On Data Engineering and Knowledge Graphs

✦✦✦

A Context-Aware Proposal for Web-Scale Knowledge Graph Completion

Agustín Borrego Díaz

University of Seville, Spain

Doctoral Dissertation

Supervised by Dr. Inma Hernández and Dr. David Ruiz

May, 2023

First published in May 2023 by
The DEAL Research Group
E.T.S. Ingeniería Informática
Av. de la Reina Mercedes s/n
41012 Seville, Spain

https://deal.us.es/team/aborrego/
borrego@us.es

Support: The author's doctoral studies have been supported by the Spanish FPU scholarship program (FPU18/00363). The work and results presented in this dissertation have been supported by the Spanish and Andalusian R&D programs (grants TIN2016-75394-R, PID2019-105471RB-I00, P18-RT-1060, US-1380565).

"El hogar no está en ningún lugar, sino en la gente. Si vuelves a casa después de que todo el mundo se haya ido, sólo verás lo que ya no existe."

— *La suerte del bufón*, Robin Hobb

A Matilde, Javier y Caleb,
Manolo y Loli,
y Ren.

Contents

List of Figures

List of Tables

Acknowledgments

"There is nothing which I can esteem more highly than being and appearing grateful. For this one virtue is not only the greatest, but is also the parent of all the other virtues."

— Marcus Tullius Cicero

Even though this is the first section of this dissertation, it was the last one to be written. Perhaps because it may be the most important one. I would like to begin by thanking my advisors, David and Inma, for having given me the opportunity to join their group many years ago, and for their insightful guidance and advice throughout all this process. More importantly, I want to thank them for leading by their example of academical and personal honesty and integrity, and for the genuine care with which they treat everyone under their supervision. In particular, I want to thank David for the trust he has placed in me and my capabilities in a multitude of occasions; and Inma, for her constant availability and support, even when she barely had the time for it (and for her many excellent restaurant recommendations!). I feel very lucky for having had the opportunity and the pleasure to work with both of them.

I would also like to thank my colleagues, teammates, and friends from Geozoco: Miguel and Fernando (and more recently, also Paula and Pepe) for the many good times and adventures that we have had together. They were the ones that made going to work not feel like going to work at all. Although we part ways for now, I leave with the certainty that those who are already pursuing their PhD, and those who should choose to do so in the future, are in very good hands. Regardless of the path they choose, I am convinced they will succeed in it.

Furthermore, I want to extend my thanks to Dr. Francesco Osborne for facilitating the research stays that were carried out throughout the course of my PhD, as well as

to the rest of his colleagues: Dr. Danilo Dessì, Dr. Diego Reforgiato Recupero, and Dr. Davide Buscaldi, for consistently finding the time to keep furthering our collaboration even to this day, for sharing their expertise in scientific Knowledge Graphs and many other fields with us, and for all the work that we have done together.

I want to also acknowledge and thank the public education and scholarships system that, even with its many flaws and imperfections, has allowed me, and many others like me, to receive a high-quality education virtually free of charge, and to achieve my highest potential when I perhaps could not have afforded it. Likewise, I would like to thank the many teachers that I have had throughout the years, both in and out of university, without whom I could not have reached this point. During my PhD scholarship, I have had the privilege of teaching in the same university halls where I was a student not too long before. I firmly believe that the best teachers are those who know there is always something new they can learn, and thus I would also like to thank the students that I have had during these years, from whom I have learned a lot. I can only hope that I was the teacher they deserved.

These final lines are dedicated to the most important people in my life, those who make me feel at home no matter where I am. To my grandparents Loli and Manolo, who did everything they could, and sometimes even more, to make sure we always had whatever we needed. To my parents Javier and Matilde, and my brother Caleb, for their absolute, unconditional, and unwavering love, support and understanding. And to my partner Ren, for an infinity of reasons that could not, and need not, be listed on these pages.

Agradecimientos

"No hay nada que pueda tener en más alta estima que ser y mostrarse agradecido. Porque esta virtud no sólo es la más grande, sino que también es la madre de todas las demás virtudes."

— Marco Tulio Cicerón

A pesar de que ésta es una de las secciones que encabeza el presente trabajo, fue la última en ser escrita. Tal vez porque, en cierto modo, es la más importante de todas. Me gustaría comenzar estas líneas dando las gracias a mis directores de tesis, David e Inma, por haberme brindado la oportunidad de unirme a su grupo de investigación hace ya algunos años, y por su guía y apoyo constante durante todo este proceso. Sobre todo, me gustaría reconocer públicamente su ejemplo de integridad académica y personal, y el cariño sincero con el que tratan a cualquiera bajo su supervisión. Particularmente, quiero agradecer a David la confianza que ha puesto en mí y en mis capacidades en una gran multitud de ocasiones, que espero haber correspondido; y a Inma, su apoyo y disponibilidad constantes, incluso en momentos en los que a duras penas tenía el tiempo para ello (y por sus muchas y excelentes recomendaciones de restaurantes para nuestras comidas de grupo). Ha sido un auténtico placer y un privilegio trabajar con ellos.

Me gustaría también agradecer a mis colegas —en todas sus acepciones— de Geozoco, Miguel y Fernando (y también a Paula y Pepe, que se han unido más recientemente) por todos los momentos y risas que hemos compartido juntos. Quizá su compañía ha hecho que trabaje algo menos de lo que debería, pero es el precio a pagar por tener tan buenos compañeros. Aunque tomemos caminos separados, estoy convencido de que encontrarán el éxito (y, más importante, la felicidad) en cualquier rumbo que decidan tomar. Tengo también la tranquilidad de saber que los que decidan iniciar su doctorado, y los que ya lo han hecho, no podrían estar en mejores manos.

Quiero también agradecer al Dr. Francesco Osborne por gestionar y facilitar las estancias de investigación que he realizado a lo largo de mis estudios de doctorado, y al resto de sus compañeros: Dr. Danilo Dessì, Dr. Diego Reforgiato Recupero, y Dr. Davide Buscaldi, por su colaboración continuada que aún se extiende al día de hoy, por compartir su gran dominio de los grafos de conocimiento científicos y muchos otros temas con nosotros, y por todo el gran trabajo que hemos hecho juntos.

Debo también agradecer al sistema público de educación y becas que, con sus muchos fallos e imperfecciones, me ha permitido tanto a mí como a muchos de mis compañeros acceder a una educación de alta calidad y alcanzar nuestro máximo potencial sin prácticamente coste, con independencia de nuestras posibilidades económicas. Me gustaría también agradecer a los muchos profesores que he tenido, tanto en la universidad como fuera de ella, ya que sin ellos nunca habría podido llegar hasta aquí. En el transcurso de mi beca de doctorado, he tenido el privilegio de enseñar en las mismas aulas donde yo mismo fui un estudiante no muchos años atrás. Creo firmemente que los mejores profesores son los que saben que siempre tienen algo nuevo que aprender de todo el mundo, por lo que quiero también agradecer a los estudiantes que han pasado por mis clases en estos años, de los que he aprendido mucho. Espero haber sido el profesor que se merecen.

Estas últimas líneas que escribo están reservadas, como no podría ser de otro modo, para las personas más importantes en mi vida: las que me hacen sentir en casa independientemente de dónde esté. A mis abuelos Manolo y Loli, que siempre hicieron lo imposible y más para que tuviéramos todo lo que necesitábamos. A mis padres Javier y Matilde, y mi hermano Caleb, por su amor y apoyo constante, inacabable e incondicional. Y a Ren, por una infinidad de motivos que no podrían, y no necesitan, ser enumerados aquí.

Abstract

"Brevity is the soul of wit."

— *Hamlet*, William Shakespeare

Nowadays, Knowledge Graphs are a widely used means to store structured information for a variety of different domains and applications. However, due to the fact that they are usually constructed using automated information extraction techniques, they are often incomplete, either because these techniques failed to extract the relevant information, or because it was not present altogether in the original sources.

The problem that we address in this dissertation is how to find this missing knowledge and complete Knowledge Graphs in an automatic manner. In the literature, there are already many proposals to perform this task. However, they have important drawbacks, namely: they rely on embedded representations, which are computationally expensive to generate and demand frequent regenerations, they require human intervention or human-provided data, they rely on external sources of information, they cannot produce new knowledge on their own, or they do not scale properly to very large Knowledge Graphs.

In this dissertation, we present a new automated proposal for completing Knowledge Graphs that does not suffer from any of the previous drawbacks. Our contribution is threefold: CHAI, a technique for automatically generating tractable sets of candidate triples; CAFE, a high-accuracy triple classification proposal; and SciCheck, a technique specifically tailored for completing scientific Knowledge Graphs. Our theoretical and practical validation suggests that our proposal is very efficient and effective in practice, and that it is able to successfully complete Knowledge Graphs of varying natures.

Resumen

"Lo bueno, si breve, dos veces bueno."

