TU WIEN Informatics

Reasoning in Knowledge Graphs: Methods and Techniques

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Logic and Computation

eingereicht von

Rebecca Jahn, BSc
Matrikelnummer 1125688

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Prof. Dr. Emanuel Sallinger
Mitwirkung: Prof. Dr. Georg Gottlob
Dr. Sahar Vahdati

Wien, 8. März 2021

Rebecca Jahn	Emanuel Sallinger

TU WIEN Informatics

Reasoning in Knowledge Graphs: Methods and Techniques

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Logic and Computation

by

Rebecca Jahn, BSc
Registration Number 1125688

to the Faculty of Informatics

at the TU Wien

Advisor: Prof. Dr. Emanuel Sallinger
Assistance: Prof. Dr. Georg Gottlob
Dr. Sahar Vahdati

Vienna, 8th March, 2021

Rebecca Jahn   Emanuel Sallinger

Erklärung zur Verfassung der Arbeit

Rebecca Jahn, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 8. März 2021

Rebecca Jahn

Danksagung

"Beginne am Anfang, und fahre fort, bis du ans Ende kommst: dann höre auf."

— *Lewis Carroll*

Ich möchte mich bei all den lieben Menschen (und Tieren) bedanken, die mir dabei geholfen haben, es bis zu diesem "Ende" zu schaffen - unter anderem meinen Studienkolleg*innen, allen voran Stefan, Klara, Anja, Julian, Pamina und Stefan. Weiters bedanke ich mich bei Emanuel Sallinger und Sahar Vahdati für ihre Betreuung und Begleitung dieser Diplomarbeit.

Besonderer Dank gilt Winnie und Peter, die diese Arbeit korrekturgelesen haben und mich auch sonst stets unterstützt und motiviert haben. Außerdem danke ich Felix, Daniela, Tilman, Tom, Quinty, Francesca, Konsti, Raphael, Nicole, Marta, Ville, Kristof, Katharina und Patricia; sowie meinen Mitbewohner*innen Jana, Stefanie, Keil, Katha, Thomas, Laura, Flora, Fionn und Lina, weil sie mich moralisch unterstützt haben und Verständnis dafür hatten, dass Pläne aufgrund meiner Masterarbeit verschoben oder verändert werden mussten. Ganz lieben Dank auch meiner Familie für ihre Unterstützung: meiner Schwester Jacqueline und ihrem Partner Michi, meiner Nichte Sophia, sowie meinen Eltern Monika und Harald.

Acknowledgements

"Begin at the beginning, and go on till you come to the end: then stop."

— *Lewis Carroll*

I want to thank all the dear people (and animals) that got me to this "end" - among others my fellow students Stefan, Klara, Anja, Julian, Pamina and Stefan. I also want to thank my supervisors Emanuel Sallinger und Sahar Vahdati for their support and constructive feedback.

Above all, I would like to thank Winnie and Peter, who proofread this thesis and supported and motivated me continuously. I am also grateful to Felix, Daniela, Tilman, Tom, Quinty, Francesca, Konsti, Raphael, Nicole, Marta, Ville, Kristof, Katharina and Patricia; as well as my flatmates Jana, Stefanie, Keil, Katha, Thomas, Laura, Flora, Fionn und Lina, for giving me support and being understanding when plans had to be changed because of this thesis. A big thanks also to my family for their support: my sister Jacqueline and her partner Michi, my niece Sophia, and my parents Monika and Harald.

Kurzfassung

Durch die rapide technologische Entwicklung in den letzten Jahren wurde es immer einfacher und dadurch üblicher, riesige Datenmengen zu sammeln, zu speichern und zu verarbeiten. Da diese Daten aus vielen unterschiedlichen Quellen stammen, sind sie häufig sehr heterogen. Knowledge Graphs verkörpern eine Technologie, die es ermöglicht, diese heterogenen Daten auf eine semantische und einheitliche Art zu repräsentieren, indem Entitäten und die Beziehungen zwischen ihnen durch ein strukturiertes, aber gleichzeitig flexibles Format ausgedrückt werden. Außer der Datenspeicherung gibt es in Knowledge Graphs üblicherweise auch Informationen auf konzeptioneller (oder ontologischer) Ebene, wie z.B. Domänenbeschränkungen, Allgemeinwissen o.Ä., die in Form von Regeln hinterlegt sind. Durch diese werden Folgerungen auf Basis der Daten und darin gefundener Muster ermöglicht.

Obwohl logisches Folgern in der Informatik schon lange ein wichtiges Thema ist, wurde es durch die Forschung an Knowledge Graphs nun wieder in den Vordergrund gerückt. Das wurde besonders deutlich durch die vielen Bestrebungen, die Mengen an wissenschaftlichen Arbeiten in diesem Bereich zu strukturieren, z.B. durch Chen et al., Al-Moslmi et al. und Hogan et al. Leider behandeln diese und andere Überblicksarbeiten oft nur wenige Arten von logischem Folgern und Aufgaben im Lebenszyklus eines Knowledge Graph, und haben häufig einen Fokus auf bestimmte Domänen, was in dieser Arbeit beleuchtet wird. Aufgrund der fehlenden umfassenden Überblicksarbeit zu diesem Thema ist es schwierig, mit den vielfältigen Ansätzen und Methoden sowie dem Stand der Technik vertraut zu werden. Außerdem sind viele Publikationen auf Domänenexpert*innen ausgerichtet und sogar Hintergrundwissen ist häufig nur verstreut zu finden.

Der Beitrag dieser Arbeit ist daher: (1) eine umfassende, leicht verständliche Zusammenfassung des relavanten Hintergrundwissens sowie eine Verortung im aktuellen Stand der Technik; (2) ein Überblick über verschiedene Methoden des logischen Folgerns in Anbetracht mehrerer Aspekte, und nicht nur beschränkt auf ein Anwendungsgebiet oder Unterthema; (3) eine umfassende Untersuchung dieser Methoden, inkl. Logik-, Statistik- und Graph-basiertem Folgern sowie Kombinationen davon und Synergien zwischen den Methoden; (4) eine Abhandlung über den ganzen Lebenszyklus betreffende Aufgaben sowie jeweils dafür geeignete Methoden. Die Definition des Lebenszyklus basiert auf Auer et al. und Pouchard, wird aber auf die drei breiteren Kategorien Wissensintegration, Knowledge-Graph-Evolution und Anwendungen reduziert.

Abstract

Owing to the rapid technical development in recent years, collecting, storing and managing massive amounts of data on and off the Web has become more feasible and therefore more common. Because of its various sources, the collected data is usually highly heterogeneous. Knowledge graphs constitute a technology that makes it possible to represent and organize such heterogeneous data in a semantic and unified way by describing entities and their inter-relations in a structured but flexible format. In addition to storing the raw data, knowledge graphs normally have a layer of conceptual knowledge (also called ontological knowledge), usually represented as a set of rules, that contains domain constraints, encodes common knowledge or enables reasoning about the data and patterns therein.

Although reasoning itself is a long-standing topic in computer science, it has become an important new focus in knowledge graph research in recent years. This has been made apparent by many attempts of structuring the massive amounts of research works in this area such as the surveys by Chen et al. and Al-Moslmi et al. or the overview provided by Hogan et al. Unfortunately, existing surveys usually only cover some types of reasoning or life cycle tasks in depth or only focus on certain domains, as will be extensively presented in the thesis. A multitude of reasoning methods and frameworks have been proposed for and applied to knowledge graphs with different results and varying degrees of success. However, because of the lack of a comprehensive, deep and diverse survey on the topic, it is challenging to gain a foothold in this area, as the target audience are often experts in the field and even background knowledge is frequently scattered throughout various pieces of literature.

The main contributions of this thesis are: (1) Providing a comprehensive background that is understandable also for non-experts in the field along with an embedding into the state-of-the-art literature. (2) Giving an overview of various reasoning paradigms and the state of the art considering multiple aspects and not just focusing on one area of application or subtopic. (3) Including different types of reasoning, like statistics-based, logic-based and graph-based methods as well as combined approaches, thereby not limiting the results by ignoring valuable insights about synergies between the methods. (4) Examining reasoning methods as they pertain to the whole life cycle and which tasks they are most suited for. The life cycle classification we use is based on Auer et al. and Pouchard, but will be reduced to three broader categories: knowledge integration, knowledge graph evolution and knowledge applications.

Contents

CHAPTER 1

Introduction

Owing to the rapid technical development in recent years, collecting, storing and managing massive amounts of data on and off the Web has become more feasible and therefore more common. Because of its various sources, the collected data is usually highly heterogeneous. Since the sheer amount of data also means it is out of the scope for human processing, it has become a focus of research how to collect, process and analyze the data automatically and efficiently. The resulting processed and analyzed data can then be used for applications and downstream tasks like entity disambiguation, semantic parsing, information extraction or question answering.

Knowledge Graphs (KGs) constitute a formalism for knowledge representation technology that makes it possible to represent and organize such heterogeneous data with regard to underlying semantics and in a unified way by describing entities and their inter-relations in a structured but flexible format. There are many ways to represent the facts stored in a KG, the most basic of which is as subject-predicate-object (SPO) triples. There are various knowledge graphs in existence (often part of entire knowledge graph management systems), some for broad domain knowledge, like the Never-Ending Language Learning (NELL) project [CBK$^+$10], DBpedia [LIJ$^+$15] or the Google Knowledge Graph [Sin12]. Others contain more specific types of knowledge, like WordNet [Mil98], which stores words and the semantic relationships between them, or social KGs like the one by Facebook, which stores relationships it maintains between people, pages etc. The term knowledge graph has also been used to describe open knowledge projects like Wikidata [VK14] and YAGO [SKW07].

In addition to storing the raw data, knowledge graphs normally have a layer of conceptual knowledge (also called ontological knowledge), usually represented as a set of rules, that contains domain constraints, encodes common knowledge or enables **reasoning** about the data and patterns therein. This makes it possible for KGs to not just organize and represent knowledge in an effective way, but to efficiently utilize this knowledge for advanced reasoning tasks.

The goal of reasoning in this context is to add more facts and logical rules to knowledge graphs as well as to improve the existing ones in an automated manner and to gain insights from the data. While data sparsity is a problem and we would like to have answers for every relevant question already contained in a knowledge graph, an increasing number of facts, i.e. a dense KG, leads to growing computational inefficiency, thereby making many tasks infeasible. The question is therefore not just how to find new facts but also how to decide which facts should be added in the first place and how to make the reasoning process more efficient.

1.1 Problem Statement

Although reasoning itself is a long-standing topic in computer science, it has become an important new focus in knowledge graph research in recent years. This has been made apparent by many attempts of structuring the massive amounts of research works in this area such as the surveys by Chen et al. [CJX19] and Al-Moslmi et al. [AMOOV20] or the overview provided by Hogan et al. [HBC$^+$20]. Unfortunately, existing surveys usually only cover some types of reasoning or life cycle tasks in depth or only focus on certain domains, as will be extensively presented in the next section.

A multitude of reasoning methods and frameworks have been proposed for and applied to knowledge graphs with different results and varying degrees of success. However, because of the lack of a comprehensive survey on the topic, it is challenging to gain a foothold in this area, as the publications are often aimed at experts in the field and even background knowledge is frequently scattered throughout various pieces of literature. Similarly, it is not easy to get an understanding of the state of the art as well as recent advances in the field without sifting through tens or hundreds of publications.

It has hence become clear that it is important for the research community for there to be a systematic overview of reasoning in KGs answering the following questions:

Research Question 1: Which reasoning approaches have been proposed over time and how can they be classified based on their underlying type of reasoning?

With this research question, we are aiming to collect a comprehensive set of research works which have been published over time concerning reasoning and knowledge graphs. We are particularly interested in the types of reasoning that have been discussed in the underlying research works, either in theory or practice. The main requirement is to develop a fair, relevant and complete set of criteria to select among the many research works the ones that are the best match for this research work. In order to do so, a selection framework has been developed. In addition to the selection criteria we also have developed classification criteria to fulfill the ultimate goals of this research question.

Research Question 2: What are the respective strengths and weaknesses of these approaches and which applications are they therefore most suited for?

In this research question, we mainly focus on a characterization of the approaches and techniques most prevalent in the previously collected works based on their strengths and weaknesses. To this end, the approaches are grouped into the categories of logic-based, statistics-based and graph-based reasoning, since members of the respective groups often share similar characteristics. The rather abstract strengths and weaknesses are then put into perspective by showcasing which tasks in a KG life cycle they are most suited for, both in terms of accuracy and efficiency.

Research Question 3: Given these strengths and weaknesses, which synergies exist between the different approaches and how can they be combined?

The strengths and weaknesses of different types of reasoning often complement each other, which is why there have increasingly been efforts to combine various approaches so as to be able to harness their advantages while hopefully being able to avoid their disadvantages. Unfortunately, many surveys focus on only one type of reasoning, thereby mostly ignoring the possible synergies with other reasoning types. The aim of this thesis is to bridge this gap and to put a particular focus on combined approaches.

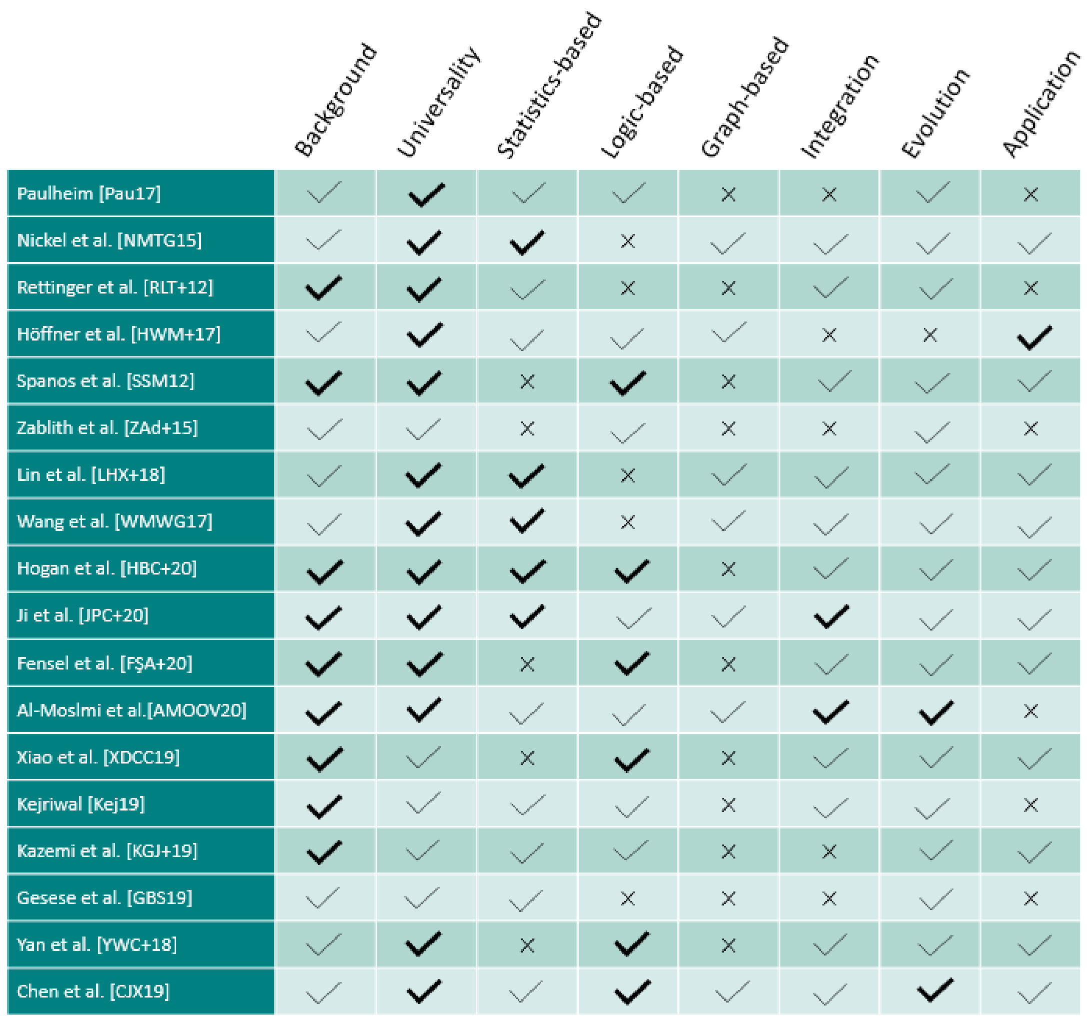
The main contributions of this thesis are therefore planned to be:

- Provide a comprehensive background that is understandable also for non-experts in the field along with citations that help to fill gaps where they may arise.
- Give an overview of various reasoning paradigms and the state of the art considering multiple aspects and not just focus on one area of application or subtopic.
- Include different types of reasoning, like statistics-based, logic-based and graph-based methods as well as combined approaches, thereby not limiting the results by ignoring valuable insights about synergies between the methods.
- Examine reasoning methods as they pertain to the whole *life cycle* and which tasks they are most suited for. The life cycle classification is based on Pouchard et al. [Pou15] and Auer et al. [ABT14], but will be reduced to three broader categories: knowledge integration [ZD07], knowledge graph evolution and applications.

While there have been multiple attempts to provide surveys about reasoning in knowledge graphs, currently available ones do not meet the standards stated above. They often fall short in at least one of these metrics, since their focus is usually too wide or too narrow, as will be discussed in the next section.

1.2 Challenges and Related Work

The state of the art with regards to reasoning in knowledge graphs has been covered in various levels of detail in a number of surveys. A selection of the most relevant and comprehensive of them will be presented here along with their focuses as well as the challenges they - and this thesis - face. The main challenge seems to be **topic broadness**, as both reasoning and KGs are ubiquitous nowadays, particularly since there is no clear definition for the latter. An additional challenge is the **interdisciplinarity** of reasoning in KGs, since it means that research is very heterogeneous and scattered among various domains and it takes a large amount of background knowledge to analyze it.



	Background	Universality	Statistics-based	Logic-based	Graph-based	Integration	Evolution	Application
Paulheim [Pau17]	✓	**✓**	✓	✓	×	×	✓	×
Nickel et al. [NMTG15]	✓	**✓**	**✓**	×	✓	✓	✓	✓
Rettinger et al. [RLT+12]	**✓**	**✓**	✓	×	×	✓	✓	×
Höffner et al. [HWM+17]	✓	**✓**	✓	✓	✓	×	×	**✓**
Spanos et al. [SSM12]	**✓**	**✓**	×	**✓**	×	✓	✓	✓
Zablith et al. [ZAd+15]	✓	✓	×	✓	×	×	✓	×
Lin et al. [LHX+18]	✓	**✓**	**✓**	×	✓	✓	✓	✓
Wang et al. [WMWG17]	✓	**✓**	**✓**	×	✓	✓	✓	✓
Hogan et al. [HBC+20]	**✓**	**✓**	**✓**	**✓**	×	✓	✓	✓
Ji et al. [JPC+20]	**✓**	**✓**	**✓**	✓	✓	**✓**	✓	✓
Fensel et al. [FŞA+20]	**✓**	**✓**	×	**✓**	×	✓	✓	✓
Al-Moslmi et al.[AMOOV20]	**✓**	**✓**	✓	✓	✓	**✓**	**✓**	×
Xiao et al. [XDCC19]	**✓**	✓	×	**✓**	×	✓	✓	✓
Kejriwal [Kej19]	**✓**	✓	✓	✓	×	✓	✓	×
Kazemi et al. [KGJ+19]	**✓**	✓	✓	✓	×	×	✓	✓
Gesese et al. [GBS19]	✓	✓	✓	×	×	×	✓	×
Yan et al. [YWC+18]	✓	**✓**	×	**✓**	×	✓	✓	✓
Chen et al. [CJX19]	✓	**✓**	✓	**✓**	✓	✓	**✓**	✓

Figure 1.1: A comparison of the coverage of related work regarding background, whether they are domain-specific or universally applicable, types of reasoning and life cycle. A bold check mark means coverage in detail or full applicability, a normal check mark means some coverage or applicability and a cross means no or little coverage or applicability.

As can be seen in Figure 1.1, many surveys focus on a single type of reasoning [YWC+18, ZAd+15, SSM12, FŞA+20] or a small subset of life cycle tasks [ZAd+15, Pau17, HWM+17]. In particular, coverage of graph-based methods like the Path Ranking Algorithm (PRA) [LMC11] is still very poor in the state-of-the-art surveys that were analyzed for this thesis. While a review of e.g. only one or two types of reasoning with regards to, say, KG integration is marked with a non-bold check mark in the respective life cycle column in Figure 1.1, the lack of a comprehensive overview is even more obvious in the following matrix representation (Table 1.1). As can be seen in this table, hardly any surveys cover more than four aspects and, again, graph-based methods are grossly neglected. This reaffirms our hypothesis of a great need to bridge the gap for a comprehensive review article discussing both KGs and multiple types of reasoning in depth.

	Integration		Evolution		Application	
Statistics-based	[NMTG15] [LHX+18] [HBC+20] [AMOOV20]	[RLT+12] [WMWG17] [JPC+20]	[Pau17] [RLT+12] [WMWG17] [JPC+20] [Kej19] [GBS19]	[NMTG15] [LHX+18] [HBC+20] [AMOOV20] [KGJ+19] [CJX19]	[NMTG15] [LHX+18] [HBC+20] [KGJ+19]	[HWM+17] [WMWG17] [JPC+20] [CJX19]
Logic-based	[SSM12] [JPC+20] [AMOOV20] [Kej19] [CJX19]	[HBC+20] [FŞA+20] [XDCC19] [YWC+18]	[Pau17] [ZAd+15] [FŞA+20] [XDCC19] [YWC+18]	[SSM12] [HBC+20] [AMOOV20] [KGJ+19] [CJX19]	[HWM+17] [HBC+20] [XDCC19] [YWC+18]	[SSM12] [FŞA+20] [KGJ+19] [CJX19]
Graph-based	[JPC+20]	[AMOOV20]	[NMTG15] [WMWG17] [CJX19]	[LHX+18] [AMOOV20]	[HWM+17]	[CJX19]

Table 1.1: A matrix view of the coverage regarding different types of reasoning and life cycle tasks. Marked in blue resp. green are surveys that at least touch on more than one type of reasoning resp. life cycle task, marked in purple are those for which both is true.

Notable exceptions, also regarding a comprehensive background section and the applicability of surveyed results to general KGs and not just domain-specific ones, are:

- Al-Moslmi et al. [AMOOV20] cover integration and KG evolution with regards to all three types of reasoning, but they focus strongly on named entity extraction and therefore do not cover applications at all.
- The surveys by Nickel et al. [NMTG15], Wang et al. [WMWG17] and Lin et al. [LHX+18] are very similar in their coverage, as they all mainly review Knowledge Representation Learning (KRL) and only cover graph-based methods insofar as they overlap with this topic, e.g. by covering methods that also embed paths. They do not cover logic-based reasoning in any detail, only mentioning methods that are able to embed rules in passing.

- While Ji et al. [JPC+20] also have a focus on KRL and only cover KG evolution and applications in this context, they do include logic-based reasoning in a bit more detail than the surveys above.
- Chen et al. [CJX19] have a similar categorization as the one used in this thesis, but they just include research from 2012 onward and focus heavily on KG completion, only covering integration and applications fleetingly. Additionally, they do not provide relevant background and many results are only cited with minimal explanation and context. While this does make their survey suitable to get a crude overview of (very) recent research approaches, it does not meet the standards set forth in the previous section and is better suited for researchers already well versed in the research area.
- The recent survey by Hogan et al. [HBC+20] comes closest to the aims set in the previous section, but it does not cover graph-based methods at all. Additionally, it is more of a “tutorial” on knowledge graphs in general, covering a lot of background information e.g. on the storage aspect of knowledge graphs while not focusing on reasoning in particular.

In summary, to the best of our knowledge there is no comprehensive survey of reasoning in knowledge graphs that covers both multiple types of reasoning and a variety of life cycle tasks in detail. This might be due to the strong interdisciplinarity of the field. Researchers in the areas of machine learning and natural language processing provide reviews of research about statistics-based reasoning, while researchers whose focus is or was on databases or the semantic web often only survey logic-based approaches in detail.

In particular, there is a gap when it comes to graph-based methods which have only recently seen more attention but show a lot of promise, especially in conjunction with other types of reasoning. Combined approaches also are not covered in particular detail in any of the listed surveys even though they could take advantage of synergies between different types of reasoning, thereby offering better performance than any single method.

1.3 Aim of the Work & Contribution

This thesis aims to provide an overview of different reasoning techniques and the current state-of-the-art with regards to topics related to reasoning on knowledge graphs, since the absence of a comprehensive review meeting the standards stated in Section 1.1 represents a hurdle for researchers who want to gain an overview of this topic.

In order to provide a systematic review of current research as well as a summary of relevant background information that adheres to these standards, reasoning in KGs is categorized along the dimensions of types of reasoning and life cycle. Since graph-based reasoning and approaches combining multiple types of reasoning have not been a focus of many surveys to date, particular care will be taken to include sufficient information about these aspects. In summary, the aim of this thesis is to answer the research questions posted in Section 1.1 by constructing a survey that includes:

- A presentation of comprehensive, easily-understandable preliminary knowledge on reasoning and knowledge graphs.
- An examination of the theory as well as the current state of the art as pertaining to the following types of reasoning and combinations thereof:
  - Statistics-based reasoning
  - Logic-based reasoning
  - Graph-based reasoning
- An overview of how these different reasoning paradigms are used to tackle the various tasks that arise during the life cycle of a knowledge graph consisting of:
  - Knowledge integration
  - Knowledge graph evolution
  - Applications
- A broad coverage of reasoning in KGs that is general enough to be applicable to a wide range of problem settings and not just limited to a specific domain or a single type of knowledge.

1.4 Methodological Approach

The methodological approach consists of a literature review for which relevant research is gathered, a conceptual analysis of the state of the art is conducted and the information is organized systematically. To this end, we have developed selection criteria as well as classification criteria, as stated in Section 1.1 and presented in the rest of this section.

The methodology and established principles are based loosely on Zaveri et al. [ZRM$^+$16] (particularly the first three steps) and adapted as follows:

- An initial literature list is supplemented through a systematic search activity on Google Scholar and other scholarly search engines like Semantic Scholar and Refseek, with search activity being author-based, title-based, keyword-based, conference-based, topic-based etc.
- Obviously irrelevant articles are removed from the list and the abstracts of the remaining retrieved articles are further examined for relevance, i.e. whether they do not just present a specific KG system or technique but have a broader scope pertaining to reasoning in knowledge graphs.

- References of fitting papers (but also rejected ones) are scanned for other relevant articles and surveys and the previous step is repeated until the thesis meets the quality criteria set out in Section 1.1, i.e. it contains a background section and relevant information on multiple types of reasoning and the life cycle tasks they are used for and is applicable to general problem settings.

- The next methodological step is to decide upon the axes on which to stratify the existing research. The axes used in Figure 1.1 and Table 1.1, namely types of reasoning and life cycle tasks, provide the best trade-off between being as general as possible and splitting the field along its natural interdisciplinary fault lines. This differentiation has also previously been used by Yan et al. [YWC+18] and others [Kaz18, TYM18].

1.5 Relevance to the Curriculum of Logic and Computation

This thesis aims at contributing a comprehensive overview of reasoning in knowledge graphs, which are related to other topics like:

- intelligent information systems, and
- knowledge representation

Knowledge graphs are based on and inspired by various other systems to store and reason over data, especially from the area of the semantic web. The reasoning aspect in particular has been covered in the courses on knowledge-based systems, which contain information about various reasoning paradigms, technologies and algorithms – some of which are extensively used in the context of knowledge graphs.

While the thesis also touches on graph-theoretic aspects involved in KG reasoning, the most directly related courses are the following:

- 184.730 Knowledge-based Systems
- 181.140 Database Theory
- 184.705 Semistructured Data
- 184.729 Semantic Web Technologies
- 188.387 Semi-Automatic Information and Knowledge Systems
- 188.399 Introduction to Semantic Systems
- 188.483 Knowledge Management

1.6 Structure of the Work

The remainder of this thesis is organized as follows:

First, an overview of knowledge graphs and relevant background required to understand the problem and models discussed is given in Chapter 2.

Chapter 3 presents an overview of knowledge *integration*, i.e. how reasoning can be used to include data from other knowledge bases and the Web into a KG.

Chapter 4 then describes how the collected data can be refined and how errors can be corrected automatically. It details how different types of reasoning are used for this knowledge graph *evolution* and what their respective strengths and weaknesses are.

In Chapter 5 we conduct a survey of various *applications* of knowledge graphs, like question answering or recommender systems, and how reasoning can be used to facilitate these tasks.

Chapter 6 gives a more detailed comparison of the various *types of reasoning* that are used for the previously discussed reasoning tasks. Additionally, approaches utilizing multiple kinds of reasoning are discussed, which aim to combine their advantages while avoiding their respective disadvantages.

Finally, in Chapter 7 we recap the results discussed throughout the thesis, draw *conclusions* from them and discuss future research directions.

CHAPTER 2

Background

Ambiguity around the notion of reasoning and inference in databases and machine leaning communities has itself always been challenging. This has been accelerated since the notion of knowledge graphs was introduced and a lot of attempts were made to clarify its meaning. Therefore, because of the interdisciplinary nature of research in general and more specifically regarding the topic of reasoning in the context of knowledge graphs, the same concept is often discussed under various notations in the literature. Before discussing more advanced concepts necessary to understand the life cycle of a KG and the various types of reasoning covered in this thesis, we need a comprehensive conceptualisation on all the related topics. To this end, we collected an analyzed a multitude of research works from various disciplines in order to establish a shared terminology that can be used in the rest of this work and hopefully provide a common platform for future works as well.

In this chapter, we therefore explain the important concepts, using the most generally accepted terminology and discussing alternative representations. Additionally, we provide the necessary background information on knowledge graphs and reasoning on which the rest of the thesis builds.

Section 2.1 gives a definition of knowledge graphs and discusses some relevant representational decisions, while Section 2.2 introduces the basics of knowledge representation. In Section 2.3, we provide an overview of reasoning in general and in the context of knowledge graphs, and touch upon some of the typical tasks that can be accomplished with it. Section 2.4 talks about the nature of incompleteness of knowledge graphs and the basic assumptions that must be made about missing knowledge, as well as an overview of different ways of dealing with it. Finally, Section 2.5 contains a discussion of less common kinds of knowledge graphs and what they are suited for.

2.1 What are Knowledge Graphs?

The term knowledge graph was popularized by Google after they used it as the name for their own Knowledge Base (KB), which was partly based on the collaborative KB Freebase [BEP+08]. It has been used in the past in a more diverse way to encompass several technologies like ontologies or just generally KBs somehow related to graphs. Lately, the term has come to usually denote the narrower concept of large KBs that contain information about entities and their interrelations in the form of asserted facts and axioms - sometimes represented as a graph - and apply a reasoner to derive new knowledge [EW16].

A knowledge graph typically contains statements about specific entities - people, objects, events or abstract concepts - and axioms about categories and the properties of the individuals contained in it [ABM+18]. These statements are constructed formally so as to express them in a clear and unambiguous way, for example as grounded unary and binary atoms typically referred to as facts [HSGE+18].

The facts and relationships contained in a KG can be generic, for example encoding common-sense knowledge, or they can be limited to a certain domain, like banking or a specific company.

KGs are often the result of a (semi-)automatic process of information acquisition and integration, with data coming from multiple, sometimes conflicting sources. Because of this, noisy and faulty data are a recurring challenge and ongoing topic of research [BDPP19], which will be discussed in Chapter 3.

2.2 Knowledge Representation

The factual knowledge in a KG can be structured in different ways, the most basic of which is in the form of so-called SPO triples consisting of a subject, an object and a predicate or relation linking the subject to the object.

