\in \mathcal{E}$ and the relation *place of birth* $\in \mathcal{R}$. The triple (*Leonhard Euler, place of birth, Basel*) corresponds to the fact that Leonhard Euler was born in Basel. Many real-world KGs come with a set of edge constraints specified in an ontology. For instance, a constraint in an ontology could state that a link of type *place of birth* must connect a being with a location.

To indicate whether triples are true or false, we consider the characteristic function $\phi : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \to \{0, 1\}$, where the function value 1 indicates that the triple is true; 0 that the triple is false. For any relation $p \in \mathcal{R}$ we denote with p^{-1} the corresponding inverse relation (i.e., $\phi(s, p, o) = 1 \Leftrightarrow \phi(o, p^{-1}, s) = 1$). For example, the triple (*Mathematics*, *has part*, *Graph Theory*) is equivalent to (*Graph Theory*, *is part of*, *Mathematics*) because the relations *has part* and *is part of* are mutually inverse. Moreover, for all $(s, p, o) \in \mathcal{KG}$, we assume $\phi(s, p, o) = 1$. That means a KG is interpreted as a collection of true facts. However, there exist different interpretations of the absence of triples. For example, the closed world assumption (CWA) implies that triples which are not contained in $\mathcal{KG}$ are interpreted as false facts. Alternatively, as specified by the Resource Description Framework (RDF) [38], under the open world assumption (OWA) it is assumed that non-observed facts are interpreted as unknown. Another perspective that is frequently adopted in machine learning approaches is the local closed-world assumption. Thereby, a KG is assumed to be only locally complete in the sense that $(s, p, o) \notin \mathcal{KG}$ is assumed to be false if and only if there exists an entity $\tilde{o} \in \mathcal{E}$ such that $(s, p, \tilde{o}) \in \mathcal{KG}$. Otherwise, the triple is interpreted as unknown.

Existing large scale KGs are constructed and maintained in different ways. For example, facts in DBpedia [2] are mostly automatically extracted from structured content from Wikipedia; NELL [11] is constructed by agents that automatically scrape and process unstructured data from more than 500 million websites. Other KGs such as Wikidata [51] and Freebase [6] are mainly community-built and curated by the crowd. No matter what method is used to build and maintain a KG, in practice, most KGs suffer from incomplete coverage in the sense that true triples in the scope of the KG (i.e., facts that can be expressed by the given set of entities and relations) are not included. Moreover, KGs often exhibit incomplete correctness in the sense that triples are contained in the KG that correspond to false facts. Therefore, canonical machine learning methods are concerned with automatic knowledge curation and refinement with the goal to infer missing facts based on observed connectivity patterns (KG completion or link prediction) or predict the truth value of triples (triple classification). While the formulations of these two tasks can differ in the literature, we consider the following definitions throughout this thesis.

Definition 2 (Triple Classification and KG completion).
Triple classification is concerned with predicting the truth value $\phi(s, p, o)$ for $(s, p, o) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. The task of KG completion is to rank entities $e \in \mathcal{E}$ or relations $r \in \mathcal{R}$ by their likelihood to form true triple given either a subject-predicate-pair $(s, p) \in \mathcal{E} \times \mathcal{R}$, a predicate-object-pair $(p, o) \in \mathcal{R} \times \mathcal{E}$, or a subject-object-pair $(s, o) \in \mathcal{E} \times \mathcal{E}$.

Hence, according to our definition, predicting the truth value of the triple *(Leonhard Euler, country of citizenship, Old Swiss Confederacy)* $\in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is a triple classification task. Given the input pair *(Leonhard Euler, country of citizenship)* $\in \mathcal{E} \times \mathcal{R}$, ranking the entities *Old Swiss Confederacy, Russian Empire* and *Kingdom of Prussia* (i.e., the nationalities that Euler held over his lifetime) high among all possible entities is a KG completion task. Next to curating and extending existing KGs, plenty of AI tasks such as question answering [36] or visual relation detection [3] can be framed as KG completion or triple classification. In Chapter 5, we consider a real-world setting where the recommendation task can be formulated as predicting links in a KG setting.

2.3 Representation Learning on Graphs

In this section, we first discuss representation learning on homogeneous graphs that contain only one type of nodes and edges. Thereby, we focus on the problem of finding embeddings for individual nodes. As a preparation for the following chapters, we review matrix factorization techniques, graph autoencoders, and graph neural networks. These techniques play an essential role in all of our published work. Then we proceed with a review of popular embedding methods for KGs. This second part focuses on tensor factorization methods for knowledge graphs, which are the foundation of our work on recommender systems in Chapter 5 and 6.

2.3.1 Homogeneous Graphs

Consider the task of learning embeddings for nodes in a graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_{|\mathcal{V}|}\}$ denotes the vertex set and $\mathcal{E}$ the edge set. A graph is said to be directed if every edge comes with an orientation in the sense that it connects a source node to a target. In that case, $\mathcal{E}$ consists of ordered pairs of vertices $(v, w) \in \mathcal{V}^2$, where v denotes the source and w the target node. In the absence of such an orientation, the edge set is given by a collection of two-sets $\{v, w\} \subset \mathcal{V}$. In that case, the graph is called undirected. To ease the notation and not having to distinguish between edges with and without orientation, we assume that graphs in this section are directed. While there are learning methods that are only applicable to undirected graphs, this assumption is not restrictive because the methods that are considered in this section can operate on both types of graphs. The presented formulas translate to an undirected setting by using the fact that undirected graphs can be regarded as directed graphs with $(v, w) \in \mathcal{E}$ if and only if $(w, v) \in \mathcal{E}$. $A \in \{0, 1\}^{|\mathcal{V}| \times |\mathcal{V}|}$ denotes the adjacency matrix where $A_{i,j} = 1$ if $(v_i, v_j) \in \mathcal{E}$ and 0 otherwise. For the moment, we assume that the graph does not come with any auxiliary features such as node attributes. The problem of embedding the nodes in G can be defined as follows.

Definition 3 (Node Embeddings).
Given a graph $G = (\mathcal{V}, \mathcal{E})$, the problem of learning node embeddings is concerned with finding a mapping that associates each node $v \in \mathcal{V}$ with its vector representation $\mathbf{v} \in \mathbb{R}^d$ where $d \ll |\mathcal{V}|$.

During training, the embedding mapping is tuned such that the embedding map condenses the relevant information about the graph structures surrounding the nodes to conduct downstream tasks such as link prediction, node classification, or node clustering. Following [20], we analyze the problem of computing node embeddings in a task-agnostic encoder-decoder framework. From this perspective, a node embedding model is composed of two mappings: The first mapping is called the encoder. It is an embedding mapping according to Definition 3. The goal is to encode structural properties of nodes in the embedding space. Subsequently, the embedding is passed to the decoder, which is trained to unfold relevant information from the embedding space. This leads to the following definition.