— Baltasar Gracián

Hoy en día, los grafos de conocimiento son una herramienta ampliamente usada para almacenar y representar información estructurada para una gran variedad de dominios y aplicaciones prácticas. Sin embargo, debido a que generalmente son construidos usando técnicas de extracción automática de información, éstos suelen estar incompletos. Esto se debe a que las citadas técnicas pueden no extraer satisfactoriamente la información deseada, o a que la fuente original no contenía suficiente información.

El problema tratado en esta tesis doctoral es cómo encontrar este conocimiento que falta y completar un grafo de conocimiento de manera automática. En la bibliografía existen numerosas propuestas para lograr este objetivo, pero tienen importantes inconvenientes, concretamente: necesitan utilizar *embeddings*, que son computacionalmente costosos de obtener y requieren ser regenerados frecuentemente, necesitan intervención humana o datos generados manualmente, tienen una dependencia fuerte con fuentes externas de información, no tienen ningún modo para generar nuevo conocimiento por ellas mismas, o no son aplicables a grafos de conocimiento muy grandes.

En esta tesis presentamos una nueva propuesta automatizada para completar grafos de conocimiento que no sufre de los problemas anteriores. Nuestra contribución tiene tres elementos principales: CHAI, una técnica para generar automáticamente conjuntos manejables de tripletas candidatas; CAFE, una propuesta de clasificación de tripletas de alta precisión; y SciCheck, una técnica especialmente diseñada para completar grafos de conocimiento científicos. Nuestra validación, tanto teórica como basada en una aplicación práctica, sugiere que nuestra propuesta es muy eficiente y efectiva en casos de uso reales, y que es capaz de completar satisfactoriamente grafos de conocimiento de todo tipo.

Part I

Preface

Chapter 1

Introduction

"Ludwig Boltzman, who spent much of his life studying statistical mechanics, died in 1906, by his own hand. Paul Ehrenfest, carrying on the work, died similarly in 1933. Now it is our turn to study statistical mechanics."

— *States of matter*, David L. Goodstein

In this dissertation, we report on our work to devise a proposal to complete large Knowledge Graphs. This chapter introduces the reader to the necessary research context behind it, and it is structured as follows: in Section 1.1, we first introduce the context of our dissertation; in Section 1.2, we present the hypothesis that has motivated it, as well as our thesis; in Section 1.3, we summarize the contributions that we make in this dissertation; in Section 1.4, we detail the collaborations with other researchers that we have conducted during the development of our work; finally, in Section 1.5, we describe the structure of the rest of the dissertation.

1.1 Research context

Today, more information is being generated, stored and published than ever before. Among the many means for storing information, Knowledge Graphs (KGs) have risen to popularity due to their ability to hold large amounts of structured knowledge, either about a specific domain or across a range of domains. In a KG, information is represented as triples, which are links between two entities through a certain relation. These links are what confers KGs their graph-like structure, and make them a very appealing option to hold complex, interconnected information in an efficient yet expressive way.

Knowledge Graphs can be constructed in a variety of ways, although the most popular ones generally work by automatically extracting information from non-structured [40, 91] or semi-structured [50, 79] sources, processing them to add semantic meaning or disambiguate the meaning of some concepts [9, 96], and then making them available online through a web interface, an API, or both [21], although private and commercial use is also common [40, 132].

Those automated means of construction allow KGs to contain very large amounts of information, however, they introduce a number of issues. First, two Knowledge Graphs that are constructed separately are generally not mutually compatible, either because they represent the same concepts in different ways [114], or just because they have a different language or modality [53, 66]. Second, the information they are built upon may be incorrectly interpreted or simply factually wrong [107], leading to incorrect facts being present in a KG. Finally, fully automated KG construction methods are prone to missing information present in the original source [16]. Additionally, no single source of information, or combination of them, explicitly contains all pieces of information about a single domain. For these reasons, Knowledge Graphs are fundamentally incomplete. The process of finding facts missing from a KG and using them to complete it is a task known as Knowledge Graph completion, and it is the focus of this dissertation.

In the literature, KG completion has been approached in a number of different ways, however, most existing proposals can be classified in one of three ways [128]. Some authors propose using logical rules to find missing facts, which analyze the existing information patterns in a Knowledge Graph to deduce rules that can then be used to generate further knowledge [43, 44, 69, 77, 90, 115, 121, 154, 156, 167]. Other authors have proposed using a variety of embedded knowledge representations, which provide a more compact way to represent the information in a KG and can be leveraged to generate more knowledge [17, 39, 46, 67, 82, 98, 140, 145, 155, 166]. A third line of techniques focuses on using information that can be found in the paths that exist between entities in a KG, and in the contexts of the entities themselves [12, 47, 48, 57, 70, 78, 80, 93, 144, 151]. In our proposal, we extend this third line of work.

Regardless of how they are classified, most existing KG completion proposals have a very limited ability to actually generate new knowledge: they either rely on a set of candidate facts being provided to them for evaluation, or require a pre-made combination of one entity and one relation to be specified [128]. For this reason, although they may yield satisfactory metrics in a theoretical evaluation, they are of little practical use. A separate challenge in KG completion is to efficiently materialize a set of candidate triples of a reasonable size, that aims to minimize the amount of clearly incorrect triples it contains while retaining as many correct ones as possible. A much smaller body of work can be found in this regard [103, 130, 169], even though it is a key step for completing a Knowledge Graph.

1.2 Research rationale

In this section, we present the hypothesis that has motivated our research work in the context of Knowledge Graph completion, and state our thesis, which we prove in the rest of the dissertation.

1.2.1 Hypothesis

Nowadays, there is an increasing interest of individuals, organizations and companies in using Knowledge Graphs to represent information about a certain domain and to exploit said KGs to provide services. In the present day, practical applications such as question answering [132], product recommendations [104, 170], data retrieval [157] and meta-research [5, 37] already rely on Knowledge Graphs. However, these applications need the data present in KGs to be as complete as possible, which makes it necessary to expand them after their creation.

According to the previous argumentation, we conclude that our hypothesis is the following:

> *Knowledge Graphs provide value to individuals and companies alike by supporting applications that are widely used today, and their usefulness is likely to keep growing. Due to their incompleteness, there is a need to refine them and find the information that they are missing in order to expand them.*

1.2.2 Thesis

There already exist a number of techniques that can be applied to perform Knowledge Graph completion, e.g., [12, 17, 39, 43, 44, 46, 47, 48, 57, 67, 69, 70, 77, 78, 80, 82, 90, 93, 98, 115, 121, 140, 144, 145, 151, 154, 155, 156, 166, 167]. Unfortunately, they do not fulfill a number of requirements for their application to large Knowledge Graphs. Those that rely on extracting and applying rules suffer from a particularly poor scalability, which has been acknowledged in the literature [44]. A large number of proposals use latent

representations such as tensors, or entity or relation embeddings. While they can be effective, they are hindered by the fact that these representations must be obtained beforehand, which is a very computationally intensive task, and they must be updated entirely when a new entity or relation is introduced to the graph, which is a frequent event. Furthermore, many other proposals for KG completion need to access external or manually-provided information, making them not fully automatic or self-contained.

In light of the previous reasoning, we conclude that our thesis is as follows:

> *In the context of Knowledge Graph completion, it is possible to overcome the problems of existing proposals and develop a new one to automatically complete the missing information in a large KG, in a way that does not rely on external information or alternative representations, but that still achieves a high effectiveness.*

1.3 Summary of contributions

To prove our thesis, we have devised a number of proposals that address different aspects of the problem of Knowledge Graph completion, namely:

- **CHAI**, a technique that is able to quickly filter out a large amount of candidate triples, leaving only the most promising ones. To do this, CHAI generates a number of rules that determine which triples have a higher chance of representing correct facts and deserve further evaluation, and which ones are most likely wrong and should be discarded immediately. Each of these rules is generated in an iterative manner, evaluating which continuation for it is the most promising. After a rule has been expanded, it is assessed again to determine whether it needs to be expanded further or not.

  Regarding this contribution, the article that describes it [19] was accepted and presented at K-CAP 2019.

- **CAFE**, a technique that is able to learn what constitutes a correct and a wrong fact, and thus can select only the correct triples provided by CHAI, which can then be added to a KG, completing it. CAFE relies upon a set of neighborhood-aware features that are able to accurately characterize the neighborhoods of a pair of entities. It then uses these features to transform all triples it is provided into feature vectors, and trains a set of neural classification models to learn to distinguish between correct and incorrect knowledge.

  Regarding this contribution, the article that describes it [20] was published in the EAAI journal.

- **SciCheck**, a proposal that is specifically designed to complete scientific Knowledge Graphs. Due to the particularities of these KGs, regular KG completion proposals do not provide satisfactory results. SciCheck overcomes these issues by considering the semantic similarities of the research concepts in a KG and by leveraging the rich ontologies that such KGs usually have.

  Regarding this contribution, the article that describes it [22] was published in the IEEE Access journal.