However, for some applications this formalism is not expressive enough [WHS16], and there are multiple extensions and also completely separate forms of representation. The most common kind of extension involves higher-arity relations that make it possible to add more information to a fact, like temporal [KGJ+19] or spatial context or the probability that the fact is indeed true [NMTG15].

In Freebase [BEP+08], beginning and end dates, geographic coordinates, changing values over time etc. could be added to some facts, using compound value type (CVT) constructs. For example, to describe the office of US president, beginning and end dates as well as the person holding the office at that time could be added to the entity using CVT statements [PTVS+16]. The YAGO2 project [HSBW13] extended the representation format to SPOTL (subject, predicate, object, time, location) so as to be able to model this additional information directly, without having to use reification as in Freebase to transform n-ary relations into several binary relations connected to a central concept.

As was mentioned in the previous section, a KG usually also contains statements about facts, i.e. an ontology, which are expressed in a language like first-order logic (FOL), or a subset thereof. In description logic (DL), instance data like the unary fact $\mathit{Person}(\mathit{BarackObama})$ or the binary fact $(\mathit{BarackObama}, \mathit{presidentOf}, \mathit{USA})$ is part of the so-called A-Box, while the subset of statements contained in the T-Box defines classes, class hierarchy, domains, and ranges for predicates [GTHS15]. This would include axioms limiting the range of subject and object in a triple like $(x, \mathit{presidentOf}, y)$ to entities of type Person and $\mathit{Country}$ respectively, or axioms defining sub-classes like $\mathit{President} \sqsubseteq \mathit{Person}$ [HBC+20], expressing that every president is a person.

2.3 Reasoning in the Context of Knowledge Graphs

An important characteristic of knowledge graphs is that they are not just (graph) databases, but also contain a layer of conceptual knowledge that offers insights about the data it encompasses and allows to reason over it. [OWW18]

This knowledge is often represented through rules encoding if–then-style consequences, which are usually of the form $\mathit{head}(\mathit{then}) \leftarrow \mathit{body}(\mathit{if})$, where head is a binary atom and body is a conjunction of, possibly negated, binary or unary atoms [HBC+20]. Equivalently, they can be seen as formulas of first-order or second-order logic, as with T-Box statements discussed in the previous section. Thus, a set of rules is effectively an ontology, and the other way around. While statistical methods, like machine learning and data mining, are well suited to deal with large amounts of data and are applied to large KBs to find patterns and thereby gain new insights, the results are usually not explainable, i.e. it is not clear how they were found and whether they contain any unwanted biases.

In contrast, rules are explicit knowledge and offer human understandable explanations to the learning results they produce [OWW18], but often struggle with efficiency when it comes to datasets in the order of magnitude of a typical knowledge graph. However, their semantic reasoning capabilities make them quite an important asset in research on KGs and big data in general, since they facilitate dealing with heterogenous data, allowing data from multiple sources to be integrated into a single KB by overcoming semantic interoperability barriers [ABM+18].

Extracting rules automatically is therefore an important and useful process in reasoning [OWW18]. To combine the advantages of statistics-based and logic-based reasoning in this context, graph-based reasoning is being used more and more often [LMC11]. It aims to utilize structural data of the graph by analyzing patterns in the reachability of nodes from each other, using logical constraints to guide the process.

Reasoning in the context of knowledge graphs is the process of inferring new (non-trivial) knowledge or identifying false knowledge from existing knowledge, and its importance is demonstrated by different tasks. Antoniou et al. [ABM+18] described a wide variety of tasks, of which the most important can be summarized as:

- Materialization or closure is the task of inferring implied facts from the given data, ontology and/or rules. It can be total (i.e. inference of all implied facts) or partial, as is usually the case in a KG setting. While forward chaining (i.e. applying axioms and rules to facts so as to derive new facts) is used for KG completion, backward chaining (i.e. finding instantiations of rule bodies with existing facts) is used for query answering.
- Consistency checking refers to the task of checking whether a KG contains a contradiction, i.e. by making sure that they do not violate consistency constraints. It is an especially critical task when dealing with heterogeneous data in large-scale applications.
- Subsumption refers to the task of discerning which concepts in an ontology are subsumed by others, i.e. because one is a specialization of the other.
- Instance checking is the task of inferring which concepts a specific individual is part of and which properties apply to it.
- Finding justifications refers to the task of finding the axioms and facts from which a particular conclusion follows.

While traditional reasoning is applied in the context of knowledge graphs, the simplicity, intuitiveness and efficiency with which they can organize and represent knowledge encourages more diverse knowledge reasoning for advanced applications [CJX19]. Usually, reasoning is first applied in settings like the aforementioned consistency checking and other tasks needed to find and repair errors in the data set, enabling further analysis and allowing for more complete and meaningful answers in downstream tasks, like querying the KG for inferred information [ABM$^+$18].

This life cycle is also the basis for the structure of this thesis, which aims to review different concepts and methods of reasoning over knowledge graphs. We first discuss the acquisition of knowledge, then the curation of the assembled knowledge and lastly present applications based on both of the previous steps. All of this is done considering three different kinds of reasoning, namely logic-based, statistics-based and graph-based, and examining which life cycle tasks they are each most suited for and why.

2.4 Incompleteness

Knowledge graphs are inherently incomplete, as it would be impossible to know every entity and every relationship between entities in the world, let alone save all this information in an efficient way. The more specialized to a (narrow) area of application, the more feasible it is to actually have a more or less complete KG but there will usually be many unknown facts nonetheless.

As such, if we call a (fictitious) complete knowledge graph only containing true facts K, then we usually deal with an incomplete KG K', that does not contain all the facts in

K and may contain some (false) facts not in K [Gar15]. Therefore, in knowledge graph completion (see Section 4.1) the aim is to infer those missing facts $F = K - K'$, that are “true” but not yet given or entailed by the knowledge graph and knowledge graph correction (see Section 4.2) aims to identify the erroneous facts $F' = K' - K$.

While adding missing entities is rather straightforward, one of the most important and challenging tasks for KG completion is link prediction, which aims to judge the probability that two entities should be linked by a relation [HBC+20]. For example, given facts like $(BarackObama, fatherOf, MaliaObama)$ and $(MaliaObama, siblingOf, SashaObama)$, a probable missing edge would be $(BarackObama, fatherOf, SashaObama)$.

However, it is just as important to decide which triples to materialize, since blindly adding more facts to the knowledge graph would often make subsequent reasoning less efficient. In general, very dense KGs are not strictly better than sparser ones, as they result in computationally very hard tasks.

A related problem is how to deal with triples that are not contained in the KG. While triples existing in the KG are always interpreted as true - at least to a certain degree, as sometimes there is a probability attached - there are different approaches to dealing with non-existing triples and how their absence should be interpreted:

- **Closed-World Assumption (CWA)**

  Under the closed-world assumption, triples that are not in the KG are interpreted as being false. If, for example, there is no edge of type *marriedTo* leading to or from the entity *BarackObama*, the conclusion would be that he is definitely not married. A respective query would therefore get a negative result, even though he is actually married to Michelle Obama and the KG is just ignorant to that fact, as it is to many others.

  The CWA is the typical assumption in standard database settings and it sets them apart from KGs, in which other approaches are more prevalent, due to their inherent incompleteness [GTHS15].

- **Open-World Assumption (OWA)**

  Under the open-world assumption, triples that are not in the KG are interpreted as being unknown, i.e. they can be either true or false [NMTG15]. We would therefore not conclude from a missing edge of type *marriedTo* that Barack Obama is NOT married.

  As stated before, in incomplete settings like KGs, this more cautious approach is justified, as even basic facts are unknown in a lot of cases. Freebase, for example, does not contain a place of birth attribute for 71 percent of the people contained in the KG [WGM+14]. In particular, RDF-style KBs (and the Semantic Web) operate under the OWA - RDF has only positive inference rules and the corresponding KBs therefore only contain positive statements and no negation [GTHS15]. It therefore completely lacks the capacity to differentiate between known false facts and unknown facts and could not reasonably operate under the CWA.

- **Local Closed World Assumption (LCWA)**

  The local closed-world assumption [DGH+14a], sometimes also called Partial Completeness Assumption (PCA) [GTHS13], is a kind of intermediate assumption between CWA and OWA. Its underlying assumption is that if a relation $r(s, o)$ exists in the KG, then any relation $r(s, o')$ that is not already in the KG is considered to be false. Basically, we assume that if we know one o for a given s and r, then we know all o for that pair. For example, if the triple (*BarackObama*, *marriedTo*, *MichelleObama*) is known, then (*BarackObama*, *marriedTo*, *RobinSmith*) is probably false, since - at least in most European countries - one can only be married to one person at a time.

  This approach is a reasonably good heuristic for functional relations, like capital cities or birth places, and relations that have a high functionality, like citizenship or spouses. It obviously works less well for multi-valued properties like siblings, but often still rather well for knowledge extracted from a single source, as it often contains either all the values for a given subject/relation pair, or none [GTHS13].

  The LCWA is often used for data extraction or training relational models [NMTG15], as for example in AMIE [GTHS15] it is used to generate useful negative/counter-examples for a rule in a less restrictive way than with standard confidence. It has proven rather effective in practice [DGH+14a], because even though it produces false negative examples, it produces far fewer than the CWA, since no assumption is made for a relation mention $r(s, o')$, if there is no knowledge of a triple $r(s, o)$ [MRHLA18].

2.5 Other Kinds of Knowledge Graphs

Many interesting extensions to “classical” KGs have been proposed that try to fix certain problems, extend the area of application, etc. While they are not the focus of this thesis, they offer some interesting insights into further opportunities and application areas for reasoning in KGs, which is why short overviews will be presented in the following:

- **Probabilistic Knowledge Graphs**

  While many systems compute the probabilities of facts being true through some kind of confidence score e.g. for the extraction of facts, they often discard those probabilities after filtering candidate facts by applying some threshold. The final output KG will still be deterministic, which some argue [WLWZ12] is not a very realistic view at data, as it is rarely completely clear whether a fact is actually true, even with human-curated data.

  Since removing all uncertainties is impossible anyway, probabilistic or uncertain KGs like ProBase [WLWZ12] or NELL [CBK+10] try to embrace them. They use probabilities or confidence scores, that represent the likelihood of a fact being true, to model the inconsistency, uncertainty and ambiguity in the data.

In ProBase the uncertainty is dealt with in two ways. First, all facts (or rather claims) are assigned a plausibility, which is supposed to indicate the confidence in the fact as well as the reliability of the source. Second, a conditional probability between a concept and its instances, which is supposed to indicate whether the instance is a typical example of the concept (a.k.a. instantiation) or vice versa (a.k.a. abstraction). This typicality score would be higher for *Cat* than for *Platypus* as instances of the category *Mammal*; and given the instance *Apple* the category *Fruit* would be more likely than the category *Company*.

This additional information obviously makes it easier to deal with ambiguity and therefore helps in reasoning applications such as question answering, as it provides the KG with a kind of general knowledge about the world. However, it comes at a price, as for example queries involving existentially quantified variables are quite inefficient to compute over probabilistic databases [NMTG15].

- **Temporal and Event-Centric Knowledge Graphs**

  Most KGs are centered around entities, i.e. both the subject and the object in triples are usually entities and events are only expressed through the predicate while temporal information is relegated to attributes, if it is saved at all [RvEV+16].

  This leads to several problems, like inconsistencies in data extraction, when seemingly conflicting facts are just facts that were true at different points in time [CPSS17]. For example, in an old text that states something like "incumbent US-president Roosevelt", the extractor would have a hard time to process this correctly and not just pass it as an inconsistent fact (and label the source unreliable) since the value in the KG and probably most other sources would indicate a different current president. In the end, while it might save a fact like $(Roosevelt, formerPresidentOf, USA)$, in an event-centric KG, Roosevelt's presidency would be the subject of the triple and it would link to the entities involved and be bound to time [RvEV+16].

  Organizing the knowledge in this way allows for easier reconstruction of historic developments (e.g. it is very easy to list all US presidents this way) and to build networks of people and other entities sharing the same events. By reasoning over the temporal aspects using inference and consistency checking constraints it is also possible to derive new and implicit facts [CPSS17], allowing the KG to learn how data changes over time. For example, it could learn to differentiate between functional data like birth dates and non-functional data, through the fact that the latter changes while the first does not or that graduation dates can only happen after birth [SGEH18]. This in turn enables new ways of consistency checking.

CHAPTER 3

Knowledge Integration

A knowledge graph is usually comprised of facts that are collected manually (i.e. from human input) or (semi-)automatically - from (semi-)structured data sources like tabular or annotated data in the form of databases etc. as well as from unstructured data sources, like text extracted from the Web [SGEH18]. While extracting information from unstructured text is obviously the most challenging, it is also arguably one of the most important tasks, as a large part of data is not available in structured form. This difference between the coverage of structured data available on the Web and its demand is sometimes called semantic gap [MMZ09].

Usually, the information extraction and integration systems produce candidate facts that are assigned a confidence value depending on their source, mode of extraction etc. where data from an in-house database would usually be considered more trustworthy than data extracted from a news article, for example. Such a "raw" kind of graph is sometimes called extraction graph [PMGC13], and it may yet contain duplicate or incorrect facts and entities or violate ontological constraints.

Since manually collecting data does not require (automated) reasoning, we will focus on (semi-)automatically extracted data in the remainder of this section. However, reasoning is used on both kinds of data when approaching problems like the aforementioned erroneous data or incompleteness. For data collection to be possible in a (semi-)automatic way, it is important to first construct the underlying structure. For example, entities and entity types have to be collected in a process called term extraction [MRHLA18], which denotes extracting the relevant entities and concepts relevant to a given domain. Additionally, the relations between them have to be established - for a scientific knowledge base we would want to have entities like *MolecularBiologist* and *Geneticist* as well as a relation pertaining to the fact that both of these are subclasses of *Biologist*.

The relations and groupings can be of different natures, such as strictly hierarchical or mutually exclusive classes of entities (e.g. *Person*, *Animal*, *Company*,...) or less strictly comparable groups like topics to which the subsumed entities relate (e.g. *Politics*, *Meteorology*,...) [MRHLA18]. This process, especially when conducted in an automatic manner, is known as taxonomy induction [WT10] and the subset of facts in the knowledge base that store this kind of taxonomic data - including, for example, that an entity cannot both be in the class *Animal* and in the class *Company* - is sometimes also called the T-Box, while simple data instances are said to be part of the A-Box [GTHS15].

This task is often tackled by tapping into the existing taxonomies included in the WordNet thesaurus [Mil98] and Wikipedia. However, both of these are neither complete nor always correct, as especially Wikipedia category hierarchies tend to have nothing to do with actual subclasses. Therefore systems like YAGO [SKW07] tend to apply an additional filtering step, where they only include Wikipedia categories that can be mapped to an existing WordNet class or one that was previously added through this process.

In the following subsections, several techniques for adding both factual knowledge (A-Box) and ontological knowledge (T-Box), and from structured as well as unstructured data, will be introduced. Section 3.1 presents wrapper induction, which is a way of learning to extract information from web sources. Sections 3.2 and 3.3 then show how entities and relations, respectively, can be extracted and linked to existing concepts in the KG. Section 3.4 presents the related concept of linking entities and relationships among different knowledge graphs. Sections 3.5 and 3.6 contain a discussion of how to determine the correctness of an extracted triple, by taking into account the quality of extractors and trustworthiness of sources and by leveraging existing triples in the KG, respectively. Finally, Section 3.7 presents extraction frameworks that combine many of the previously discussed tasks while optimizing extraction performance.

3.1 Wrapper Induction

Most web sources are built from database-backed content that is used to construct semi-structured elements like HTML headings, lists, etc. Some pages like Wikipedia even contain info-boxes and a taxonomy, which makes it possible to extract the underlying data by constructing or automatically inferring wrappers for it [WT10]. Wrappers are rules or procedures that are usually specialized for each source, so as to be able to translate their content into a regular form from which values and features can be extracted [YC03].

Since most web sources present their content uniformly across all their pages, one wrapper per source usually suffices. But other web sources have different interface and output formats, which is why a different wrapper is needed for each site [YC03]. The goal is therefore to automatically learn the underlying structures from examples, like the simple contextual pattern that URLs are usually found within HTML tags of the form $< a\ href = ... >$. This process is called wrapper induction and it can be achieved through various means.

The predominant techniques used to provide some hand-crafted extraction rules in combination with a set of manually labeled web pages, from which the extraction wrappers would be learned [YWC+18]. However, this neither scales very well nor can it handle greatly differing sources, which is why more recent methods using unsupervised or semi-supervised learning have been developed [WC07]. They often do not need labeled training samples as they automatically learn wrappers from a set of similar sources [YWC+18].

Obviously, wrapper induction is harder for pages with less rigid structure and/or uncommon or complicated content, but recent algorithms are rather effective at handling this kind of complication well. DIADEM [FGG+14] is an automatic full-site extraction system that explores websites, identifies relevant data and induces wrappers without requiring supervision. Instead it uses a combination of phenomenological and ontological knowledge as well as a self-adaptive network of relational transducers in order to construct effective wrappers automatically.

3.2 Entity Extraction and Linking

entity extraction and linking (EEL), also referred to as record linkage or entity disambiguation, is the task of linking or mapping each entity mention detected in a text or e.g. on a web page with its corresponding entity in a knowledge base. This could also mean creating the entity if it is not yet contained in the knowledge base or adding the mention to a list of aliases of an entity. For example, if we come upon the word "Obama" in a text and we have deduced from the context that it refers to *BarackObama* and not *MichelleObama* or some other entity, we might add a *sameAs* relationship between *Obama* and *BarackObama* to our knowledge base, as a way of keeping track of the various surface forms that are in use for that entity [YWC+18].

While entities collected from other KGs or databases usually adhere to some kind of naming standard or may even use unique identifiers (like DBpedia IRIs), extracting them from text means dealing with a lot of ambiguity. Usually, texts are pre-processed through named entity recognition (NER), which is the task of identifying named entities and classifying them into different groups, like locations or people. The entities identified in this way are often ambiguous due to name variations (as in the example above) or multiple real world entities sharing a name. Therefore, another processing step is needed to map the entities to a so-called dictionary, a step often referred to as coreference resolution or named entity disambiguation. [DBCL18]

The dictionary maps surface forms to identifiers in a many-to-many fashion, since a surface form like "Obama" can refer to multiple entities, and on the other hand multiple different surface forms like "Barack Obama", "Obama", "President Obama" can all refer to the same entity. Coreference resolution therefore usually involves computing a so-called support or score for each candidate entity a term could refer to, either picking the one with the highest score or saving the options along with their support [MRHLA18].

This is often done by looking at other entities occurring in the input text and by leveraging knowledge already contained in the knowledge base about the different entities that could be referred to by a certain name. For example, in a text segment like “Obama’s visit to Honolulu, his birthplace, ...” we can use the knowledge stored in the knowledge graph to deduce that the text refers to Barack Obama, and not Michelle Obama, who was born in Chicago, and would also not be referred to with male pronouns. A dictionary might therefore contain contextual information to help disambiguate entities, which comes in two forms [MRHLA18]:

- **Structured contextual features.**

  These features include type or attribute information associated with an entity, related entities, the centrality of the entity in the graph structure of the KG, etc. For example, it may help to disambiguate a mention of *Boston* in a text if we know that it can be of type *City* or *MusicGroup* and to have related entities like the mayor of Boston, band members of the band, and so forth.

  Structured features are usually extracted directly from a structured or semi-structured source, often from the same reference KG from which facts were also extracted.

- **Unstructured contextual features.**

  Unstructured features like text patterns and statistics, on the other hand, have to be extracted from textual corpora. They usually provide information about patterns in text and other entities usually found in the vicinity of the entity mention in question, or statistics about how often certain entities are mentioned and in which kinds of documents etc. For example, given the above disambiguation, it would help to know that mentions of the city Boston (or cities in general) often co-occur with words like *Population* or *University* while mentions of the band would occur near words like *Concert* or *Record*.

While many approaches to entity extraction and linking are statistics based since they are closely related to natural language processing, there have been recent efforts to incorporate structured knowledge as described above in more logic-based approaches to named entity disambiguation [DS15]. This approach is based on the extraction of commonalities between pairs of entities, by comparing one or several unambiguous entities in the neighbourhood of the entity to be resolved, and comparing them to the context that is to be expected for each of the candidate options. The correct entity is then identified by using various sets of neighbor entities until no meaningful commonalities are found between them and all but one candidate meaning.

3.3 Relation Extraction

Relation(ship) extraction is the process of classifying semantic relationship mentions (i.e. in a text) - it is therefore used to establish the kinds of relationships that can exist between entities in a KG. As such, a step in the extraction pipeline involving EEL (see

Section 3.2) is usually a prerequisite for efficient relation extraction and therefore also link prediction, which is a task in knowledge graph completion involving the prediction of missing relationships between entities. It is especially important to capture coreferences, as many potential relations would be lost otherwise - e.g. in sentences like “Obama arrived in Berlin on Monday. He met with [...]” the relation information in the second sentence would be lost if we would not possess knowledge of who “He” refers to [MRHLA18].

While EEL has traditionally been performed independently of relation extraction, there are recent efforts [DBCL18] to perform them in a joint task, as this makes it possible to leverage knowledge from both processes in conjunction and maximise the evidence available for tasks like coreference resolution etc.

In general, having previously extracted the entities *BarackObama* and *MichelleObama* and given sentences like “Barack Obama’s wife, Michelle Obama [...]” and others like the ones in Figure 3.1 we would like to extract a relationship like *spouseOf* between the Obamas as well as a relationship *presidentOf* between Barack Obama and the USA. As for link prediction, which will be discussed in detail later (see Section 4.1.1), given knowledge of a relationship like *firstLadyof* (i.e. because it was also extracted but in a different context) we would like to predict whether this applies to Michelle Obama and the USA.

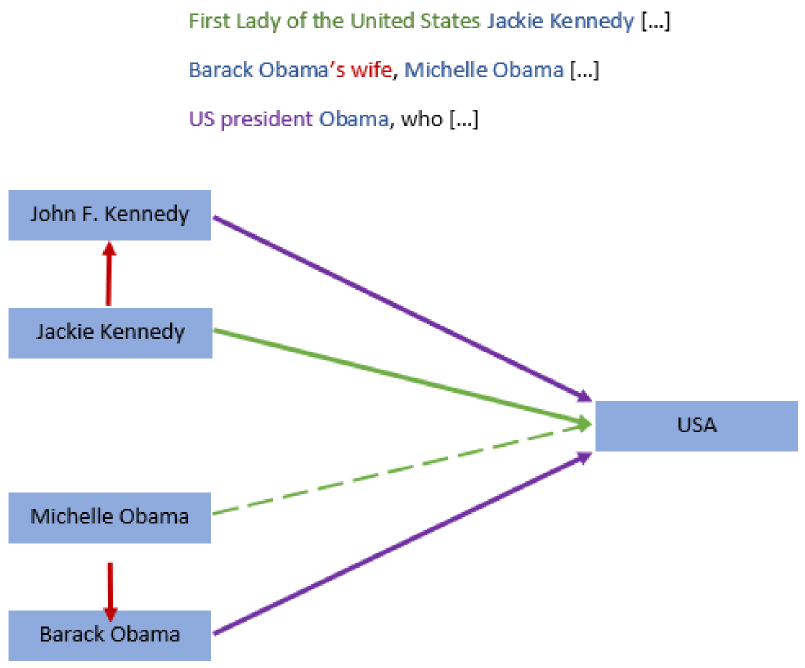

First Lady of the United States Jackie Kennedy [...]
Barack Obama’s wife, Michelle Obama [...]
US president Obama, who [...]
John F. Kennedy
Jackie Kennedy
USA
Michelle Obama
Barack Obama

Figure 3.1: An example of a relation extraction task taking into account textual information.

Well typed entities make both of these tasks easier, as a system might use the knowledge that e.g. *spouseOf* can only be a relation between entities of the type *Person* to extract relationships and link entities correctly.

Unfortunately, just saving relationships as discussed in the example would lead us to lose a lot of valuable information. Indeed a simple optimization has already been added by using the (symmetric) relationship *spouseOf* instead of simply using *wifeOf*, thereby making it easier to get the answer "Michelle Obama" in a query like "Who is Barack Obama married to?". Additionally we would save certain attributes of relationships when coming across them during the extraction, like Obama having been the 44th president of the USA and the start and end date of his tenure.

While attributes are sometimes saved in a triple like this with binary relations, it is often easier or even necessary to link more than two entities. Therefore relations are usually represented as n-ary tuples of entities. In this way, we can consider entities as atomic elements, concepts like $Person(BarackObama)$ as unary predicates, and relations as n-ary ($n \geq 2$) predicates, where some form of reification is used for tuples of arity 3 and above [MRHLA18]. In the case of Freebase, this is done via a so-called mediator, which is a relation instance that shares a triple with each of the n arguments [Gar15]. Freebase would, for example, save an abstract entity like *Performance* to link a movie, an actor starring in the movie, the character played by that actor, a performance type, etc.

There are different approaches to relation extraction, some of which will be presented in the following subsections.

3.3.1 Logic-based Relation Extraction

Many logic-based approaches use higher-level theories of language understanding like frame semantics, which postulates that humans parse sentences not just using the words themselves but also concepts they evoke [MRHLA18]. For example, in a sentence about someone receiving "payment", various related components come to mind, including the giver and the receiver of the money, the amount and the service for which it was paid. Structuring all of this information in the form of frames, the components would then be grouped around the original word evoking the frame, in this case "payment". There are various collections of frames that are meant to guide annotation of frame elements in text, most notably among them FrameNet [BFL98]. This is especially useful for parsing n-ary relations.

Another approach specialized in complex n-ary relation extraction is Discourse Representation Theory (DRT) [Kam81], which aims to structure claims made in language in a formal, logic-based way by using a FOL style form of representation. In addition to n-ary relations it is able to deal with negation, disjunction, equalities, implication and discourse spanning multiple sentences by modeling the intrinsic coreferences. This is done using existential quantification and a focus on event- or frame-spanning modeling, decomposing complex n-ary relations into conjunctions/disjunctions of unary and binary relations. For example, the sentence "Barack Obama met with Angela Merkel in Berlin in 2018" could be modeled as follows:

$$\exists e : meet(e), Agent(e, BO), CoAgent(e, AM), Theme(e, BER), Time(e, 2018)$$

3.3.2 Distant Supervision

While traditional supervised methods achieve high precision and recall on small hand-labeled datasets, this obviously does not scale well. Since accurate training data is often not easily available, some research has therefore turned in the direction of more data-driven approaches like weak [HZL+11] or distant supervision. Particularly distant supervision, which was popularized by Mintz et al. [MBSJ09], has led to a host of new research in the area of relation extraction.

The underlying idea behind this method is that existing KG relations/triples can be used to learn how relationships are usually mentioned in a text, since the phrasing often follows certain patterns. Many of these approaches are mostly used to extract new triples with previously extracted relations (see link prediction, Section 4.1.1) rather than to extract new types of relations, but the following is a short list of frameworks that are at least in part used for relation extraction [MRHLA18].

Research on this technique is very varied - PROSPERA [NTW11], for example, learns patterns in the form of POS-tagged n-grams between entities by first extracting entities and later relations in text via EEL and filtering out negative or inconsistent relations via constraint-based reasoning. Other approaches use the concept of Local Closed World Assumption (see Section 2.4)) to generate useful negative examples for training, since missing pairs can usually not be assumed to be negative examples [MRHLA18]. Fan et al. [FZZ+14] try to overcome the shortcomings of distant supervision, such as sparse and noisy features and incomplete labels in the data, by interpreting it as a low-rank matrix completion problem. They are thereby able to recover information from an incomplete set of entries and enhancing the robustness to data noise.

3.3.3 Embedding-based Relation Extraction

A shortcoming of distantly supervised methods is that they do not leverage the information contained in the structure and the reasoning capability of the KG itself, rather extracting information solely from plain text [LHX+18]. Embedding-based techniques, however, aim to identify possible relationships holding between pairs of entities by representing them as vectors, which are learned by minimizing a pairwise ranking loss between them [WMWG17].

One recent method by Weston et al. [WBYU13] proposed to use both sources, combining a text-based extractor with the embedding model TransE [LLS+15], thereby leveraging triples from the known KG. In the training phase, a text-based extractor is learned from a text corpus and a TransE model from a KG aligned to said corpus, using a ranking-based embedding framework. A text-based extractor predicts relations from their textual mentions by scoring the similarity between each relation and its textual mention, while the TransE model scores the probability for each missing fact between two entities to be true. This way, relation mentions, entities as well as relationships are embedded into one low-dimensional vector space, where composite scores are calculated, favoring predictions that agree not only with the textual mentions but also with the KG.

The framework by Riedel et al. [RYMM13] also jointly embeds plain text and KGs, they are however represented in a matrix. Each row of the matrix stands for a pair of entities, which were retrieved from both pre-existing structured databases and textual corpora. Each column stands for a relation, which comes from the union of textual surface forms and database/KG relations - the corresponding entries are set to one if the two entities co-occur with the relation and to zero otherwise. Collaborative filtering techniques are employed to factorize the input matrix in order to learn vector embeddings for textual mentions, relations and entity tuples. The model is then trained on both textual mentions and KG relations, so that it learns correlations between them and is then able to extract relationships solely from text input.

3.4 Schema Alignment

A related problem is schema alignment (also referred to as ontology matching or link discovery), where information in the form of facts or just entities/relationships from existing knowledge graphs is (partially) incorporated by mapping it onto the knowledge graph that is to be constructed or enhanced. This is often achieved by comparing attribute values of records, like birthday and location for people, and assuming they refer to the same entity if the attributes exceed a certain similarity measure.

While it would be desirable to build KGs in a way that includes links to external knowledge, maybe even including IRIs or identity links, only about half of the datasets added to the Linked Open Data (LOD) Cloud are linked with others, and in many cases only sparsely [NHNNR17]. There are, however, efforts to change this, as links could enable or aid in processes like federated queries or complex question answering tasks, among others. Unfortunately, due to the often large amounts of data in KGs like DBpedia [LIJ$^+$15] or YAGO, even a relatively fast exhaustive search comparing any two entities between them would take years to finish [NHNNR17]. Additionally, while some entity types, like countries or books, naturally come with unique identifier schemes, which makes linking them across KGs easier, most entity types do not have anything of the like and the entities are often referred to with (greatly) varying names and identifier schemes.

In the related task of link prediction (see Section 4.1.1), local examples of the links to predict already exist and can be learned from. However, due to the situation stated above, for schema alignment it is often necessary to manually add linkage rules or a small set of alignment seeds, i.e. example links between synonymous entities in different KGs, to start the alignment process [HBC$^+$20].

Since manually defining explicit linkage rules specifying the conditions entities have to meet so as to be considered to refer to the same real-world object is prohibitively inefficient, especially in KGs with widely varying types of data, there are now efforts to generate such rules automatically. GenLink [IB12], for example, is a supervised learning algorithm which uses genetic programming for learning expressive linkage rules from a set of existing record links. The generated rules can select attributes for comparison that are likely to be discriminative, normalize these attribute values, evaluate their similarity based on appropriate measures and thresholds and aggregate comparison results.

Another approach is to use embedding-based alignments, which calculate the similarity between embeddings of pairs of entities. Systems like IPTransE [ZXLS17], for example, encode the entities and relations of multiple KGs into a unified low-dimensional semantic representation space under a joint embedding model. IPTransE is based on one of the most widely-used knowledge embedding methods TransE [LLS+15] (see Section 6.2). It makes use of a small seed set of aligned entities, so as to be able to embed entities according to their semantic distance, resulting in a semantic space where entities with similar or identical meanings tend to be close together.