Definition 4 (Encoder, Decoder, and Node Embedding Model).
An encoder is an embedding mapping

$$Enc : \mathcal{V} \to \mathbb{R}^d \,. \tag{2.1}$$

A decoder takes as input the embedding of two nodes and maps it to a real-valued output

$$Dec : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R} \,. \tag{2.2}$$

The mapping $Dec \circ (Enc \otimes Enc)$ is called a node embedding model.

Typically, the goal is to tune the trainable parameters of an embedding model such that it approximates some node proximity measure. Concretely, that means

$$\mathrm{Dec}\left(\mathrm{Enc}(v_i), \mathrm{Enc}(v_j)\right) \approx S_{i,j} \,, \tag{2.3}$$

where $S \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ is called node proximity matrix. Equation 2.3 resembles a self-supervised learning setting since the node proximity matrix constitutes a supervision signal that is automatically generated from the graph data. A common node proximity matrix is given by the adjacency matrix. In that case, neighboring nodes are enforced to be close in the embedding space preserving so-called first-order proximity. Among others, this proximity measure is a common choice for link prediction tasks since the decoder's output is a proxy for the edge reconstruction probability. Considering higher powers of the adjacency matrix as similarity measure induces a higher-order proximity measure that captures long-range dependencies [30].

Many pioneering works in graph learning belong to the class of matrix decompositions. These techniques can be roughly divided into Graph Laplacian eigenmaps and inner-product methods. We focus on the latter because they play an essential role in Chapter 5. Inner-product methods employ a parameter free decoder consisting of a dot product. That means the objective function is typically of the form

$$\min_{V \in \mathbb{R}^{|V| \times d}} \|S - VV^\intercal\|_F^2 = \sum_{i,j} \|S_{i,j} - \mathbf{v}_i^\intercal \mathbf{v}_j\|_2^2 \,, \tag{2.4}$$

where $V \in \mathbb{R}^{|V| \times d}$ is an embedding matrix and the rows of V contain embeddings for the corresponding nodes. That means the embeddings are obtained via a simple lookup

$$\mathrm{Enc}(v_i) = \mathbf{v}_i = V^\intercal \mathbb{1}_i \,. \tag{2.5}$$

The differences between existing inner product methods mainly lie in the formulations of the loss functions and regularizations imposed on the optimization problem in Equation (2.4) to enforce additional constraints. These modifications include non-negativity constraints [56], regularized Gaussian matrix factorization [1], or the deployment of max-margin losses [49]. Inner-product methods are still among the most frequently used techniques in representation learning in graphs. However, they do suffer from a set of short-coming: First, parallelization of inner product methods is non-trivial since the embeddings of the nodes require frequent synchronization. Consequently, without taking additional measures (e.g., graph partitioning), inner-product methods often face challenges scaling to massive graphs with billions of nodes and edges. Moreover, it is not straightforward to incorporate and encode context information in the embedding space. Another shortcoming is that the shallow encoder does not employ a weight sharing mechanism which is one of the most effective regularization techniques in machine learning (see [20] for a discussion).

In contrast to shallow embeddings that employ a simple embedding lookup (Equation 2.5), deep learning techniques allow to build expressive encoders that can leverage node features. Neural networks have shown impressive performance in various disciplines, such as computer vision and NLP. Substantial efforts have been devoted to transfer deep learning techniques to graphs efficiently. One major difficulty is that, in contrast to image, video, or text data, graph data is not sampled on a regular grid. For example, the variable number of nodes in a graph (or in a given neighborhood of a vertex) and the absence of a straightforward up- or downsampling operation imposes a challenge for feedforward neural network architectures that consider vectors with fixed dimensionality as input. Moreover, the lag of a natural order of the vertices prohibits the straightforward use of recurrent neural networks that process the input signal in a sequential manner. Similarly, convolutional neural networks cannot be applied directly to the graph domain because sliding a fixed filter mask over the input data requires a directional structure that allows to define a shift operator. Nevertheless, there exists neural network models for graph learning that aim to directly apply existing network architectures to graphs. An example of such kind of adoptions is the class of graph autoencoders (GAEs). Autoencoders are a popular tool for unsupervised learning tasks consisting of two neural network components: An encoder and a decoder network. The encoder network is usually a dense neural network that computes a non-linear mapping of the input into a low dimensional space. The decoder mirrors the encoder's action in the sense that it takes the encoding of the input and aims to recover the original input. GAEs are a special case of autoencoders that take as input a row of the similarity matrix $S_i \in \mathbb{R}^{|\mathcal{V}|}$ and produce a d-dimensional encoding $\mathbf{v_i}$ of the corresponding node. The decoder network aims to recover S_i from $\mathbf{v_i}$. In other words, an encoder network $f_{\text{enc}} : \mathbb{R}^{|\mathcal{V}|} \to \mathbb{R}^d$ compresses the structural information contained in S_i into a low-dimensional vector, while the decoder aims to reconstructs S_i from that embedding via the neural network $f_{\text{dec}} : \mathbb{R}^d \to \mathbb{R}^{|\mathcal{V}|}$. The GAE is given by

$$f = f_{\text{enc}} \circ f_{\text{dec}} \,. \tag{2.6}$$

The parameters of f_{enc} and f_{dec} are trained jointly to reconstruct the rows of some similarity matrix. Wang et al. [52] propose the structure deep network embedding (SDNE), which employs the adjacency matrix A as similarity measure and an additional regularization term that enforces neighboring nodes to have similar embeddings. Cao et al. [10] employ positive pointwise mutual information (PPMI) as similarity matrix. The entries of this matrix corresponds to a probabilistic co-occurrence measure based on random walks that captures structural information. To incorporate node attributes, Tran [47] considers a vector of node features along with the similarity matrix as input to the GAE. All of these methods have in common that the input dimension is at least the number of nodes. That can be extremely costly in terms of time and space complexity such that GAEs

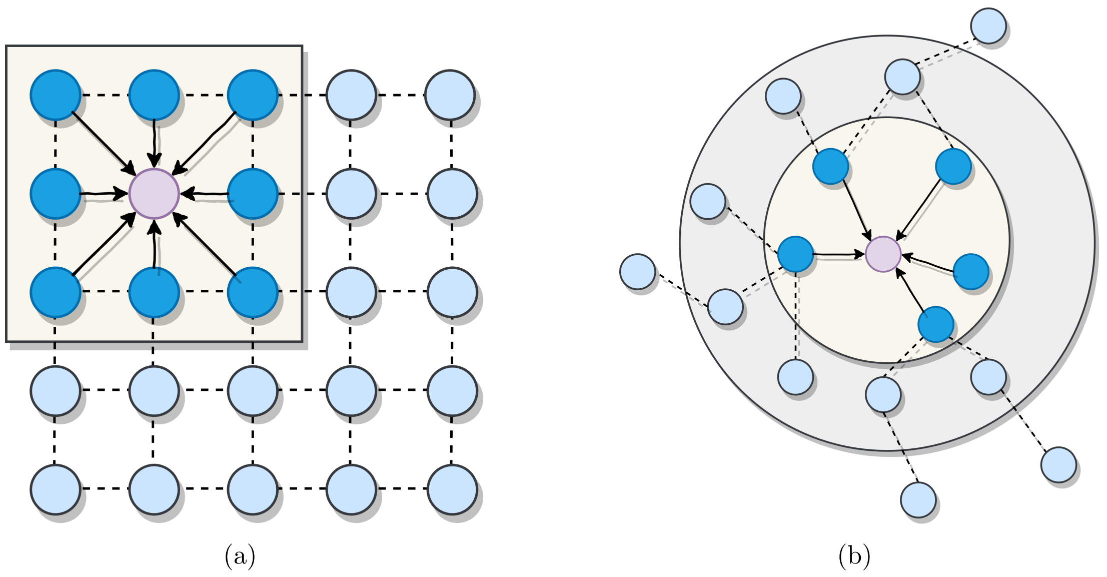

Figure 2.2: In analogy to a CNN on a regular grid (a), GNNs (b) aggregate features from its neighboring nodes. The inner circle indicates the receptive field with one layer that corresponds to the one-hop neighborhood. The outer circle indicates the receptive field after adding an additional layer corresponding to the two-hop neighborhood.