1.4 Collaborations

Two research visits were carried out, from September 3 to December 4, 2021; and from September 1 to December 1, 2022; to the research group of Dr. Francesco Osborne at the Knowledge Media Institute (KMI), which is a research institution within The Open University (United Kingdom). Dr. Osborne and his colleagues are experts in the creation, refinement, and exploitation of Knowledge Graphs [5, 34, 35, 36, 37, 109, 122, 123, 124]. During these stays, we applied our proposal to some of the scientific Knowledge Graphs they had created in order to complete them and find missing research knowledge, resulting in an improved version of our technique specifically tailored for scientific Knowledge Graphs, as well as in a more complete version of one of their scientific KGs. Additionally, we carried out some work to apply KG completion techniques to detect possible future research hypotheses, based on the scientific knowledge available today.

1.5 Structure of this dissertation

This dissertation is structured as follows:

Part I: Preface. It encompasses this introduction and Chapter 2, in which we provide the motivation for our research work and we conclude that the existing proposals for automatically completing Knowledge Graphs have a number of drawbacks.

Part II: Background Information. It provides information about Knowledge Graphs and the different proposals to complete them that can be found in the literature. In Chapter 3, we introduce the concept of Knowledge Graph, their main characteristics and the current open challenges regarding them. In Chapter 4, we present different proposals to complete KGs that rely on latent triple representations. In Chapter 5, we provide an overview of the KG completion proposals that use information found in paths between entities and, in Chapter 6, we summarize the existing proposals to perform this task using logical rules.

Part III: Our Proposal. It reports on the main contribution of this dissertation. In Chapter 7, we define a common framework of definitions and concepts. In Chapter 8, we present our proposal for automated candidate triple filtering, which is able to quickly discard a large number of incorrect triples. In Chapter 9, we describe our proposal for triple classification, which can assess the correctness of a triple with a high efficacy. Finally, in Chapter 10, we present our technique for completing scientific Knowledge Graphs, along with a practical use case.

Part IV: Final Remarks. It contains Chapter 11, which concludes this dissertation and presents some possible future research directions.

Chapter 2

Motivation

"The worthwhile problems are the ones you can really solve or help solve, the ones you can really contribute something to. No problem is too small or too trivial if we can really do something about it."

— Richard Feynman

Despite Knowledge Graph completion being a very active research topic, the current proposals for carrying it out still have some drawbacks that hinder their applicability in practice, and which should be solved. Our goal in this chapter is to present the problems that arise in practice when completing Knowledge Graphs, and to motivate the need for a new proposal. This chapter is organized as follows: Section 2.1 introduces it and provides the necessary background knowledge, Section 2.2 presents the problems of KG completion in detail, Section 2.3 analyzes the current approaches and their main problems, Section 2.4 explains how none of the existing proposals solves all practical problems at a time, Section 2.5 introduces our contributions and compares them with the existing proposals in the literature; finally, Section 2.6 summarizes the chapter.

2.1 Introduction

Nowadays, there is an increasing interest of individuals, organizations and companies in Knowledge Graphs, in order to organize, store and publish their data. This, in turn, can support many practical applications in a variety of domains, such as commerce [104, 170], education [3, 27], research [34, 37, 153], or healthcare [51, 157, 171], to cite a few.

Ideally, these Knowledge Graphs should be as complete as possible, to make sure that they include all pieces of knowledge that may be relevant to the organization that manages it or the users they support. However, due to the way in which they are constructed, it is well-known that this is not the case [62, 107, 109, 128]. Those that are automatically built from external knowledge sources rely on the completeness of the original source and the capabilities of automated information extractors, which are far from perfect [16, 91]. Additionally, KGs that are built manually, either by their creators or by a crowdsourced process, tend to be much smaller in size [89]. Due to these reasons, there exists a large number of incomplete KGs under active use today, and a need to complete them [75, 101, 110, 131, 132].

In the literature, there are different proposals to address the problem of completing Knowledge Graphs, e.g., [12, 39, 43, 46, 47, 57, 69, 70, 77, 80, 82, 90, 93, 121, 140, 144, 145, 151, 154, 156, 166, 167]. Unfortunately, these proposals have a number of drawbacks that hinder their applicability in practice. Consequently, it is still necessary to research on the field of Knowledge Graph completion, which is our purpose in this dissertation.

2.2 Problems

Completing Knowledge Graphs is not a trivial task and, if not performed correctly, it may reduce the quality of the knowledge contained in them. In this section, we present the problems that must be addressed by proposals that perform this task in order to be useful in practice. These problems are as follows:

(P1) To rely on embedded representations of entities and/or relations: Many of the current proposals rely on generating, or being provided with, embedded representations of the entities and/or relations in a Knowledge Graph. These embedded representations are more compact versions of the elements they represent, usually as a vector that encodes the position of an entity or relation in an N-dimensional space. Embedded representations can be useful because they can capture meaningful semantic similarities between different elements, however, their use hinders the practical application of a proposal. Those proposals that require that embedded representations be provided to them are no longer stand-alone, they instead have a strong dependence with the tools that

generates the embeddings and the quality of them. Even those that generate them themselves suffer from another issue: the need to completely remake them —incurring in a very high computational cost— whenever a new entity or relation is added to the KG, which is an event that tends to happen frequently [40, 91].

(P2) To depend on external sources of information: An external source of information can be a structured or semi-structured repository of information, an information retrieval or query system, another Knowledge Graph or, in general any means for automatically accessing information besides the Knowledge Graph that is to be completed. Depending on these external sources introduces a single point of failure in the KG completion process that is not admissible in many practical and commercial applications. Furthermore, depending on the nature of the information source, its contents or means of access may change without any previous warning, rendering the KG completion proposal ineffective or, at least, requiring extensive maintenance.

(P3) To need user-provided data or supervision: Knowledge Graphs can store large amounts of information; some of the most well-known KGs used nowadays contain millions of triples, and some of them can reach even higher orders of magnitude [132]. For this reason, it is not reasonable to rely on any sort of human-provided information in order to complete them. The required volume of human input for a standard Knowledge Graph would be unattainable for just a few human experts, and using a crowdsourcing process would greatly decrease the quality of the information that is fed to the KG completion process.

(P4) To not have any means to automatically generate new knowledge: Many of the existing KG completion proposals in the literature are able to determine whether a triple is correct or not, which is undoubtedly an important step, but they do not have any mechanisms to autonomously generate new knowledge. Instead, they rely on a set of possible facts being passed on to them for evaluation, but they do not specify how this set of facts should be created. Other proposals have a limited way to do this, by suggesting which entities are more likely to appear in a triple along with another given entity and relation. Although it is an improvement, it is still not practical to assess all possible pairs of entities and relations in a KG.

(P5) To not be applicable to large Knowledge Graphs: As mentioned previously, most Knowledge Graphs contain large volumes of information. Even if a proposal relies entirely on information present in the graph, some of them use approaches that are known not to be scalable enough to be applied to some of the most popular Knowledge Graphs available nowadays.

2.3 Analysis of current solutions

There already exist a number of proposals for completing Knowledge Graphs in the literature. In Table 2.1, we summarize them and the problems they suffer from, and in the following, we discuss them in more detail:

Bordes et al. [17] devised a proposal that learns embedded representations of the entities in a Knowledge Graph, in order to place them in an N-dimensional space while transforming semantical similarities into physical closeness in said space. In this proposal, the correctness of a triple can be checked by evaluating the relative positions of its two entities in the embedded space.

Galárraga et al. [43] proposed a rule extractor that is able to capture common patterns in a Knowledge Graph using Inductive Logic Programming (ILP), and express them using first-order rules. Once these rules have been mined, they can be applied to materialize new knowledge in the KG.

Gardner and Mitchell [47] introduced a technique that defines a series of features to characterize the path between two entities, and then analyzes a large number of said paths to learn to identify a possible direct connection between the entities. However, it requires the manual introduction of an "Alias" relation, which indicates that two entities in a KG refer to the same concept in the real world.

Guo et al. [55] presented a proposal that combines the use of entity embeddings and logical rules. It provides a shared framework in which rules and embeddings can directly interact with each other. This is done by representing the triples in a KG mathematically, and defining a series of operators. The semantic information present in the entity embeddings helps expand the predictive capabilities of the rules that this proposal produces.

Jiang et al. [69] proposed another approach that uses ILP to find and exploit rules in a Knowledge Graph, more particularly, by analyzing the intervals of validity of the facts contained within it and reasoning when other related facts will start or stop being valid. This requires the introduction of temporal annotations in the KG which, generally, must be manually provided.

Proposal	P1	P2	P3	P4	P5
Bordes et al. [17]	X	✓	✓	X	✓
Galárraga et al. [43]	✓	✓	✓	✓	X
Gardner and Mitchell [47]	✓	✓	X	X	✓
Guo et al. [55]	X	✓	X	X	✓
Jiang et al. [69]	✓	X	✓	✓	X
Kazemi and Poole [72]	X	✓	✓	X	✓
Lao et al. [78]	✓	✓	✓	X	✓
Lin et al. [82]	X	✓	✓	X	✓
Nickel et al. [98]	X	✓	✓	✓	X
Shi and Weninger [130]	X	X	✓	✓	✓
Trouillon et al. [145]	X	✓	✓	✓	X
Wang et al. [155]	X	✓	✓	X	✓

P1 = To rely on embedded representations of entities and/or relations; P2 = To depend on external sources of information; P3 = To need user-provided data or supervision; P4 = To not have any means to automatically generate new knowledge; P5 = To not be applicable to large Knowledge Graphs.
A ✓ means that the proposal is free from a problem, while X means that it is present.