A similar idea can also be used to simplify merging ontologies by mapping them into the same semantic vector space [SS17]. The advantage of such an approach is that even if concepts are not mapped to the exact same term but only to a nearby one, that can still be good enough for finding equivalences and enabling chains of reasoning. For example, if one ontology contains the statement $(Bears, eat, Salmon)$ and the other contains $(Fish, liveIn, BodyOfWater)$ then even the connection between bears and rivers could be made based on the semantic similarity of concepts.

Additionally, there are approaches that try to add multi-modal information in the KG as well as during the alignment process [LLGD+19]. In this case, images are added to most entities and embeddings of these images are later used during alignment along with other "experts" (i.e. different similarity measures) like an embedding expert, a relational expert and a numerical expert whose scores are combined. This multi-modality reportedly leads to stronger results while also amplifying the available data in KGs.

3.5 Data Fusion and Knowledge Fusion

Ideally, when querying multiple sources, they would all return the same values for any given fact, like the number of people at the inauguration of a politician. Unfortunately, this is often not the case and we have to deal with the problem of resolving those conflicting values and, if possible, of actually finding the underlying true values.

Data fusion approaches were typically rule-based, like using the value from the source that was most recently updated, or calculating a metric for numerical values, like average/median or maximum/minimum. Some recent solutions have applied semi- or unsupervised learning to this task and can roughly be classified into three classes [DGH+14b]:

- **Voting.** Voting is the simplest strategy and therefore mostly just used as a baseline. Here, the value that is being returned by the most sources is taken.
- **Quality-based.** Quality-based methods measure trustworthiness of sources according to different evaluation techniques, like Bayesian methods that calculate the probability of each of the values being true, or methods that calculate the

source trustworthiness based on how close its values are to the correct values on known instances. They then assign a higher score to facts originating from more trustworthy sources.

- **Relation-based.** Relation-based methods (additionally) consider the relationships between sources, i.e. giving lower credence to sources that copied from others or counting values that appear multiple times, but in closely related subsets of sources, only once.

Knowledge fusion [DGH+14b] is a related concept, where we are trying to identify which of the candidate triples extracted from multiple sources by multiple extractors are true. Since both extractors and sources might provide conflicting information, binary decisions are hard to make with any certainty in this context and a triple is instead assigned a truthfulness probability between 0 and 1, which can be used in multiple ways. Triples with high probabilities can be assumed to be true and used in applications, while triples with lower probabilities can be used to improve the extraction and learning systems, i.e. by using them as negative examples.

Compared with data fusion, where we only have to consider the two dimensions of value and source to make a binary decision, in knowledge fusion we also want to evaluate different extractors and output a probability score. This means there is an additional potential error source, as triples are not necessarily extracted correctly or linked incorrectly in a previous processing step. A simple way to mitigate this is by assigning a higher score to triples whose value is supported by multiple sources and/or extractors.

More advanced models try to estimate the accuracy of classifiers using unlabeled data [PSNK14] and some methods also incorporate logical constraints [BBM13], in the case of Platanios et al. [PPMH17] in the form of probabilistic logic. Logical constraints help to guide the classification into target classes, e.g. by marking mutually exclusive classes as such. In general, agreeing classifiers are assumed to be more likely to be correct and when classifiers make predictions that are violating constraints at least one classifier is assumed to be making an error. For example, in the NELL project [CBK+10], if two classifiers which predict whether a noun phrase represents an animal or a city, both classify the noun phrase as belonging to their respective category and these two categories are mutually exclusive according to the used ontology, at least one of the classifiers must be wrong.

These logical constraints can be "hard" or "soft", i.e. strongly enforced or only enforced in a probabilistic manner, respectively. It is, for example, rather unlikely that a noun phrase refers to a company as well as a fruit, but we all know at least one that does. Platanios et al. [PPMH17] use these rules as priors when performing probabilistic inference, so as to be able to approximate both the accuracy/error rates of the classifiers as well as the underlying, correct classifications, which are not observed. They do this using probabilistic logic, which combines classical logic with probabilistic reasoning. Instead of being boolean, truth values of rules, their ground predicates etc. are continuous and lie in the interval [0, 1], representing the probability that they are true. This also means that the boolean logic operators (AND($\wedge$), OR($\vee$), etc.) have to be redefined.

3.6 Triple Classification

Triple classification is the task of determining the correctness of an unseen triple fact (h, r, t). While it is normally seen as a binary classification problem, it is usually based on a probability calculated through a scoring function as well as a threshold defining the value from which on a fact is assumed to be correct. [JPC+20]

If an embedding model (see Section 6.2 has already been learned on the knowledge graph, any triple can be evaluated as long as h, t $\in$ E and r $\in$ R, i.e. both entities and the type of relation are already known to the KG [WMWG17]. The task then only involves checking all the candidate triples against the model and calculating their score, where higher scores indicate true facts. It then additionally has to be established which threshold is used, but it is possible to use different thresholds for different relations. As it would depend on how well we expect the KG to be able to reason about the relation, a lower threshold could be used for relations where the data is rather sparse. However, as pointed out in [JPC+20], vector-based embedding usually can not deal with 1-to-n relations.

Suchanek et al. [SSW09] proposed SOFIE (Self-organizing framework for information extraction, which was developed to enable automatic growth of YAGO [SKW07] while retaining its high level of near-human quality. It uses a set of weighted clauses representing known and candidate facts, constraints etc. and aims to solve the problem of finding truth values so as to maximise the total weight of satisfied clauses. Although that is NP-hard, it has many good approximation algorithms as it can leverage the plethora of work done in the area of (Max)SAT solvers. Additionally, since it uses entity typing based on YAGO, it can easily falsify wrongly-typed facts such as ParisHilton capitalOf France, while also making grounding more efficient, since only constants of the correct types have to be considered.

Another approach, using reinforcement learning (RL), is presented in [HSM+20]. They use debate dynamics, framing the task as a game between two agents of which one is trying to prove the fact and the other is trying to disprove it. To this end, they each extract paths from the knowledge graph that support their argument, which is then judged by a binary classifier as being true or false. Although it is a black-box method, the arguments produced by the agents provide interpretable evidence for both sides and can therefore help the users understand the decision.

3.7 Declarative Extraction

While there are systems that combine many of the extraction tasks discussed so far into monolithic black boxes that can be employed off-the-shelf, this severely limits the expressiveness and customizability of the programs that are to be developed [SDNR07].

A popular approach is therefore to use separate systems for each of the extraction tasks, sometimes also in the form of off-the-shelf black boxes but alternatively by using hand-crafted solutions where customization is needed. The results are then combined and subjected to some post-processing.

Although this approach is quite powerful, it generates large programs that are hard to understand, modify, debug and optimize. Optimization, however, is increasingly important in the context of information extraction and knowledge graphs, as the huge amounts of data involved can mean that suboptimal systems are simply not usable in practice, running for days or weeks.

Frameworks like Cimple [DRC+06] aim to solve this problem by providing an end-to-end rule-based extraction framework that is based on database-style declarative Information Extraction systems. Non-declarative program parts, e.g. for text-pattern analysis, can be encapsulated into declarative programs in a Datalog-based language called XLog [SDNR07], with constraints formulated as first-order logical rules to check both existing and candidate facts from the extraction process for consistency.

They demonstrate how query processing for extraction and consistency-centered inferencing can be combined into a unified framework. This enables a largely automated construction and maintenance of knowledge bases by providing a plug-and-play approach. Additionally, they provide optimization based on the respective data, which can lead to speedups of several orders of magnitude.

3.8 Discussion on the Chapter

In this chapter we surveyed various approaches that are current state of the art for extracting knowledge from heterogeneous sources both on and off the Web. We further discussed approaches for integrating the data collected from structured, semi-structured and un-structured, or semi-structured sources into a uniform representation like knowledge graphs. This has been done by gathering and analyzing a representative selection of relevant research chosen from well-known scholarly search engines.

We have looked at how to automatically induct rules that can be used to extract information from web pages and other (semi-)structured sources in Section 3.1 on wrapper induction. We have then shown how entities and relations can be extracted and linked to concepts in the knowledge graph in Sections 3.2 and 3.3 on entity and relation extraction, respectively. After that we discussed the related concept of linking relationships and entities to those in other KGs.

Subsequently, we have talked about how to deal with conflicting sources and extractors in Section 3.5 on data and knowledge fusion and discussed how to leverage existing triples in the KG in order to estimate the truthfulness of extracted ones in Section 3.6 on triple classification. Finally, we presented how extraction frameworks can optimize extraction performance when combining multiple tasks in Section 3.7 on Declarative Extraction.

All of these subtopics have been chosen as the most relevant because they represent different strategies of dealing with data scattered across the Web, with different types of linked, open data that have to be taken into consideration [BL5]. Like Tim Berners-Lee, this chapter aimed to answer the question of how to unify the various types of data that can be found on the Web into an all-encompassing representation that can deal with the diversity arising in this context.

CHAPTER 4

Knowledge Graph Evolution

Even though knowledge integration techniques are often quite sophisticated and aim to reduce the number of errors, extraction graphs usually still contain false or duplicate facts (spurious and missing nodes and edges, and missing or inaccurate node labels). This is compounded by KGs becoming so big that they are themselves mined for information, thereby propagating errors [GTHS15]. Because of this and the nature of incompleteness of knowledge graphs, some post-processing is needed to improve the quality and quantity of facts contained in a knowledge graph.

This post-processing is known under various names, like identification, refinement, discovery, or the term used in the context of this thesis, evolution. Pujara et al. [PMGC13] define knowledge graph identification as "the task of removing noise, inferring missing information, and determining which candidate facts should be included into a knowledge graph". While this definition is a good starting point, it should be noted that these concepts also apply to the improvement of the body of ontological knowledge in the form of rules that a knowledge graph contains.

Many tasks that would not strictly be considered knowledge integration, as they do not just aim at gathering data but also at improving it, are now already part of the extraction pipeline. They can therefore not easily be classified as either fully part of knowledge integration or knowledge graph evolution, which aims at improving a given knowledge graph, with the main difference being that for the latter knowledge from the KG is leveraged as well [NRM15]. Consequently, this classification is only tentative and some of the techniques already described in Section 3 will factor in this section as well, but with a greater focus on more sophisticated reasoning techniques.

The following section will therefore provide an overview of the reasoning tasks that are necessary for the refinement of knowledge graphs. Knowledge graph evolution can be divided into the subject areas completion and correction, which will be discussed in Sections 4.1 and 4.2 respectively. Completion aims at adding missing knowledge to the

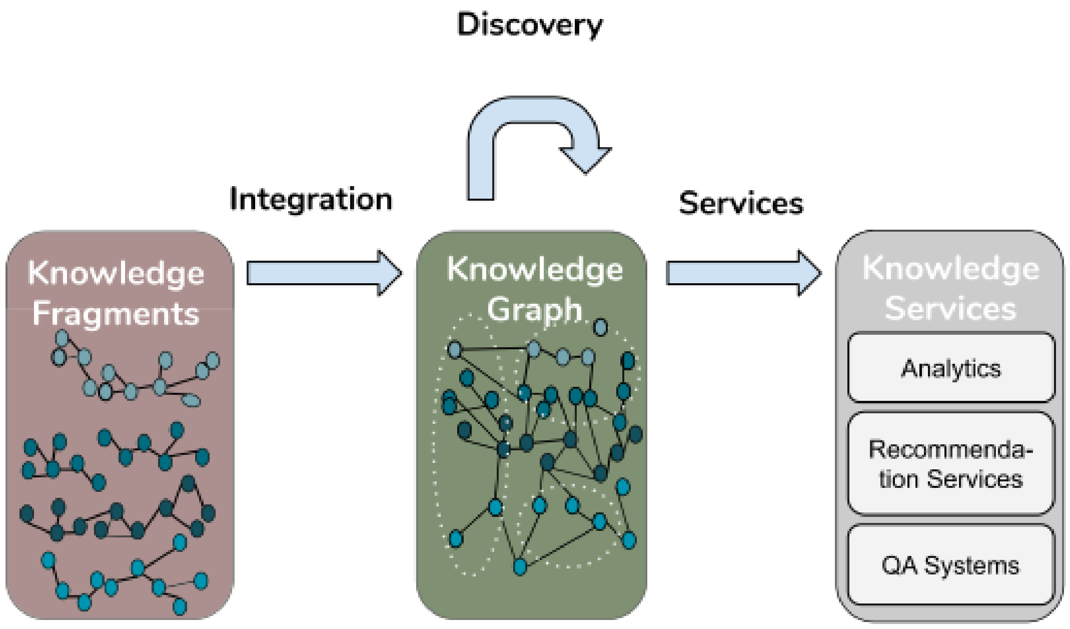


Figure 4.1: A schematic of the life cycle stages of a knowledge graph, showing how knowledge integration and evolution (called discovery here) are intertwined. "Knowledge Services" are referred to as "Applications" in the context of this thesis. This figure was taken from Bellomarini et al. [BSV20]

KG, thereby making it more comprehensive, while correction aims at improving the quality of the existing knowledge by reasoning about it, resolving conflicting data points and removing incorrect data.

4.1 Knowledge Graph Completion

Knowledge graph completion is the process of adding missing triples or parts of triples to a knowledge graph, i.e. ones that are deemed true but are neither given nor entailed by the KG yet. Due to their incomplete nature, there is a huge number of potential facts that could still be included in a knowledge graph. Since we are not generally interested in just adding any facts, due to limitations in storage and computing power and since many KGs are specific to a certain field, the real task in KG completion is not just in finding facts, but in deciding which ones to find and keep.

Many of the typical subtasks, like entity/relation prediction, entity/triple classification etc. have already been covered in Chapter 3 and will only be featured again in the second part of this section about types of reasoning, when talking about specific systems that implement them. Instead, the focus here is on link prediction, which is the task of predicting missing relationships between the entities in the KG, and rule learning, which aims to find and make explicit the patterns in the data, and is sometimes used to aid with link prediction.

4.1.1 Link Prediction

Arguably one of the most important tasks in the context of KGs and therefore a focus of much of KG-related research, is link prediction [BDPP19], which aims to assess missing relations between entities. It comes in two forms - relation prediction aims to find relationships r that hold between two given entities h and t, i.e. (h, ?, t), and entity prediction (or entity ranking) aims to find entities t that are connected via a (given or any) relationship to a given entity h, i.e. (h, r, ?) [WMWG17].

The task can be viewed from multiple angles, and some of the tasks already discussed in the context of knowledge integration (see Section 3) can also be seen as special cases of the link prediction problem that are concerned with specific kinds of links. Entity classification aims to learn type links, while for entity linking and schema alignments the goal is to learn identity links, and most general purpose link prediction techniques are concerned with learning regular relationship links between entities.

- The most general case involves edges with regular relationships between entities, like the example in Section 3.3, predicting the relationship *firstLadyOf* between *MichelleObama* and *USA*, based on other connections within the KG and given previous knowledge of the relation. Most of the techniques surveyed later in this section will concern the general case, although many could be used for the following types of link prediction as well.
- Type links, like (*BarackObama*, *isA*, *Person*), which categorize entities into semantic categories, are often contained in the KG in a similar way as general links. This task has previously been mentioned as term extraction in Section 3, but it is also sometimes known as entity classification [WMWG17].
- Identity links connect entities with edges denoting a *sameAs* relationship, i.e. they indicate that two nodes refer to the same entity [HBC+20]. This concept has already been touched upon in Sections 3.2 and 3.4 in the context of entity linking and schema alignments, respectively, where links between entities within the same or across knowledge graphs are established [HBC+20].

While all of these tasks can be addressed with general link prediction techniques, the particular semantics of the more specific tasks can often be better addressed with custom techniques [HBC+20]. When entity and relation representations have already been learned (see Section 6.2), link prediction often comes down to a ranking procedure. For example, when predicting a relation between a given entity pair (h, ?, t), one can take every relation known to the KG as a candidate answer, calculate a score for each one, and pick the one with the highest score [WMWG17].

These ranked scores can also be used to evaluate a system, as the rank of the correct answer(s) can be recorded to see whether it/they rank before incorrect ones. Given *BarackObama* and *USA* as h and t, respectively, if our system were to return a ranked list like (*presidentOf*, *primeMinisterOf*, *citizenOf*,...), the correct answers on rank 1 would be positive, but one should be worried about the erroneous relation on rank 2 appearing before another correct one on rank 3. There are various evaluation metrics that can be used for such ranked systems, e.g. mean rank, mean reciprocal rank (the average of reciprocal ranks), AUC-PR (the area under the precision-recall curve), and Hits@n (the proportion of ranks smaller than n) [WMWG17].

Link prediction can also be addressed using techniques from the area of statistical relational learning (SRL), which predict the probability of correctness of missing edges and relations. However, as this method, along with its applications, will be covered in more detail separately (see Section 6.4.4), the focus in the following paragraphs will be on other techniques.

Statistics-based methods. Link prediction can be interpreted as a classification problem that can be solved with machine learning methods, such as vector embedding- or tensor/matrix decomposition-based methods [MQH+19]. They allow to condense more complex representations like graphs into feature representations of varying dimensions by trying to only encode the relevant part of the data.

In many practical applications, there is a focus by users on some parts of the domain, which can be used for personalized tensor decomposition methods [LHCS14], as they are computationally complex, especially for dense data sets. Even though most KGs are rather sparse, therefore making this method more usable, being able to customize the decomposition process to specific areas of interest makes it possible to trade accuracy for performance in a targeted way. Accuracy can be very high for focused areas, while for the rest of the KG the focus in on speed. On average, the method is only slightly above average in performance as well as accuracy, but it is distributed in a way such that the significance of important data is taken into account.

A similar idea lies behind matrix decomposition methods, which aim to divide the link prediction problem into subproblems of smaller size, so that they can be solved efficiently, e.g. through the use of latent factor methods [DAM+16]. This ensemble enabled approach can enable performing a global search over the entire graph instead of having to limit it to specified subsets of links, even on very large KGs.

One of the most straightforward approaches is to represent each link prediction as a translation vector from the head entity to the tail entity in what is called a translational distance model, the most well known example of which is TransE [BUGD+13]. Both entities and relations are embedded into the same space, so that for each triple (h, r, t) that holds, $h + r \approx t$. The scoring function can then be defined as the (negative) distance between $h + r$ and t [WMWG17], leading to a ranked list as described above.

While most statistical approaches, and especially vector space embedding, are relatively effective and scalable, one of the main problems with them is that the results are not usually explainable [BDPP19]. Also, while there have recently been some efforts to include the capability to reason over multi-step relationships, this was for the most part not possible in early research efforts [JPC+20]. The following techniques aim to ameliorate these problems at least to some degree.

Path-based reasoning methods. One of the first link prediction techniques based on paths, or more specifically, random walks through the graph, was the path ranking algorithm (PRA) [LMC11]. It learns specific relation path features and to classify them using logistic regression by training a model for each relation r in the KG. Given an entity h, the aim is to find all other entities y which potentially are connected to h via r. Known triples (h, r, t) are used as labeled positive examples in the training process so as to learn to predict other triples of the form (h, r, t').

While most embedding-based methods do not take paths into account, there are some that try to combine the merits of both approaches, as the path ranking algorithm (PRA) in particular struggles with data sparseness [MQH+19]. PTransE [LLL+15] does not just consider direct (one-step) relations between entities as translations for representation learning, but also multi-step relation paths, which are represented via semantic compositions of relation embeddings.

Das et al. [DNBM16] proposed a model that can deal with longer paths, using the semantic vectors of the binary relation of the path to represent distributed vectors of entities. They learn to jointly reason about entities, entity types and relations using recurrent neural networks (RNNs) combined with multi-step inference. These combined approaches (see Section 6.4) have the advantage of being able to also predict relations that do not appear in the supervised training set and, like all path-based methods, have good interpretability [MQH+19].

Weak/distant supervision. As mentioned in Section 3.3 weak or distantly supervised learning is also often used to predict relationships between pairs of entities. This technique aims at finding the underlying patterns in the way relationships are usually mentioned in texts. If the triple $(BarackObama, presidentOf, USA)$ is already contained in the KG and can be linked to a sample mention like "President Obama gave a press conference in front of the White House today, ...", other text can be searched for similar patterns to add more triples of the form $(x, presidentOf, USA)$ [MRHLA18].

4.1.2 Rule Mining

Since it would be difficult to manually define a complete set of rules that hold in a given body of knowledge - especially since one might not even be aware of some of them - mining frequent patterns from a KG and inferring new knowledge from it in the form of rules is an important task in the context of KG completion. Mining such rules is the underlying principle of both logic-based and path-based reasoning methods and is often also incorporated into statistics-based approaches because of its merits, prime among them explainability of inference results. Also, even when statistical data mining methods are used, rules, e.g. in the form of ontologies, can help to make them faster by pruning useless candidates earlier [JŁŁ10].

The resulting rules are of many different forms, not just between the areas of logic-based systems and path-based systems, but also within them. Different kinds of rules will be surveyed in detail in Section 6.1, but some of them are more relevant to KG completion than others. For example, learning descriptive rules might be useful for KG correction of and for analyses over the data, but to be able to add new data to the KG, prescriptive rules are more important. Similarly, deductive reasoning is very useful for extracting implicit knowledge from the KG that might be considered "common knowledge" and is therefore used for many knowledge application tasks, but the focus in KG completion is on inductive reasoning, i.e. the automatic extraction of hypothesis about the data from knowledge contained within the KG.

Rule mining in the context of KG completion can be seen as a generalization of link prediction, where not just links between specific entities are predicted, but rather between classes of entities, and general rules and patterns within the data are unearthed and made explicit. This induced knowledge can indicate new knowledge but it is potentially imprecise, as the data it is extracted from might contain errors or biases. For example, a KG containing facts about the heads of state of different countries might only or disproportionately contain data on male heads of state (which is not too unlikely, given that the vast majority of them is male) and therefore learn an erroneous rule that predicts only male heads of state.

Jozefowska et al. propose mining frequent patterns in a language combining ontologies and rules, or more specifically combining description logic with DL-safe rules [JŁŁ10]. They aim to understand how the choice of different settings and semantics of the representation formalism impact the task of frequent pattern discovery.

The most well-known and one of the most efficient rule miners is AMIE+ [GTHS15], which is a logic-based approach. But recently there have been some techniques using embedding-based approaches [NRM15, YYH+14], since they promise to provide better performance on very large knowledge graphs. Omran et al. [OWW18] aim to combine both ideas and further improve upon them, by using embeddings to reduce the search space and thereby guide the extraction of rules.

4.1.3 Question Answering for Knowledge Graph Completion

While question answering is mostly used as a way to run queries on top of the knowledge contained in a KG and as such is described in detail in Section 5.2, there are recent techniques that aim at using it for KG completion, in particular to learn specific attribute values of entities [WGM+14]. However, in this case the goal is not to learn how to answer questions, but rather to learn which questions to ask so as to get the right answer, i.e. the missing data point.

The underlying idea is to leverage existing question answering systems, like web search engines, to fill in gaps in the KG in a targeted way [WGM+14] by learning which questions need to be asked to maximize the probability that the answers contain the correct values for the given attributes. For example, if the birth date of a person is not contained in the KG but the birthplace is, the system would learn to include it in the query for the birth date since the attributes are related and often mentioned together, while e.g. adding the gender would probably not help in this case.

Usually multiple different questions will be posed and the results compared, so as to decrease the likelihood that the phrasing or specific context lead the system astray. This could happen, for example, when they include a rather obscure science term that also has a different, more popular meaning that completely dominates search results, as would be the case for the Sonic the Hedgehog gene. However, asking too many questions can also hurt accuracy by introducing false positives, so the system needs to learn how many queries to ask for each attribute.

There are different techniques to filter and aggregate the returned candidate answers, the simplest of which would be a majority voting mechanism. In the end, a probabilistic prediction should be returned for all the possible values of a given attribute and above a given threshold probability, the highest scoring candidate can be included into the KG [WGM+14].

4.2 Knowledge Graph Correction

Although accuracy of extracted facts is usually an important factor in knowledge integration and completion, the focus is nonetheless often on adding as many (relevant) facts as possible. Especially for heuristic methods, this trade-off between coverage and correctness can not be avoided, as even with a strong focus on correctness it is unlikely that the KG will be fully correct [Pau17]. Unfortunately, spurious facts can be just as detrimental to the usability of data in KGs as missing facts, if not more, since they can lead to all results becoming less reliable, e.g. by causing reasoners to infer incorrect statements [FK15].

While steps for KG correction can be taken alongside those for knowledge integration and KG completion and by the same teams, it can be beneficial if the tasks are tackled independently of each other, i.e. by an outside group taking an existing KG and trying to increase its correctness through various means [Pau17]. This way, methods are more likely to work on other KGs as well and not be - voluntarily or involuntarily - optimized for one KG alone, and the results can allow for a better understanding of the effectiveness of different approaches because there is a cleaner separation of effects.

The main approaches for KG correction are fact validation, which tries to assess the plausibility of a given edge, and inconsistency detection and repairs, which use ontological axioms to resolve inconsistencies found in the KG. For fact validation in particular, external sources like text corpora etc. are often used to help make decisions [HBC+20]. While there are also approaches that use external knowledge in the form of human input [Pau17], such as crowd-sourcing and game-based approaches, they are not covered in the following subsections, as their application does not involve automatic reasoning.

4.2.1 Fact Validation

Fact validation or fact checking is the task of assigning a plausibility score to edges and possibly deleting them if their score falls below a certain threshold. The task is related to triple classification (see Section 3.6), with the difference being that the KG can be used as a basis for reasoning over triples for fact validation [Pau17]. That is often not yet possible for the knowledge integration task of triple classification, or rather only in the form of the given ontology, since depending on the stage of knowledge integration the triple is encountered at, there might not be many other triples present in the KG yet. Another minor difference is that triple classification involves the decision of whether or not to include the triple in the KG rather than whether or not to delete it, but there will obviously be a lot of overlap in techniques.

Fact validation is also related to link prediction - a fact can also be seen as an edge in a KG [SW17] and therefore both methods rely on assessing the plausibility of edges. To use link prediction techniques for fact validation, edges are often "deleted" and then the plausibility of the missing edge is assessed. Shi et al. [SW16b] propose a method that learns alternative paths for a generalized statement of the edge in question to verify it. For example, given a statement like $(\mathit{Paris}, \mathit{capitalOf}, \mathit{France})$ it would first use the known ontological type information to generalize it to a statement like $(\mathit{City}, \mathit{capitalOf}, \mathit{State})$ and then find known patterns that hold for such pairs of entities. For example, capital cities are more likely than just general big cities to be the seats of government agencies, and they are more likely to have an airport than other cities their size.

With this "transformation" between fact checking and link prediction in mind, obviously many of the same numeric- and symbol-based techniques can be applied for both problems, but in general fact validation more often considers online and out-of-KG assessments as input [HBC$^+$20]. Because of this, the approaches will be split into those using only internal knowledge and those also using external knowledge, which may be unstructured or structured in nature:

Fact validation using internal knowledge. One of the ways that data contained in a KG can be used to validate facts is through outlier detection, i.e. the identification of instances that deviate from the charactaristics of the majority of the data [Pau17]. Since this approach usually deals with numeric data, numeric literals are an obvious target for it. An example of it is used in Wienand et al. [WP14], where various univariate outlier detection methods are applied to DBpedia. While outliers are not automatically errors, but can also be unusual correct data points, a vast majority of the indentified outliers were actual errors in DBpedia, resulting from number format and string parsing errors etc.

Similarly, outlier detection can also be used for finding erronous type assertions by computing the characteristic distribution of subject and object types for each relation and marking edges in the graph with strongly deviating subject and/or object types as potential errors [PB14]. It can even be used on the KG structure, e.g. by encoding the interlinks between KGs in a multi-dimensional feature vector with the types of the respective entities as binary features [Pau14]. Such a model could then learn solely through methods like cluster-based outlier detection that a *sameAs* assertion between entities of such radically different types as *book* and *animal* is implausible, since the combination is bound to be infrequent in the overall distribution of all links [Pau17].

Fact validation using unstructured external knowledge. Most works on fact validation use external reference sources and when these sources are unstructured there is often the need for a verbalisation function that can translate edges into natural language and vice versa [HBC$^+$20]. DeFacto [LGMN12], for example, is a system that uses a database of lexicalizations for relations so that it can transform statements into natural language sentences and feed them to a search engine. Sentences are then judged according to the amount of web pages that is found supporting them and assigned a confidence score accordingly.

Once such a verbalisation function is established, e.g. using rule-based approaches or encoder–decoder architectures, so-called fact finders [PR11] can be used to compute the plausibility of facts. Many such fact finding algorithms construct a bipartite/n-partite graph, with fact-nodes on one side, source-nodes on the other, and an edge between a given source and fact if the source provides evidence for said fact, i.e. because it contains - with sufficient confidence - a matching text snippet for the verbalisation of the given edge.

Subsequently the plausibility of facts and the trustworthiness of the source are calculated in a mutually-dependent way. This computation is done differently by each fact finder, but it has been generalised into a single multi-layered graph-based framework [PR11], which also takes into account the similarity between sources and facts and uses weighs to express uncertainty when it comes to a source supporting a fact.

Fact validation using structured external knowledge. When it comes to structured sources used for fact validation, many systems leverage the rich information of other knowledge graphs, often by trying to find paths that provide evidence for a given fact. Some approaches extract features from other KGs that they can use to train classification models to assign edges a validity score, just as they would with internal knowledge. An important set of such features are metapaths [SHY$^+$11], which are type-based "patterns" of paths that can be explicitly specified by users or automatically learned, as in the case of PredPath [SW16a]. For example (see Figure 4.2), a metapath between a president and his or her vice president might consist of the types *Politician - Country- Politician* as in

$$(KamalaHarris, vicePresidentOf, USA)$$
$$(JoeBiden, presidentOf, USA)$$

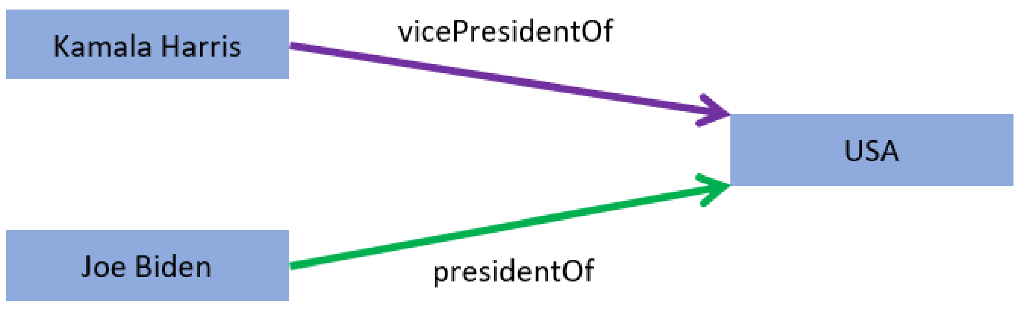


Figure 4.2: An example of a metapath connecting a president and their vice president

Another approach is to use interlinks between KGs (see schema alignments 3.4) for detecting erroneous numerical values. Links between identical resources are used by comparing their properties in the respective sources using various matching functions [LdM15]. E.g. if multiple other sources have a consensus for a value other than the one in the given fact, it is assumed to be wrong.

4.2.2 Inconsistency Detection

While other facts can often help to classify given facts and thereby possibly detect inconsistencies, knowledge graphs by design contain another resource that can be leveraged for this task, namely their ontologies and the axioms they contain. These define the possible types of edges and nodes and the constraints that hold on them, like disjointness between two concepts [Pau17]. They are a kind of fortified, rule-based version of the type of knowledge that also goes into fact validation - in fact, ontologies are sometimes learned (at least in part) from the data in a KG. In contrast to the primarily statistics-based approaches used in fact checking, a sufficiently rich ontology makes it possible to also leverage logic-based reasoning.