may not scale to large graphs. Moreover, since the autoencoder requires an input vector with constant dimensionality where every entry corresponds to a fixed vertex, GAEs are bound to the graphs they are trained on (up to isomorphisms) and cannot be transferred to other unseen graphs or nodes.

Convolutions on regular grids are the basic building blocks of convolutional neural networks (CNNs) [35]. CNNs extract localized spatial features by sliding trainable, but invariant filter masks over regular grids. This imposes not only location invariance but also constitutes a parameter sharing mechanism that acts as a powerful regularizer. The extracted features are subsequently combined by feeding them through multiple layers to build new expressive representations. Motivated by the achievements of CNNs in computer vision, different graph neural networks (GNNs) were developed in parallel that aim to transfer convolutional operators to the graph domain. These attempts resulted in two different frameworks: One branch of methods perform graph Fourier transforms to employ trainable filters in the spectral domain (such as in [8] or [14]). The other conceptually simpler line of research is based on spatial convolutions and employs a message-passing heuristic between neighboring nodes (e.g., [42]). One of the most influential methods is the graph convolutional network (GCN) introduced in [31]. GCNs are a unification of both spectral and spatial methods in the sense that they can be derived as a special, simplified case of both the message passing GNN proposed in [42] and the spectral method ChebNet from [14]. Subsequently, a variety of extensions to GCNs were proposed. For example, [12] propose FastGCN that allows for efficient training on massive graphs, [50] add attention mechanisms to GCNs to increase the expressivity leading to graph attention networks (GAT), and GraphSage [19] can operate in the inductive setting (i.e., nodes can be embedded during deployment which are not seen during training).

In what follows, we review GNNs that fit into the framework of spatial message passing [18]. Thereby, an embedding $\mathbf{v} \in \mathbb{R}^d$ for a node $v \in \mathcal{V}$ is formed by first aggregating node features from the neighborhood of v denoted by $\mathcal{N}_v := \{w \in \mathcal{V} | (v, w) \in \mathcal{E}\}$. The current node of interest v is usually called the center node. In contrast to shallow encoders that consist of a simple lookup function (see Equation 2.5), GNNs can naturally leverage auxiliary node features. Concretely, it is assumed that every node v comes with a set of initial features denoted by $\mathbf{x}_v \in \mathbb{R}^F$. These initial features may contain additional context information that describes the entity corresponding to the node (e.g., demographic information in a social network or technical attributes of an item in a recommender system), embeddings obtained from a different data modality (e.g., word embeddings from a textual description of the node), or generic graph features such as the node degree, centrality measures, and clustering coefficients. In a first step, these features from the nodes adjacent to the center node $v \in \mathcal{V}$ are aggregated

$$\mathbf{h}_{\mathcal{N}_v} = f_{\text{aggregate}} \left(\{\mathbf{x}_w | w \in \mathcal{N}_v\} \right) , \tag{2.7}$$

where $f_{\text{aggregate}}$ is a trainable function shared by all nodes that aggregates the features from its neighbors. In general, $f_{\text{aggregate}}$ operates on an unordered set of vectors with variable size. Hence, $f_{\text{aggregate}}$ typically involves a commutative pooling operation with constant output dimension such as summation, averaging or max-pooling. Subsequently, the aggregated feature representation $\mathbf{h}_{\mathcal{N}_v}$ is combined with the input features of the center node to obtain an embedding $\mathbf{h}_v \in \mathbb{R}^d$. Concretely, this leads to

$$\mathbf{h}_v = f_{\text{combine}} \left(\mathbf{x}_v, \mathbf{h}_{\mathcal{N}_v} \right) . \tag{2.8}$$

The successive application of $f_{\text{aggregate}}$ and f_{combine} in Equations (2.7) and (2.8) defines one layer of the encoder network. Note that in Equations (2.7), each center node aggregates information from its immediate neighbors. Information beyond this one-hop neighborhood is discarded in the aggregation step. Thus, the so-called receptive field is equal to the one-hop neighborhood of a each node. Multiple layers of GNNs can be stacked on top of each other to increase both the expressivity and the receptive field. In particular, in a GNN with L layers each center node receives information from the nodes L hops away, broadening the receptive field to the L-hop neighborhood (see Figure 2.2). The trainable parameters in Equations (2.7) and (2.8) are typically tuned via backpropagation to minimize a task-specific loss. However, due to the modularity of encoder-decoder architectures, GNNs can also be combined with a generic decoder and trained in an unsupervised setting.

Various GNNs can be distinguished according to the formulation of $f_{\text{aggregate}}$ and f_{combine}. For example, the GCN is given by

$$\mathbf{h}_{\mathcal{N}_v} = \sum_{w \in \mathcal{N}_v} \frac{1}{\sqrt{d_v d_w}} W \mathbf{x}_w \tag{2.9}$$

and

$$\mathbf{h}_v = \sigma\left(\frac{1}{d_v} W \mathbf{x}_v + \mathbf{h}_{\mathcal{N}_v}\right), \tag{2.10}$$

where $W \in \mathbb{R}^{d \times F}$ is a trainable weight matrix, $d_w := |\mathcal{N}_w|$ denotes the degree of w, and σ denotes a non-linear activation function. More compactly, a GCN with one layer is given by

$$\text{Enc}_{\text{GCN}}(v) = \sum_{w \in \mathcal{N}_v \cup \{v\}} \frac{1}{\sqrt{d_v d_w}} W \mathbf{x}_w \,. \tag{2.11}$$

Note that when computing the embedding for a node according to Equation 2.11, the same weight matrix W is applied to all adjacent nodes. However, it may be the case that certain neighbors carry more useful information than others. Based on this insight, Veličković et al. [50] proposed the graph attention network (GAT), which employs a multi-head attention mechanism. Concretely, an adaptive attention weight is assigned to a neighboring node depending on the current embedding of the center node and the neighboring node. For two adjacent nodes $v, w \in \mathcal{V}$ the corresponding attention weight (up to normalization) for the k-th head is given by

$$\alpha_{v,w}^{(k)} = a^k\left(W_a^k \mathbf{x}_v, W_a^k \mathbf{x}_w\right), \tag{2.12}$$

where $a^k : \mathbb{R}^F \times \mathbb{R}^F \to \mathbb{R}$ is the attention mechanism given by a single-layer feedforward neural network. Combining the aggregation and combination step, the encoder of a one-layered GAT is given by

$$\mathrm{Enc}_{\mathrm{GAT}}(v) = \left[\sigma \left(\sum_{w \in \mathcal{N}_v} \alpha_{v,w}^{(k)} W^{(k)} \mathbf{x}_w \right) \right]_{k=1}^{K} , \tag{2.13}$$

where $[\cdot]_{k=1}^{K}$ denotes the concatenation of K vectors. The GCN's aggregation and combination step can be thought of as a low-pass filter on graph signals. Thus, using multiple layers of a GCN may smooth over informative signals on a densely connected graph. The GAT's attention mechanism can be seen as a remedy to this problem allowing to use multiple layers and increasing the receptive field and extract more expressive features.