Table 2.1: Comparison of current proposals for KG completion

Kazemi and Poole [72] also leverage entity and relation embeddings, and propose adding an extra inverse relation for every one that is already present in a KG. This allows their proposal to reach a higher degree of expressivity, while its simple embedding model allows it to be applied to large KGs.

Lao et al. [78] presented a technique that uses random walks to traverse the space in the KG between the two entities of a triple. By analyzing examples of correct and incorrect triples, their proposal is able to learn whether two entities should be connected according to the possible paths that can be traced between them.

Lin et al. [82] proposed a different use of embedding spaces, by defining two groups of them, one exclusively for entities and one for every different relation present

in a KG. Their model is able to express triples as transformations between relation spaces through the use of projection matrices. The increased number of embedding spaces makes it more computationally complex, but also more effective.

Nickel et al. [98] suggested using tensors to represent a Knowledge Graph, and then factorizing those tensors to obtain more compact representations of the knowledge in them. Their approach works by using a third-order binary tensor that holds information about which entities are connected in the graph and through which relation, and then performing a series of mathematical operations that result in another tensor, with confidence levels for any possible fact that could be introduced in the KG, given its current entities and relations. However, the size of the tensor scales quadratically with respect to the number of entities, making it a poor choice for large KGs.

Shi and Weninger [130] introduced a proposal that uses not only the embeddings of an entity, but also of its textual description, to look for additional levels of semantic similarity. Additionally, their proposal does not rely on generating negative examples, like most other existing state-of-the-art proposals do. However, it relies on external sources of information to retrieve the descriptions of the entities.

Trouillon et al. [145] proposed a technique that factorizes the tensor representation of a KG, but instead of binary values, it uses complex numbers. This can be seen as generating two separate tensors: one which contains the real parts of the values, and another one that contains the imaginary parts. The usage of complex numbers allows it to simplify some of its internal calculations, making it slightly more efficient.

Wang et al. [155] proposed using translation on hyperplanes to change the embedded representation of an entity, depending on which relation is being used to link it with another entity. However, these translations, just like the overall embedding space, must be re-learned whenever a change in the KG occurs.

2.4 Discussion

The previous proposals have problems that hinder their applicability in practice. Regarding the use of embeddings (P1), most of the existing KG completion techniques in the literature rely on them to some extent [17, 55, 72, 82, 98, 130, 145, 155]. This is problematic because, as discussed, most KGs are in constant expansion, and adding any entity or relation to the KG must trigger a re-computation of said embeddings, which is not feasible in practice.

Jiang et al. [69] and Shi and Weninger [130] propose techniques that depend on outside information (P2), making them vulnerable to changes in the sources of external information. Furthermore, the proposals made by Gardner and Mitchell [47] and Guo et al. [55] are not fully automated, and require manual intervention by the users (P3).

Regarding the automated generation of new knowledge (P4), there are few techniques that are able to do it independently. The approaches proposed by Galárraga et al. [43] and Jiang et al. [69] provide a straight-forward way to achieve this, since they generate first-order logical rules that can be used to materialize new knowledge in the KG. The tensor factorization approaches introduced by Nickel et al. [98] and Trouillon et al. [145] also enable this by traversing the resulting tensor, which contains confidence scores for all possible facts, and adding those whose score exceeds a certain threshold (although this may be time-consuming given the size of the tensors). Finally, Shi and Weninger [130] use a simplified way of generating candidate triples, by generating and examining those in which the left-hand entity and relation already appear together somewhere else in the KG.

Most existing proposals can be applied to large Knowledge Graphs (P5), although there are some exceptions. The proposals made by Galárraga et al. [43] and Jiang et al. [69] rely upon rule mining through ILP, which is known to scale poorly to large collections of facts [128]. Additionally, tensor factorization techniques, such as those proposed by Nickel et al. [98] and Trouillon et al. [145], need to generate tensors that contain $R \cdot E^2$ elements, where R is the number of distinct relations in a KG and E is the amount of different entities. This size can quickly become unmanageable when the KG contains more than a few thousand entities.

2.5 Our proposal

In this dissertation, we present a proposal for Knowledge Graph completion that does not suffer from the previously discussed problems. Our proposal consists of three different elements, each of which solves different problems regarding KG completion:

To determine the correctness of a triple, we introduce CAFE, our triple classification technique. It relies solely on path and entity neighborhood information present in the graph, which, contrary to embeddings, does not require any pre-computation or significant re-generation when the graph changes (P1). It is also able to operate taking only a Knowledge Graph as input, without any additional dependencies on external sources of information (P2). CAFE works by transforming the triples in a KG into a set of labeled feature vectors, using a novel set of context-aware features, and then learning to differentiate between correct and incorrect triples in a completely automated manner, without any user intervention (P3).

We also present CHAI, our proposal for automatically generating candidate triples. CHAI is able to materialize sets of possible facts of a reasonable size, that include most of the information that is missing in a KG. These facts can be passed on to a triple classification technique such as CAFE, thus solving the problem of generating new knowledge (P4).

Finally, to prove the applicability of our proposal, we introduce SciCheck, an extension of CAFE specifically tailored to scientific Knowledge Graphs. We show that SciCheck can be applied to AI-KG, a large scientific KG with over 14 million triples, yielding very satisfactory results and taking considerably less time than other existing approaches in the literature (P5).

2.6 Summary

In this chapter, we have motivated the reason for this dissertation. We have analyzed the problems of completing Knowledge Graphs and the current proposals in the literature to carry out this task, and we have concluded that none of these proposals solves all of the presented problems at a time.

Part II

Background Information

Chapter 3

Knowledge Graphs

"Knowledge is hot water on wool. It shrinks time and space."

— *House of Leaves*, Mark Z. Danielewski

Knowledge Graphs (KGs) are collections of facts that are represented in a graph-like structure, with entities and connections between them. This chapter provides an introduction to KGs, and it is structured as follows: Section 3.1 introduces the reader to their history and main characteristics. Section 3.2 provides an overview of the most prominent KGs in use nowadays. Section 3.3 presents some of the many practical applications of Knowledge Graphs. Section 3.4 reflects on the main current challenges regarding KGs. At last, Section 3.5 summarizes and concludes this chapter.

3.1 Introduction

Representing and storing structured domain-specific knowledge has been an active research topic since at least 1970, when the first relational databases were introduced [29]. Given their indisputable success, many alternative means of representing knowledge in an structured manner have been proposed over the years. One of such proposals was using what was coined as a *Knowledge Base* (KB) [60], a collection of facts that are stored as direct relations between concepts. Contrary to relational databases, which need to go through a normalization process that introduces a number of indirections to represent a piece of knowledge, KBs were considered more straightforward to operate and reason about [119]. A number of Knowledge Bases were thus created and maintained throughout the subsequent years by multiple organizations and research entities, which were both general-purpose [14, 25, 79, 87, 111, 148] and domain-specific [26, 71, 141, 158, 159].

The information inside a Knowledge Base was stored in the form of entities, which represented real-world or domain-specific concepts, and relations that link these entities together. This is known as a triple: a combination of two entities by means of a relation, which usually contains a verb. For example, a KB can represent the fact that Magnus Carlsen is a chess player using the triple *(Magnus Carlsen, plays, Chess)*. Such triples were generally encoded in a KB using the RDF/XML format [31], or an extension of it called RDFS, which allows for a higher degree of expressivity. However, since RDF/XML is mainly intended for machine use, later KBs also used other formats, like N3, Turtle, or RDF/JSON, which are more human-readable.

However, the introduction of the Google Knowledge Graph in 2012 [132] was a pivotal point for both the industrial use and academic research of Knowledge Bases. Rather than a simple collection of relations between names of entities, they started to be seen as a rich, interconnected structure of elements (“*things, not strings*”) with an enormous potential for practical and commercial applications. Many other large companies of the likes of Amazon, Facebook, Microsoft and eBay soon followed suit [75, 101, 110, 131], and the term Knowledge Graph (KG) rose to the popularity it still enjoys nowadays, replacing the denomination “Knowledge Base”.

In the following sections, we present the most popular Knowledge Graphs that are under active use today and the ways in which they are constructed. We then discuss the many practical applications that can, and have been, derived from the usage of KGs in commercial and academical contexts. However, despite their many benefits, Knowledge Graphs still present several challenges that must be addressed to improve their functionality and usefulness. Therefore, we also provide an overview of the main open challenges that need to be addressed to refine and improve Knowledge Graphs.

3.2 Current Knowledge Graphs

Nowadays, there are a number of popular Knowledge Graphs under active use, either commercial or academical. Some of the most prominent ones are as follows:

- **DBpedia [79]:** Initially, the DBpedia project aimed to obtain a graph-like structure from semi-structured information sources, mainly, from the infoboxes present in Wikipedia articles (see Figure 3.2). Once this was achieved, DBpedia was further enhanced by adding links to external KGs and general open resources. DBpedia is both a multi-domain and multi-language KG, with a rich and publicly available ontology that enhances the contents in it. It also supports live synchronization with dynamic Wikipedia articles, to ensure that its knowledge is consistently up-to-date.