Type and disjointness constraints can be used to discover new information and detect inconsistencies in a straightforward way, for example, a fact like

$$(BarackObama, marriedTo, Hawaii)$$

could easily be identified as a contradiction by a strong enough ontology, given that Hawaii is a state and states and persons are disjoint, but *marriedTo* is a relation that is limited to entities of type *Person*. However, the ontologies of many knowledge graphs are not rich enough to perform this kind of inference, so many approaches trying to exploit reasoning for inconsistency detection are combined with methods aiming at enriching ontologies [Pau17].

This can e.g. be done through statistical approaches, as proposed by Toepper et al. [TKS12], whose method aims to derive missing domain and range restrictions as well as class disjointness axioms from the DBpedia dataset. For example, they fill in missing domains by taking the type that occurs most often, using the most specific one if two types are equally frequent and in a subclass relationship and a generic *Thing* class if there is no clear dominating type. For disjointness axioms they use the Vector Space Model to map all classes to vectors in the same dimension and compute the similarity of two classes based on their relative position therein. If the similarity score is sufficiently low, disjointness can be assumed.

Tools like OntoCmaps [ZGH11] aim to extract deep semantic representations in the form of axioms from corpora in an unsupervised way by applying open ontology learning. To rate the correctness and importance of the extracted axioms they use filtering mechanisms, such as voting schemes, that are based on metrics used in graph theory.

Another way to uncover implicit ontological knowledge from data is association rule mining (see Section 6.1.1 for a more in-depth consideration). The ORE system [LB10] learns formal descriptions of classes from inferred instances in the ontology, i.e. they find an expression that exactly covers all instances of a class. These learned concepts are then candidates for adding class equivalence of subset relation axioms to the ontology, or - in case they are already present - to modify them through specialization/generalization.

Ma et al. [MGWQ14] also use association rule mining to learn disjointness axioms. They focus on subclass relationships, since these are often useful for determining whether a negative association rule of the form $A \rightarrow \neg B$ (which in this context is equivalent to $A \cap B = \emptyset$) is correct or not. They first mine positive association rules to learn subclass relationships (i.e. get all the parent classes of a given concept) and then build a transaction table showing which instances are in which classes and whether there are overlaps, which they then use to mine negative association rules. Even though they aim to always use the most general form and therefore take parent classes where applicable, the generic class *Thing* is excluded from results, as the rules containing it would not be meaningful.

While for association rule learning to yield meaningful results the data set has to be of high quality, i.e. representative of the real world, which is often not the case, the authors found that a lot of wrong classifications came down to simple errors. For example, there were various classes in DBpedia that were only overlapping in one individual (often at least one of them erroneously extracted or classified), which led to their types not being classified as disjoint. Therefore, a reasoner would often just state a possible contradiction, i.e. that one out of a few axioms in the knowledge graph has to be wrong, and a human expert would decide how to resolve it or whether a new axiom would be added to the ontology in the end.

Inconsistency Repairs In some ways, repairing a KG once a reasoner has turned up possible inconsistencies is even harder than finding them in the first place. Even in a simple case like an entity being classified as two disjoint classes, repairing the inconsistency means determining which class is the incorrect one, which is not a trivial task [HBC$^+$20]. This is exacerbated by the fact that one edge can be involved in multiple inconsistencies and vice versa, and that often not just the erroneous type has to be removed, but it has to also be prevented from being re-entailed in case it was the result of an erroneous axiom. As mentioned before, it often comes down to a human expert having to at least decide between one of the several ways a system suggests the inconsistency could be repaired through, like removing a domain/range/disjointness constraint, removing a type membership, etc [TKS12].

One of the few methods that automatically performs inconsistency repairs was proposed by Bonatti et al. [BHPS11]. Their logical framework annotates inferences with varying "strengths", depending on the trustworthiness and centrality of the source, to judge facts. It then uses minimal hitting sets, i.e. sets that are a minimal explanation for an inconsistency. The choice which edge to remove is then based on the calculated trustworthiness of their sources as well as the amount of minimal hitting sets they are either part of or participate in entailing elements of.

An option that avoids repairing the inconsistencies altogether and rather works around them is the evaluation of queries under inconsistency-aware semantics [LMS13]. For example, a system operating under this kind of semantics would return consistent answers that are valid under every possible inconsistency repair.

CHAPTER 5

Knowledge Applications

A common characteristic of the life cycle tasks presented in the previous chapters is that their main purpose is to make a knowledge graph more comprehensive and improve its quality. This is done by either directly identifying and adding missing facts or by analyzing the knowledge contained in a KG in order to find relational and structural patterns and further extract rule-based knowledge. These kinds of tasks are often also categorized as "in-KG" applications, while "out-of-KG" applications merely apply the knowledge from the KG to external problems and do not add new knowledge to it [WMWG17].

Due to their powerful intelligent reasoning capabilities, knowledge graph reasoning methods can infer previously unknown knowledge from the existing triples in the KG and efficiently discover correlations and other patterns in large-scale heterogeneous data. Knowledge graphs can therefore be used for various downstream tasks such as intelligent question answering and recommender systems and their qualities make them quite useful for supporting automatic decision making, community detection and data mining [ZYL+16]. Additionally, KGs are used for many applications in the field of computational linguistics, like sentiment analysis, document categorization and plagiarism detection [CJX19].

Wilcke et al. [WBDB17] proposed to use KGs as the default data model for machine learning, since their ability to deal with heterogeneous knowledge makes them perfectly suited for a wide array of related tasks. Their statement about the suitability of KGs for this context is summarized as:

> *"a) they allow for true end-to-end-learning by removing the need for feature engineering, b) they simplify the integration and harmonization of heterogeneous knowledge, and c) they provide a natural way to integrate different forms of background knowledge."*

This proves the importance and the impact of KGs in modern machine learning tasks, where dealing with heterogeneous knowledge spread over the Web and other resources is commonplace and can be unified by such a formalism.

In the rest of this chapter we aim at providing a closer look at some of the important applications of KGs. It will be structured as follows: First, we will discuss recommender systems in Section 5.1, which try to estimate the likelihood of a future user-item interaction. Second, question answering systems will be reviewed in Section 5.2, along with a short review of the natural language processing capabilities needed for this task.

5.1 Recommender Systems

Nowadays, there are huge amounts of data available about people, users and commodities, which leads to customers expecting companies to know what they want before they know it themselves. Therefore, personalized recommendation has become the core of many real-world applications, be it music recommendation, social networks, or e-commerce [WHC20]. The underlying recommender systems construct predictive models whose goal it is to predict the likelihood of a user-item interaction, be it an actual transaction or just a closer examination. In many cases it is also desirable to explain such a recommendation, i.e. to make explicit on which past interactions it is based and how strongly they influenced the prediction results.

One of the earliest techniques in this area was collaborative filtering [KBV09], which we will discuss first. After that we will review hybrid approaches that incorporate auxiliary information in addition to users' historical information in order to make recommendations. Among these, we will investigate approaches that exploit the graph structure, like path-based techniques, and statistics-based approaches, as well as approaches combining these two kinds of reasoning.

5.1.1 Collaborative Filtering

The idea behind collaborative filtering is to collect the preferences or past interactions of many users and filter out the ones that are useful for making the prediction at hand. The underlying assumption is that if two people agree on one issue or a set of issues, then they are more likely to agree on some other issue than they are to agree with some randomly chosen other person. The two main research areas in this domain are neighbourhood methods and latent factor models [KBV09].

Neighbourhood methods either compute the similarity between items or users. A user's neighbours are those that tend to have similar ratings for the same items. An item's neighbours are those that tend to get similar ratings from the same user. As an example for the item-oriented approach, consider a Sherlock Holmes novel. Neighbouring books would probably include other Sherlock Holmes novels, other books by Arthur Conan Doyle as well as other detective novels. To make a prediction about a user's rating for a

Sherlock Holmes novel, we would evaluate how neighbouring items were previously rated by the user [KBV09].

While they produce good results, neighbourhood methods can not incorporate auxiliary information like temporal effects, confidence levels and implicit feedback. Latent factor models, on the other hand, characterize both items and users on a number of factors that can automatically be inferred from patterns in previous ratings. For Sherlock Holmes, this might include obvious factors like genre, but also hard-to-define ones like how much of a classic a book is or dimensions that are not interpretable by humans at all [KBV09].

Koren et al. [KBV09] use matrix factorization to embed users and items into a joint space, such that the vector representation of items measures how they score on the various factors. The vector representation of users measures how important these factors are for them and whether they prefer items that score high or low on any given factor. To learn the vectors, they minimize the regularized squared error on the set of known ratings using stochastic gradient descent and alternating least squares.

The main problem of these approaches is that they are quite dependent on explicit feedback from users. Unfortunately, any given user usually only rates or buys very few items and in an environment where the item set is very large, many items will have few to no ratings, which leads to very sparse feature matrices [ZYL+16]. Some users do not rate anything at all and new users and items obviously do not have ratings or a history of purchases yet either, leading to what is known as the “cold start problem” [JPC+20]. Therefore, many recent recommender systems use some sort of auxiliary information, for example from a knowledge graph, in order to augment the recommendations for users and items with few interactions in particular. These systems are then called hybrid systems and will be discussed next.

5.1.2 Hybrid Systems

Using only one or two kinds of relationship information, like purchase or rating history in a commercial setting, or friendships in social networks, can lead to very limited prediction results since a lot of useful information is ignored [YRS+14]. Some recent approaches therefore aim to leverage the additional knowledge present in heterogeneous information networks like KGs. These hybrid recommender systems combine collaborative filtering with the rich information provided by a KG, like item types or structural data. Integrating knowledge graphs into the recommendation process enables such systems to have the ability of commonsense reasoning [JPC+20].

Most approaches in this area utilize KGs by exploring relational paths and the structure of knowledge graphs or by learning latent representations. Additionally, there are approaches that utilize both kinds of reasoning in order to combine their advantages.

- **Relational Path-based Approaches**

  Beyond direct connections between users and items, like whether a user purchased, rated, or otherwise directly interacted with the item, there are many more indirect connections that can be used for recommendation. For example, if we know that a user has read and enjoyed the book Pride & Prejudice by Jane Austen we could recommend other books by the same author, the movie adaptation of Pride & Prejudice or that of another Jane Austen book, or other books and movies in this genre.

Wang et al. [WWX+19] propose Knowledge-aware Path Recurrent Network (KPRN), which examines the entity-relation paths between users and items, modeling both the sequential dependencies within a path as well as its holistic semantics. To this end, they first employ a long short-term memory network (LSTM) network to capture sequential dependencies and then use an attention mechanism in the form of a weighted pooling operation to aggregate the representations of paths. This attention mechanism allows them to discriminate the contributions of various paths for the prediction, giving their model the capability to explain why certain predictions were made and what influenced them the most.

Xian et al. [XFM+19] propose a reinforcement learning approach called Policy-Guided Path Reasoning (PGPR), casting the recommendation problem as a deterministic Markov Decision Process (MDP). First, a policy-guided graph search algorithm is used to effectively and efficiently sample reasoning paths for recommendation. After that, reinforcement learning using a soft reward strategy based on a multi-hop scoring function is used on these paths to learn the correctness of an item for a user. Since the action space can be quite large for nodes with many outgoing edges, they employ a user-conditional action pruning strategy to to conduct an efficient exploration and find promising reasoning paths. Generated recommendations are supported by an interpretable causal inference procedure, making it possible to explain recommendations by providing specific influential paths from the KG. The general framework can also be extended to other graph-based tasks like product search and social recommendation [CJX19].

- **Statistics-based Approaches**

  Many recent approaches incorporate representation learning into recommender systems alongside collaborative filtering in order to leverage the reasoning capabilities of both. One of the state-of-the-art systems in this area is Collaborative Knowledge Base Embedding (CKE) [ZYL+16], which integrates collaborative filtering with various semantic representations of items, like textual, visual and structural information. They jointly train the embedding model TransR [LLS+15] (see Section 6.2), stacked denoising auto-encoders and stacked convolutional auto-encoders to extract the structural, textual and visual representations of items, respectively, and combine them with the latent representations from collaborative filtering.

  In order to apply standard machine learning techniques to categorical variables, like the gender and age of users, they have to be converted into highly sparse one-hot feature vectors. In order to effectively learn from such sparse data, it is important to account for the interactions between features. He et al. [HC17] propose Neural

Factorization Machine (NFM) for predictions under this setting, building upon the previously used methods of regular Factorization Machines (FMs) and deep learning. FMs can efficiently model second-order feature interactions, but since they model interactions in a linear way, they have trouble capturing the non-linear, complex structure of real-world data. Deep neural networks are able to model higher-order feature interactions non-linearly, but they can be difficult to train because of their deep structure. NFM combines both models, making it more expressive than a regular FM while being easier to train and tune than deep learning methods.

Cao et al. [CWH+19] aim to endow their embedding-based approach with a degree of explainability, while also tackling the problem of incomplete knowledge graphs at the same time. They jointly model item recommendation and KG completion, learning the representations of users, items, entities and relations, and use a translation-based model to transform implicit preferences into new relations between users and items. Thus, if a user read multiple books by the same author, the system could make recommendations based on that fact while also adding an edge like "likesBooksOf" between user and author to the KG.

- **Combined approaches**

  As will be discussed in Section 6.4, combining multiple types of reasoning in the right way can help to profit of their respective advantages while avoiding their disadvantages. Some approaches in the area of recommender systems aim to do just that by utilizing both of the predominant kinds of reasoning, namely graph-based approaches like relation-path reasoning and statistics-based approaches like knowledge representation learning. The main advantage of relation-path reasoning is that the results are generally more explainable and can combine the information of multiple paths from user to item [WHC+19], while statistics-based approaches allow to optimize the recommendation objective through trainable parameters [WHC20].

  Yu et al. [YRS+14] use data on users, items, item attributes and relationships for entity recommendation, including implicit user feedback data and personalized recommendation models. To represent the connections between users and items along different paths, they extract meta-path based latent features from the network structure. Based on these representations, they define both global and personalized recommendation models and apply collaborative filtering based on Bayesian ranking optimization.

  HOP-Rec [YCWT18] is a unified method that aims to capture both the direct interactions between users and items that are often translated into user preferences by factorization and indirect preferences that can be extracted through graph-based models. This is achieved by harvesting high-order information from neighbouring nodes for each user through random walks and using this information to enrich a sparse user-item interaction matrix. This matrix is in turn used as input for a confidence weighting parameter which is used instead of factorization since it can simulate all high-order information simultaneously, considering different orders of items at once.

Another promising line of research uses graph neural networkss (GNNs), which synthesize information from connections in a KG by aggregating and encoding information from a node's neighbors in order to enrich its representation [WHC20]. Wang et al. [WHC+19] propose Knowledge Graph Attention Network (KGAT), aiming to better encode the collective behaviour of users and find more complex relations among items etc. by utilizing this information about a node and its neighbours, like attributes and types. For example, the author of a book could be the subject of another book, which could lead to fans of said author to also be interested in the book about them, but these kinds of multi-relational connections are hard to find when focusing only one one type of relation or when not encoding side information.

KGAT explicitly models such high-order connections in an end-to-end fashion. It recursively refines a node's embedding by propagating and aggregating the embeddings of neighboring nodes, using an attention mechanism in order to estimate the importance of the neighbors. Other approaches in this area, which also utilize GNNs for recommendation, include KGCN [WZX+19] and KGNN-LS [WZZ+19].

5.2 Question Answering

Many recent applications, from search engines like Google, Bing and Yahoo, to social networks like Facebook and LinkedIn and actual conversational question answering systems like Apple's Siri and Amazon's Alexa, need to answer natural language questions or queries. They increasingly often utilize KGs to do so [Don19]. This allows users to access large amounts of heterogeneous structured data without having to learn a formal query language or knowing about the specific vocabulary of the data source they want to query [HWM+17].

This semantic gap between the representation of data and the way users express their information needs is one of the main challenges for question answering (QA) systems, since natural language is complex and ambiguous [FC14]. If a user asks their voice assistant for help because their "apple crashed" we need to perform entity identification, i.e. discern that they are talking about a piece of technology and not a fruit, and relation identification, i.e. whether there was a malfunction or something fell down, and link both to the respective concepts in the KG.

As is apparent from an example like this, identifying a user's intent often requires complex reasoning and a QA system needs to factor in any context clues given in the question, like using the possible meanings for the entity to eliminate some of the candidate meanings for the relation and vice versa. Sometimes an educated guess can only be made through previously asked questions or common knowledge, or the system even has to choose a meaning just based on the popularity of the various options at the time. The last step, after all entities and relations have been correctly linked to the knowledge graph, is to generate the formal query used for information retrieval [DBCL18].

Additional challenges arise through multilingualism and when complex operators like comparisons, superlatives and negations are used [HWM+17], as they require constraint reasoning and general world knowledge [FQT+20]. These are all common challenges in the area of Natural Language Processing (NLP), and the two areas of research are strongly interconnected as a result. While questions with complex semantic structures can be hard to deal with, most questions are actually rather simple, i.e. they can be answered with a single KG fact and often do not require multi-hop reasoning or constraint inference, with the biggest challenge being the correct identification of entity and predicate [HZLL19].

In general, recent QA systems can be differentiated into information retrieval-based methods and semantic parsing-based methods [FQT+20]. The former generate distributed representations of questions and candidate answers and use scoring functions to rank the answers based on their distance to the question. They are usually quite efficient and easy to use, but most of them can only deal with simple questions and the results are not interpretable [FQT+20]. Semantic parsers, on the other hand, aim to translate natural language queries directly into executable queries in some logical formalism that can then be used to retrieve the answer from the KG. They are able to deal with complex questions that contain multiple relationships or constraints but they often require supervision during training.

5.2.1 Traditional Methods

In addition to the two categories of QA systems mentioned above, there are some approaches that predate both of them but can mainly be seen as predecessors of the semantic parsing-based methods. These traditional methods rely on manually defined templates and rules in order to parse questions and obtain their logical forms, which means they are less scalable and researchers need to be knowledgeable in NLP [FQT+20].

Berant et al. [BCFL13] built a bottom-up parser, for which they constructed a lexicon mapping questions to entities and relations in a KG and a text corpus. The parser then recursively makes derivations based on this lexicon and the four manually defined operations Join, Interaction, Aggregate, and Bridging. In order to reduce the search space and improve the quality of derivations a log-linear model over the hand-crafted features is used.

To improve the number and complexity of the questions template-based methods can handle, recently some researchers have been aiming to learn the templates directly from the data sets in a semi- or fully automatic fashion. Abujabal et al. [AYRW17] proposed QUINT, an automated template generation model utilizing both query and question templates, constructing both the lexicon and the templates using distant supervision. During training, dependency parsing is performed on questions to generate a template library and the system learn rules for mapping question templates to query templates.

When a user issues a question, QUINT constructs its dependency parse tree and matches it against the template library. It then instantiates the corresponding query template to candidate queries, ranks those candidates and outputs the final answer obtained by the top query. For simple questions these models have relatively good performance, but similar to information retrieval-based methods they can not generally answer complex questions [WZF19].

5.2.2 Information Retrieval-Based Methods

Information retrieval-based methods compare the semantic features of candidate answers and questions by first generating distributed representations of both and then calculating the matching scores between them to predict the final answer. Like other methods it has to determine entities and relations of interest mentioned in questions and link these to entities in the KG, but instead of using rules and templates it extracts topic-entity-centric subgraphs and ranks the possible answer nodes in the subgraph based on the other features extracted from the question [FQT$^+$20].

This makes them easier to train without much background knowledge, but it comes at the cost of lacking interpretability and the inability to answer complex questions requiring constraint inference [FQT$^+$20]. Some approaches can handle moderately complex questions by using multi-hop reasoning but most can only answer simple questions, which can usually be answered through an easy translation to the tail entity when the correct head entity and predicate have been identified [HZLL19].

Yao et al. [YVD14] proposed a feature engineering approach where syntax analysis is performed to extract four types of features from a question's dependency parse result, namely the question word (qword), question focus (qfocus), question topic (qtopic), and central verb (qverb). These four features are then combined to form a question graph which is compared to the KG subgraph induced by the topic entity, by measuring the pairwise semantic similarity of nodes through their feature vectors. Which node or nodes answer the question is determined by a classification model that is trained beforehand.

This kind of approach still necessitates manually defined and extracted features, so more recent approaches aim to learn to represent questions and candidate answers in a common vector space automatically, learning to capture important semantic features on the go. QA is treated as a semantic matching calculation between the distributed representation of the question and those of the candidate answers, with some methods also incorporating external knowledge in order to mitigate KG incompleteness [FQT$^+$20].

One of the earliest works in this area is that by Bordes et al. [BCW14], who compose the meanings of questions and answers through a simple summation approach. Question representations are constructed by summing up the vectors of question words and candidate answers are represented by summing up the embeddings of the answer entity, the relationship between answer and topic entity and that of the subgraph induced by the answer entity. The similarity between the representation of the question and that of the candidate answers is calculated by dot product and during training a margin ranking loss between positive and negative examples is used to learn the model parameters.

Recently, researchers have tried to take advantage of neural networks for learning distributed representations, with Dong et al. [DWZX15] proposing multi-column convolutional neural networks (MCCNNs) to combine different aspects of questions and answers. Specifically, the model learns a question representation by combining the embeddings of its constituent words, with the embeddings of entities and relations being learned jointly to improve their representation. For candidate answers, embeddings for three features are learned, namely the path from the answer entity to the question topic, the one-hop context around this path and the type of the answer entity. A score layer is used to rank the candidate answers, which sums up the three dot-product results between the answer features and the question representations.

In contrast to approaches calculating question representations through bag-of-words techniques, MCCNNs are able to retain word order information, thereby improving performance [FQT+20]. However, to put more emphasis on question representation instead of using a simple fixed-length vector, Hao et al. [HZL+17] propose a cross-attention based end-to-end neural network model. They use the same features for candidate answers as Dong et al. [DWZX15], but add a simple embedding of the answer entity as a fourth feature. Then they dynamically learn to capture the correlation between the answer features and question words via the attention mechanism, by having it learn the weights of the various aspects of an answer.

To harness deep learning capabilities for QA, Bordes et al. [BUCW15] leverage memory networks [WCB14], since they are scalable, capable of multi-hop reasoning and applicable to strong and weak supervision. Memory networks have a long-term memory component, which can be used to store KG triples, as well as an inference component, which can be split into modules. Bordes et al. use an input module for processing and embedding questions, an output module for choosing the most relevant supporting memory, an answer module for transforming the triple memory into an object and outputting it, as well as a generalization module for testing the network's generalization ability.

To better model the two-way interaction between questions and KG triples, Chen et al. [CWZ19] propose BAMnet, which employs a bidirectional attention memory network. They jointly learn representations for questions and the KG in the same word vector space and consider all entities connected to the topic entity within a certain number of hops as candidate answers. Similar to Dong et al. [DWZX15], for these candidate answers they learn embeddings for the three features: a) answer type, b) answer path and c) answer context. The bidirectional attention mechanism helps to enhance both question and entity representations, while it also makes the model relatively explainable. However, deep learning techniques like these inevitably increase the model complexity [JPC+20].

5.2.3 Semantic Parsing-Based Methods

Semantic parsing-based methods (SP-based methods) aim to transform unstructured natural language questions into executable query languages so that the answer to the question can be retrieved by simply executing the query [ZDK+18]. Sometimes they are also referred to as neural semantic parsing-based methods (NSP-based methods) in order to distinguish them from traditional methods also using rules to parse questions, since they do not rely on handcrafted rules and templates and instead use neural networks to improve scalability and parsing capability [FQT^{+}20].

Usually, questions are first mapped to some structured logical form, like a query graph, which are then analysed in a bottom-up manner and converted to executable queries, thereby eliminating ambiguity. The main problem to solve is the semantic gap between the way humans express a question and the kind of logically unambiguous statements needed to query KGs [WZF19]. In this way, NSP-based methods combine the advantages of information retrieval-based methods, like not having to manually define templates and rules, and those of template based methods, like interpretability, while also being able to handle complex questions requiring constraint inference [FQT^{+}20].

Query graphs are often used to represent questions, since they share topological commonalities with KGs and can also be directly mapped to lambda calculus [WZF19], a formalism that can easily be translated into executable queries. Yih et al. [YCHG15] proposed Staged Query Graph Generation (STAGG), which represents the question as well as the KG in a query graph that also contains existential variables, aggregation functions and a lambda variable for the answer. The query graph is constructed in three stages, with the current partial query graph being scored after each stage using a log-linear model. First, the question is analyzed to obtain candidate entities and calculate their scores and then the paths between topic entity and answer node are explored and scored by a deep convolutional neural network (CNN). The third stage consists of attaching constraint nodes to the relation path according to heuristic rules and after scoring the final query graph the highest-rated query is chosen to query the KG. STAGG leverages the KG to prune the search space early and thereby increase performance.

Dong et al. [DL16] propose an attention-enhanced encoder-decoder model to convert natural language into a logic formalism, reducing semantic analysis to a seq2seq (Sequence-to-Sequence) problem, i.e. sequence transformation. They encode questions into vector representations and decode their logical forms by conditioning the output sequences on these vectors using RNNs with LSTM, with an attention layer that helps to learn soft alignments between natural language and logical forms. They present two variants of the model, a classical seq2seq model and a seq2tree model that uses a hierarchical tree-structured decoder to capture the compositional structure of the logical representation of questions.

While these methods do not rely on predefined rules or templates, they do require many annotated training examples, and this kind of strong supervision is sometimes infeasible [FQT^{+}20]. Liang et al. [LBL^{+}16] propose a method called Neural Symbolic Machine (NSM), a seq2seq model trained with reinforcement learning on simple question-answer pairs. Utterances are mapped to programs with the seq2seq model and a symbolic computer executes the programs, helping to find good programs through a reward mechanism and by pruning the search space. Intermediate results are stored in a key-variable memory that enables reusing them and reasoning upon them further, helping to handle compositionality.

In addition to these end-to-end QA frameworks, several subtasks of QA require advanced reasoning techniques as well. A prerequisite for QA is that all entities and relations are correctly linked to the KG, but as mentioned in Section 3.2, entity linking often relies on the textual context of a mention to find the correctly disambiguated named entity. In QA, utterances often only span one sentence and might only contain one entity, so there is very minimal context.

Dubey et al. [DBCL18] propose a system called EARL (Entity and Relation Linker) that aims to improve Linking accuracy by jointly linking entities and relations. By treating this as a single task and considering entity and relation candidates for the input question in tandem instead of sequentially, the error rate caused by the mutual dependence can be reduced, since the information available for the selection process is maximized.

In this section, a representative selection of relevant research was presented in order to give an overview of current research directions in the area of QA on KGs. For an in-depth analysis and more background information on the topic we refer the reader to the surveys by Fu et al. [FQT$^+$20] and Wu et al. [WZF19].

CHAPTER 6

Types of Reasoning

While multiple reasoning techniques and approaches to address the tasks arising in the context of knowledge graphs have been mentioned in the previous chapters, they have until now only been described insofar as they and their characteristics were relevant for the task at hand. In this section, they will be presented in more detail so that there can be a better understanding of their respective advantages and shortcomings on: a) how they can be applied and b) for which applications their respective strengths can be used best. Lastly, it will be discussed how they can be combined to harness the strengths of multiple approaches while avoiding their weaknesses.

These techniques can be grouped in various different ways and along different axes - some [TYM18] differentiate first on whether only facts, i.e. entities and their directly neighboring entities, connected by a relation edge (so called one-hop relationships) are considered or whether longer paths between entities, i.e. multi-hop relationships, are taken into consideration [WMWG17]. While the resulting approaches are often different, this differentiation will not be made here since one-hop relationships are just a special case of multi-hop relationships and the respective limitations will instead be mentioned on a case-by-case basis.

The distinction that is most often made [Kaz18, YWC$^+$18, TYM18] is between the following three broad types of reasoning:

- **Logic-based.** This includes rule learning using logical inference, e.g. through deductive or inductive reasoning.
- **Statistics-based.** This includes statistical techniques such as tensor factorization, or embedding based inference [SGEH18].
- **Graph-based.** This includes graph-based inference and learning algorithms, like path-ranking based methods or other approaches taking into account the graph structure.

Additionally, there is increasingly often a discussion on combined approaches using multiple types of reasoning, which will also be included in this section. Statistical relational learning (SRL) approaches (see Section 6.4.4), including Markov Logic Networks (MLN), are sometimes [YWC+18] discussed as a separate category but in this thesis they are instead listed under combined approaches as they are a prime example of a technique that aims to incorporate the advantages of multiple types of reasoning.

The remainder of this section is organized as follows: First, an overview of logic-based reasoning and methods based on logical inference is given in Section 6.1. It includes a discussion of different reasoning paradigms, with a focus on deductive and inductive reasoning, as well as various types of logics and rules, like Horn rules, Probabilistic Soft Logic [KBB+12], etc. While these methods are powerful, highly expressive and their results are explainable, they often have problems with efficiency when scaling to the huge size of knowledge graphs.

Statistics-based methods, which are described next in Section 6.2, come in various forms and include many machine learning techniques. They learn a projection mapping triples and other elements of the graph structure into a lower dimensional space, so they can efficiently reason over the data. The various ways of encoding the knowledge as well as the different techniques employed to reason over the resulting numerical data are therefore equally important and broached in this subsection. While the results of these methods are mostly not explainable and they often can not perform the kind of complex reasoning involved in logic- and graph-based methods, and specifically only lately have become capable of dealing with multi-step paths, they scale far better and are also more robust to noise and sparse training data [Gar15].

Subsequently, graph-based methods will be discussed in Section 6.3. These often use similar information and techniques as logic-based systems and share the advantage of interpretable results, but they build upon the structure of the knowledge graph and therefore usually scale better. Similarly, while path-based reasoning has recently also been incorporated in statistics-based methods like embedding-based reasoning, its strengths can best be harnessed by graph-based methods which reason over (multi-step) paths naturally. The subsection includes a discussion of random walk inference models like the Path Ranking Algorithm (PRA) [LMC11] and methods using reinforcement learning to identify useful paths, along with other techniques taking into account the graph structure.

Lastly, in Section 6.4 the discussion will turn to combined approaches in their various constellations. These aim to unite the advantages of two or more types of reasoning, while avoiding as many of their downsides as possible. These combinations come in various forms as the methods can benefit from each other in multiple, often surprising, ways. For example, when it comes to methods combining rule learning and embeddings, there are approaches using logical rules for guiding the embedding process, ones that use embeddings to learn better rules and others that actually embed rules.

6.1 Logic-based Reasoning

A popular class of techniques for reasoning in data mining and knowledge graphs are logic- or rule-based methods, which try to find regularities in data and express them in the form of a rule that can be used to describe the data set, to find irregularities or even to infer new knowledge that can again be incorporated into the knowledge graph [FK15].

As knowledge graphs usually contain millions or even billions of facts while also being highly incomplete, classical rule induction techniques often do not provide adequate results or are simply intractable [SGEH18]. Therefore, heuristics and evaluation mechanisms specifically tailored to the application area of knowledge graphs have to be developed.

Even reasonably complete knowledge bases contain a lot of implicit semantic information that can not be used without being mined and brought into an explicit form [WL18]. Finding rules that describe common patterns and correlations in the data is therefore an important task that makes it possible to efficiently use the data contained in the knowledge graph.

For example, mining a rule like

$$spouseOf(x, y) \rightarrow spouseOf(y, x)$$

which captures the fact that marriage is a symmetric relationship can serve multiple purposes [GTHS15]. First, it can be used in knowledge graph completion to derive new facts by applying it to the data - e.g. if we know that Barack Obama is married to Michelle Obama, we can deduce that the opposite must be true as well. Second, rules like this can be used to identify potential errors in the KG - e.g. if there is a triple stating that Michelle Obama is married to someone other than Barack Obama, then this statement is probably wrong (since polygamy is not legal/customary in most countries) or it at least refers to some other point in time. Third, rules help to better understand the data by describing general regularities like this.