We make use of GATs in Chapter 4 where we consider the task of visual question answering. In that context, the purpose of using a GAT is to compute context-aware node embeddings in a scene graph. Thereby, the nodes correspond to detected object and the initial node features are given by word embeddings of the predicted classes. We employ GATs to produce embeddings for the depicted objects that pool information from neighboring objects to facilitate path-based reasoning on the scene graph.

2.3.2 Knowledge Graph Embeddings

Machine learning methods for KG reasoning are studied under the umbrella of statistical relational learning (SRL) [38]. Similar to the graph learning methods discussed in the previous section, KG embeddings have become the dominant approach for AI applications on KGs. The underlying idea is that features that explain the connectivity pattern between entities can be encoded in low-dimensional vector spaces. In the embedding spaces the interactions between the embeddings of entities and relations can be efficiently modelled to produce scores that predict the validity of a triple. Moreover, since the learned embeddings also contain rich semantic information similar to word embeddings, the embeddings can also be used for clustering or KG visualization tasks. In analogy to homogeneous graphs in the previous section, we can define KG embeddings as follows.

Definition 5 (KG Embeddings).
Given a knowledge graph $\mathcal{KG} \subset \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, the problem of learning embeddings is concerned

with finding mappings that associate each entity $e \in \mathcal{E}$ with a vector in $\mathbb{R}^{d_\mathcal{E}}$ and each relation $r \in \mathcal{R}$ with a vector in $\mathbb{R}^{d_\mathcal{R}}$.

In order for this definition to subsume all KG embeddings models that we consider in this thesis, we assume that $\mathbb{R}^{m\times n} \cong \mathbb{R}^{mn}$, where a matrix in $\mathbb{R}^{m\times n}$ is identified with a vector in $\mathbb{R}^{mn}$ via the natural linear isomorphism (i.e., via reshaping). Most KG embedding methods can be analyzed within the encoder-decoder-framework. This leads to the following definition.

Definition 6 (Encoder, Decoder, and Node Embedding Model).
KG encoders are mappings from the symbolic objects in $\mathcal{E}$ and $\mathcal{R}$ to low-dimensional vector spaces

$$Enc_\mathcal{E} : \mathcal{E} \to \mathbb{R}^{d_\mathcal{E}} \quad \text{and} \quad Enc_\mathcal{R} : \mathcal{R} \to \mathbb{R}^{d_\mathcal{R}}\,. \tag{2.14}$$

A KG decoder is given by

$$Dec : \mathbb{R}^{d_\mathcal{E}} \times \mathbb{R}^{d_\mathcal{R}} \times \mathbb{R}^{d_\mathcal{E}} \to \mathbb{R}\,. \tag{2.15}$$

Composing the encoders and the decoders $Dec \circ (Enc_\mathcal{E} \otimes Enc_\mathcal{R} \otimes Enc_\mathcal{E})$ leads to a KG embedding model. The encoders and the decoder can be trained jointly such that their decomposition approximates a similarity measure $\mathcal{S}_{\mathcal{KG}} : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \to \mathbb{R}$ leading to

$$Dec\,(Enc_\mathcal{E}(s), Enc_\mathcal{R}(p), Enc_\mathcal{E}(o)) \approx \mathcal{S}_{\mathcal{KG}}(s,p,o)\,. \tag{2.16}$$

Usually, the embeddings are initialized randomly and updated by solving the optimization problem induced by the similarity measure. Since most KG embedding methods are tuned for the KG completion task, the similarity measure scores the plausibility of triples by approximating the characteristic function ϕ. Ideally, this scoring function assigns high values to missing triples, which are actually true and low values to false triples. Hence, it can be used to discover unknown but true facts or detect false positives.

While there are attempts that generalize GNNs to multi-relational data [43], most KG embedding models to this day employ a simple embedding lookup as encoder. That means

$$\mathrm{Enc}_\mathcal{E}(e_i) = \mathbf{E}^\intercal \mathbb{1}_i\,. \tag{2.17}$$

and

$$\mathrm{Enc}_{\mathcal{E}}(r_j) = \mathbf{R}^{\intercal}\mathbb{1}_i \,. \tag{2.18}$$

where **E** and **R** are embeddings matrices for the entities and the relations, respectively. In that case, the decoder distinguishes different KG embedding methods and determines how the embeddings are combined to score an input triple. Most methods can be categorized according to the decoder's interaction mechanisms in multiplicative, additive, and neural network models. Additive models such as TransE [7] project both entities and relations into the same vector space and interpret relations as translations between in the embedding space. An early neural network-based approach is the neural tensor network (NTN) introduced in [46], which combines both tensor products and neural networks to model the interaction between entities. However, simple neural network architectures such as ER-MLP [16] that concatenate shallow embedding vectors of a triple and pass it through a dense neural network are more parameter efficient but achieve similar performance. More recently, ConvE [15], which learns 2D-convolutional networks over the embedding spaces to obtain feature maps, achieved state-of-the-art-approach. After the convolution operation, the latent features are fed through a linear layer followed by a bilinear product with all the candidate object entities.

In this thesis, we focus on multiplicative scoring functions that are based on tensor factorizations. A knowledge graph has a natural representation in terms of a binary, three-way adjacency tensor $\mathbf{X} \in \{0,1\}^{n_E \times n_E \times n_R}$. Each entry of $\mathbf{X}$ indicates the abscence or the presence of a triple: 1 corresponds to an observed triple and 0 to an unobserved one. For most tensor factorizations, $\mathbf{X}$ serves directly as similarity measure (i.e., $\mathcal{S}_{\mathcal{KG}}(s,p,o) = \mathbf{X}_{s,o,p}$). The rationale is to compute a low-rank decomposition of this tensor by associating an embedding vector to each entity and a matrix to each relation and compute a bi-linear form induced by the relation matrix with the two entities as input.