Figure 3.1: The entity *Magnus Carlsen* in DBpedia

- **YAGO [111]:** A Knowledge Graph that shares a number of similarities with DBpedia. Just like it, YAGO is also a KG that extracts semi-structured information from Wikipedia to create, as its authors put it, a *"light-weight and extensible ontology with high quality and coverage"* [137]. A number of refinements are applied to the knowledge stored in YAGO, such as canonicalization, which removes possibly duplicate elements from the graph; or type checking, which prevents invalid combinations of entities from being formed.
- **FreeBase [14]:** A large-scale, multi-domain KG that aimed to collect human knowledge in order to facilitate its integration, usage and standardization. Unlike the previously discussed KGs, which use Wikipedia as their main sources of information, FreeBase was a collaborative project that relied on human editors and curators to add knowledge to it. It was also one of the first multimodal

KGs, since it allowed users to include text, images and media files. FreeBase was acquired by Google in 2010 and, upon the release of the Google Knowledge Graph, it was discontinued and put in a read-only mode. However, it is still a popular Knowledge Graph in the academic domain, since many different research works use it to benchmark their proposals.

Figure 3.2: A Wikipedia infobox

- **WikiData [148]:** After KGs such as DBpedia and YAGO successfully leveraged the information in Wikipedia's infoboxes, the Wikimedia Foundation soon realized that the information contained in them could be very useful. However, the data in these boxes needed to be manually added, edited and updated by human editors, and constantly kept up-to-date in as many languages as the original articles were available in. To alleviate this issue, they introduced WikiData as a centralized and crowdsourced source of information, to supply Wikipedia articles and a number of other applications with data. Contrary to most other KGs, and following the philosophy of Wikipedia, all data entered into WikiData is not immediately considered a fact, but rather a claim that must be supported by references to outside sources. Both entities and relations in WikiData are denoted with codes, which makes it language-agnostic, but can reduce its human interpretability if they are not mapped to their human-readable names. Figure 3.3 shows a small excerpt of WikiData, centered around the entity *Artist*.

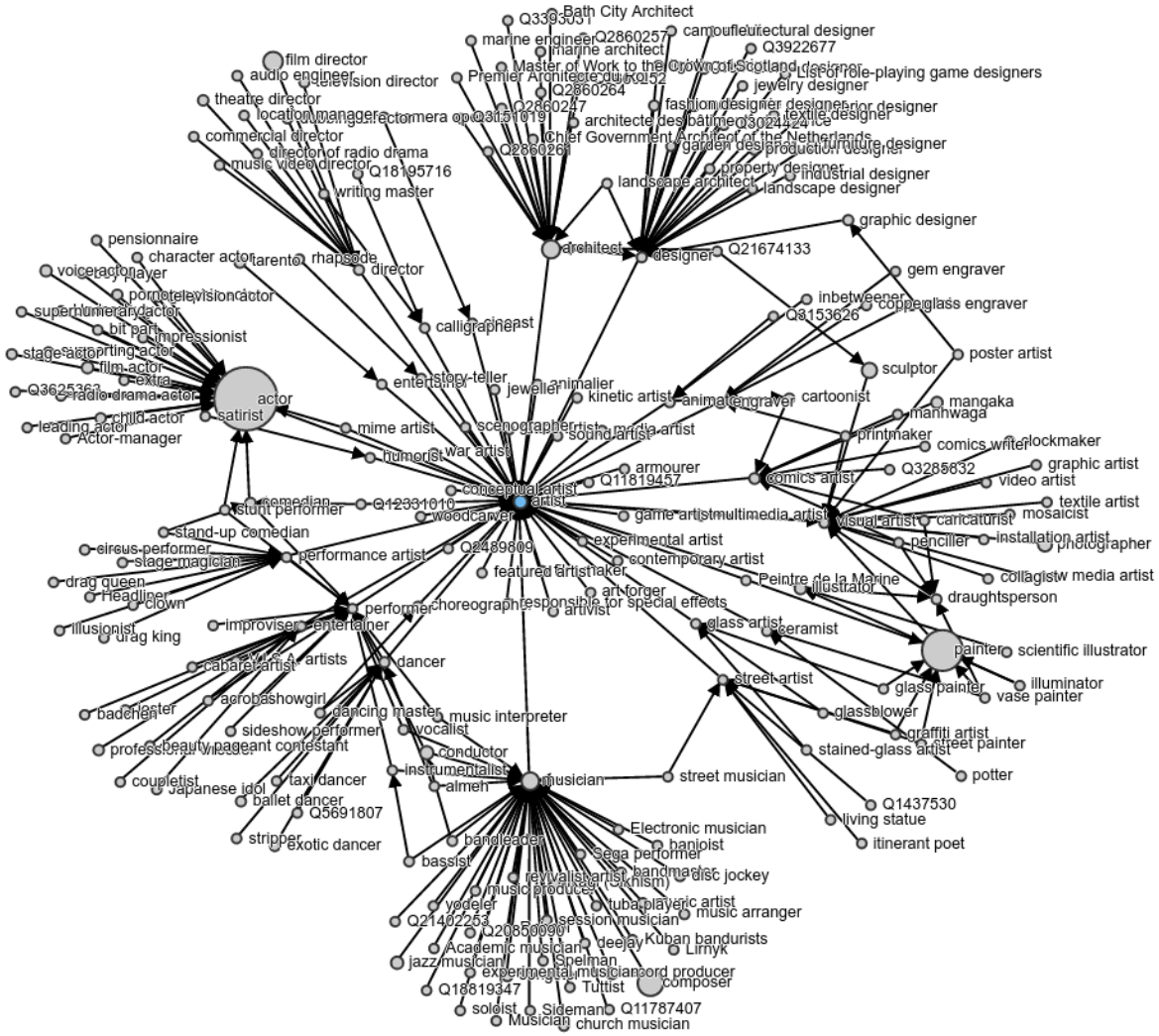


Figure 3.3: An excerpt of WikiData

- **NELL [91]:** The Never-Ending Language Learning project, or NELL for short, is a fully-automated system that scans the entire web, reads the contents of a webpage, and attempts to extract information from plain text written by humans. The information it extracts is then structured in the form of a triple and inserted in a KG. This Knowledge Graph is thus, by its very definition, under constant expansion. The NELL project allows users to rate an extracted fact positively or negatively, depending on whether or not the fact makes sense and is correct, and this allows the extractor to learn over time which facts are being successfully extracted, consistently improving its capabilities. NELL can also assign a confidence score to the facts it extracts, although only ~6% of its ~50 million facts have a high confidence score.
- **MAKG [133]:** A prime example of a domain-specific KG, the Microsoft Academic Knowledge Graph is a very large collection of over 8 billion triples with meta-information about research, such as academic publications, authors, institutions, and domains of study, among others. The information in it is obtained directly

from two main sources: academic publishers, such as ACM and IEEE, and websites indexed by the Bing search engine, from which the information must be extracted from a plain-text format. In total, more than 83 million papers and 20 million authors are present in this academic Knowledge Graph.

3.3 Applications

It has been established that Knowledge Graphs are a powerful tool to organize and connect information in a semantically meaningful way, making it easier for machines and services to understand and interpret complex data. The use of Knowledge Graphs has expanded rapidly across a wide range of fields, including healthcare, education, e-commerce, and many others [109]. By providing a structured representation of data, Knowledge Graphs enable more accurate and efficient decision-making processes, support natural language processing, and enhance the ability of machines to understand the context and relationships between different entities.

In this way, Knowledge Graphs have proven to be an invaluable resource for a variety of practical applications that we take for granted nowadays, which require complex data management and analysis. Some of the most popular practical applications of Knowledge Graphs are as follows:

- **Question answering:** Many search engines and personal assistants that are ubiquitous today, such as Siri, Alexa, Cortana, and Google Assistant, combine natural language processing (NLP) techniques, which helps them understand a query posed by a user, with Knowledge Graphs, which are used to navigate between entities and find the correct answer. Although most of the KGs that are used for this purpose are not made public for commercial reasons, they often contain a high amount of relations between both real-world and fictional entities and people. An example of this in practice is shown in Figure 3.4.

- **Product recommendations:** Many authors [54, 104, 150, 170] have analyzed and proposed using Knowledge Graphs to issue product or media recommendations to users. By storing the relevant elements in a KG, such as the products that are on sale and the relations between them, and the users that bought or rated a product positively, one can apply a number of techniques to find a product that will be appealing to a new user in the system. The same concept can be applied for books or movies, a prime example of this is the movie information site IMDb, whose information has been used by Amazon to build a movie recommendation engine [113].