Modern algorithms like KGRL [WLX16] use semantic inference to deduce the theory closure of knowledge graphs in an efficient way, but they often have to do a full re-reasoning whenever new triples are added to the KG [WL18]. For large KGs that are updated regularly, this can be prohibitively time-consuming. Wang et al. [WL18] therefore propose an incremental reasoning algorithm that avoids completely re-reasoning over the entire data by deducing the theory close of the updated KG from the theory closure of the original KG and the added triples. They use filtering algorithms to reduce the amount of data that needs to be considered while keeping the relative completeness of the final deduction results.

Scalability is generally an issue with rule-based methods, as determining rule structures and searching for support triples over the huge search space of a knowledge graph is computationally expensive [ZPW+19]. The search space is usually exponential in the number of relations/triples so even very small KGs containing a few hundred entities and relations the number of possible structures for a relatively short rule containing three relations is several millions and the number of possible supports is in the trillions. Additionally there is an inherent trade-off between the expressiveness of the formalism and its efficiency, as for example rules containing higher-order logic or existentials can quickly become intractable.

However, rule-based reasoning is accurate [ZPW+19] and since rules are explicit knowledge (as opposed to neural networks etc.) they can provide human-understandable explanations for their inference results [OWW18]. This interpretability also makes them more robust when it comes to transfer tasks - while methods that learn embeddings often do so for specific knowledge graph entities and have to be retrained when the underlying data changes, rules stay usefully accurate and can immediately be applied to the new knowledge [YYC17].

Logic-based Reasoning Paradigms

There are various reasoning paradigms, each with their own specific merits for reasoning over different types of data and in different contexts, like paraconsistent reasoning, inductive reasoning, normative reasoning, analogical reasoning, deductive reasoning, abductive reasoning etc. which some [BDPP19] argue, should not be studied in isolation. Particularly in the context of large KGs whose data is integrated from multiple sources of information, multiple reasoning paradigms could be needed at the same time, which gives rise to the need for a unified framework integrating the different paradigms. However, as efforts in this direction are not yet very advanced, the focus in the following will be on the two major reasoning paradigms deployed in the context of KGs, namely deductive reasoning and inductive reasoning.

- **Deductive Reasoning**

  Deductive reasoning is the process of making implicit knowledge explicit by deriving new data from logical axioms in the form of rules about the world or a part thereof, and the facts in the KG, which form the premises for these rules. Depending on the generality of the premises and rules, this is denoted as “commonsense knowledge” or “domain knowledge” [HBC+20] and it is usually present in a KG in the form of an ontology or T-Box (see Section 2.2).

  Expressing and automating complex entailments is an important research topic in the area of knowledge graphs, as it can be used for many tasks in the life cycle of a KG, like completion, query answering, classification, finding inconsistencies, etc. Additionally, this type of reasoning provides justification for the derived knowledge, but the derived conclusions will only be as accurate as the data they are based on [RP17]. In contrast to numerical methods, these rules form a symbolic representation of knowledge [JHNT12].

  The inference rules can be of many different formalisms and logical frameworks, like first-order logic, description logic, answer set programming etc. but will generally in some way encode an “if ... then” relationship. If we can substitute the variables in the body (if-part) of a rule with terms from the KG, then a valid entailment can be made by using the same replacement for the variables in the head (then-part). For example, given a (tiny) ontology and some facts like the following:

$$(x, presidentOf, USA) \rightarrow (x, citizenOf, USA)$$
$$(BarackObama, presidentOf, USA)$$
$$(JohnF.Kennedy, presidentOf, USA)$$
$$(BarackObama, citizenOf, USA)$$

we could then deduce that John F. Kennedy must have been a citizen of the United States as well. Indeed, if we are interested in the deductive closure with respect to our set of axioms and facts (or even the whole KG), we would need to add this fact, since deductive closure is only achieved if, whenever a tuple or tuples satisfy the body of a rule, the head is also satisfied with the same variable replacement [Coh16]. Typically, the variables appearing in the head must be a subset of those appearing in the body so that the conclusion leaves no variables unreplaced [HBC$^+$20]. This leads to rules that correspond e.g. to (positive) Datalog [CGT89] or Horn clauses (see the discussion on different kinds of rules later in this subsection).

Adding inferred triples is also called materialization (see Section 2.3), and it can either be done on demand for a part of the KG or on the whole data, by recursively applying rules and adding the generated conclusions to the graph until nothing more can be added and a fixpoint is reached [HBC$^+$20]. While there are scalable algorithms for achieving this [FVHA$^+$08], albeit based on less expressive ontologies, this is often not possible or desirable in fact [JHNT12], as adding all triples that can be derived via deductive reasoning would often lead to KGs that are "bloated" and too large to manage [HBC$^+$20].

Among the various ontology languages, the most popular is arguably the Web Ontology Language (OWL) [C$^+$12], which is partially based on description logics (DLs). For example, ome OWL dialects are being restricted in a way such that entailment becomes decidable as is the case for many DLs but not generally for deductive reasoning [HBC$^+$20]. Both formalisms allow to define various relationships between and rules for properties and classes, like domains, ranges, inverses, disjointness, transitivity, subclass/-property relationships etc. They also allow for the multiplicity of properties to be defined as functional (many-to-one), inverse-functional (one-to-many) or one-to-one.

While some OWL standards define many such rules, they can not capture all the relationships that we might want to express on our data and are therefore incomplete. They generally can not express negation, aggregation, existentials or universal quantification but there are other rule languages that do support such features, like Disjunctive Datalog [RKH08] (disjunction) or Datalog$^\pm$ [BGS18] (existentials, see the discussion on different kinds of rules later in this subsection) [HBC$^+$20].

However, including additional features always comes as a trade-off with efficiency and decidability.

In general, entailment given the features described so far is undecidable, with most practical reasoning algorithms for ontologies either accepting that certain entailments will be missed, restricting the features of the input ontologies or accepting that certain inputs will not produce an answer [HBC^{+}20]. While the first option might not be acceptable in more sensitive domains, it is useful in the context of KGs, where the data is incomplete and having some entailments using efficient and scalable algorithms is more valuable than potentially having all possible entailments.

One technique that aims to be complete is based on proof by refutation, i.e. negating a statement and trying to find a contradiction resulting from that (when trying to prove a statement) or reasoning from the original statements (when trying to find inconsistencies in the data) [RP17]. For example, given a question like "Did Barack Obama live in the White House?" one could assume that he did not, which would entail that he was neither a US president, nor the spouse or child of a US president. Given a fact stating that he was indeed a US president, this would lead to a contradiction, so that the answer could be assumed to be that he did live in the White House after all.

While this kind of automated theorem proving used to be infeasible for the amounts of data present in KGs, it has in recent years become fast enough to be practical and has been used by systems like SUMO [PSST10], which contain tens of thousands of logical statements.

- **Inductive Reasoning**

  In inductive reasoning, the main task is the automatic extraction of hypotheses about the data from knowledge contained in the knowledge graph by identifying patterns, be it by learning from positive and negative examples or interpretations. In contrast to deductive reasoning it can be used in a broader group of reasoning types, like graph neural networks or statistics-based methods [LB10], and is the underlying principle of machine learning [RP17]. In the context of logic-based reasoning the task is often referred to as automatic rule induction or rule learning [SGEH18].

  While deductive reasoning is characterised by precise logical arguments, as entailments are formally proven and preserve the truth in the data by merely making knowledge explicit, inductive reasoning involves learning new but potentially imprecise predictions from the data [HBC^{+}20]. The conclusions only likely follow from the premises and can therefore only be made with a certain probability that is necessarily lower than that of the premises [RP17].

  However, since KGs are highly incomplete, noisy and biased and cannot represent the real world in all of its complexity anyway, it is not possible to automatically learn a perfect representation of the real world from the data. Additionally, inductive approaches are better able to deal with the huge amounts of data that arise in the context of knowledge graphs, as they scale better and are more noise-tolerant [JŁŁ10]. Consequently, inductive methods, like the machine learning approaches mentioned above, are used to complement deductive ones.

As an example, consider a KG containing information on all US presidents, where we might observe the pattern that almost all of them have been white and male. Therefore, given knowledge of a new president, we could predict that they are likely also male (our KG does not contain any counterexamples and in fact there are none yet) and white. The latter prediction could however only be made with a confidence level of about 0.978, as the respective pattern is only true for 44 out of the 45 past presidents.

One of the important applications of inductive reasoning is automatic rule induction, or rule learning, the aim of which is to represent knowledge gained from the data in some kind of human-understandable description language [SGEH18]. As mentioned above, these automatically learned rules do not usually aim to be universally correct, with the goal instead often being to predict a sufficiently large portion of true facts.

One of the techniques used for this is Inductive Logic Programming (ILP), in which the goal is to learn complete and consistent hypotheses about unseen instances by learning from positive and negative examples, background knowledge in the form of facts and rules and some syntactic restrictions [SGEH18]. As will be discussed in more detail in the next section on rules, a KG does not usually contain negative examples and some kind of workaround has to be used to be able to use a semblance of negative examples for this kind of technique.

Using inductive methods to complement deductive ones, as is done e.g. in ontology mining [FdE08], where ILP methods are applied to description logic (DL) knowledge bases, aims to combine the respective advantages of the underlying reasoning paradigms. Jozefowska et al. [JŁŁ10] propose a method for mining frequent patterns in knowledge bases using Datalog-rules that tries to exploit the combined knowledge by utilizing the meaning of the represented knowledge as well as the semantics of the chosen language to improve the mined rules.

In general, inductive methods utilize varying degrees of supervision, with supervised ones learning to generalize patterns from labelled in- and output so they can be applied to unlabelled data in the future. Since human interaction is always costly, the goal is usually to minimize the necessary supervision. Therefore some processes instead generate the input-output pairs used as training data from the input alone and only then use a supervised process to learn from them [HBC$^+$20]. This kind of self-supervision is also often utilized for KG embeddings to learn a low-dimensional numeric model that assigns plausibility scores to e.g. assess whether a missing edge is true by generating training data from the KG itself.

Other approaches also use the structure of the graph itself by detecting clusters, interesting paths or central nodes and edges or, as in the context of graph neural networks, by learning models that can be applied over neighbourhoods in the graph, so as to generate outputs for nodes.

6.1.1 Rules

As has been covered in the previous section, rules are used in both deductive and inductive reasoning to represent knowledge in a symbolic, human-interpretable form. Rule mining is the central task of Inductive Logic Programming (ILP) and has been researched in detail in recent years, with approaches varying in the types of rules that are examined, the types of data used to learn the rules, the types of application and so on [GTHS15].

Generally, rule learning can be differentiated along another dimension, namely by whether the rules merely aim to describe a pattern observed in the data, as is done in descriptive rule learning, or whether they actually aim to add data to or improve the data in the KG, which is called predictive rule learning [FK15]. Descriptive rule learning has applications for systems aiming to learn about general world or domain knowledge, however, a larger focus in research has been on predictive rule learning. Building complete sets of rules for the representation of knowledge is a daunting task that can hardly be done manually. Therefore the potential offered by rule learning algorithms to automate this process is intriguing.

Rules in the context of KGs normally encode if-then-style consequences of the form $head(then) \leftarrow body(if)$, indicating that if the variables of the body can be replaced with terms from the data resulting in a valid statement, then using the same variable replacement in the head leads to a valid entailment. For example, given a KG containing the following rule and tuples

$$(x, fatherOf, z) \leftarrow (x, fatherOf, y) \wedge (y, siblingOf, z)$$
$$(BarackObama, fatherOf, MaliaObama)$$
$$(MaliaObama, siblingOf, SashaObama)$$

we could instantiate the rule with x being $BarackObama$, y being $MaliaObama$ and z being $SashaObama$, which would allow us to make the prediction that Barack Obama is the father of Sasha Obama, adding a respective fact to the KG [GTHS15]. The body of a rule r can sometimes contain negative statements such as $not\, female(x)$, in which case the negative part is referred to as $body-(r)$ while the positive part is denoted as $body+(r)$ [SGEH18]. The rule can then be written as $head(r) \leftarrow body+(r)$, $not\ body-(r)$ [HSGE+18].

There are various restrictions to which type of atom (variable, constant, with/without negation, ...) can be used in which part of the rule, which lead to different types of rules that will be discussed in the following. Typically, the variables in the head must be a subset of the variables appearing in the body so that no variables are left unreplaced [HBC+20]. Many systems extract so-called closed rules, i.e. rules in which every variable appears at least twice, so as to be able to predict actual facts and not just their existence [SGEH18]. Horn rules, which adhere to this structure, are therefore often used in the context of rule mining in KGs and will be discussed in more detail later in this section.

However, Horn rules are not well suited to represent incomplete real-world knowledge as they cannot adequately capture exceptions. Usually, when rules are automatically learned, statistical measures like confidence and support are used to assess their quality, which might be misleading in settings with incomplete data [HSGE$^+$18]. For example, confidence is calculated as the fraction of facts predicted by a rule that are actually present in the KG, but under the Open World Assumption (OWA) missing facts cannot automatically be assumed to be false. Especially in the context of domain-specific KGs, rules might be learned which do not represent general relationships, since standard statistical measures cannot reflect the patterns in the missing facts [SGEH18].

For instance, from a KG containing data about American heads of state, we could extract a Horn rule like $heterosexual(x) \leftarrow (x, headOfStateOf, y)$, stating that heads of state are usually heterosexual and in the Americas (so far) exlusively so. However, this rule is obviously heavily biased and not true in general and if we include data from Europe, there would actually be counterexamples, so the rule should look more like this:

$$heterosexual(x) \leftarrow (x, headOfStateOf, y) \land \neg(y, locatedIn, Europe)$$

Since Horn rules disallow negative atoms in the body, this exception could not be expressed and different formalisms are necessary for more complex rules like these.

One of the ways this can be tackled is by learning non-monotonic logic programs under the answer set semantics [GL88]. Several other approaches will be presented in the rest of this section, as well as a general discussion of different kinds of rules along with their advantages and disadvantages:

- **Horn rules**

  AMIE [GTHS15] and other systems aim to mine Horn rules, which are based on association rules, but try to overcome some of the problems they face on large knowledge graphs. Association rules [AIS93] are used for finding frequent co-occurrences in sets of items, like the tendency of people who buy butter to also buy bread, and to predict this relation in the future. However, it is difficult to apply the standard measures for support and confidence to these rules in the setting of knowledge graphs, since that would normally require counterevidence and KGs usually do not contain negative statements. Since they are also usually operating under the OWA, absent statements can not serve as negative evidence either.

  Therefore, Horn rule mining is often used instead, and it is shown in [GTHS13] to correspond to association rule mining on a database with an exponentially large maximal number of variables of the rules. Following the definition from above, a rule r of the form *head* ← *body* is Horn if there are no negative atoms in the body and all head variables occur in the body as well.

  An example for an ILP system that uses Horn rules is QuickFOIL [ZPP14], which learns hypotheses from positive and negative examples. It adds clauses to rules greedily by maximizing a scoring function depending on the support and confidence of the resulting refined rule candidates and afterwards removes the positive examples covered by the new rule, starting the induction process again on the remaining facts. Since it uses an aggressive pruning strategy it can scale to very large problem instances. However, like similar systems such as WARMR [DT01]/WARMER [GVdB02] and Sherlock [SEWD10], it has difficulties in dealing with large knowledge graphs operating under the OWA as it requires explicit negative examples.

AMIE [GTHS15] was explicitly developed to overcome these obstacles by introducing a new measure for confidence that works better under the OWA. Specifically, they calculate confidence using a so-called partial completeness assumption (PCA), which is equivalent to the Local Closed World Assumption (LCWA) discussed in Section 2.4. Like other ILP systems, it does not mine general Horn clauses but rather constrains the form of the mined rules so as to restrict the size of the search space with so-called language biases. Restricting the search to shorter rules, for example, leads to a smaller search space and therefore faster mining. However, this comes at a price, as longer rules are typically more expressive since they can capture more complicated correlations.

AMIE has a number of language biases:

- Rules have to be connected. A rule is connected if every atom in it is connected transitively to the others by sharing a variable or entity. This avoids mining rules with completely unrelated atoms like starredIn(w, x) → directorOf(y, z)

- Rules have to be closed. A rule is closed if all the variables it contains appear at least twice in the rule. This avoids mining rules that only predict the existence of a fact, like starredIn(w, x) → ∃z : genreOf(x, z)

- Rules can not be reflexive. Reflexive rules of the form r(x, x) do not usually provide relevant information in real-world settings.

In contrast to some other ILP systems, AMIE does however allow recursive rules, i.e. ones that contain the head relation in the body. While some of the language biases and a multi-threaded approach aim to make mining rules with AMIE faster, the KB is stored and indexed in memory, which leads to a high memory usage [BBL16].

- **Frequent Predicate Cycles (FPCs)**

  While AMIE+ mines one rule at a time and has a relatively long running time, especially when learning long rules in very large KBs, Wang et al. [WL15] proposed a system called RDF2Rules. Although it also mines Horn rules, it aims to have a better performance by taking into account entity type information and mining multiple inference rules at once from frequent patterns in the form of closed paths, which they call Frequent Predicate Cycles.

  They define predicate paths as sequences of entities or entity variables connected by predicates or relations, like

$$(BarackObama, livesIn, Washington)$$
$$(Washington, locatedInCountry, USA)$$

and predicate cycles as predicate paths that start and end at the same entity variable. If, for example, there is a triple $(BarackObama, livesIn, USA)$ as well and there are similar cycles for other people, the system could mine the rule that a person who lives in a city can also be said to live in the country the city is located in.

The more frequently a predicate cycle occurs in a KG, i.e. the more support it has, the more reliable and useful its pattern and the rules generated from it are. Whether a predicate cycle is classified as frequent is determined based on whether its support exceeds a certain predefined threshold. Additionally, entity type information is added to rules to make them more accurate. For example, the rule $(x, bornIn, y) \rightarrow (x, livesIn, y)$ might not have a lot of support in general. However, that changes significantly when the entity instantiating y is a country and not just a town or a city.

By utilizing an efficient algorithm with an effective pruning strategy which allows for parallel execution on multi-core machines, RDF2Rules can run faster than AMIE+. Closed path rules are also used by Lao et al. [LMC11] in combination with constrained, weighted, random walks - this and other combined approaches are discussed in Section 6.4.

- **Semantic Association Rules**

  SWARM (Semantic Web Association Rule Mining) [BBL16] is an approach that mines association rules from RDF data. Similar to RDF2Rules, SWARM also includes schema-level (i.e. ontological) knowledge like type and subclass relationships when mining rules so as to add semantic information and thereby increase rule quality. AMIE's quality measures are calculated using only instance-level knowledge, which can negatively impact the mining result as well as the interpretation of discovered rules. Semantic Web Association Rule Mining (SWARM) uses both instance-level and schema-level knowledge to calculate the support, confidence and lift of rules. It is more efficient than ILP-based approaches, and in contrast to them it does not require counter examples [FK15].

  The SWARM framework consists of a pre-processing module and a mining module [BBL16]. In the pre-processing module, commonalities of entities are extracted by identifying relationships that they share. For example, $\{ BarackObama, JohnF.Kennedy, BillClinton \}$ would be grouped together as a semantic item, since they are all connected via the relationship *presidentOf* to the entity *USA*. Among these semantic items, subgroups that feature in multiple common relationships are identified. For example, the three presidents from above were/are also all members of the Democratic Party, and the algorithm would therefore classify them into the same common behavior set that also contains information on the relationships they share.

The mining module receives the common behavior sets from the pre-processing module and generates association rules from them by combining the featured relationships in all possible ways. E.g. the generated rules for our presidents are:

$$(x, \mathit{presidentOf}, \mathit{USA}) \leftarrow (x, \mathit{memberOf}, \mathit{DemocraticParty})$$

$$(x, \mathit{memberOf}, \mathit{DemocraticParty}) \leftarrow (x, \mathit{presidentOf}, \mathit{USA})$$

Subsequently, statistical measures are applied to test whether the generated rules are actually supported by the data, which would lead to the above rules being discarded (except in a KG only containing records of presidents from the Democratic Party).

Both during pre-processing and when measuring the rule quality, subclass relationships and other ontological knowledge are being used to improve the quality of the mined rules. This could for example lead to more general rules about heads of state when knowledge from other parts of the world is included, by taking into account that presidents and emperors are both heads of state, or that England, the US and Japan are all entities of the type *Country*. The aim is to reveal common behavioural patterns within the data automatically, using all of the available information and not just a part of it.

- **Probabilistic Soft Logic (PSL) rules**

  While many KGs assign probabilities to (candidate) facts and other approaches can deal with the resulting continuous truth values, Probabilistic Soft Logic (PSL) [BBHG17] is explicitly built to work with them. This way, noisy extractions (because of untrustworthy sources or extractors) can be represented by assigning soft-truth confidence values to extracted relations. The resulting candidate facts can then be processed, removing inconsistencies via ontological constraints in the form of PSL rules [PLGC15].

  These rules are usually in the form of (universally quantified) weighted first-order logic rules. They can be as simple as applying a threshold based on extractor/source trustworthiness values or provide additional meta knowledge about entities, relations and facts. As an example, consider the following extracted candidate facts and labels, as well as relevant ontological rules concerning domain (DOM) and range (RNG) of a relation, relations between entities (REL) and an entities class restrictions (LBL):

$$\mathit{Person}(\mathit{BarackObama})$$

$$\mathit{Company}(\mathit{BarackObama})$$

$$(\mathit{BarackObama}, \mathit{presidentOf}, \mathit{USA})$$

$$DOM(presidentOf, Person)$$

$$RNG(presidentOf, Country)$$

$$LBL(E1, L) \leftarrow DOM(R, L) \wedge REL(E1, E2, R)$$

The system could try to resolve the conflicting labels for the entity *BarackObama* based solely on their respective trustworthiness score. However, given the additional evidence of an extracted fact featuring this entity and the limitation of the position in the rule to entities of type *Person*, it can make a more informed decision. Specifically, the system would have to compare the probabilities of the two extracted type declarations and the extracted rule to make its decision, as proposed by Pujara et al. [PLGC15]. Their system uses a continuous-valued hinge-loss Markov random field in combination with a PSL modeling framework [BMG12] and formulates inference as a continuous optimization problem, allowing it to scale much better than other probabilistic models [Puj16].

They define a probability distribution over possible knowledge graphs, essentially grounding the rules with atoms from the KG and applying soft logic to determine the truth values of logical formulas that have to deal with soft-truth values. For example, if the truth value of $person(BarackObama)$ is determined to be 0.9, then the truth value of $\neg person(BarackObama)$ is 0.1 etc. According to these rules and transformations, each possible KG is assigned a truth value or so-called distance to satisfaction, which makes it possible to infer the most probable KG, i.e. the one with the highest probability score [PLGC15].

One of the drawbacks of PSL, as a model based on declarative rules, is that a lot of the common sense and domain knowledge that is used as input for rule establishment has to be acquired manually. Zhang et al. [ZZL$^+$19] therefore proposed an automatic rule-building method, which extracts rules from RDF using the AMIE+ algorithm and then transforms these rules into adaptive probabilistic soft logic models. To improve efficiency, multilevel reasoning was modeled, whereby the inference results of one predicate can be used as the input condition of other predicates.

- **Existential rules**

  There are several extensions to Datalog that aim to make the language more expressive, while trying to keep the resulting language decidable and tractable at the same time. One such extension is the addition of existentials, as is the case for Datalog$^\pm$, which allows existential quantification in rule heads [BSG18]. The resulting rules are first-order sentences of the following form, where φ and ψ are conjunctions of atoms with constants and variables:

$$\exists z \psi(x, z) \leftarrow \forall x \forall y \varphi(x, y)$$

  The intended meaning of such a rule is that whenever there is a fact $\varphi(x, y)$ occurring in a KG, then there exists a tuple z of constants and nulls such that the facts

$\psi(x, z)$ are also in that KG. E.g. the following rule and facts formalize the notion that, if a country has a First Partner, they are the spouse of that country's head of state, so the country must have a head of state:

$$\exists z(z, \mathit{headOfStateOf}, y) \leftarrow \forall x \forall y(x, \mathit{firstPartnerOf}, y) \land \mathit{Country}(y)$$
$$\mathit{Country}(\mathit{USA})$$
$$(\mathit{MichelleObama}, \mathit{firstPartnerOf}, \mathit{USA})$$

While there are many possible entities z that would make this statement true (in fact, unless we take type limitations for the relationships *spouseOf* and *headOfStateOf* into account, any entity in the KG would do so), we are usually interested in the most general answer. This will usually be one that contains a null value in place of z, such that this null value can be mapped to any other possible answer, and the answer is therefore called universal [BFGS19]. To find these universal answers, the chase procedure [BV84] can be applied, which adds new facts (possibly containing null values) until the KG satisfies all the existential rules. However, the result can be infinite, since tuples containing null values generated during the chase can again give rise to new generated tuples, and determining whether a certain tuple is part of the answer is in general undecidable [BGPS18].

Several possible restrictions to Datalog make the above problem decidable, each of them giving rise to a new Datalog$^{\pm}$ language. One of these dialects is Warded Datalog$^{\pm}$ [GP15], which restricts the way "dangerous" variables - i.e. ones that could lead to the unrestricted propagation of nulls and therefore infinite chase sequences - can be used. Wardedness is given if all the dangerous variables in a rule coexist within a single body atom (called the ward), and this atom can only share harmless variables with the rest of the body, i.e. variables that are only unified with constants during the chase [BSG18]. For example, the following set of existential rules is not warded:

$$\exists z(x, Q, z) \leftarrow \forall x P(x)$$
$$P(y) \leftarrow \forall x \forall y(x, Q, y)$$

Variable y in the body of the second rule is dangerous, since it can be instantiated with a null by the first rule, and because it also appears in the head of the rule, any tuple P(x) in the KG would lead to an infinite chase sequence [BGPS18].

Warded Datalog$^{\pm}$ is used as the logical core of VADALOG [BGPS18], a recently developed Knowledge Graph Management System (KGMS). The language is extended with additional features to make it more suitable for common use cases, e.g. (monotonic) aggregation, probabilistic reasoning and data binding primitives to connect the system to external data sources. It captures both plain Datalog and SPARQL queries, generalizing ontology languages like the OWL 2 QL profile of OWL, making it suitable for ontological reasoning and querying RDF graphs [BFGS19].

6.2 Statistics-based Reasoning

Since the development of Machine Learning (ML) has been promising in many domains like computer vision and NLP, it has also become an endeavour to apply it to KGs. To be able to harness the possibilities of ML techniques, the data of the KG as well as its structure (like local neighbourhood structures) have to be represented, or encoded, in a way so that ML models can be applied to them. While the heuristics used to extract these features usually had to be defined by hand, this is both expensive and inflexible. Recently, techniques based on deep learning and nonlinear dimensionality reduction have been employed to learn such heuristics automatically [HYL17b]. This is called Knowledge Representation Learning (KRL).

The resulting representations are usually in the form of low-dimensional embeddings, but they differ in what is embedded (entities/relations, graph structure or entire (sub)graphs), whether all features are embedded into the same latent space or into separate ones and the dimensionality of those latent spaces. The goal is to make data manipulation simpler and more efficient while preserving the inherent structure of the KG [WMWG17]. Usually the aim is to find a representation such that geometric relations in the latent space correspond to relations in the original graph. For example, in Translational Distance Models semantically similar items are supposed to also be close to each other in the semantic space, e.g. when embedding entities, movies should be closer to each other than to animals, and movies by the same director should be closer still (see Figure 6.1 for an example).

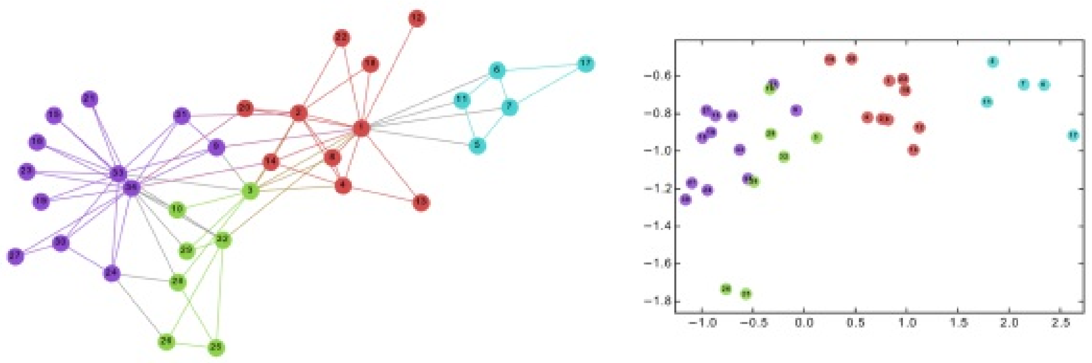


Figure 6.1: Left, a graph structure of a social network, where two nodes are connected if the respective individuals are friends. Right, the generated node embeddings of this graph using DeepWalk [PARS14], represented in a two-dimensional space. This figure was taken from Perozzi et al. [PARS14]

There are many other methods used for KRL which can be classified along several dimensions, e.g. by whether they only embed one node at a time or entire subgraphs, what class of techniques is used or whether, in addition to the node/subgraph itself, additional information in the form of rules, type attributes etc. is used. Many of the slightly different classifications in the literature can be summarized by an encoder/decoder framework as proposed by Hamilton et al. [HYL17b] and adopted by Kazemi et al. [KGJ$^+$19] and others.

This framework makes it easier to deal with the methodological and notational diversity of existing approaches by classifying them according to their role in the embedding process [KGJ$^+$19]. The encoder functions map nodes or entire subgraphs to low-dimensional representations based on their features, while the aim of the decoder functions is to decode structural information about the graph from these learned embeddings. Thus, by jointly optimizing encoder and decoder functions and thereby learning to decode high-dimensional graph information from encoded low-dimensional embeddings, the embeddings should be sufficient for downstream applications and information can therefore be stored in a greatly compressed way [HYL17b].

To be able to learn embeddings automatically and rate the accuracy of encoding and decoding functions, a proximity function measuring "closeness" of nodes as well as a loss function measuring the difference between the original proximity value and the one after the en- and decoding process have to be defined. One of the advantages of this encoder-decoder framework is the possibility of creating new models by combining encoders and decoders from different models [KGJ$^+$19]. For many encoder-decoder pairs, the components can be trained end-to-end, with parameters for both components initialized randomly. Then, through several epochs of stochastic gradient descent, the parameters are gradually updated based on the loss function computed in each epoch until some criterion is met.

- **Encoders**

  The encoding process consists of a pairwise proximity function defined over the nodes of the graph that measures the "closeness" between nodes based on some metric as well as the actual encoder function that generates the node embeddings [HYL17b]. The encoder function contains a number of trainable parameters that are being optimized in the training phase.

  The approaches can be divided into the simpler direct encoding approaches which include factorization and random walks and the more complex encoders utilizing deep learning. In both cases the aim is to learn a projection that maps triples into a low-dimensional representation space [LC19], like a point-wise space, complex vector space or Gaussian distribution. The choice of representation space is an important step when developing a KRL model, as it has to match the nature of the encoding/decoding methods and be as expressive as possible while not adding unnecessary computational complexity [JPC$^+$20].

  - **Factorization-based methods**

    Factorization-based methods represent connections between nodes via matrices, tensors etc. and factorize those to obtain an embedding. The type of connection can be anything from node adjacency to node transition probability and the factorization techniques vary based on the properties of the matrix or tensor [GF18] and the type of connection involved.