An early tensor factorization, the canonical polyadic decomposition (CANDECOMP), was proposed in the 1920s [27]. CANDECOMP was later rediscovered by Harshman [22] and popularized under the name PARAFAC. That is why CANDECOMP/PARAFAC is often abbreviated with CP. When directly applied to KGs, CP computes one embedding for each relation and two embedding vectors for each entity. Thereby, one embedding represents the entity when it appears as subject in a triple and the other is the object representation. The decoupling of subject and object embedding leads to poor performances when CP is applied directly to KG reasoning tasks (see for example the results

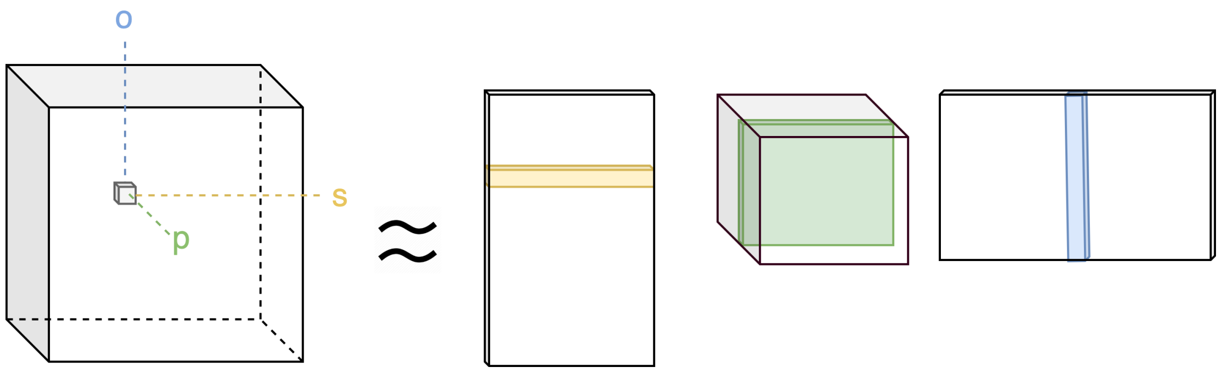

Figure 2.3: A visualization of the low-rank tensor decomposition performed by RESCAL (depiction based on a figure in [33]).

in [48]). In particular, the absence of a weight sharing mechanism hinders the flow of information between triples where the same entity appears as subject in different triples.

RESCAL was proposed in [37] and, in contrast to CP, it employs the same embeddings for entities no matter at what position of a triple they appear. RESCAL is a special case of the Tucker decomposition (see [32] for more details) which factorizes the adjacency tensor into a core tensor multiplied twice by the same factor matrices along the first and the third mode. Concretely, this leads to

$$\mathrm{Dec}(\mathrm{Enc}(s), \mathrm{Enc}(p), \mathrm{Enc}(o)) = \mathbf{s}^\intercal \mathbf{R}_{:,:,p}\mathbf{o}. \tag{2.19}$$

with $\mathbf{R} \in \mathbb{R}^{d_\mathcal{E} \times d_\mathcal{E} \times d_\mathcal{R}}$. In particular, this means that given a query relation $p \in \mathcal{R}$, the scoring function is given by a bilinear form induced by $\mathbf{R}_{:,:,p}$. Hence, the connectivity prediction between a pair of entities is channeled through the core tensor where each frontal slice contains an embedding of a different relation leading to $d_\mathcal{R} = d_\mathcal{E}^2$. This leads to one of the disadvantage of RESCAL: The number of trainable parameters grows quadratically in the latent dimension $d \in \mathbb{N}$ as the number of relations increases. As a remedy to this problem the core tensor of DistMult [55] is constrained to be diagonal. Concretely, the decoder of DistMult is given by

$$\mathrm{Dec}(\mathrm{Enc}(s), \mathrm{Enc}(p), \mathrm{Enc}(o)) = \sum_{i=1}^{d} \mathbf{s}_i \mathbf{p}_i \mathbf{o}_i = \mathbf{s}^\intercal \mathrm{diag}(\mathbf{p})\mathbf{o}\,, \tag{2.20}$$

where $\mathbf{s}, \mathbf{p}, \mathbf{o} \in \mathbb{R}^d$ and $\mathrm{diag}(\mathbf{p})$ is diagonal matrix with diagonal entries given by $\mathbf{p}$. While this lowers the burden on the number of trainable parameters, the scoring function of DistMult is symmetric and asymmetric relations cannot be modelled. The factorization method ComplEx [48] offers a solution to this limitation by extending DistMult over the field of complex numbers. More concretely, by imposing that subject and object embeddings of the same entities are complex conjugates, ComplEx is able to break the symmetry of DistMult and allows to take asymmetric relations into account. The decoder is given by

$$\mathrm{Dec}(\mathrm{Enc}(s), \mathrm{Enc}(p), \mathrm{Enc}(o)) = \sum_{i=1}^{d} \mathrm{real}\left(\mathbf{s}_i \bar{\mathbf{p}}_i \bar{\mathbf{o}}_i\right) = \mathrm{real}\left(\mathbf{s}^\intercal \mathrm{diag}(\bar{\mathbf{p}}) \bar{\mathbf{o}}\right) , \qquad (2.21)$$

where $\mathbf{s}, \mathbf{p}, \mathbf{o} \in \mathbb{C}^d$ and $\bar{\ } : \mathbb{C} \to \mathbb{C}$ denotes the complex conjugate.

In our published work, we employ tensor factorization approaches to perform the recommendation task in a KG setting. In particular, in Chapter 5, we develop a context-aware recommender engine based on RESCAL and test it with respect to its real-world applicability in a specific industrial use case.