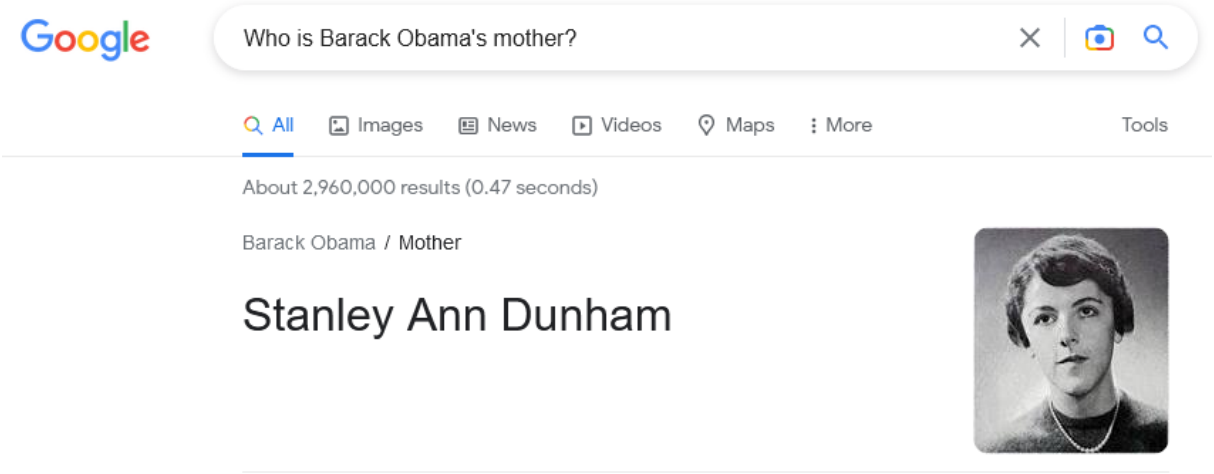


Figure 3.4: Question answering using the Google Knowledge Graph

- **Advanced information retrieval:** Traditionally, information retrieval systems work by building and maintaining an index of a collection of documents and, when a user submits a query to the system, returns the documents that match the query more closely. However, this purely text-based approach falls short when it comes to semantical problems, for example, disambiguating an entity in the query, or interpreting it correctly. For this reason, some authors have proposed using Knowledge Graphs to enhance this process: for example, during the recent COVID-19 outbreak, Wise et al. [157] introduced a COVID-19 KG that aggregated the most recently published scientific works regarding this disease. This KG could be queried to quickly retrieve relevant research efforts, facilitating a worldwide collaboration to quell the pandemic.

- **Education:** Multiple KG-centered proposals have been made in recent years to aid in several aspects of education. For example, Chen et al. [27] presented KnowEdu, a tool made for the purpose of building Knowledge Graphs out of educational materials. It is able to extract the main concepts of a subject or course, and then builds relations between them according to the activities and evaluation activities that the students will have to carry out in the future. Aliyu et al. [3] proposed building a KG containing the teachers, courses and scientific literature for a university, and using it to recommend relevant courses and books to students.

- **Research:** Besides the previously discussed MAKG, a number of other Knowledge Graphs of research and academic entities have been built to help researchers explore possible lines of investigation and find possible relevant related materials. AceKG, introduced by Wang et al. [153], is a similar KG, containing more than 3 billion triples with information about authors, papers, venues, and so on. The Artificial Intelligence KG [34], and its extension, the

Computer Science KG [37], also contain a large volume of information pertaining research concepts, materials, tasks, and other similar entities in these fields. Furthermore, Angioni et al. [5] introduced the Academia/Industry KG, to help researchers identify trends in the constant exchanges between these two realms.

- **Healthcare:** It is undoubtable that having an efficient and effective healthcare system is a cornerstone of a functional modern society. To help manage the ever-growing amount of medical data and records that healthcare professionals have to deal with, some proposals to integrate KGs in this field have been made. Zhang et al. [171] presented HKGB, a KG builder aimed towards healthcare professionals that leverages their expertise to construct medical Knowledge Graphs. Gong et al. [51] proposed constructing KGs with information about medicines, diseases and patients, in order to use it to suggest medicines to patients with particular incompatibilities or intolerances.

3.4 Open challenges

Despite the fact that Knowledge Graphs are already a well-established concept in research and industry, a number of challenges about their use and refinement still remain [62, 107, 109]. In the following subsections, we delve into some of the most prominent ones in more detail.

3.4.1 Integration

It is common that more than one Knowledge Graph contain information about the same real-world concept or entity. While each individual KG may include different additional information complementing it, it would be desirable that said information be integrated together, so as to have a single source of truth that contains a more comprehensive set of facts. The process of joining two or more Knowledge Graphs together is known as KG integration or fusion.

The most challenging step of this process is determining which entities in different KGs refer to the same one in the real world, a task known as entity alignment [114]. Although this has been a very active research topic, some future directions can still be further explored. For example, it is still not clear how this can be done reliably when the KGs to be integrated have different languages, which could be useful for multilingual recommendation systems or question answering [66]. An attempt in this regard has been made by Xu et al. [164] using neural networks, however, the alignment accuracy they obtain is still not high enough to perform this process reliably.

Moreover, most KG fusion approaches assume that the Knowledge Graphs to be fused together are of the same modality. This falls short in the presence of multi-modal

KGs, for example, the previously discussed FreeBase. Joining together two or more KGs which entities can be represented in different formats is still an open challenge, although some authors have made recent proposals. Guo et al. [53] have presented HMEA, a multi-modal entity alignment technique by using hyperbolic spaces. Cheng et al. [28] proposed MultiJAF, a framework for multi-modal entity alignment. Despite this, the wide array of possible modalities that could be represented through a KG make multi-modal alignment a very challenging and still unsolved task.

Furthermore, a related problem is entity disambiguation. This problem presents itself during the process of building the KG, and consists in determining the specific meaning of an entity written in natural text in the presence of ambiguity. For example, contextual information is required to determine whether *"Armstrong"* refers to the astronaut, the jazz musician, or the cyclist. If this is not solved correctly, even a successful entity alignment can be poisoned by the fact that the entities that were aligned were not disambiguated properly upon their creation, and thus do not actually refer to the same concepts.

3.4.2 Correction

The previous analysis of the most popular Knowledge Graphs shows that automatic construction is the most common way to build a KG, due to the sheer amount of facts that they are intended to contain. It is therefore inevitable that these automated methods introduce a certain degree of incorrect information in the KG, either because it was not correctly interpreted, or because the original source of information was wrong.

Refining a KG after its creation by detecting wrong facts is known as Knowledge Graph correction. The most common way to perform this task is to do fact validation [107], which assigns a confidence score in the interval $[0, 1]$ to every triple in the KG. Then, the triples that do not meet a minimum confidence threshold are purged from the graph.

There exist a number of research works in the literature that propose different ways to do fact validation. Pasternack and Roth [105] propose converting the facts in a KG into natural language sentences, and then using more general methods for fact checking. The same authors [106] also propose an alternative method, which groups up similar facts in a KG that provide support for each other, and then joins these groups to create more general justifications for the possible correctness of a fact. Gerber et al. [49] have proposed using classifiers, which can learn which facts are wrong by using a number of features. Of course, this requires that a manually-annotated set of correct and wrong triples is provided, which can be an arduous task. Another classification-based approach is proposed by Syed et al. [139], in this case, based on textual evidence.

A more refined —and challenging— approach to KG correction is to not only detect which facts are wrong, but to amend them if possible so that they represent correct knowledge. This is known as fact repairing. Due to the increased difficulty of the task, a smaller body of work can be found in the literature. Töpper et al. [142] proposed leveraging the ontology of a Knowledge Graph to detect and fix inconsistencies in triples where the domain or range restrictions of the relation are violated, however, their proposal requires a human to step in and select the correct version of it out of the suggestions provided by the system. Bonatti et al. [15] proposed a fully-automated method for fixing triples, but it requires provenance and trust annotations to be present in the KG, which are relatively rare.

3.4.3 Completion

As previously discussed, most KGs are commonly built by extracting non-structured or semi-structured information from web sources, though some KGs can be manually curated by domain experts. When information extraction systems are applied to extract knowledge from online sources, that information is then semantized [7, 96] and stored in a KG as triples in the KG. Regardless of the specific process by which a KG is constructed, the resulting graph usually lacks a certain amount of information, either because said information was not originally present in the information source, or because it was unsuccessfully extracted or semantized [16].

Because of this inherent incompleteness, KGs operate under the Open World Assumption, i.e., a piece of information that is not present in a KG is not considered to be incorrect, but rather just unknown [43]. Therefore, it is mandatory to refine KGs after their creation in order to expand the knowledge they contain [107].

Deriving additional knowledge from an existing KG with the goal to augment it is a task known as Knowledge Graph completion [107, 128]. In KG completion, the goal is to identify triples that are missing from the KG and have a chance of being correct that is as high as possible. To achieve this goal, a series of steps are usually carried out, which are visually depicted in Figure 3.5.

First, the Knowledge Graph must be pre-processed in order to add negative triples, in the case they are not present, and split them between a training and a testing set [6]. Then, a KG completion model is trained and evaluated using the sets of triples generated in the previous step. In parallel, a set of plausible candidate triples is materialized. If the KG completion model yields a satisfactory efficacy after its evaluation, it is applied on the candidate triples, to identify which ones are correct and which ones should be discarded [19, 20, 128]. Finally, the triples considered correct are added back to the Knowledge Graph, enriching it.