For example, Gaussian embeddings are well suited to expressing uncertainties of entities and relations while embedding in complex vector space is particularly well suited to connectivity patterns like symmetry/antisymmetry [JPC+20]. Laplacian matrices and other positive semidefinite matrices can be factorized via eigenvalue decomposition, while for unstructured matrices an embedding can be obtained in linear time by factorization via gradient descent methods [GF18].

While tensors are more expressive, operations are generally less computationally expensive on lower dimensional representations like matrices or vectors. To be more computationally efficient, connectivity measures like adjacency tensors can therefore be reshaped into matrices by associating rows with subjects and columns with relation-object pairs or by associating rows with subject-object pairs and columns with relations etc. [NMTG15]. The matrices are then factorized to obtain low-rank representations [LHX+18]. However, this reduction in dimensionality invariably results in a loss of information compared to the tensor representation [NMTG15].

One of the applications of these factorization approaches is learning entity and relation representations. Under the Closed World Assumption (CWA) this can be done by minimizing some loss function (like square loss, logistic loss [NT13] or absolute loss [EM13]), which enforces facts contained in the KG to have scores close to 1, while non-observed facts score close to 0 [WMWG17]. Minimizing the squared loss amounts to the factorization of a three-mode tensor encoding the relationships in the KG.

A well-known example of a factorization model is RESCAL [NTK11], a collective tensor factorization model, which reduces modeling the KG's structure to a tensor factorization operation.

– **Random Walk based algorithms**

While random walk based approaches can also be classified as direct encoders, they optimize embeddings to encode the statistics of random walks instead of fixed deterministic distance measures [HYL17b]. They thereby learn mappings of nodes to a low-dimensional feature space that maximises the likelihood of preserving the nodes' neighborhoods [GL16].

Two of the leading techniques in this area are DeepWalk [PARS14] and node2vec [GL16]. The basic idea behind them is to learn embeddings which encode the probability of visiting node B on a random walk starting at node A, with the length usually in the range of two to ten hops. In contrast to factorization-based approaches, this probability measure is both asymmetric and stochastic [HYL17b]. Since naively evaluating the cross-entropy loss would be too inefficient, both node2vec and DeepWalk use optimizations to compute approximations of it.

DeepWalk [PARS14] uses simple unbiased random walks over the graph, whereas node2vec uses a more flexible definition of a node's neighborhood and a biased random walk procedure [GL16]. Specifically they introduce random walk hyperparameters that control the likelihood of the walk immediately revisiting a node or a node's one-hop neighborhood. This allows for a smooth interpolation between walks more closely resembling depth-first search and those resembling breadth-first search. Depending on how the parameters are set, the model learns embeddings emphasizing local structural roles or community structures [GL16].

An approach that can be seen as a kind of meta-strategy is Hierarchical Representation Learning for Networks (HARP) [CPHS17], which improves upon various random walk based approaches by extending them via graph pre-processing. The graph is "coarsened" by collapsing related nodes into common supernodes, which can be done at varying levels of coarseness. The resulting graphs and their embeddings are then used as input for random walk algorithms like DeepWalk and node2vec, where embeddings of supernodes are used as initial values for the embeddings of the nodes it emcompasses. By repeating this process in a hierarchical manner, HARP can improve the performance of the specific random-walk based approaches that are used for the embeddings [HYL17b].

– **Deep Learning**

The encoding approaches discussed so far can be classified as direct [HYL17b] or shallow [KGJ$^+$19] encoders, since they generate unique embeddings for each node independently. This means that they do not share parameters between nodes and they do not use node attributes during encoding, which means that a lot of potentially important information is not used for the embedding process and it is also rather inefficient. Parameter sharing would enable better regularization of embeddings and also reduce the number of parameters needed overall.

Additionally these methods are inherently transductive, meaning that they are not generally able to generate embeddings for nodes that were not present during the training process or they at least need to perform additional rounds of optimization for such nodes. While this is less of a problem for relatively static graphs, it does not make it possible to generalize to new graphs after training and to deal with graphs too big to be fully stored in memory or graphs that evolve quickly [HYL17b].

To address these issues, more complex encoders are needed. Neighbourhood autoencoder methods use a deep learning approach called autoencoders [HS06] to explicitly incorporate graph structure into the encoding process. Examples of this are Deep Neural Graph Representations (DNGR) [CLX16] and Structural Deep Network Embeddings (SDNE) [WCZ16]. While they are able to compress and use information about the local neighbourhood of nodes, they are still transductive since the structure and size of the autoencoders is fixed [HYL17b].

One solution to these problems is to use encoders that generate embeddings by aggregating information from a node's local neighbourhood, relying on node features or attributes to do so [HYL17b]. This neighbourhood aggregation approach is taken by several recent frameworks, like the GraphSAGE algorithm [HYL17a] and various graph convolutional network (GCN) approaches [KW16, SKB$^+$18]. For example, relational GCNs (R-GCNs) [SKB$^+$18] apply a relation specific transformation to a node's neighbours in the aggregation function, i.e. one that depends on the direction and the label of the edge [KGJ$^+$19].

The main difference compared to direct encoding approaches is that the trainable parameters - aggregate functions and weight matrices - are shared across all nodes. These parameters specify how to aggregate information from the local neighbourhood of a node and they are used to generate embeddings for all nodes, with the input node attributes and neighbourhood structure being the only parameters dependant on the specific node being embedded [HYL17b]. Since the number of parameters is therefore independent of the size of the graph, these approaches are more efficient while also providing regularization and making it possible to generate embeddings for nodes that were not present during the training phase [HYL17a].

A similar approach is used by GNNs [SGT$^+$08], which is more directly aimed at embedding entire subgraphs. For GNN, subgraphs specify a "compute graph", i.e. a blueprint for accruing information and passing it between nodes [HYL17b]. In the original framework, nodes are initialized with random embeddings and simple neural network layers are used to accumulate inputs from nodes' neighbors until the embeddings converge. They are then aggregated and the aggregated embedding is then used for subgraph classification. Li et al. [LTBZ15] extended the framework, with their version being able to utilize node attributes and converging faster. While GNN is very expressive, it is also computationally expensive compared to convolutional approaches and is therefore mainly used for complex, smaller scale tasks [HYL17b].

- **Decoders**

  The decoding process consists of a pairwise loss function that defines how decoded values can be compared to the "true" values of nodes and how to measure the quality of the reconstructions as well as the actual decoder function. Its aim is to reconstruct the proximity values of nodes from the embeddings generated by an encoder. In contrast to encoders, it usually does not contain trainable parameters [HYL17b].

  The three main categories of decoding frameworks are distance-based, semantic matching and techniques utilizing deep learning. Distance-based and semantic matching scoring functions are some of the earliest approaches in KRL while more recent models focus heavily on deep learning and neural networks, rarely using distance-based approaches [JPC$^+$20]. As mentioned previously, the different decoders and scoring functions can be combined with varying encoding frameworks, with the resulting system optionally being optimized using stochastic gradient descent (SGD) [HYL17b]. This whole process is usually unsupervised, with the model being trained over sets of node pairs.

– **Translational Distance**

Translational distance models, an example of which was shown in Figure 6.1, use distance-based scoring functions, which measure the plausibility of a fact as the distance between two entities in the representation space [WMWG17]. The idea is based on distributed word representation learning [MYZ13], which aims to capture linguistic regularities like the following between major policies that US presidents are known for:

$$\mathit{MoonLanding} - \mathit{JFK} \approx \mathit{AffordableCareAct} - \mathit{BarackObama}$$

One of the first and most representative translational distance models is TransE [BUGD+13], which represents entities and relations as vectors in the same space. The relation r in a fact (h, r, t) is interpreted as a translation vector that connects the two embedded entities h and t with low error, such that when (h, r, t) holds, $h + r \approx t$ [WMWG17]. This simple and efficient approach inspired similar techniques, like TransH [WZFC14b] and TransR [LLS+15], which aim to improve on TransE by providing a better way to deal with 1-to-N, N-to-1, and N-to-N relations [JPC+20]. Since an entity has multiple aspects and different relations may focus on different aspects, a common space for relations and entities is sometimes insufficient for modelling these higher-order relations [LLS+15].

For a 1-to-N relation like *MajorPoliciesOf*, TransE might learn similar vector representations for entities like *AffordableCareAct* and *DACA* (Deferred Action for Childhood Arrivals), since they are both major policies enacted during the Obama administration, even though they are quite different entities [WMWG17]. TransR and TransH deal with complex relations by allowing an entity to have different representations with respect to different relations, such that the embeddings of *AffordableCareAct* and *DACA* might still be very close given the relation *MajorPoliciesOf*, but distant from each other given other relations.

While TransH retains most of the efficiency of TransE, modelling relations as vectors and only adding relation-specific hyperplanes [WMWG17], TransR uses relation-specific spaces instead [LLS+15]. Each relation is associated with a specific space and for any given fact (h, r, t), TransR projects the entity representations into the space corresponding to the relation r and subsequently learns translations between the projected entities. The projection matrix needed for this adds complexity and increases the number of parameters needed per relation by a factor of k (the number of relations) [LLS+15].

So while it is a powerful improvement for modeling complex relations, it is computationally expensive.

There are various other translational distance approaches, such as the recently proposed TransMS [YTZ+19], an approach that models multi-directional semantics, or TransD [JHX+15], which decomposes the projection matrix into a product of two vectors, thereby simplifying TransR. A detailed survey of these and many other approaches can be found in Wang et al. [WMWG17]

– **Semantic Matching or (Bi-)Linear Models**

Semantic matching or (bi-)linear models use similarity-based scoring functions, which usually compare the latent semantics of entities and relations through their vector space representations. They work in a similar way as translational distance models, using a scoring function to measure plausibility of facts, where facts observed in the KG have higher scores than those that are not, and solving an optimization problem maximizing the total plausibility of observed facts to learn embeddings [WMWG17].

However, while translational distance models usually represent entities in d-dimensional vector space and distance-based scoring functions measure the distance between them using additive translation with relations as h + r ≈ t, semantic matching models often use plain vector space (e.g. HolE [NRP15]) or relational projection matrices (e.g., ANALOGY [LWY17]). Similarity-based scoring functions measure the plausibility of facts through a multiplication-based function like $h^T M_r \approx t^T$ [JPC+20].

The linear and bilinear decoders differ mainly in the restrictions they impose on the matrix M_r, which models the pairwise interactions between latent entities. The bilinear model RESCAL [NTK11] does not impose any restrictions, making it fully expressive with respect to link prediction but prone to overfitting, due to the many parameters per relation, [KGJ+19].

Yang et al. [YYH+14] proposed a unified learning framework using neural networks to learn low-dimensional vectors for entities and (bi-)linear mapping functions for relations. They show that their framework can generalize many existing models like neural tensor network (NTN) [SCMN13] and TransE [BUGD+13] and propose DistMult, a multiplicative version of TransE. DistMult also uses a bilinear scoring function but simplifies RESCAL by restricting M_r to be diagonal so as to reduce the number of parameters. However, it thereby loses expressivity and can only model symmetric relations because its scoring function does not distinguish between source and target vector [KGJ+19].

ComplEx [TWR+16] is an extension of DistMult that manages to preserve the ability to model asymmetric relations by introducing complex-valued embeddings instead of real-valued ones [KGJ+19]. Since it involves redundant computations, SimplE [KP18] was developed with the aim to be a more efficient version of ComplEx. However, both ComplEx and SimplE reduce the number of parameters in RESCAL while still being able to guarantee full expressiveness under certain embedding dimensionality bounds [JPC+20].

HolE (short for Holographic Embeddings) [NRP15] is an approach that aims to combine the simplicity and efficiency of DistMult with the expressive power of RESCAL. As mentioned previously, it represents both entities and relations in plain vector space and it has been shown to be equivalent to ComplEx [HS17]. Lastly, ANALOGY [LWY17] employs a bilinear scoring function like RESCAL, but restricts the matrix M_r to be block-diagonal. As a result, it is better suited for modeling analogical properties of entities and relations, like "The Moon landing is to JFK as the Affordable Care Act is to Barack Obama" [WMWG17].

– **Deep learning-based Decoders**

Deep learning approaches usually have the same goals as the previously mentioned decoding techniques - i.e. learning tensor factorizations, performing semantic matching, etc. - but they utilize deep learning to achieve them, frequently in the form of neural networks [KGJ+19]. Often previously learned distributed representations are used to learn richer representations via complex neural structures like feed-forward or convolutional neural networks, tensor networks [SCMN13] or GCNs.

The model introduced by Socher et al. [SCMN13] can be used for link prediction using only existing facts already in the KG. It represents entities as vectors and defines relations through the parameters of NTNs, which are able to explicitly relate two entity vectors. The technique utilizes standard forward propagation and back propagation techniques modified for the NTN as well as several bilinear components. It generalizes several previous neural network models, like the one by Yu et al. [YDS12], which uses tensor layers for speech recognition but is only applicable in a more restricted setting. The model can be viewed as learning tensor factorizations, similar to RESCAL [NTK11], but it tends to perform better [SCMN13].

Semantic matching and linear/bilinear approaches can also be modeled by neural networks. A representative neural model for semantic matching is semantic matching energy (SME) [BGWB14], which uses linear and bilinear matching to calculate the semantic similarity between entity-relation pairs. Other neural models include neural association models (NAM) [LJE+16] and multi-layer perceptrons (MLPs) [DGH+14a]. All of these approaches have in common that they use deep neural networks to compute a semantic matching score from entities and/or relations. In the case of MLP, entities and relations are encoded together into a fully-connected layer and a second layer with sigmoid activation is used for scoring triples.

These deep learning models, whether they are performing semantic matching or learning tensor factorization etc. have very competitive predictive performance [JPC+20]. However, as is typical for deep learning, their results are not transparent and lack interpretability. While there are attempts to design interpretable deep learning models, most approaches focus on large-scale pre-training or more powerful neural architectures, because the performance of deep learning approaches is frequently superior to conventional techniques regarding other various metrics [JPC+20].

A closely related area of research is that of SRL, which will be discussed in detail in Section 6.4.4. While SRL is also statistics-based as it uses statistical inference, it also incorporates rules and does not conform to the encoder/decoder framework [KGJ+19]. It naturally captures uncertainty about facts and relations while also being more interpretable than most KRL approaches, at the price of efficient computability. Since SRL utilizes both rules and statistical inference, it will be discussed again in the section on combined approaches (see Section 6.4).

Similarly, there are other related methods involving representation learning with graph-structured data, like latent space models of social networks [HRH02], manifold learning algorithms [LV07], and geometric deep learning [BBL+17]. They will also be discussed in the context of combined approaches in Section 6.4.2. Other types of additional information can be incorporated into the embedding process as well, as will be laid out in the following subsection.

6.2.1 Incorporating Additional Information

Most of the techniques mentioned so far perform embeddings using only facts observed in the KG. Usually, the embeddings learned in this way only have to be compatible within individual facts and are therefore sometimes not predictive enough for certain downstream tasks [WWG15]. As a result, there have been many attempts to facilitate more effective knowledge representation by incorporating additional, sometimes external information [WMWG17] such as textual descriptions, entity types, visual information, relation paths or logical rules. While combinations of embeddings with the latter two types of additional information will be discussed separately in Section 6.4, the remainder of this subsection will be dedicated to other kinds of multi-modal embeddings.

- **Textual Descriptions** In many KGs, entities have textual descriptions that provide additional semantic information and there have been a few attempts at embedding this unstructured textual information together with the structured knowledge contained in the KG. Joint loss functions are often used in this context [JPC+20], e.g. by Wang et al. [WZFC14a], who proposed two alignment models using entity names and Wikipedia anchors to jointly embed entities and words into the same continuous vector space. By aligning entity space and word space in this way, they aim to preserve the relationship between entities and the co-occurrences in the text corpus.

  Description-Embodied Knowledge Representation Learning (DKRL) [XLJ+16] is an extension of TransE [BUGD+13] that includes entity descriptions into the embedding process. In addition to learning the translation from head entity to tail entity represented by each relation, the model also learns to maximize the likelihood of predicting the description of an entity. To represent the semantics of entity descriptions, continuous bag-of-words (CBOW) and a deep convolutional neural model are used, with the neural model also taking word order and local interactions of words into account.

The semantic space projection (SSP) model [XHMZ17] builds interactions between triples and textual descriptions by projecting them into a semantic subspace. Entities co-occurring in a triple are embedded into a semantic space constructed by the associated textual semantics such that semantically relevant entities, as well as the loss vector between them, lay on a consistent hyperplane. SSP also uses a joint loss function , specifically a two-component objective function in the form of $L = L_{embed} + \mu L_{topic}$, where L_{embed} is the embedding-specific loss, L_{topic} is the topic-specific loss and μ is a weighting parameter.

- **Type Information**

  Knowledge graphs usually contain hierarchical or type information about the entities and relations contained in them, i.e. they represent type or subclass relationships in some way. This makes it possible to learn a lot of new knowledge from one fact, e.g. if we know that US presidents have to be natural born citizens and that Barack Obama was a US president, we can conclude that he was born in the US. In this way, one new fact concerning a superclass entity can lead to dozens of new pieces of information. Ignoring such type information means missing out on valuable knowledge, which is why there have been several attempts at incorporating type information into the embedding process.

  Type-embodied Knowledge Representation Learning (TKRL) [XLS16] is a model that aims to take advantage of hierarchical entity types by considering them as projection matrices for entities and utilizing them through two type encoders designed to model such hierarchical structures. They also use type information for relation-specific type constraints.

  Semantically Smooth Embedding (SSE) [GWW$^+$15] is a more general framework that uses additional semantic information and enforces semantic smoothness of the embedding space such that entities from the same semantic category lie close together. Two manifold learning algorithms formulated as geometrically based regularization terms are used for constraining the embedding task. While it was originally designed to incorporate entities' types or semantic categories, it can also be used to incorporate other information and can be imposed on many common embedding models.

  Zhang et al. [ZZQ$^+$18] focus on hierarchical relation structure instead, where relations can have sub-relations and semantically similar relations may form relation clusters. For example, a relation like *fatherOf* between two entities implies that they are also connected by *relatedTo* and we can infer that relations like *secretaryOfStateOf* and *attorneyGeneralOf* are semantically related since they both describe a relationship between a politician and a country. The authors therefore extend existing KRL models like TransE [BUGD$^+$13], TransH [WZFC14b] and DistMult [YYH$^+$14] to leverage this information by splitting the relation parameter in their scoring functions into a relation cluster embedding, a relation-specific embedding and a sub-relation embedding.

- **Visual Information**

  While not as common as type or attribute information, some KGs store multi-modal information in the form of pictures with entities. Liu et al. [LLGD$^+$19] use various "experts" (i.e. similarity measures) for schema alignments (see Section 3.4), with one of these experts being based on embeddings of images connected to entities. However, since the experts are scored independently and are only later combined using weighting parameters, the connection between entities and images is not as strong.

  Image-embodied Knowledge Representation Learning (IKRL) [XLLS16], which contains cross-modal, structure-based and image-based representation, encodes images into the same embedding space as entities and uses a translational distance-based scoring function. They first use a neural image encoder to construct representations of all images connected to an entity and then integrate them into an aggregated image-based representation using an attention-based method. The cross-modal representations are used to make sure that image-based and structure-based representations are in the same embedding space. [JPC$^+$20]

- **Structural Information**

  Structural information, like the global position of a node in the KG or the structure of a node's local graph neighborhood, can also lead to valuable insights. It has therefore become a recent focus in KRL to learn representations where the geometric relationships in the embedding space reflect the structure of the original graph in some way [HYL17b].

  One important area of applications for this are KGs with multiple layers or various types of nodes and edges, like recommender system graphs, whose nodes are split into the categories users and content, or biological networks like protein-protein interaction graphs. In the latter case, the same proteins (nodes) can occur across multiple tissues (layers) and this multi-layer structure can be utilized by making sure that embeddings for the same node is similar across the various layers.

  Sharing information in this way helps to regularize learning across layers and makes it possible for a node's embedding in one layer to be informed by its embedding in different layers. To solve this problem, Zitnik et al. proposed OhmNet [ZL17], which combines node2vec [GL16] with a regularization penalty, thereby encouraging the sharing of similar features among proteins activated in similar tissues and proteins with similar network neighborhoods.

Another task in this area is to learn embeddings that represent the structural roles of nodes in the graph rather than their global positions in the graph. For example, in transportation or communication networks, it is often useful to know which role nodes fulfill, i.e. whether they are periphery nodes, members of a clique, centers of stars etc. [HGER+12]. While there exist specialized approaches to solve this problem [RSF17], general purpose techniques like node2vec [GL16] can also be used by using biasing for the random walks to improve their sensitivity to structural roles.

6.2.2 Comparison and Areas of Application

As can be seen from the variety of models presented in the previous section, there is no singular best KRL model, since the performance depends on the data and the downstream task it is being used for. However, it has been shown that more complex models which consequently appear more expressive do not always perform better. Yang et al. have demonstrated that NTN [SCMN13] performed worse than TransE [BUGD+13] and DistMult [YYH+14], some of the simplest KG models. This might be due to the large number of parameters required for more expressive models and their tendency to overfit on smaller datasets [WMWG17].

A common characteristic of KRL approaches is that they project representations into a low-dimensional semantic space. They thereby give entities, relations etc. much denser representations compared to other types of reasoning, leading to lower computational complexity in their applications. This is mostly due to the fact that in a well-designed KRL system, measuring the similarity of the embeddings is sufficient to determine the similarity between the embedded entities and relations [LHX+18]. In contrast to traditional feature engineering, the representations do not have to be designed by hand, which greatly increases the efficiency during the design phase and makes it possible to apply KRL models to unknown datasets [HYL17b].

There are a number of challenges that arise in this context. While it is convenient not to have to design features by hand, automatically identifying relevant features results in a kind of black box where the output is usually not interpretable. Additionally, it leads to fundamental limitations and underlying biases being difficult or impossible to identify. The resulting systems can therefore often make predictions, but cannot derive general knowledge about the data that would be able to explain why the predictions were made [GTHS15].

A related problem is the difficulty of encoding sparse entities. Since patterns can be learned with greater confidence when there are many example values, entities and relations that have many triples in the KG correlate with more and stronger results in tasks like link prediction [ZPW+19]. This is particularly problematic since a large portion of entities in a typical KG only have a few triples, and these sparse entities have much worse prediction results than frequent ones. With time, this effect increases the difference between sparse and frequent entities and leads to prediction results that mostly focus on already well-populated areas of the KG.

Current embedding approaches are usually not well suited for dealing with dynamic, temporal graphs like financial transaction graphs as they can not usually incorporate timing information about edges [HYL17b]. They also often lack the ability to capture multi-hop relationships and reasoning patterns contained in multi-step paths [LC19]. Both of these areas have become the focus of several recent research works that aim to incorporate logical rules, temporal information and multi-hop paths into the embedding process [JPC+20]. Approaches that incorporate multiple types of reasoning like this in order to combine their advantages will be discussed in Section 6.4.

With these advantages and disadvantages, it has become clear that some of the applications KRL is best suited for are visualization, clustering, node classification, link prediction, and relation extraction [HYL17b]. Node embeddings offer a powerful tool for clustering related nodes, which can help to find patterns in the data and discover related nodes, hidden structures and communities. This can be of use in and of itself, but it is also very useful for data visualization. Node classification and link prediction are also well suited to the great pattern matching capabilities that are inherent to node embeddings. With node classification, the goal is to learn to label the full graph based on a small initial prelabeled seed set, and the aim of link prediction is to predict, based on known facts, which edges between entities are not yet contained in the KG but should be [HYL17b].

The challenges in designing a good KRL model are therefore choosing which property or properties of the graph the embedding should preserve, determining how scalable the model must be and finding the optimal dimensionality that strikes a good balance between precision and complexity. Many of these decisions will depend on the application, since they involve trade-offs. For example, it might be possible to obtain better link prediction accuracy when using less dimensions if the chosen model only encodes local connections between nodes [GF18]. Additionally, special care should be taken to make results interpretable and to ensure that the model actually learns to represent relevant graph information and does not just optimize for certain statistical patterns [HYL17b].

6.3 Graph-based Reasoning

Even though embedding-based approaches achieve impressive results in some areas of applications, most of them do not factor in structural information, fail to model complex relation paths and cannot capture multi-hop reasoning patterns [LC19]. Logic-based approaches, on the other hand, struggle to make complex inferences from automatically-extracted, imperfect knowledge and often suffer from scalability issues [LMC11].

Graph-based methods like relation path reasoning aim to address these limitations by explicitly considering path information over the graph structure as reasoning evidence. Relation paths are usually defined as arbitrarily long sequences of relations connecting a pair of entities in a graph [WMWG17], so e.g. $childOf \rightarrow presidentOf$ would be a length-

2 path connecting the daughters of Barack and Michelle Obama to the United States of America. In contrast to most embedding-based approaches, graph-based methods can also offer logical insights about the KG and the results typically provide better interpretability [LSX18].

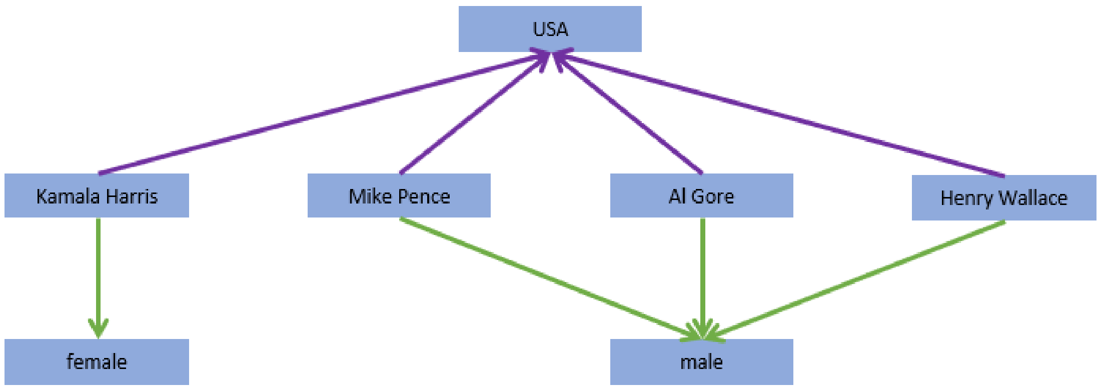


Figure 6.2: An example KG representing a subset of the US vice presidents, where green arrows denote the relationship *hasGender* and purple arrows denote *vicePresidentOf*

An important concept in this area is random walk inference, which is the process of learning to rank graph nodes relative to each other, based on how likely it is to end up at the target node when taking a k-step walk from the source node [LMC11]. The walks are usually constrained in some way, i.e. by only being able to move along edges of certain types. As an example, suppose we have a KG that contains all the vice presidents of the United States as well as their gender (Figure 6.2 shows a subset of this graph).

A random walk starting at the node *KamalaHarris* would have a 50 percent chance each of reaching the nodes *female* and *USA* after one step. However, if n is the number of vice presidents, a 2-step random walk starting at *USA* would only have probability 1/n of reaching *female* (assuming there is a constraint that the same edge cannot be taken twice). As such, the ranking associated with this path represents the prior probability of a US vice president x to be of gender y. This might not by itself be a useful, high-precision inference rule but it can for example be used as a feature in a combined ranking method [LMC11].

Early work in this area, like the PRA, treat this as a link prediction problem and perform maximum-likelihood classification over discrete path features, while some more recent models [DNBM16, GML15] perform the classification over vector space representations [LSX18]. We will discuss these approaches first and then move on to models that treat multi-hop reasoning as a sequential decision problem, using reinforcement learning to perform effective path search [LSX18]:

- **Path-Ranking Algorithm and related approaches**

  The PRA [LMC11] uses random walks for path-finding and making probabilistic inference over KGs. While connecting nodes in a graph through a path had been done previously [CBK$^+$10], these approaches considered just one query during path finding, whereas the PRA uses statistics over multiple queries, combining them using a learned logistic function. This makes it possible to make statements about the importance of a path and in combination with efficient approximation schemes learning and inference are much more efficient than in the previous logic-based approaches.

Previous iterations of the algorithm enumerated the paths between query and target nodes completely, but this is only viable when the number of edges and edge types is small [LMC11]. To achieve the greater efficiency necessary for processing large data sets like KGs, in recent versions the paths are determined through training queries so that only paths that are potentially useful for the task at hand are used. The requirements are, among others, that nodes created during path finding need to be supported by a certain fraction of training queries and that paths need to retrieve at least one target entity in the training set and do not exceed a certain length.

To learn an inference model for link prediction, the PRA finds sequences of edge types that frequently connect nodes that are also connected by the relation being predicted. In this case, the paths used in the training phase would be further restricted to those including nodes connected by this target relation. The PRA would then use the random walk probabilities to compute the values of a feature matrix and perform relation-specific classification, using these features in a logistic regression model to predict missing edges [CJX19].

Gardner et al. [GTKM14] proposed a variation on the PRA which incorporates textual content to relieve feature sparsity issues that the PRA sometimes faces. Their method combines this textual information and the KG relations into a single vector space embedding that the feature similarity is then computed on. To further reduce the number of paths and improve upon the performance of the PRA, related paths according to their relations' embeddings are clustered together and unseen paths are mapped to semantically close paths that were seen at training time [NRM15]. This combination of distributional similarity and symbolic logical inference draws on the advantages of both kinds of reasoning and will be discussed alongside other combined approaches in Section 6.4.

- **Neural multi-hop relational path modeling**

  Neural multi-hop models use neural networks to compose the semantics of relational paths. In contrast to KRL models that deal with path queries by recursively applying their embedding models and therefore suffer from cascading errors, these models use an integrated, compositional training objective [GML15]. Guu et al. [GML15] show that this technique can be applied to various embedding models that involve an intermediate vector representation that does not implicitly encode the target relation and can therefore be decomposed, like TransE [BUGD$^+$13] and many bilinear models.

Neelakantan et al. [NRM15] use RNNs to compose the distributed semantics of relations in multi-hop paths. At each step in the path the neural network consumes both the vector representation of the path so far and of the next relation and outputs a composed vector which will again serve as input for the next step. This way, after consuming a path, the RNN should output a vector that is semantically similar to the relation between the first and last entity of that path. For example, when consuming a relation chain like *MichelleObama* - *marriedTo* - *BarackObama* - *presidentOf* - *USA* - *presidentLivesIn* - *WhiteHouse* - *locatedInCity* - *WashingtonDC*, their method would produce a vector close to the relation *livesInCity*.

Their approach is related to that of Gardner et al. [GTKM14], but does not require classifiers for each predicted relation type based on atomic-path features. This makes it possible to predict links from paths that were not seen during training. While they originally trained separate RNNs for each relation type, by learning a single high-capacity composition function for all relations instead, it could even perform zero-shot learning, i.e. predict relation types for which the composition function was never explicitly trained.

Das et al. [DNBM16] proposed Chain-of-Reasoning which aims to do just that. They use a single, high-capacity RNN that represents logical composition across all relations, while also extending the approach by Neelakantan et al. by reasoning jointly about relations, entities, entity-types and text since the same relation can connect entities of varying types which affect the meaning of the connection. Additionally, they use a neural attention mechanism to be able to incorporate multiple paths connecting the same entities. Overall, this leads to increased accuracy and better efficiency through shared RNN parameters during the training stage.

- **Reinforcement Learning**

  While particularly the neural multi-hop models have achieved very good results, the paths they use for the training phase are gathered through random walks, which can be rather noisy [LC19]. Reinforcement learning based approaches were designed to address this problem and they are now the state-of-the-art method in this area [ZCGZ19]. These approaches model multi-hop path completion as sequential decision making, specifically a Markov decision process (MDP), and use reinforcement learning to learn a policy that guides the agent in choosing the entity and relation pair to jump to next so that the expected reward is maximized.