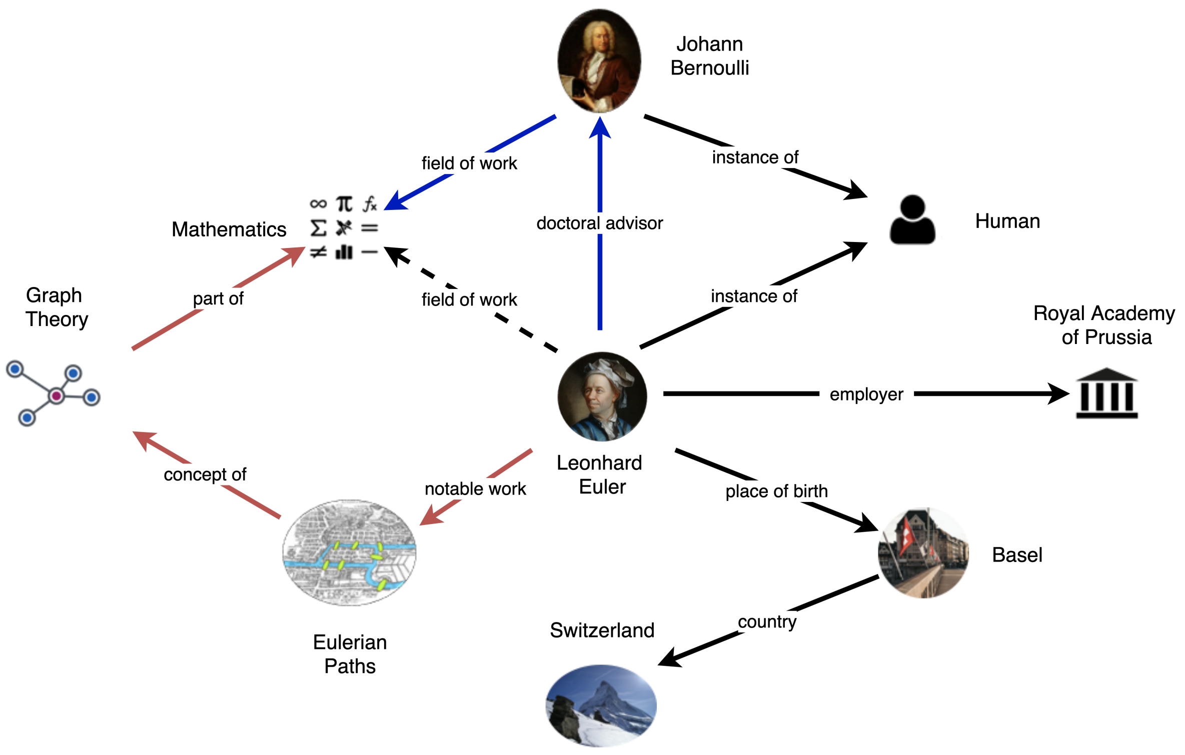

Figure 2.4: When predicting the *field of work* of Leonhard Euler (i.e., the dashed arrow), the two paths highlighted with red and blue arrows can serve as explainable features.

2.4 Path-based Reasoning for Knowledge Graphs

Despite achieving good results in KG reasoning tasks, a fundamental problem with many embedding based methods that fit into the encoder-decoder framework is their nontransparent nature in the sense, that it remains hidden to the user what contributed to the model's prediction. Path-based reasoning methods (also known as multi-hop reasoning methods) follow a different philosophy than embedding-based methods. The underlying idea is to infer missing knowledge based on sequentially extended inference paths on the KG. Thereby, these methods have an inherent transparency mechanism by providing explicit reasoning chains that can be analyzed by the user. Moreover, path-based reasoning methods can naturally capture the compositionality expressed by long reasoning chains allowing them to perform complex reasoning tasks.

Path-based reasoning methods naturally connect machine learning methods with the rule mining and logical reasoning literature. Classical methods rely on applying explicit logical rules. That means, during test time, inference is conducted by plugging instance data into the learned formulas in order to infer missing knowledge (see [53, 39]). However, massive scales and diverse topologies of real-world KGs often result in combinatorial complexities that prevent using these symbolic approaches. Besides, symbolic methods tend to be sensitive to the presence of noisy, corrupted facts. In this thesis, we focus on a class of methods based on randomly sampled reasoning paths. More concretely, these methods rely on stochastic paths that correspond to trajectories of random walks.

The pioneering method Path Ranking Algorithm (PRA) proposed in [34] performs triple classification based on random walks that connect a subject entity and a candidate object entity. More concretely, for a given query triple $(s, p, o) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, PRA samples connecting paths on the KG from s to o. Each path is regarded as a feature and a weighted combination of all paths is processed by a relation specific elastic net to predict $\phi(s, p, o)$.

While PRA employs uniformly at random extracted paths on a KG between two entities, the pathfinding problem can also be posed in a reinforcement learning framework. More specifically, pathfinding can be framed as a sequential decision problem where subsequent transitions are added to the current inference path. This procedure can be modeled in terms of a Markov decision process (MDP).

Definition 7 (Markov decision process).
A discounted MDP consists of a quintuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$, *where* $\mathcal{S}$ *denotes the state space, the set* $\mathcal{A}$ *contains all available actions,* $\mathcal{P}\left(S_{t+1} \middle| S_t = s, A_t = a\right)$ *is called the transition probability defined over the state space conditioned on an action* a *and a state* s, *and* $\mathcal{R}(s, a) \in \mathbb{R}$ *defines the expected reward of the agent for every state-action pair.* $\gamma \in (0, 1]$ *is called the discount factor and balances rewards across time.*

Goal-directed path extraction on KGs can be naturally conducted via policy-guided random walks modeled in terms of MDPs. Thereby, the environment evolves deterministically in the sense that the transition probabilities (given an action and a state) $\mathcal{P}\left(S_{t+1} \middle| S_t = s, A_t = a\right)$ are degenerate point measures. In addition to information about the query, the state contains a representation of the agent's location (i.e., the entity where the agent is currently located). Given the current location, the available actions correspond to the set of outgoing edges along with their target entities. Roughly speaking, reinforcement learning methods can be categorized into value function and direct policy search techniques. Value function-based methods aim to estimate the long-term utility of decisions and derive a behavioural policy by selecting actions that lead to the highest utility estimate. Policy search methods directly fit a policy function that maps each state to an action distribution. For navigating over a KG, policy search methods have the advantage of exhibiting superior convergence behavior given the high-dimensional state and action spaces. Furthermore, policy-based methods naturally induce stochastic policies that balance between exploration and exploitation [54]. Thereby, the agent is represented by its policy $\pi(s,a) = \mathbb{P}(a|S_t = s)$ for $a \in \mathcal{A}$, and $s \in \mathcal{S}$, where $\mathbb{P}(\cdot|S_t = s)$ is a probability distribution over $\mathcal{A}$ from which the next action is sampled. The goal of the training process is to find a policy that maximizes the expected discounted, cumulative rewards of the agent given by

$$\mathbb{E}_{\mathcal{P}}\left[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t)\pi(S_t, A_t)\right] = \sum_{t=0}^{\infty} \sum_{S_t \in \mathcal{S}} \sum_{A_t \in \mathcal{A}} \gamma^t R(S_t, A_t)\pi(S_t, A_t)\mathcal{P}\left(S_{t+1}|\, S_t, A_t\right), \tag{2.22}$$

where $\mathbb{E}_{\mathcal{P}}$ denotes the expectation with respect to the transition probabilities $\mathcal{P}$.

Xiong et al. [54] extend the idea from PRA and frame the task of path extraction as a reinforcement learning problem. They propose Deeppath, which substitutes the nearest-neighbor random walk from PRA with a policy-guided random walk. Deeppath operates in the triple classification setting (see Definition 1). Concretely, a subject and an object entity are shown to the agent that has the goal to find multiple connecting paths between these two entities. Once the paths are extracted, they are fed into a relation specific classifier that predicts the truth value of the triple. However, this setup leads to scalability issues on large KGs, since a different classifier needs to be trained for every relation.

The path-based method MINERVA [13] overcomes this issue by combining the path-finding and prediction task in one reasoning module. In contrast to Deeppath, MINERVA operates in the link prediction setting (see Definition 1). That means a subject-predicate pair $(s,p) \in \mathcal{E} \times \mathcal{R}$ are presented to the agent. The agent's goal in MINERVA can be roughly described as follows: Start at the subject entity and walk to the correct answer. More formally, starting from the subject entity s, the objective of the agent is to extract a relational path in $\mathcal{KG}$ to an unknown object entity $o \in \mathcal{E}$ that forms a correct triple (i.e., $\phi(s,p,o) = 1$). After the agent has terminated its walk, a positive reward is assigned if the agent reaches the desired target entity. By maximizing these rewards, the agent's policy implicitly incorporates sequences of relations that allows to find correct object entities.