In opposition to the previous two challenges, which focus on information that is already present in the graph, KG completion, by definition, focuses on finding

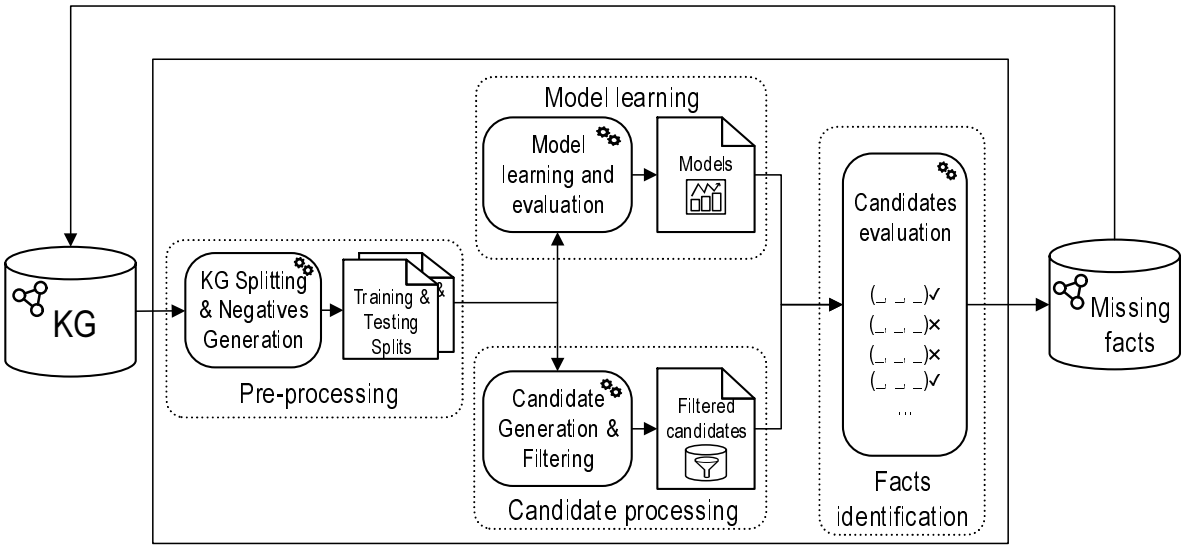


Figure 3.5: Knowledge Graph completion workflow

knowledge that is not yet present. For this reason, it is particularly hard, and a number of authors have made different proposals to tackle it. Since the focus of this dissertation is on Knowledge Graph completion, the following chapters provide a more in-depth analysis of the existing proposals in the scientific literature to complete Knowledge Graphs.

3.5 Summary

This chapter has introduced the reader to Knowledge Graphs. It has provided a brief summary of their history and main characteristics, as well as a summary of the most prominent KGs that can be found today. Furthermore, it has presented the reader with an ample repertoire of practical applications of Knowledge Graphs, many of which we use in our daily lives. Finally, it has introduced some of the challenges regarding KGs that still remain open to this day, namely: Knowledge Graph integration, correction, and completion.

Chapter 4

Latent triple representations

"All problems in computer science can be solved by another level of indirection, except for the problem of too many layers of indirection."

— David Wheeler

A latent representation of a triple is one that exposes previously hidden knowledge, such as semantic similarity to some other triple. Obtaining such a representation is a popular way to perform Knowledge Graph completion, and it can be achieved in a number of ways. In this chapter we introduce the most prominent KG completion methods in the scientific literature that rely on latent representations. This chapter is structured in the following manner: Section 4.1 provides an introduction to the matter, Section 4.2 presents the methods that perform tensor factorization, Section 4.3 introduces the models that are centered around embedded translations, Section 4.4 discusses the number of ways in which neural networks can be used in this regard; finally, Section 4.5 provides a summary of the contents of the chapter.

4.1 Introduction

A popular approach to Knowledge Graph completion consists on changing the representation medium of entities and relations entirely: instead of elements with a semantic meaning in a graph structure, they are represented in a numerical way. This then allows for the application of numerical methods to find missing entities, or missing connection between the existing entities. This is known as a latent representation.

One such way is by using tensors, a mathematical structure that can hold data in any number of dimensions. Consequently, through a series of transformations, the original tensor representing the Knowledge Graph is turned into another that materializes some knowledge that was not previously readily available.

Another popular latent representation is by creating an N-dimensional space and determining a position in it for every entity in a KG. An entity can then be referred to as the N-dimensional vector that represents its position. Such a space is known as an embedded space, and the position vectors are known as entity embeddings. In the embedded space, the plausibility of any given relation can be checked by performing a series of translations in it and evaluating the result, or by finding more complex relations between the entity embeddings thanks to the use of neural networks.

4.2 Tensor factorization models

Tensors are a generalization of scalar numbers, vectors, matrices, and so on. Broadly, a tensor of order N represents an N-dimensional collection of elements, where N indices are necessary to address the position of an element. Any given Knowledge Graph can be represented using a third-order tensor of size $|E| \times |E| \times |R|$, namely $X = \{0,1\}^{|E|\times|E|\times|R|}$, where E is the set of entities in the KG, and R is the set of possible relations in it. Using this representation, every element of the tensor is a binary number $\{0,1\}$ denoting whether or not a given relation exists between a pair of entities. This concept is visually represented in Figure 4.1.

Once this representation is achieved, the goal of tensor factorization models is to compute a complementary tensor $Y = [0,1]^{|E|\times|E|\times|R|}$ representing the correctness confidence of all possible combinations of entities and relations. Note that the elements in X are binary, while those in Y can take any real values between 0 and 1. Finally, to complete the KG, the facts with the highest confidences in Y that are not still present in the Knowledge Graph are added into it.

One of the first models in this line of work is RESCAL, which was proposed by Nickel et al. [98]. RESCAL represents the data in a Knowledge Graph using a tensor with the same structure as that shown in Figure 4.1. Then, it creates 2-dimensional

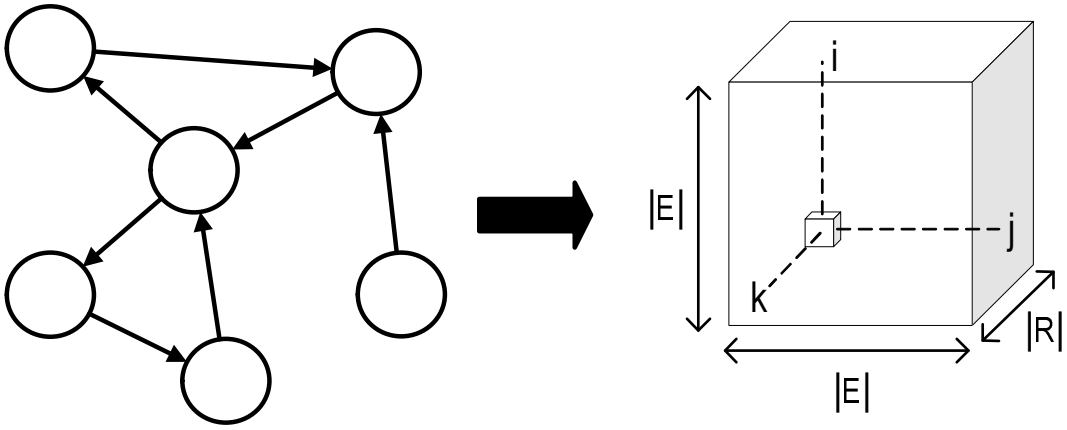


Figure 4.1: Representing a KG as a third order tensor

slices of this tensor, one per each relation, which are illustrated in Figure 4.2. Each slice, X_r, is factorized as $X_r = AB_rA^T$, where A is a matrix that contains the latent representations of the entities, and B is another matrix that represents the interactions of said entities for the relation r. Then, to predict the existence of a relation in them, the confidence scores are looked up in the computed latent representations.

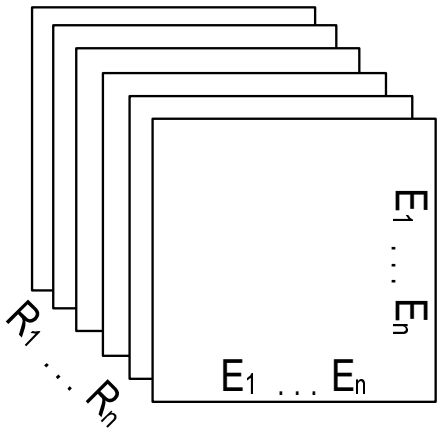


Figure 4.2: A third order tensor, sliced up in the RESCAL model

Jenatton et al. [67] proposed the LFM model, an extension of RESCAL that scales better to Knowledge Graphs with a higher number of relations. Similarly to the model it builds upon, LFM represents the tensor as a series of X_r slices, but introduces a more complex factorization based on a bilinear structure that is able to capture one-, two-, and three-way interactions between the components of a triple. Additionally, the matrix that represents the interactions between the latent representations of the entities is further decomposed into smaller elements, which reduces the number of parameters required for its computation and thus increases the efficiency of the model.