  One of the first deep reinforcement learning models proposed in this area was DeepPath [XHW17], which parametrizes its policy-based agent with a fully-connected neural network whose continuous states are based on fixed pretrained embeddings for the entities and relations. The agent reasons in vector space by sampling the most promising relation for extending a path, and the gathered paths are then fed to the PRA. DeepPath uses a novel, manual reward function that takes accuracy, path diversity and efficiency into account. A linear combination of these criteria are used as the positive reward during the training phase, while a hand-crafted constant is applied as the negative penalty [LC19].

One of the drawbacks of DeepPath is that their MDP needs to know the target entity in advance and their path finding strategy is therefore dependent on knowing the answer entity [DDZ+17]. This restricts the applications it can be used for in a similar way to random walks, which are also impractical in settings with an unknown target since there are too many possible paths from any given start node. Both methods are useful for link prediction and triple classification, but inefficient when it comes to answering questions where the relation is known, but only one entity. To solve this issue, MINERVA [DDZ+17] was proposed, a neural reinforcement learning approach which uses the REINFORCE algorithm [Wil92] to train an end-to-end model for multi-hop query answering over KGs.

MINERVA parametrizes its policy-based agent with a LSTM and uses a binary reward, with path validity as the only reward criterion. During the training phase it evaluates whether the current path reaches the target entity, thereby learning to navigate the graph and find predictive paths based on the input query. The trained agent, when given a query relation and a source entity, will then find candidate answers without access to any pre-computed paths [LSX18]. MINERVA does not need pretraining and it can deal with paths of various lengths while being computationally efficient, since it avoids ranking all entities in the KG by searching in a small neighbourhood around the query entity [DDZ+17].

One of the problems with binary reward functions is that they suffer from sparse rewards, i.e. the agent only receives a reward at the end of a search path, having correctly of incorrectly predicted a target entity, and does not receive any feedback on the intermediate steps. This can lead to false negatives and false positives in the training data, e.g. when an agent is misled by spurious search trajectories incidentally leading to the correct answer [LSX18].

M-Walk [SCH+18] is a similar approach as MINERVA and also uses binary rewards, but aims to alleviate the problem of sparse rewards by using a deep RNN to capture the history of traversed nodes and Monte Carlo Tree Search (MCTS) for effective path generation. The MCTS is used in combination with the neural policy to generate trajectories yielding more positive rewards. The network is then improved via Q-learning, which modifies the RNN policy via parameter sharing. By applying this policy-improvement step repeatedly, the training phase can be completed much faster than e.g. in methods using the REINFORCE method, since more positive rewards are available faster.

A different attempt at solving the problem of sparse rewards is Multi-Hop [LSX18], which uses a soft reward mechanism instead of a binary reward function. To reduce the impact of false negative supervision, a pretrained one-hop embedding model is used for estimating the reward of unobserved facts, while edges are randomly masked during training to force the agent to explore more diverse paths and therefore reduce the sensitivity to spurious paths.

While RL-based methods generally need very minimal manual tuning, the reward function is still often adjusted for different datasets to achieve good performance, which makes it difficult to adapt to changing environments and fast-evolving KGs. DIVINE [LC19] is a plug-and-play framework that can be used to enhance existing approaches in this regard, by using generative adversarial imitation learning (GAIL) [HE16] to learn policies and reward functions self-adaptively.

The reasoner consists of a generator and a discriminator, where the generator is one of the policy-based agents from an existing approach and the discriminator works like a self-adaptive reward function. It is trained from demonstrations automatically sampled from KGs and makes it possible for the reward function to be automatically tuned to an approximately optimal performance, thereby reducing the need for reward engineering and manual intervention.

One of the properties that unites graph-based methods and one of their biggest advantages is their interpretability. Relation paths can be viewed as the bodies of weighted rules, with the weight specifying how predictive the body is for the head [NMTG15]. On the other hand, many of them share the disadvantage of only being able to deal with paths up to a certain length, which might be shorter than the actual distance between some source and target entities. This can lead an algorithm to deduce that entities are not connected even though there is a path from one to the other, leading to them not being considered for further reasoning. As a result they are then implicitly missing from the inference results [ZCGZ19].

6.4 Combined Approaches

Many of the approaches that we discussed above can be considered to combine multiple types of reasoning - for example by using KRL to embed paths or rules. However, this is a relatively recent trend, as the reasoning types used to mostly be studied and developed in isolation, with few attempts at understanding the synergies between them and combining them [Kaz18]. Because the different reasoning types are often well-suited to specific types of KG or patterns therein - i.e. clusters, local neighbourhoods, strongly connected components etc. - and a KG contains varying patterns, none of the state-of-the-art models can be said to be superior in general [NMTG15]. To achieve good results on general purpose knowledge graph it is therefore beneficial to combine multiple reasoning approaches.

Table 6.1 shows a high-level comparison of the advantages and disadvantages of statistics-based, logic-based and graph-based reasoning. These aspects will be discussed in detail in the following subsections, where we will be contrasting each type of reasoning with the other two types and show possible synergies between them.

	Statistics-based	Logic-based	Graph-based
Advantages	+ generalization + able to represent semantic similarity + fast, when small number of latent variables is needed	+ interpretable + support complex reasoning + can deal better with sparse entities	+ interpretable + naturally factors in graph structure + fast, when reasoning over short paths/small neighbourhood is sufficient
Disadvantages	- not interpretable - difficulty learning good representations for sparse entities - difficulty dealing with strongly connected components	- dependence on background knowledge - difficulty dealing with imperfect knowledge and noise	- difficulty dealing with low-connectivity graphs

Table 6.1: A comparison of the different types of reasoning along with their advantages and disadvantages. References can be found in the respective subsections.

6.4.1 Combination of Logic- and Statistics-based Approaches

The main problem of statistics-based approaches is that they do not usually make it possible to find out how and why conclusions are drawn. On the other hand, their ability to represent semantic similarity, ability to generalize and efficiency are quite valuable for many reasoning tasks [CJX19]. Logic-based approaches do, in contrast, provide interpretable results, are able to capture the rich semantics of natural language and support complex reasoning but they are dependant on logical background knowledge and less scalable because of the huge search space.

Therefore, approaches aiming to combine the advantages of these two types of reasoning have received some attention lately [WMWG17]. Rule injection, i.e. the incorporation of rules into other learning systems like embeddings, can increase efficiency by turning rule learning from discrete graph search into vector space calculations while using deductive rules to learn additional triples for sparse entities improves the performance of embedding models, since they rely heavily on data richness [ZPW$^+$19].

Since representing knowledge in a continuous semantic vector space can help to overcome the incompleteness and brittleness of a KG, Summers et al. [SS17] propose a method for performing deductive reasoning directly in such a vector space. They aim to discover chains of reasoning that connect a premise to a conclusion, using analogy, association and deduction to find semantically close entities that can help to answer queries. For example, when trying to find a connection between the two concepts *BarackObama* and *Citizens*, if the KG contains facts like (*BarackObama*, *isA*, *USPresident*) and (*federalOfficial*, *represents*, *Citizens*) the proposed method would return a combination of paths like *BarackObama* → *USPresident* and *federalOfficial* → *citizens*. Even though they do not strictly form a chain of reasoning, *USPresident* being close to *federalOfficial* in the semantic space would be enough to form a connection.

Most of the other techniques in this area combine the two kinds of reasoning by finding a way to efficiently inject rules into the embedding process, be it by using them to guide representation learning or by actually embedding rules. One of the earliest of these approaches is that by Rocktäschel et al. [RSR15], which uses first-order logic rules to perform pre-training inference on the data, thereby gaining additional information and making the embedding process more robust. Additionally they use a joint optimization objective to guide embeddings by rewarding predictions that satisfy some predefined logical background knowledge. Wang and Cohen [WC16] use matrix factorization to learn how to map training examples and inference rules into one matrix, using these embeddings in turn to learn parameters for the rules.

Wang et al. [WWG15] propose an approach that formulates inference as an integer linear programming (ILP) problem where the objective function is generated from embedding models and the rules are transformed into constraints. Solving the ILP problem leads to facts that comply with all the rules and that the embedding models prefer the most. Since the rules and embeddings are modeled separately, with rules serving only as a post-processing step, they do not improve the quality of the embeddings [WMWG17].

Guo et al. [GWW+16] proposed KALE, a method that jointly embeds facts and logical rules, modelling both in a unified framework. They represent triples as atomic formulae, using the translational distance model TransE [BUGD+13] to measure semantic differences. Rules are represented as complex formulae and t-norm fuzzy logic [Háj13] is used to model the composition functions of the logical connectives conjunction, disjunction and negation in order to define the truth values of the formulae. KALE then learns entity and relation embeddings by minimizing a global loss over both atomic and complex formulae. These embeddings are compatible with both the triples in the KG and the rules, enabling the prediction of new facts that could not be directly inferred by pure logical inference [CJX19].

Yang et al. [YYC17] propose NeuralLP, an approach combining neural and symbolic models to learn probabilistic FOL rules in an end-to-end differentiable way. Their proposed neural controller system learns to compile inference tasks into sequences of differentiable operations and then compose these operations in various ways by integrating an attention mechanism and auxiliary memory, learning the parameters in a continuous space and the structure in a discrete one. By combining the parameter and structure learning in this way, they are enabling the application of gradient-based optimization to inductive logic programming [JPC+20].

RuLES [HSGE+18] is an end-to-end rule learning system guided by a precomputed embedding model over the KG and external information sources like text. Since statistical quality measures for learned rules often depend on a reasonably complete KG, it is hard to learn high-quality rules from facts in the KG alone. RuLES therefore constructs an approximation of an ideal graph by learning entity and relation embeddings from the KG as well as additional information sources, that then serves as a probabilistic representation of missing facts. They then iteratively extend rules induced from a KG by relying on feedback from this embedding model for ranking and pruning rule candidates, with the feedback helping to distinguish exceptions from noise [SGEH18].

A similar approach, called IterE [ZPW$^+$19], proposes an iterative training strategy that consists of the three components embedding learning, axiom induction and axiom injection. In the fist component, embeddings are learned based on triples already included in the KG, with some of them inferred by rules that were predefined or found in a previous step of the iteration. Axiom induction then generates a pool of possible axioms and scores them with a calculation based on the embeddings. The highest scoring axioms, or rules, are then used to infer new triples and they are injected into the KG in the third component in order to improve sparse entity embeddings. This process is repeated iteratively during training, so that rules and embeddings are learned at the same time.

Qu et al. [QT19] propose a probabilistic logic neural network (pLogicNet), which aims to combine the advantages of Markov logic networks and KRL methods to learn rules and assign a weight to them that represents how predictive they are. The Markov logic network is used to model the joint distribution of all possible triples and it can be efficiently optimized with the variational EM algorithm [NH98]. During the E-step, missing triples are inferred using an embedding model, while during the M-step these new inferred triples are used along with previously existing ones to update the weights of logic rules.

6.4.2 Combination of Statistics- and Graph-based Approaches

Neither statistics-based nor graph-based models can be said to be superior for reasoning over knowledge graphs, as their strengths are complementary [TC15] and they are suited for different kinds of data and tasks. Statistics-based methods are well suited for modeling global relational patterns and when triples can be explained with few latent variables they are quite computationally efficient [NMTG15]. Graph-based methods on the other hand are better suited for modeling local and quasi-local graph patterns and when triples can be explained from such a small neighborhood of an entity or from a short path, they are also computationally efficient.

Nickel et al. [NJT14] have shown that tensor factorization is often inefficient when dealing with relational data consisting of a large number of strongly connected components, while graph-based methods can often handle these relations efficiently. These kinds of contrasting strong points therefore led many researchers to try to combine statistics- and graph-based models so as to increase the predictive performance. One of the challenges these combined approaches face is how to represent paths and other graph features in a vector space along with entities and relations [WMWG17]. Paths, for example, are often represented as a composition of their constituent relations' representations, with the composition typically being done through addition, multiplication or RNNs.

Path-based TransE (PTransE) [LLL+15] is an extension of TransE [BUGD+13] that considers relation paths as translations between entities. To this end, the objective $h + r \approx t$ is generalized to also regard multi-step relation paths as connections in addition to direct relations. Since many relation paths are not reliable or meaningful, with relation paths that lead to various possible tail entities being most unreliable, PTransE uses a path-constraint allocation algorithm to select the most reliable and useful paths.

Nickel et al. [NJT14] designed a generic framework to make tensor and matrix factorization algorithms more efficient by determining the optimal rank for a given data set using both latent and observable variable approaches. They show that the maximum and minimum rank of an adjacency tensor can be determined through the structure of the KG. Since the rank of a factorization is an important factor in determining runtime, with approaches often scaling cubical with respect to the rank, their approach can reduce runtime and memory complexity significantly by reducing the required rank.

Another line of research combines KG embedding with relation-path inference methods, like the PRA (see Section 6.3). Dong et al. [DGH+14a] proposed a fusion method to combine the PRA with a MLP, i.e. a kind of neural network where entities and relations are encoded together into an input layer, mapped to a non-linear hidden layer and an output layer with sigmoid activation is used for scoring triples. They build two prior models based on MLP and the PRA and for any given pair of entities their scores are combined, with the respective weights learned through training a binary classifier.

Neural multi-hop relational path modeling approaches, which were discussed in Section 6.3, also use neural networks to compose the semantics of relational paths. Both Neelakantan et al. [NRM15] and Das et al. [DNBM16] train RNNs that take, at each step in the path, the vector representation of the path so far and of the next relation and compose them into a new vector, which is again used as input for the next step. While Neelakantan et al. train separate RNNs for each relation type, Das et al. use a single, high-capacity RNN across all relations, which enables them to perform zero-shot learning.

Both of these approaches have difficulty dealing with sparse data, which is why some related approaches try to improve upon the data by incorporating other external content. Gardner et al. [GTKM14] combine relations and textual content into a single graph representation, and incorporate vector space similarity into random walk inference by clustering together semantically close paths, thereby reducing feature sparsity issues.

While the approaches above are mostly optimized toward link prediction, random walks can also be used to approximate other properties of the graph, like node centrality or similarity. Several embedding techniques, like DeepWalk [PARS14] and node2vec [GL16], aim to encode this kind of information about the graph and the neighbourhood of nodes by encoding the statistics of random walks. While DeepWalk includes both predecessors and successors of a node in the embedding and uses simple unbiased random walks, node2vec only considers a node's successors but employs are more flexible definition of a node's neighbourhood by using biased random walks. These allow, through the manipulation of the associated hyperparameters, to learn different embeddings that emphasize local structural roles or community structures and ideally preserve both [GF18].

Perozzi et al. [PKCS17] extend DeepWalk by skipping over a varying number of steps in random walks in an approach called WALKLETS. They aim to capture a node's latent community hierarchy in this way, since nodes may have similar far-away neighbours that are never reached with ordinary random walks, which are usually rather short. Another extension to various random walk based approaches is HARP [CPHS17], which adds a graph pre-processing step during which related nodes are collapsed into supernodes. Regular random walk algorithms can then be run on an embedding of this coarsened graph, with the embeddings of supernodes being used as initial values for the embeddings of their constituent nodes. This process can be repeated in a hierarchical manner and can improve the performance of the approaches it is used in conjunction with by preventing them from being stuck in local optima [GF18].

6.4.3 Combination of Logic- and Graph-based Approaches

There have been quite a few recent works aiming to understand the relationship between and combining statistics-based approaches with both logic- and graph-based approaches, as exemplified in the previous subsections. However, there are far fewer works that combine logic- and graph-based reasoning, which might be due to the fact that they are a lot more similar to each other and therefore do not have quite such contrary advantages and disadvantages. Specifically, relation paths can actually be viewed as the bodies of weighted rules, where the weight specifies how predictive the body is for the head [NMTG15].

Kazemi et al. [Kaz18] compare the PRA with relational logistic regression (RLR), a weighted rule learning approach. They provide a normalization technique for relations and prove that a restricted subset of RLR using such normalized relations, called RC-RLR, generalizes PRA models. They go on to show that these results can be extended to other weighted rule learning and graph random walk models. Performance gains are mostly achieved by limiting one paradigm in ways that are tailored to the task at hand, retaining only the necessary reasoning capabilities while excluding more complex but computationally expensive ones.

6.4.4 Combination of Logic-, Statistics- and Graph-based Approaches

To the best of our knowledge, Ma et al. [MQH$^+$19] is the only approach so far, which explicitly uses all three kinds of reasoning. As described in the previous subsection, relation paths can be interpreted as being the bodies of weighted rules and relation path learning is therefore a special case of rule learning, while paths are obviously just generalizations of relations, which can be interpreted as length-1 paths. Still, these special cases have different strong points and Ma et al. aim to combine all of them within their link prediction approach called ELPKG.

They first use a combination of the PRA and tensor factorization to represent relations between query entities in order to measure both their semantic similarity and structural similarity. Based on this combined representation, they then use probabilistic soft logic to infer the probability of a missing relation between the entities, since it allows them to better deal with uncertain, non-deterministic knowledge.

However, also on the intersection between all three kinds of reasoning is statistical relational learning, which is used as a blanket term for models that can deal with both complex, relational structure and uncertainty. Other names for these and related models are relational machine learning (RML) and statistical relational artificial intelligence (StaR-AI). They comprise many methods combining probability and predicate logic in various forms [KNP11], aiming to unite the high precision of rule-based techniques with the high recall of Machine Learning or pattern-based approaches [WT10].

To model the uncertainty, probabilistic graphical models like Bayesian or Markov networks are often used as they can be used to perform joint reasoning, i.e. reasoning over sets of facts and their joint probability distribution. Markov Logic Networks (MLN) are arguably the most well-known method and probably the most versatile in combining the strengths of statistics- and rule-based approaches. They are using first-order logic (FOL) and probabilistic graphical models to elegantly handle the noise in both logic rules and knowledge graphs.

In the Markov Logic framework, FOL rules are grounded against base and candidate facts to produce a set of propositional-logic clauses, whose literals are interpreted as binary random variables, with a dependency between literals appearing in the same clause. This probabilistic structure forms a Markov Random Field (MRF), which can be viewed as the machine-language representation of MLNs, and is then used by MLN solvers to compute the joint probability distribution of all variables, or to determine the most likely joint assignment of truth values [WT10].

Unfortunately, as is the case for most probabilistic graphical models, inference and learning in MLNs, or rather the underlying MRFs, is computationally expensive since the cost of constructing the ground Markov network is exponential and the optimization problem is NP-complete. Therefore, efficient approximations and techniques need to be developed to improve the basic approach in both accuracy and efficiency, for example Markov Chain Monte Carlo (MCMC) sampling [PDS08]. Still, to date MLNs struggle to handle large-scale knowledge bases in practice, but there are many approaches that aim to tackle the inefficiencies in various ways.

ExpressGNN [ZCY$^+$20] aims to overcome the scalability issues of MLNs using both an efficient stochastic training algorithm as well as a compact posterior parametrization with graph neural networks, which encode local graph structures and node attributes and are thereby able to learn effective representations of nodes. Advantages of this approach include the compactness of the model and the capability to employ inductive learning, which is making it possible to leverage domain or human knowledge that is encoded in logic rules.

Probabilistic Soft Logic (PSL) [KBB^{+}12] also uses FOL to describe features that define MLNs. However, PSL makes it possible to specify rich probabilistic models over continuous-valued random variables rather than binary variables. At the same time, most probable explanation (MPE) inference is cast as a convex optimization problem instead of a combinatorial one, thereby making it solvable in polynomial rather than exponential time [PMGC13].

Additionally, there are many related methods in the areas of inductive logic programming, weighted rule learning, probabilistic inference and random walks on graphs that can be seen as belonging to SRL [Kaz18]. For a more detailed discussion on SRL in the context of knowledge graphs, we refer the reader to the survey by Nickel et al. [NMTG15]

CHAPTER 7
Conclusion

Even with the recent surge of research in the area of reasoning in the context of knowledge graphs, there was to date no comprehensive survey encompassing multiple types of reasoning as well as the whole life cycle of a KG. As discussed in Section 1.2, most existing surveys only cover the subject from the view of a certain domain like machine learning, or they just focus on a particular life cycle stage. In order to fill this gap, this thesis aimed to provide a survey covering all of these issues and in doing so, to answer the research questions posed in Section 1.1:

Research Question 1: Which reasoning approaches have been proposed over time and how can they be classified based on their underlying type of reasoning?

With this research question, we were aiming to collect a comprehensive set of research works which have been published over time concerning reasoning and knowledge graphs. To this end, we first identified relevant research works based on the criteria and methodology set forth in Section 1.4. We then developed classification criteria and established the dimensions by which we were going to structure the multitude of research works, categorizing them according to the type(s) of reasoning used as well as the relevant life cycle stage(s). We then analyzed the collected research works and surveys based on the utilized reasoning techniques and provided a structured overview of logic-based, statistics-based and graph-based reasoning in Chapter 6.

Research Question 2: What are the respective strengths and weaknesses of these approaches and which applications are they therefore most suited for?

The focus of this research question was a characterization of the approaches and techniques most prevalent in the previously collected research works based on their strengths and weaknesses. Since members of a specific type of reasoning often share similar characteristics, we grouped the approaches into the categories of logic-based, statistics-based and graph-based reasoning and discussed their advantages and disadvantages in Sections 6.1, 6.2 and 6.3, respectively.

We then put the rather abstract strengths and weaknesses into perspective by showcasing which tasks in a KG life cycle the various types of reasoning are most suited for, both in terms of accuracy and efficiency. We have addressed the life cycle stages of knowledge integration, knowledge graph evolution and applications, based on a simplified version of the classifications of Auer et al. [ABT14] and Pouchard [Pou15].

We explored knowledge integration in Chapter 3, talking about the various ways of extracting knowledge from external sources and integrating them into a unified representation. Knowledge graph evolution was presented in Chapter 4, divided into techniques aiming to complete the KG in terms of coverage and those aiming to improve the existing facts by employing error correction. Finally, in Chapter 5 we reviewed which out-of-KG applications KGs can be used for, with a particular focus on recommender systems and question answering. In each area, we presented state-of-the-art techniques and discussed challenges arising in their respective context.

Research Question 3: Given these strengths and weaknesses, which synergies exist between the different approaches and how can they be combined?

Because the strengths and weaknesses of different types of reasoning often complement each other, combining them can help to exploit various synergies between them. Unfortunately, many surveys focus on only one type of reasoning, thereby mostly ignoring this potential. However, there have increasingly been efforts to combine various approaches so as to be able to harness their advantages while hopefully being able to avoid their disadvantages.

Since they are so underrepresented in previous surveys, we aimed to bridge this gap and put a particular focus on combinations of various types of reasoning. To this end, we have examined several combined approaches in detail in Section 6.4. These approaches, which incorporate features from two or more types of reasoning, are often neglected in spite of their copious advantages and their ability to unite the strong points of various types of reasoning. In order to provide readers with a cursory overview of the possible beneficial combinations, we have summarized the advantages and disadvantages of logic-based, statistics-based and graph-based reasoning in Table 6.1.

In addition to answering these research questions, we summarized important background information and explained underlying concepts in Chapter 2. We hope to facilitate future research by establishing a common terminology throughout the thesis and by additionally providing pointers to more detailed surveys for several subtopics.

7.1 Outlook and Future Work

In addition to the fact that most surveys only cover one type of reasoning or one aspect of the life cycle in depth, they are furthermore often only familiar with and cite research from within their own discipline. This leads to strongly segregated research areas, mostly along the types of reasoning involved, and research about topics at their intersection can be found in either discipline, but is rarely connected because of the differences in vocabulary etc. What would therefore be needed for such an interdisciplinary topic are common formalisms and benchmarks to make research results more comparable and relatable. We expect that this will be one of the most fruitful and important future developments for the field, and hope to have provided a useful overview of the different fields that will facilitate reaching this objective.

There are many other interesting open problems in the area of reasoning on knowledge graphs, the solutions of which could help KGs to become the ensemble of human knowledge that they were designed to be. For example, the addition of common sense and procedural knowledge could enable them to be adopted for many additional applications, invariably leading to new challenges that are waiting to be solved. Another open problem is the implicit bias that is incorporated into the knowledge graph when learning from data automatically and in an undifferentiated way. Going forward, it will be one of the most important issues to find ways of eliminating or at least reducing this bias.

List of Figures

List of Tables

Acronyms

CNN	convolutional neural network. 54
CVT	compound value type. 12
CWA	Closed World Assumption. 73
DKRL	Description-Embodied Knowledge Representation Learning. 79
DL	description logic. 13, 38, 61
EEL	entity extraction and linking. 21–23
FM	Factorization Machine. 49
FOL	first-order logic. 13, 24, 90
GAIL	generative adversarial imitation learning. 88
GCN	graph convolutional network. 75, 78
GNN	graph neural networks. 50, 75
HARP	Hierarchical Representation Learning for Networks. 74, 93
IKRL	Image-embodied Knowledge Representation Learning. 81
ILP	Inductive Logic Programming. 63–67
KB	Knowledge Base. 12, 13, 15, 66
KG	Knowledge Graph. 1–9, 11–17, 20–22, 25–27, 29, 30, 33–43, 45–55, 59–65, 67–71, 73, 77–79, 81–85, 87–92, 97–99, 101
KGAT	Knowledge Graph Attention Network. 50
KGMS	Knowledge Graph Management System. 70

KRL Knowledge Representation Learning. 5, 6, 71, 72, 75, 79, 81–83, 85, 88, 91

LCWA Local Closed World Assumption. 66

LSTM long short-term memory network. 48, 54, 87

MCCNN multi-column convolutional neural network. 52, 53

MCTS Monte Carlo Tree Search. 87

MDP Markov decision process. 86, 87

ML Machine Learning. 71

MLP multi-layer perceptron. 78, 92

NFM Neural Factorization Machine. 48, 49

NLP Natural Language Processing. 51, 71

NTN neural tensor network. 77, 78, 82

OWA Open World Assumption. 65, 66

PRA path ranking algorithm. 37, 84–86, 92–94

PSL Probabilistic Soft Logic. 68, 69

PTransE Path-based TransE. 92

QA question answering. 50–53, 55

RDF resource description framework. 15, 67, 69, 70

RL reinforcement learning. 29, 88

RLR relational logistic regression. 93

RNN recurrent neural network. 37, 54, 86, 87, 92

SGD stochastic gradient descent. 76

SME semantic matching energy. 78

SPO subject-predicate-object. 1, 12

SRL statistical relational learning. 36, 79, 94, 95

SSE Semantically Smooth Embedding. 80

SSP semantic space projection. 80

STAGG Staged Query Graph Generation. 54

SWARM Semantic Web Association Rule Mining. 67

TKRL Type-embodied Knowledge Representation Learning. 80

Bibliography

[ABM+18] Grigoris Antoniou, Sotiris Batsakis, Raghava Mutharaju, Jeff Z Pan, Guilin Qi, Ilias Tachmazidis, Jacopo Urbani, and Zhangquan Zhou. A survey of large-scale reasoning on the web of data. *The Knowledge Engineering Review*, 33, 2018.

[ABT14] Sören Auer, Volha Bryl, and Sebastian Tramp. *Linked Open Data–Creating Knowledge Out of Interlinked Data: Results of the LOD2 Project*, volume 8661. Springer, 2014.

[AIS93] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 207–216, 1993.

[AMOOV20] Tareq Al-Moslmi, Marc Gallofré Ocaña, Andreas L Opdahl, and Csaba Veres. Named entity extraction for knowledge graphs: A literature overview. *IEEE Access*, 8:32862–32881, 2020.

[AYRW17] Abdalghani Abujabal, Mohamed Yahya, Mirek Riedewald, and Gerhard Weikum. Automated template generation for question answering over knowledge graphs. In *Proceedings of the 26th international conference on world wide web*, pages 1191–1200, 2017.

[BBHG17] Stephen H Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. Hinge-loss markov random fields and probabilistic soft logic. *The Journal of Machine Learning Research*, 18(1):3846–3912, 2017.

[BBL16] Molood Barati, Quan Bai, and Qing Liu. Swarm: an approach for mining semantic association rules from semantic web data. In *Pacific rim international conference on artificial intelligence*, pages 30–43. Springer, 2016.

[BBL+17] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[BBM13] Nina Balcan, Avrim Blum, and Yishay Mansour. Exploiting ontology structures and unlabeled data for learning. 2013.

[BCFL13] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544, 2013.

[BCW14] Antoine Bordes, Sumit Chopra, and Jason Weston. Question answering with subgraph embeddings. *arXiv preprint arXiv:1406.3676*, 2014.

[BDPP19] Piero Andrea Bonatti, Stefan Decker, Axel Polleres, and Valentina Presutti. Knowledge graphs: New directions for knowledge representation on the semantic web (dagstuhl seminar 18371). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[BEP$^+$08] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie 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.

[BFGS19] Luigi Bellomarini, Daniele Fakhoury, Georg Gottlob, and Emanuel Sallinger. Knowledge graphs and enterprise ai: the promise of an enabling technology. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 26–37. IEEE, 2019.

[BFL98] Collin F Baker, Charles J Fillmore, and John B Lowe. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics, 1998.

[BGPS18] Luigi Bellomarini, Georg Gottlob, Andreas Pieris, and Emanuel Sallinger. Swift logic for big data and knowledge graphs. In *International Conference on Current Trends in Theory and Practice of Informatics*, pages 3–16. Springer, 2018.

[BGS18] Luigi Bellomarini, Georg Gottlob, and Emanuel Sallinger. The vadalog system: Datalog-based reasoning for knowledge graphs. *arXiv preprint arXiv:1807.08709*, 2018.

[BGWB14] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259, 2014.

[BHPS11] Piero A Bonatti, Aidan Hogan, Axel Polleres, and Luigi Sauro. Robust and scalable linked data reasoning incorporating provenance and trust annotations. *Journal of Web Semantics*, 9(2):165–201, 2011.

[BL5] Tim Berners-Lee. 5-star open data. *URL: https://5stardata. info*, 5.

[BMG12] Matthias Brocheler, Lilyana Mihalkova, and Lise Getoor. Probabilistic similarity logic. *arXiv preprint arXiv:1203.3469*, 2012.

[BSG18] Luigi Bellomarini, Emanuel Sallinger, and Georg Gottlob. The vadalog system: Datalog-based reasoning for knowledge graphs. *PVLDB*, 11(9):975–987, 2018.

[BSV20] Luigi Bellomarini, Emanuel Sallinger, and Sahar Vahdati. Reasoning in knowledge graphs: An embeddings spotlight. In *Knowledge Graphs and Big Data Processing*, pages 87–101. Springer, Cham, 2020.

[BUCW15] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*, 2015.

[BUGD+13] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013.

[BV84] Catriel Beeri and Moshe Y Vardi. A proof procedure for data dependencies. *Journal of the ACM (JACM)*, 31(4):718–741, 1984.

[C+12] World Wide Web Consortium et al. Owl 2 web ontology language document overview. 2012.

[CBK+10] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka, and Tom M Mitchell. Toward an architecture for never-ending language learning. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[CGT89] Stefano Ceri, Georg Gottlob, and Letizia Tanca. What you always wanted to know about datalog (and never dared to ask). *IEEE transactions on knowledge and data engineering*, 1(1):146–166, 1989.

[CJX19] Xiaojun Chen, Shengbin Jia, and Yang Xiang. A review: Knowledge reasoning over knowledge graph. *Expert Systems with Applications*, page 112948, 2019.

[CLX16] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *AAAI*, volume 16, pages 1145–1152, 2016.

[Coh16] William W Cohen. Tensorlog: A differentiable deductive database. *arXiv preprint arXiv:1605.06523*, 2016.

[CPHS17] Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. Harp: Hierarchical representation learning for networks. *arXiv preprint arXiv:1706.07845*, 2017.