Hence, the policy embodies an inductive bias to encode structural rules.

Outline This section on path-based reasoning methods completes the background material of this thesis. The remainder of this work contains our publications. In the following two chapters, we consider two different settings that employ path-based reasoning using reinforcement learning: First, in Chapter 3, we propose the idea of debate dynamics where two agents take opposing positions and extract paths on a KG that serve as adversarial, sparse features for and against the validity of a query triple. Second, in Chapter 4, we develop a new scene graph reasoning module, where a reinforcement learning agent is trained to navigate on a scene graph to provide a conclusive reasoning path that allows answering questions about an image. Chapter 5 and 6 are centered around an industrial application concerning the configuration of control systems. In Chapter 5 we introduce RESCOM, a recommender systems that is based on the tensor factorization method RESCAL (see Chapter 2.2). The architecture and underlying data model of RESCOM are designed so that RESCOM can operate in a partially inductive setting and deal with the cold start problem. In the follow-up work presented in Chapter 6, we introduce NECTR, which couples an autoencoder-like neural network with a tensor factorization to produce rich embeddings that also capture non-linear interactions between configured components. Chapter 7 concludes and sketches directions for future works.

Chapter 3

Reasoning on Knowledge Graphs with Debate Dynamics

This chapter contains the publication

> Marcel Hildebrandt, Jorge Andres Quintero Serna, Yunpu Ma, Martin Ringsquandl, Mitchell Joblin, and Volker Tresp. *Reasoning on Knowledge Graphs with Debate Dynamics.* In Proceedings of the AAAI Conference on Artificial Intelligence, 2020.

Chapter 4

Scene Graph Reasoning for Visual Question Answering

This chapter contains the publication

> Marcel Hildebrandt, Hang Li, Rajat Koner, Volker Tresp, and Stephan Günnemann. *Scene Graph Reasoning for Visual Question Answering.* In International Conference on Machine Learning: Workshop GRL+, 2020.

Chapter 5

Configuration of Industrial Automation Solutions Using Multi-relational Recommender Systems

This chapter contains the publication

> Marcel Hildebrandt, Swathi Shyam Sunder, Serghei Mogoreanu, Ingo Thon, Volker Tresp, and Thomas Runkler. *Configuration of industrial automation solutions using multi-relational recommender systems.* In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 2018.

Chapter 6

A Recommender System for Complex Real-World Applications with Nonlinear Dependencies and Knowledge Graph Context

This chapter contains the publication

> Marcel Hildebrandt, Swathi Shyam Sunder, Serghei Mogoreanu, Mitchell Joblin, Akhil Mehta, Ingo Thon, and Volker Tresp (2019, June). *A recommender system for complex real-world applications with nonlinear dependencies and knowledge graph context.* In Proceedings of the European Semantic Web Conference, 2019.

Chapter 7

Conclusion

In this thesis, we examined machine learning on graph-structured data in three different settings. In what follows, we summarize our main contributions and point out directions for future works.

Explainable Knowledge Graph Reasoning We developed the idea for debate dynamics for explainable KG reasoning. The resulting method, R2D2, models the triple classification task as a debate between two opposing reinforcement learning agents. Both agents extract arguments in favor of a query fact being true or false, respectively. The agents' objective is to shift the judge's decision, a binary classifier, towards their position. Since the judge's decision is solely based on the extracted arguments, the user can examine the arguments and trace back the classification decision. We evaluated the performance of R2D2 in the triple classification setting on the benchmark datasets FB15k-237 and WN18RR. Our findings show that R2D2's performance is at least as good as several popular baseline methods with respect to the triple classification accuracy. To analyze the interpretability aspect of R2D2 in a systematic setting, we conduct a survey where 44 respondents took the role of the judge classifying the truthfulness of facts based on automatically extracted arguments. Thereby, we find that nine out of ten facts are classified correctly by the majority respondents and that for each fact the respondents' predictions agree with the decision of R2D2's judge. These findings indicate that the arguments of the agents are informative and the judge is aligned with human intuition. Hence, R2D2 allows building tools where users interact with the system via deriving their own conclusions based on the presented arguments or input new arguments. Such a tool can lead to higher acceptance of KG reasoning methods, in particular in sensitive domains where transparency and robustness are crucial.

The idea to employ debate dynamics for deriving predictions is not restricted to graph-structured data. In future works, we want to explore whether using debating agents as sparse, adversarial feature generators also leads to explainable and robust predictions in other areas such as computer vision or natural language processing.

Scene Graph Reasoning for Visual Question Answering Based on the insight that current methods for visual question answering (VQA) lack compositional reasoning abilities, we develop a new scene graph reasoning module. Our approach consists of the following two-step procedure: First, we generate a scene graph from an image. Second, conditioned on a questions, a reinforcement learning agent is trained to navigate on the scene graph until a reasoning path is obtained that allows to answer the question. The reasoning module contains state-of-the-art components from graph representation learning and natural language processing that allow the agent to form instrumental embeddings of the scene graph and the question, respectively. In the first preliminary work that is part of this thesis, we conduct experiments with manually curated scene graphs. Thereby, our method reaches human-like performance on a challenging VQA dataset.

In future works, we will integrate our own scene graph reasoning module into our method such that we can cover the whole VQA task in the sense that we map an image-question pair to a candidate answer.

Multirelational Recommender System for Industrial Automation Solutions We developed the recommendation engine RESCOM. The underlying idea is that we merge databases with historical configurations of industrial control systems and technical attributes of the components in a KG. Subsequently, we formulate the recommendation task in a link prediction setting. RESCOM is based on the existing tensor factorization method RESCAL. In particular, we derive a novel projection map that allows to compute recommendations in a partially inductive setting. This is crucial for the method's real-world applicability since it allows RESCOM to operate in real-time without having to retrain the embedding model. Incorporating the technical attributes of items allows dealing with data sparsity issues, which are particular pronounced in our user case (e.g., the sparsity measured in terms of the relative frequency of zero entries in the adjacency tensor is more than an order of magnitude higher than in the popular benchmark datasets FB15k and WN18). Control systems are complex systems where multiple, highly differentiated components are available. RESCOM allows aligning components that have similar types and technical information in the embedding space. As a consequence, the recommendation task is lifted to a technical level lowering the dependency on exact historical matches for computing recommendations. This mechanism also allows tackling the cold start problem, which arises when no or very few historical configurations for new items are available. Next to experiments in the canonical recommendation setting, we perform dedicated experiments to examine the cold start performance and show that RESCOM can compute reasonable recommendations in the absence of any historical information for a set of novel items. For example, we find that RESCOM outperforms other popular baseline methods with respect to the mean rank, the mean reciprocal rank, and the Hits@10% by at least 29%.