[CPSS17] Melisachew Wudage Chekol, Giuseppe Pirrò, Joerg Schoenfisch, and Heiner Stuckenschmidt. Marrying uncertainty and time in knowledge graphs. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[CWH+19] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *The world wide web conference*, pages 151–161, 2019.

[CWZ19] Yu Chen, Lingfei Wu, and Mohammed J Zaki. Bidirectional attentive memory networks for question answering over knowledge bases. *arXiv preprint arXiv:1903.02188*, 2019.

[DAM+16] Liang Duan, Charu Aggarwal, Shuai Ma, Renjun Hu, and Jinpeng Huai. Scaling up link prediction with ensembles. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 367–376, 2016.

[DBCL18] Mohnish Dubey, Debayan Banerjee, Debanjan Chaudhuri, and Jens Lehmann. Earl: joint entity and relation linking for question answering over knowledge graphs. In *International Semantic Web Conference*, pages 108–126. Springer, 2018.

[DDZ+17] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. Go for a walk and arrive at the answer: Reasoning over paths in knowledge bases using reinforcement learning. *arXiv preprint arXiv:1711.05851*, 2017.

[DGH+14a] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei 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, 2014.

[DGH+14b] Xin Luna Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Kevin Murphy, Shaohua Sun, and Wei Zhang. From data fusion to knowledge fusion. *Proceedings of the VLDB Endowment*, 7(10):881–892, 2014.

[DL16] Li Dong and Mirella Lapata. Language to logical form with neural attention. *arXiv preprint arXiv:1601.01280*, 2016.

[DNBM16] Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. Chains of reasoning over entities, relations, and text using recurrent neural networks. *arXiv preprint arXiv:1607.01426*, 2016.

[Don19] Xin Luna Dong. Building a broad knowledge graph for products. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 25–25. IEEE, 2019.

[DRC$^+$06] AnHai Doan, Raghu Ramakrishnan, Fei Chen, Pedro DeRose, Yoonkyong Lee, Robert McCann, Mayssam Sayyadian, and Warren Shen. Community information management. *IEEE Data Eng. Bull.*, 29(1):64–72, 2006.

[DS15] Eugenio Di Sciascio. A logic-based approach to named-entity disambiguation in the web of data. In *AI* IA 2015 Advances in Artificial Intelligence: XIVth International Conference of the Italian Association for Artificial Intelligence, Ferrara, Italy, September 23-25, 2015, Proceedings*, volume 9336, page 367. Springer, 2015.

[DT01] Luc Dehaspe and Hannu Toivonen. Discovery of relational association rules. In *Relational data mining*, pages 189–212. Springer, 2001.

[DWZX15] Li Dong, Furu Wei, Ming Zhou, and Ke Xu. Question answering over freebase with multi-column convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 260–269, 2015.

[EM13] Dóra Erdos and Pauli Miettinen. Discovering facts with boolean tensor tucker decomposition. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 1569–1572, 2013.

[EW16] Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, 48, 2016.

[FC14] Andre Freitas and Edward Curry. Natural language queries over heterogeneous linked data graphs: A distributional-compositional semantics approach. In *Proceedings of the 19th international conference on Intelligent User Interfaces*, pages 279–288, 2014.

[FdE08] Nicola Fanizzi, Claudia d'Amato, and Floriana Esposito. Statistical learning for inductive query answering on owl ontologies. In *International Semantic Web Conference*, pages 195–212. Springer, 2008.

[FGG$^+$14] Tim Furche, Georg Gottlob, Giovanni Grasso, Xiaonan Guo, Giorgio Orsi, Christian Schallhart, and Cheng Wang. Diadem: thousands of websites to a single database. *Proceedings of the VLDB Endowment*, 7(14):1845–1856, 2014.

[FK15] Johannes Fürnkranz and Tomáš Kliegr. A brief overview of rule learning. In *International symposium on rules and rule markup languages for the semantic web*, pages 54–69. Springer, 2015.

[FQT+20] Bin Fu, Yunqi Qiu, Chengguang Tang, Yang Li, Haiyang Yu, and Jian Sun. A survey on complex question answering over knowledge base: Recent advances and challenges. *arXiv preprint arXiv:2007.13069*, 2020.

[FŞA+20] Dieter Fensel, Umutcan Şimşek, Kevin Angele, Elwin Huaman, Elias Kärle, Oleksandra Panasiuk, Ioan Toma, Jürgen Umbrich, and Alexander Wahler. *Knowledge Graphs*. Springer, 2020.

[FVHA+08] Dieter Fensel, Frank Van Harmelen, Bo Andersson, Paul Brennan, Hamish Cunningham, Emanuele Della Valle, Florian Fischer, Zhisheng Huang, Atanas Kiryakov, Tony Kyung-il Lee, et al. Towards larkc: a platform for web-scale reasoning. In *2008 IEEE International Conference on Semantic Computing*, pages 524–529. IEEE, 2008.

[FZZ+14] Miao Fan, Deli Zhao, Qiang Zhou, Zhiyuan Liu, Fang Zheng, and Edward Y Chang. Distant supervision for relation extraction with matrix completion. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 839–849, 2014.

[Gar15] Matthew Gardner. Reading and reasoning with knowledge graphs, 2015.

[GBS19] Genet Asefa Gesese, Russa Biswas, and Harald Sack. A comprehensive survey of knowledge graph embeddings with literals: Techniques and applications. In *DL4KG@ ESWC*, pages 31–40, 2019.

[GF18] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, 2018.

[GL88] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *ICLP/SLP*, volume 88, pages 1070–1080, 1988.

[GL16] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[GML15] Kelvin Guu, John Miller, and Percy Liang. Traversing knowledge graphs in vector space. *arXiv preprint arXiv:1506.01094*, 2015.

[GP15] Georg Gottlob and Andreas Pieris. Beyond sparql under owl 2 ql entailment regime: Rules to the rescue. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[GTHS13] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd international conference on World Wide Web*, pages 413–422, 2013.

[GTHS15] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M Suchanek. Fast rule mining in ontological knowledge bases with amie

. *The VLDB Journal*, 24(6):707–730, 2015.

[GTKM14] Matt Gardner, Partha Talukdar, Jayant Krishnamurthy, and Tom Mitchell. Incorporating vector space similarity in random walk inference over knowledge bases. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 397–406, 2014.

[GVdB02] Bart Goethals and Jan Van den Bussche. Relational association rules: getting w armer. In *Pattern Detection and Discovery*, pages 125–139. Springer, 2002.

[GWW$^+$15] Shu Guo, Quan Wang, Bin Wang, Lihong Wang, and Li Guo. Semantically smooth knowledge graph embedding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 84–94, 2015.

[GWW$^+$16] Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. Jointly embedding knowledge graphs and logical rules. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 192–202, 2016.

[Háj13] Petr Hájek. *Metamathematics of fuzzy logic*, volume 4. Springer Science & Business Media, 2013.

[HBC$^+$20] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, et al. Knowledge graphs. *arXiv preprint arXiv:2003.02320*, 2020.

[HC17] Xiangnan He and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 355–364, 2017.

[HE16] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *arXiv preprint arXiv:1606.03476*, 2016.

[HGER$^+$12] Keith Henderson, Brian Gallagher, Tina Eliassi-Rad, Hanghang Tong, Sugato Basu, Leman Akoglu, Danai Koutra, Christos Faloutsos, and Lei Li. Rolx: structural role extraction & mining in large graphs. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1231–1239, 2012.

[HRH02] Peter D Hoff, Adrian E Raftery, and Mark S Handcock. Latent space approaches to social network analysis. *Journal of the american Statistical association*, 97(460):1090–1098, 2002.

[HS06] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[HS17] Katsuhiko Hayashi and Masashi Shimbo. On the equivalence of holographic and complex embeddings for link prediction. *arXiv preprint arXiv:1702.05563*, 2017.

[HSBW13] Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61, 2013.

[HSGE+18] Vinh Thinh Ho, Daria Stepanova, Mohamed H Gad-Elrab, Evgeny Kharlamov, and Gerhard Weikum. Rule learning from knowledge graphs guided by embedding models. In *International Semantic Web Conference*, pages 72–90. Springer, 2018.

[HSM+20] Marcel Hildebrandt, Jorge Andres Quintero Serna, Yunpu Ma, Martin Ringsquandl, Mitchell Joblin, and Volker Tresp. Reasoning on knowledge graphs with debate dynamics. *arXiv preprint arXiv:2001.00461*, 2020.

[HWM+17] Konrad Höffner, Sebastian Walter, Edgard Marx, Ricardo Usbeck, Jens Lehmann, and Axel-Cyrille Ngonga Ngomo. Survey on challenges of question answering in the semantic web. *Semantic Web*, 8(6):895–920, 2017.

[HYL17a] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.

[HYL17b] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

[HZL+11] Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics, 2011.

[HZL+17] Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 221–231, 2017.

[HZLL19] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. Knowledge graph embedding based question answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 105–113, 2019.

[IB12] Robert Isele and Christian Bizer. Learning expressive linkage rules using genetic programming. *arXiv preprint arXiv:1208.0291*, 2012.

[JHNT12] Xueyan Jiang, Yi Huang, Maximilian Nickel, and Volker Tresp. Combining information extraction, deductive reasoning and machine learning for relation prediction. In *Extended Semantic Web Conference*, pages 164–178. Springer, 2012.

[JHX$^+$15] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*, pages 687–696, 2015.

[JŁŁ10] Joanna JÓzefowska, Agnieszka Ławrynowicz, and Tomasz Łukaszewski. The role of semantics in mining frequent patterns from knowledge bases in description logics with rules. *Theory and Practice of Logic Programming*, 10(3):251–289, 2010.

[JPC$^+$20] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S Yu. A survey on knowledge graphs: Representation, acquisition and applications. *arXiv preprint arXiv:2002.00388*, 2020.

[Kam81] Hans Kamp. A theory of truth and semantic representation. *Formal semantics-the essential readings*, pages 189–222, 1981.

[Kaz18] SM Kazemi. Bridging weighted rules and graph random walks for statistical relational models. *Computer Science Department, University of British Columbia, Vancouver, BC, Canada*, 2018.

[KBB$^+$12] A Kimmig, SH Bach, M Broecheler, B Huang, and L Getoor. A short introduction to 302 probabilistic soft logic. In *Proceedings of the NIPS Workshop on Probabilistic Program-303 ming*, volume 304, 2012.

[KBV09] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[Kej19] Mayank Kejriwal. *Domain-Specific Knowledge Graph Construction*. Springer, 2019.

[KGJ+19] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Relational representation learning for dynamic (knowledge) graphs: A survey. *arXiv preprint arXiv:1905.11485*, 2019.

[KNP11] Kristian Kersting, Sriraam Natarajan, and David Poole. Statistical relational ai: logic, probability and computation. In *Proceedings of the 11th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'11)*, pages 1–9, 2011.

[KP18] Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. In *Advances in neural information processing systems*, pages 4284–4295, 2018.

[KW16] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[LB10] Jens Lehmann and Lorenz Bühmann. Ore-a tool for repairing and enriching knowledge bases. In *International Semantic Web Conference*, pages 177–193. Springer, 2010.

[LBL+16] Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. *arXiv preprint arXiv:1611.00020*, 2016.

[LC19] Ruiping Li and Xiang Cheng. Divine: A generative adversarial imitation learning framework for knowledge graph reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2642–2651, 2019.

[LdM15] Shuangyan Liu, Mathieu d'Aquin, and Enrico Motta. Towards linked data fact validation through measuring consensus. In *LDQ@ ESWC*, page 21, 2015.

[LGMN12] Jens Lehmann, Daniel Gerber, Mohamed Morsey, and Axel-Cyrille Ngonga Ngomo. Defacto-deep fact validation. In *International semantic web conference*, pages 312–327. Springer, 2012.

[LHCS14] Xinsheng Li, Shengyu Huang, Kasim Selçuk Candan, and Maria Luisa Sapino. Focusing decomposition accuracy by personalizing tensor decomposition (ptd). In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 689–698, 2014.

[LHX+18] Yankai Lin, Xu Han, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. Knowledge representation learning: A quantitative review. *arXiv preprint arXiv:1812.10901*, 2018.

[LIJ$^+$15] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[LJE$^+$16] Quan Liu, Hui Jiang, Andrew Evdokimov, Zhen-Hua Ling, Xiaodan Zhu, Si Wei, and Yu Hu. Probabilistic reasoning via deep learning: Neural association models. *arXiv preprint arXiv:1603.07704*, 2016.

[LLGD$^+$19] Ye Liu, Hui Li, Alberto Garcia-Duran, Mathias Niepert, Daniel Onoro-Rubio, and David S Rosenblum. Mmkg: Multi-modal knowledge graphs. In *European Semantic Web Conference*, pages 459–474. Springer, 2019.

[LLL$^+$15] Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. Modeling relation paths for representation learning of knowledge bases. *arXiv preprint arXiv:1506.00379*, 2015.

[LLS$^+$15] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[LMC11] Ni Lao, Tom Mitchell, and William W Cohen. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 529–539. Association for Computational Linguistics, 2011.

[LMS13] Thomas Lukasiewicz, Maria Vanina Martinez, and Gerardo I Simari. Complexity of inconsistency-tolerant query answering in datalog+/–. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 488–500. Springer, 2013.

[LSX18] Xi Victoria Lin, Richard Socher, and Caiming Xiong. Multi-hop knowledge graph reasoning with reward shaping. *arXiv preprint arXiv:1808.10568*, 2018.

[LTBZ15] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[LV07] John A Lee and Michel Verleysen. *Nonlinear dimensionality reduction*. Springer Science & Business Media, 2007.

[LWY17] Hanxiao Liu, Yuexin Wu, and Yiming Yang. Analogical inference for multi-relational embeddings. *arXiv preprint arXiv:1705.02426*, 2017.

[MBSJ09] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International*

Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2, pages 1003–1011. Association for Computational Linguistics, 2009.

[MGWQ14] Yanfang Ma, Huan Gao, Tianxing Wu, and Guilin Qi. Learning disjointness axioms with association rule mining and its application to inconsistency detection of linked data. In *Chinese Semantic Web and Web Science Conference*, pages 29–41. Springer, 2014.

[Mil98] George A Miller. *WordNet: An electronic lexical database*. MIT press, 1998.

[MMZ09] Peter Mika, Edgar Meij, and Hugo Zaragoza. Investigating the semantic gap through query log analysis. In *International Semantic Web Conference*, pages 441–455. Springer, 2009.

[MQH$^+$19] Jiangtao Ma, Yaqiong Qiao, Guangwu Hu, Yanjun Wang, Chaoqin Zhang, Yongzhong Huang, Arun Kumar Sangaiah, Huaiguang Wu, Hongpo Zhang, and Kai Ren. Elpkg: A high-accuracy link prediction approach for knowledge graph completion. *Symmetry*, 11(9):1096, 2019.

[MRHLA18] Jose L Martinez-Rodriguez, Aidan Hogan, and Ivan Lopez-Arevalo. Information extraction meets the semantic web: a survey. *Semantic Web*, (Preprint):1–81, 2018.

[MYZ13] Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751, 2013.

[NH98] Radford M Neal and Geoffrey E Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998.

[NHNNR17] Markus Nentwig, Michael Hartung, Axel-Cyrille Ngonga Ngomo, and Erhard Rahm. A survey of current link discovery frameworks. *Semantic Web*, 8(3):419–436, 2017.

[NJT14] Maximilian Nickel, Xueyan Jiang, and Volker Tresp. Reducing the rank in relational factorization models by including observable patterns. *Advances in Neural Information Processing Systems*, 27:1179–1187, 2014.

[NMTG15] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1):11–33, 2015.

[NRM15] Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. Compositional vector space models for knowledge base completion. *arXiv preprint arXiv:1504.06662*, 2015.

[NRP15] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. Holographic embeddings of knowledge graphs. *arXiv preprint arXiv:1510.04935*, 2015.

[NT13] Maximilian Nickel and Volker Tresp. Logistic tensor factorization for multi-relational data. *arXiv preprint arXiv:1306.2084*, 2013.

[NTK11] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Icml*, volume 11, pages 809–816, 2011.

[NTW11] Ndapandula Nakashole, Martin Theobald, and Gerhard Weikum. Scalable knowledge harvesting with high precision and high recall. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 227–236, 2011.

[OWW18] Pouya Ghiasnezhad Omran, Kewen Wang, and Zhe Wang. Scalable rule learning via learning representation. In *IJCAI*, pages 2149–2155, 2018.

[PARS14] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[Pau14] Heiko Paulheim. Identifying wrong links between datasets by multi-dimensional outlier detection. In *WoDOOM*, pages 27–38, 2014.

[Pau17] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8(3):489–508, 2017.

[PB14] Heiko Paulheim and Christian Bizer. Improving the quality of linked data using statistical distributions. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 10(2):63–86, 2014.

[PDS08] Hoifung Poon, Pedro M Domingos, and Marc Sumner. A general method for reducing the complexity of relational inference and its application to mcmc. In *AAAI*, volume 8, pages 1075–1080, 2008.

[PKCS17] Bryan Perozzi, Vivek Kulkarni, Haochen Chen, and Steven Skiena. Don't walk, skip! online learning of multi-scale network embeddings. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, pages 258–265, 2017.

[PLGC15] Jay Pujara, Ben London, Lise Getoor, and William W Cohen. Online inference for knowledge graph construction. In *Workshop on statistical relational AI*, 2015.

[PMGC13] Jay Pujara, Hui Miao, Lise Getoor, and William Cohen. Knowledge graph identification. In *International Semantic Web Conference*, pages 542–557. Springer, 2013.

[Pou15] Line Pouchard. Revisiting the data lifecycle with big data curation. 2015.

[PPMH17] Emmanouil Platanios, Hoifung Poon, Tom M Mitchell, and Eric J Horvitz. Estimating accuracy from unlabeled data: A probabilistic logic approach. In *Advances in Neural Information Processing Systems*, pages 4361–4370, 2017.

[PR11] Jeff Pasternack and Dan Roth. Making better informed trust decisions with generalized fact-finding. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[PSNK14] Fabio Parisi, Francesco Strino, Boaz Nadler, and Yuval Kluger. Ranking and combining multiple predictors without labeled data. *Proceedings of the National Academy of Sciences*, 111(4):1253–1258, 2014.

[PSST10] Adam Pease, Geoff Sutcliffe, Nick Siegel, and Steven Trac. Large theory reasoning with sumo at casc. *Ai Communications*, 23(2-3):137–144, 2010.

[PTVS$^+$16] Thomas Pellissier Tanon, Denny Vrandečić, Sebastian Schaffert, Thomas Steiner, and Lydia Pintscher. From freebase to wikidata: The great migration. In *Proceedings of the 25th international conference on world wide web*, pages 1419–1428, 2016.

[Puj16] Jay Pujara. *Probabilistic models for scalable knowledge graph construction.* PhD thesis, 2016.

[QT19] Meng Qu and Jian Tang. Probabilistic logic neural networks for reasoning. In *Advances in Neural Information Processing Systems*, pages 7710–7720, 2019.

[RKH08] Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler. Description logic reasoning with decision diagrams. In *International Semantic Web Conference*, pages 435–450. Springer, 2008.

[RLT$^+$12] Achim Rettinger, Uta Lösch, Volker Tresp, Claudia d'Amato, and Nicola Fanizzi. Mining the semantic web. *Data Mining and Knowledge Discovery*, 24(3):613–662, 2012.

[RP17] Stephen K Reed and Adam Pease. Reasoning from imperfect knowledge. *Cognitive Systems Research*, 41:56–72, 2017.

[RSF17] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 385–394, 2017.

[RSR15] Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. Injecting logical background knowledge into embeddings for relation extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1119–1129, 2015.

[RvEV$^+$16] Marco Rospocher, Marieke van Erp, Piek Vossen, Antske Fokkens, Itziar Aldabe, German Rigau, Aitor Soroa, Thomas Ploeger, and Tessel Bogaard. Building event-centric knowledge graphs from news. *Journal of Web Semantics*, 37:132–151, 2016.

[RYMM13] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, 2013.

[SCH$^+$18] Yelong Shen, Jianshu Chen, Po-Sen Huang, Yuqing Guo, and Jianfeng Gao. M-walk: Learning to walk over graphs using monte carlo tree search. In *Advances in Neural Information Processing Systems*, pages 6786–6797, 2018.

[SCMN13] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934, 2013.

[SDNR07] Warren Shen, AnHai Doan, Jeffrey F Naughton, and Raghu Ramakrishnan. Declarative information extraction using datalog with embedded extraction predicates. In *Proceedings of the 33rd international conference on Very large data bases*, pages 1033–1044. VLDB Endowment, 2007.

[SEWD10] Stefan Schoenmackers, Oren Etzioni, Daniel S Weld, and Jesse Davis. Learning first-order horn clauses from web text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1088–1098. Association for Computational Linguistics, 2010.

[SGEH18] Daria Stepanova, Mohamed H Gad-Elrab, and Vinh Thinh Ho. Rule induction and reasoning over knowledge graphs. In *Reasoning Web International Summer School*, pages 142–172. Springer, 2018.

[SGT$^+$08] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[SHY$^+$11] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu, and Tianyi Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment*, 4(11):992–1003, 2011.

[Sin12] Amit Singhal. Introducing the knowledge graph: things, not strings, may 2012. *URL http://googleblog. blogspot. ie/2012/05/introducing-knowledgegraph-things-not. html*, 2012.

[SKB$^+$18] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018.

[SKW07] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706, 2007.

[SS17] Douglas Summers-Stay. Deductive and analogical reasoning on a semantically embedded knowledge graph. In *International Conference on Artificial General Intelligence*, pages 112–122. Springer, 2017.

[SSM12] Dimitrios-Emmanuel Spanos, Periklis Stavrou, and Nikolas Mitrou. Bringing relational databases into the semantic web: A survey. *Semantic Web*, 3(2):169–209, 2012.

[SSW09] Fabian M Suchanek, Mauro Sozio, and Gerhard Weikum. Sofie: a self-organizing framework for information extraction. In *Proceedings of the 18th international conference on World wide web*, pages 631–640, 2009.

[SW16a] Baoxu Shi and Tim Weninger. Discriminative predicate path mining for fact checking in knowledge graphs. *Knowledge-based systems*, 104:123–133, 2016.

[SW16b] Baoxu Shi and Tim Weninger. Fact checking in heterogeneous information networks. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 101–102, 2016.

[SW17] Baoxu Shi and Tim Weninger. Proje: Embedding projection for knowledge graph completion. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[TC15] Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66, 2015.

[TKS12] Gerald Töpper, Magnus Knuth, and Harald Sack. Dbpedia ontology enrichment for inconsistency detection. In *Proceedings of the 8th International Conference on Semantic Systems*, pages 33–40, 2012.

[TWR$^+$16] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. International Conference on Machine Learning (ICML), 2016.

[TYM18] Chujun Tian, Hongzhi Yu, and Xianghe Meng. Knowledge reasoning based on knowledge graph. In *2018 3rd International Conference on Advances in Materials, Mechatronics and Civil Engineering (ICAMMCE 2018)*. Atlantis Press, 2018.

[VK14] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

[WBDB17] Xander Wilcke, Peter Bloem, and Victor De Boer. The knowledge graph as the default data model for learning on heterogeneous knowledge. *Data Science*, 1(1-2):39–57, 2017.

[WBYU13] Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. Connecting language and knowledge bases with embedding models for relation extraction. *arXiv preprint arXiv:1307.7973*, 2013.

[WC07] Richard C Wang and William W Cohen. Language-independent set expansion of named entities using the web. In *Seventh IEEE international conference on data mining (ICDM 2007)*, pages 342–350. IEEE, 2007.

[WC16] William Yang Wang and William W Cohen. Learning first-order logic embeddings via matrix factorization. In *Ijcai*, pages 2132–2138, 2016.

[WCB14] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

[WCZ16] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234, 2016.

[WGM$^+$14] Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd international conference on World wide web*, pages 515–526, 2014.

[WHC$^+$19] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 950–958, 2019.

[WHC20] Xiang Wang, Xiangnan He, and Tat-Seng Chua. Learning and reasoning on graph for recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 890–893, 2020.

[WHS16] Gerhard Weikum, Johannes Hoffart, and Fabian Suchanek. Ten years of knowledge harvesting: Lessons and challenges. 2016.

[Wil92] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[WL15] Zhichun Wang and Juanzi Li. Rdf2rules: Learning rules from rdf knowledge bases by mining frequent predicate cycles. *arXiv preprint arXiv:1512.07734*, 2015.

[WL18] Yifei Wang and Jie Luo. An incremental reasoning algorithm for large scale knowledge graph. In *International Conference on Knowledge Science, Engineering and Management*, pages 503–513. Springer, 2018.

[WLWZ12] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q Zhu. Probase: A probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 481–492, 2012.

[WLX16] Yuze Wei, Jie Luo, and Huiyuan Xie. Kgrl: an owl2 rl reasoning system for large scale knowledge graph. In *2016 12th International Conference on Semantics, Knowledge and Grids (SKG)*, pages 83–89. IEEE, 2016.

[WMWG17] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

[WP14] Dominik Wienand and Heiko Paulheim. Detecting incorrect numerical data in dbpedia. In *European Semantic Web Conference*, pages 504–518. Springer, 2014.

[WT10] Gerhard Weikum and Martin Theobald. From information to knowledge: harvesting entities and relationships from web sources. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 65–76, 2010.

[WWG15] Quan Wang, Bin Wang, and Li Guo. Knowledge base completion using embeddings and rules. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[WWX$^+$19] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. Explainable reasoning over knowledge graphs for recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5329–5336, 2019.

[WZF19] Peiyun Wu, Xiaowang Zhang, and Zhiyong Feng. A survey of question answering over knowledge base. In *China Conference on Knowledge Graph and Semantic Computing*, pages 86–97. Springer, 2019.

[WZFC14a] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1591–1601, 2014.

[WZFC14b] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Aaai*, volume 14, pages 1112–1119. Citeseer, 2014.

[WZX$^+$19] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. Knowledge graph convolutional networks for recommender systems. In *The world wide web conference*, pages 3307–3313, 2019.

[WZZ$^+$19] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 968–977, 2019.

[XDCC19] Guohui Xiao, Linfang Ding, Benjamin Cogrel, and Diego Calvanese. Virtual knowledge graphs: An overview of systems and use cases. *Data Intelligence*, 1(3):201–223, 2019.

[XFM$^+$19] Yikun Xian, Zuohui Fu, S Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. Reinforcement knowledge graph reasoning for explainable recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 285–294, 2019.

[XHMZ17] Han Xiao, Minlie Huang, Lian Meng, and Xiaoyan Zhu. Ssp: semantic space projection for knowledge graph embedding with text descriptions. In *Thirty-First AAAI conference on artificial intelligence*, 2017.

[XHW17] Wenhan Xiong, Thien Hoang, and William Yang Wang. Deeppath: A reinforcement learning method for knowledge graph reasoning. *arXiv preprint arXiv:1707.06690*, 2017.

[XLJ$^+$16] Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. Representation learning of knowledge graphs with entity descriptions. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[XLLS16] Ruobing Xie, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. Image-embodied knowledge representation learning. *arXiv preprint arXiv:1609.07028*, 2016.

[XLS16] Ruobing Xie, Zhiyuan Liu, and Maosong Sun. Representation learning of knowledge graphs with hierarchical types. In *IJCAI*, pages 2965–2971, 2016.

[YC03] Jaeyoung Yang and Joongmin Choi. Knowledge-based wrapper induction for intelligent web information extraction. In *Web Intelligence*, pages 153–172. Springer, 2003.

[YCHG15] Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. Semantic parsing via staged query graph generation: Question answering with knowledge base. 2015.

[YCWT18] Jheng-Hong Yang, Chih-Ming Chen, Chuan-Ju Wang, and Ming-Feng Tsai. Hop-rec: high-order proximity for implicit recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 140–144, 2018.

[YDS12] Dong Yu, Li Deng, and Frank Seide. Large vocabulary speech recognition using deep tensor neural networks. In *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[YRS$^+$14] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. Personalized entity recommendation: A heterogeneous information network approach. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 283–292, 2014.

[YTZ$^+$19] Shihui Yang, Jidong Tian, Honglun Zhang, Junchi Yan, Hao He, and Yaohui Jin. Transms: Knowledge graph embedding for complex relations by multidirectional semantics. In *IJCAI*, pages 1935–1942, 2019.

[YVD14] Xuchen Yao and Benjamin Van Durme. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 956–966, 2014.

[YWC$^+$18] Jihong Yan, Chengyu Wang, Wenliang Cheng, Ming Gao, and Aoying Zhou. A retrospective of knowledge graphs. *Frontiers of Computer Science*, 12(1):55–74, 2018.

[YYC17] Fan Yang, Zhilin Yang, and William W Cohen. Differentiable learning of logical rules for knowledge base reasoning. In *Advances in Neural Information Processing Systems*, pages 2319–2328, 2017.

[YYH$^+$14] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.

[ZAd⁺15] Fouad Zablith, Grigoris Antoniou, Mathieu d'Aquin, Giorgos Flouris, Haridimos Kondylakis, and Enrico Motta. Ontology evolution: a process-centric survey. *The knowledge engineering review*, 30(1):45–75, 2015.

[ZCGZ19] Yunan Zhang, Xiang Cheng, Heting Gao, and Chengxiang Zhai. Cooperative reasoning on knowledge graph and corpus: A multi-agentreinforcement learning approach. *arXiv preprint arXiv:1912.02206*, 2019.

[ZCY⁺20] Yuyu Zhang, Xinshi Chen, Yuan Yang, Arun Ramamurthy, Bo Li, Yuan Qi, and Le Song. Efficient probabilistic logic reasoning with graph neural networks. *arXiv preprint arXiv:2001.11850*, 2020.

[ZD07] Patrick Ziegler and Klaus R Dittrich. Data integration—problems, approaches, and perspectives. In *Conceptual modelling in information systems engineering*, pages 39–58. Springer, 2007.

[ZDK⁺18] Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander Smola, and Le Song. Variational reasoning for question answering with knowledge graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[ZGH11] Amal Zouaq, Dragan Gasevic, and Marek Hatala. Towards open ontology learning and filtering. *Information Systems*, 36(7):1064–1081, 2011.

[ZL17] Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14):i190–i198, 2017.

[ZPP14] Qiang Zeng, Jignesh M Patel, and David Page. Quickfoil: Scalable inductive logic programming. *Proceedings of the VLDB Endowment*, 8(3):197–208, 2014.

[ZPW⁺19] Wen Zhang, Bibek Paudel, Liang Wang, Jiaoyan Chen, Hai Zhu, Wei Zhang, Abraham Bernstein, and Huajun Chen. Iteratively learning embeddings and rules for knowledge graph reasoning. In *The World Wide Web Conference*, pages 2366–2377, 2019.

[ZRM⁺16] Amrapali Zaveri, Anisa Rula, Andrea Maurino, Ricardo Pietrobon, Jens Lehmann, and Soeren Auer. Quality assessment for linked data: A survey. *Semantic Web*, 7(1):63–93, 2016.

[ZXLS17] Hao Zhu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. Iterative entity alignment via joint knowledge embeddings. In *IJCAI*, pages 4258–4264, 2017.

[ZYL⁺16] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 353–362, 2016.

[ZZL+19] Jia Zhang, Hui Zhang, Bo Li, Chunming Yang, and Xujian Zhao. A probabilistic soft logic reasoning model with automatic rule learning. In *CCF Conference on Big Data*, pages 33–45. Springer, 2019.

[ZZQ+18] Zhao Zhang, Fuzhen Zhuang, Meng Qu, Fen Lin, and Qing He. Knowledge graph embedding with hierarchical relation structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3198–3207, 2018.