In a follow-up work, we introduce NECTR, which couples a tensor factorization with a graph autoencoder. The latter component is a remedy to one of the shortcomings of RESCOM, which fits a bilinear functional to model interactions between entities during training. However, during inference, when the embeddings for existing entities are fixed, computing recommendations boils down to applying a linear mapping. Thus, RESCOM cannot model any non-linear effects such as interactions among items and non-linear scaling effects. While the tensor decomposition component of NECTR incorporates the technical attributes, the autoencoder component produces a representation of the configured components that takes non-linear effects into account. Both components are coupled via a simple weight-sharing mechanism that allows end-to-end training. We show experimentally that NECTR leads to a significant performance increase compared to RESCOM, underlining the importance of higher-order effects when recommending components for industrial engineering solutions.

One of the shortcomings of RESCOM and NECTR is that both approaches are agnostics towards the control systems' topological structure. Concretely, while the components' technical attributes are modeled via triples in a KG, links between configured components corresponding to physical connections are not considered. We plan to incorporate this information via a GNN-based encoder. This approach allows not only to recommend which items should be configured in a control system but also how and where to connect the new component.

Bibliography

[1] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*, pages 37–48, 2013.

[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[3] S. Baier, Y. Ma, and V. Tresp. Improving visual relationship detection using semantic modeling of scene descriptions. In *International Semantic Web Conference*, pages 53–68. Springer, 2017.

[4] T. Berners-Lee. Linked data-design issues, 2006. URL https://www.w3.org/DesignIssues/LinkedData.html. Online; accessed 22-July-2020.

[5] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific american*, 284(5):34–43, 2001.

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

[7] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013.

[8] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[9] H. Cai, V. W. Zheng, and K. C.-C. Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637, 2018.

[10] S. Cao, W. Lu, and Q. Xu. Deep neural networks for learning graph representations. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[11] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka, and T. M. Mitchell. Toward an architecture for never-ending language learning. In *Twenty-Fourth AAAI conference on artificial intelligence*, 2010.

[12] J. Chen, T. Ma, and C. Xiao. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*, 2018.

[13] R. Das, S. Dhuliawala, M. Zaheer, L. Vilnis, I. Durugkar, A. Krishnamurthy, A. Smola, and A. McCallum. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. In *ICLR*, 2018.

[14] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.

[15] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[16] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610. ACM, 2014.

[17] D. Fensel, U. Şimşek, K. Angele, E. Huaman, E. Kärle, O. Panasiuk, I. Toma, J. Umbrich, and A. Wahler. *Knowledge graphs: Methodology, tools and selected use cases.* Springer Nature, 2020.

[18] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.

[19] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.

[20] W. L. Hamilton, R. Ying, and J. Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

[21] X. Han and J. Zhao. Structural semantic relatedness: a knowledge-based method to named entity disambiguation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 50–59. Association for Computational Linguistics, 2010.

[22] R. Harshman. Foundations of the parafac procedure: Models and conditions for an" explanatory" multi-mode factor analysis. *UCLA Working Papers in Phonetics*, 16: 1–84, 1970.

[23] M. Hildebrandt, S. S. Sunder, S. Mogoreanu, I. Thon, V. Tresp, and T. Runkler. Configuration of industrial automation solutions using multi-relational recommender systems. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 271–287. Springer, 2018.

[24] M. Hildebrandt, S. S. Sunder, S. Mogoreanu, M. Joblin, A. Mehta, I. Thon, and V. Tresp. A recommender system for complex real-world applications with nonlinear dependencies and knowledge graph context. In *European Semantic Web Conference*, pages 179–193. Springer, 2019.

[25] M. Hildebrandt, H. Li, R. Koner, V. Tresp, and S. Günnemann. Scene graph reasoning for visual question answering. *ICML Workshop GRL+. arXivpreprint arXiv:2007.01072*, 2020.

[26] M. Hildebrandt, J. A. Q. Serna, Y. Ma, M. Ringsquandl, M. Joblin, and V. Tresp. Reasoning on knowledge graphs with debate dynamics. In *AAAI*, pages 4123–4131, 2020.

[27] F. L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927.

[28] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutierrez, J. E. L. Gayo, S. Kirrane, S. Neumaier, A. Polleres, et al. Knowledge graphs. *arXiv preprint arXiv:2003.02320*, 2020.

[29] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. Shamma, M. Bernstein, and L. Fei-Fei. Image retrieval using scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3668–3678, 2015.

[30] S. M. Kazemi, R. Goel, K. Jain, I. Kobyzev, A. Sethi, P. Forsyth, and P. Poupart. Representation learning for dynamic graphs: A survey. *Journal of Machine Learning Research*, 21(70):1–73, 2020.

[31] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

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

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

[34] N. Lao and W. W. Cohen. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81(1):53–67, 2010.

[35] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

[36] D. Lukovnikov, A. Fischer, J. Lehmann, and S. Auer. Neural network-based question answering over knowledge graphs on word and character level. In *Proceedings of the 26th international conference on World Wide Web*, pages 1211–1220. International World Wide Web Conferences Steering Committee, 2017.

[37] M. Nickel, V. Tresp, and H.-P. Kriegel. A three-way model for collective learning on multi-relational data. In *ICML*, volume 11, pages 809–816, 2011.

[38] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2015.

[39] P. G. Omran, K. Wang, and Z. Wang. Scalable rule learning via learning representation. In *IJCAI*, pages 2149–2155, 2018.

[40] H. Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8(3):489–508, 2017.

[41] R. H. Richens. Preprogramming for mechanical translation. *Mech. Transl. Comput. Linguistics*, 3(1):20–25, 1956.

[42] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[43] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018.

[44] E. W. Schneider. *Course Modularization Applied: The Interface System and Its Implications For Sequence Control and Data Analysis.* Distributed by ERIC Clearinghouse, 1973.

[45] A. Singhal. Introducing the knowledge graph: things, not strings, 2012. URL https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html. Online; accessed 03-August-2020.

[46] R. Socher, D. Chen, C. D. Manning, and A. Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934, 2013.

[47] P. V. Tran. Learning to make predictions on graphs with autoencoders. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 237–245. IEEE, 2018.

[48] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, and G. Bouchard. Complex embeddings for simple link prediction. In *International Conference on Machine Learning (ICML)*, volume 48, pages 2071–2080, 2016.

[49] C. Tu, W. Zhang, Z. Liu, M. Sun, et al. Max-margin deepwalk: Discriminative learning of network representation. In *IJCAI*, volume 2016, pages 3889–3895, 2016.

[50] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[51] D. Vrandečić and M. Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[52] D. Wang, P. Cui, and W. Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234, 2016.

[53] Z. Wang and J. Li. Rdf2rules: Learning rules from rdf knowledge bases by mining frequent predicate cycles. *arXiv preprint arXiv:1512.07734*, 2015.

[54] W. Xiong, T. Hoang, and W. Y. Wang. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, Copenhagen, Denmark, September 2017. ACL.

[55] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations*, 2015.

[56] J. Yang, S. Yang, Y. Fu, X. Li, and T. Huang. Non-negative graph embedding. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.