A combination of the previously discussed two approaches is proposed by García-Durán et al. [46], who introduced the Tatec model. Tatec has features of both RESCAL and LFM, learning two different embedded representations of the same Knowledge

Graph and combining their confidences together as $score(s,r,t) = s_1(s,r,t) + s_2(s,r,t)$, where (s,r,t) can be any triple, and s_1, s_2 are the score functions of a bilinear and trilinear tensor factorization model, respectively. Naturally, the scores emitted by both models must be normalized before they can be combined. The authors also suggest and benchmark a series of possible alternative ways to combine the scores of the two constituent models within Tatec.

Liu et al. [83] proposed the ANALOGY model, which is also an extension of RESCAL. ANALOGY derives its name from its focus on modelling and analyzing the analogical properties that are captured in the embedded representations of the entities. ANALOGY imposes further restrictions upon the relationship matrices of RESCAL, B_r, by demanding that they be normal ($B_r B_r^T = B_r^T B_r$) and commutative ($B_r B_{r'} = B_{r'} B_r$). This enables them to efficiently be block-diagonalized into a set of smaller matrices, which the authors show that enables them to be used in mathematical formulations of a lower complexity. Additionally, the factorization process of ANALOGY is guided by a fully-differentiable goal, which increases its computational scalability.

A similar effort was carried out by Yang et al. [166], who introduced the DistMult model. The authors of DistMult propose replacing the dense B_r matrix of the RESCAL model with a diagonal matrix, which greatly reduces its number of parameters and allows for a much speedier execution.

Tay et al. [140] introduced the REST factorization model, which is especially tailored for large Knowledge Graphs. REST relies upon random walks to sample small subgraphs within the KG, and represents these subgraphs using tensors. The subgraphs are further split up using a relation sparsification technique, yielding even smaller tensors. The tensors representing different parts of the KG are combined together to produce an ensemble model that can be representative of the entire Knowledge Graph. This ensemble architecture is visually depicted in Figure 4.3. REST has provided satisfactory results in practice, and its ability to work on-demand make it a convenient choice for Knowledge Graphs that are in continuous expansion.

Trouillon et al. [145] proposed the ComplEx model, which is the first one to use complex instead of real values in the tensors. This can also be seen as splitting an embedded representation of the entities into two: one which contains the real values of the numbers, and one that contains the imaginary parts. This increases the expressiveness of the model, and makes it better suited to handle symmetric and antisymmetric relations, which are traditionally more challenging for the previously discussed models, due to the fact that they greatly increase the number of parameters they must deal with [98, 134]. Perhaps counterintuitively, the use of complex numbers simplifies its score function, since it only uses the Hermitian dot product, which is the equivalent in $\mathbb{C}$ of the standard dot product in $\mathbb{R}$.

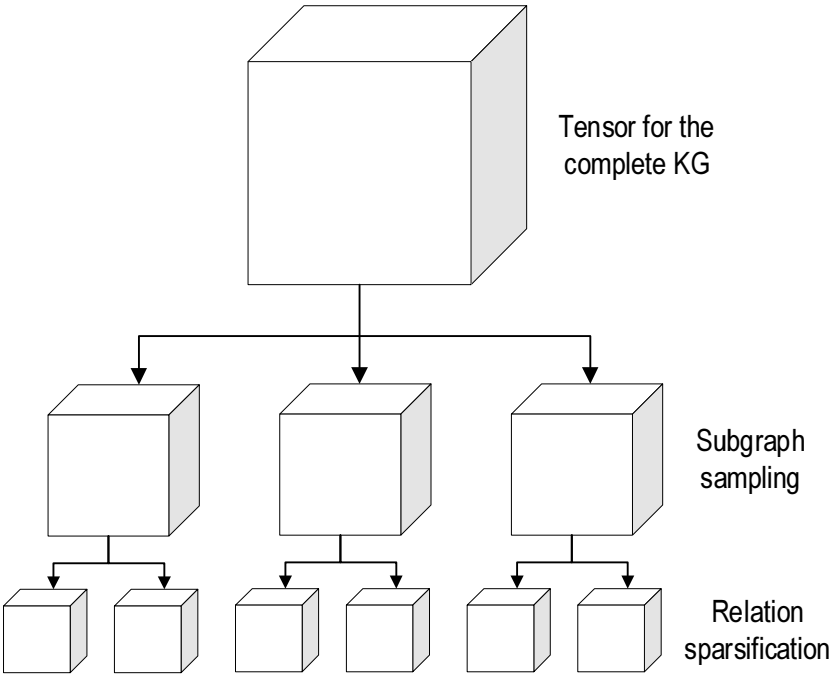


Figure 4.3: An ensemble of tensors, as proposed by the REST model

Perhaps inspired by the naming of the previous model, Kazemi and Poole [72] introduced the SimplE model, which aims to be more performant than other approaches in this category. For each relation r present in a KG, SimplE considers an additional inverse relation r^{-1} to it. Both are then represented using the vectors v_r and v_r^{-1}, respectively. Thus, the confidence score for a triple (s, r, t) is computed as the average of the confidence of (s, r, t) and (s, r^{-1}, t). This configuration allows the embedded representation of a relation and its inverse to be obtained independently, resulting in a higher expressivity and a good performance in practice.

4.3 Translational models

A separate group of models represent entities as vectors in an N-dimensional space, called embeddings, and relations as translations in that space. From this starting point, their main goal is to define the transformations from entities to vectors and the translations in such a way that, for every triple (s, r, t) in the KG, applying the translation defined by r to the entity s should result in a vector as close as possible to that of t. These models can then be used to provide a confidence value for the correctness of any triple, by evaluating to what extent this concept holds true.

Bordes et al. [17] started this line of work with the TransE model. TransE defines a single N-dimensional space to which all entities in a KG are mapped, and in which every relation turns into a translation vector. Thus, it aims to generate this space in such a way that, for any given triple (s, r, t), $v_s + v_r \approx v_t$, where v_s, v_t are the vectors representing the s and t entities respectively, and v_r represents the translation

carried out by the relation r. The main intuition behind TransE is visually depicted in Figure 4.4 for a simple 2D space.

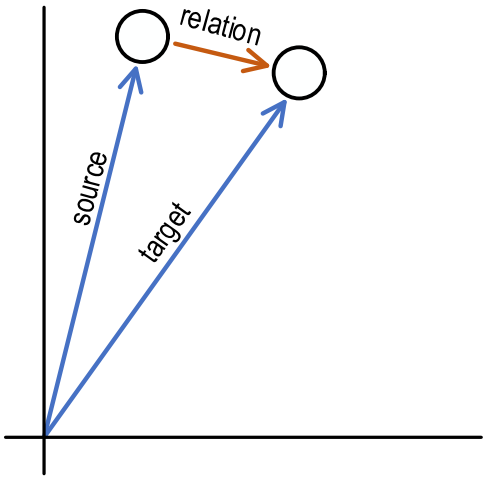


Figure 4.4: Visual representation of the TransE model in a 2D space

Wang et al. [155] proposed the TransH model, which improves TransE by considering that it may desirable for an entity to have a different embedded representation depending on the particular relation that is being considered. TransH achieves this by defining a particular hyperplane w_r for every relation r, and then projecting the entities onto these planes as $e_\perp = v_e - w_r^T v_e w_r$, where e can be any entity in the KG. Then, similarly to TransE, it optimizes the embedded space to ensure that $s_\perp + v_r \approx t_\perp$ for any triple (s, r, t). A similar visual representation is provided in Figure 4.5.

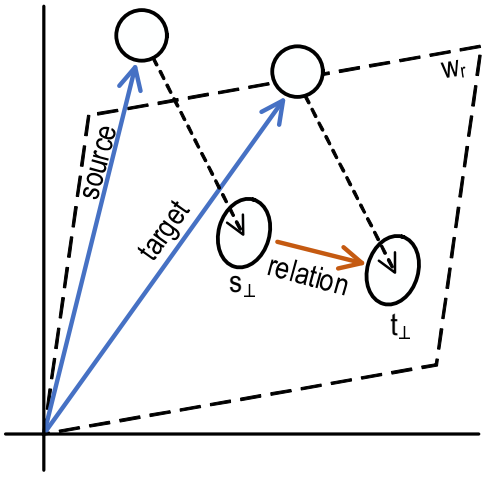


Figure 4.5: Visual representation of the TransH model in a 2D space

Following up with the previous idea, Lin et al. [82] proposed the TransR model. It acknowledges that an entity and a relation have a very different semantic meaning and, for this reason, TransR makes a twofold contribution. On the one hand, it defines two separate embedded spaces, one for entities and one for relations. On the other hand, it creates a different entity embedded space for each relation. The transition

from the entity space to the relation space is performed through a relational projection matrix M_r. The translation goal thus becomes $v_s M_r + v_r \approx v_t M_r$. A graphical example of this is provided in Figure 4.6.

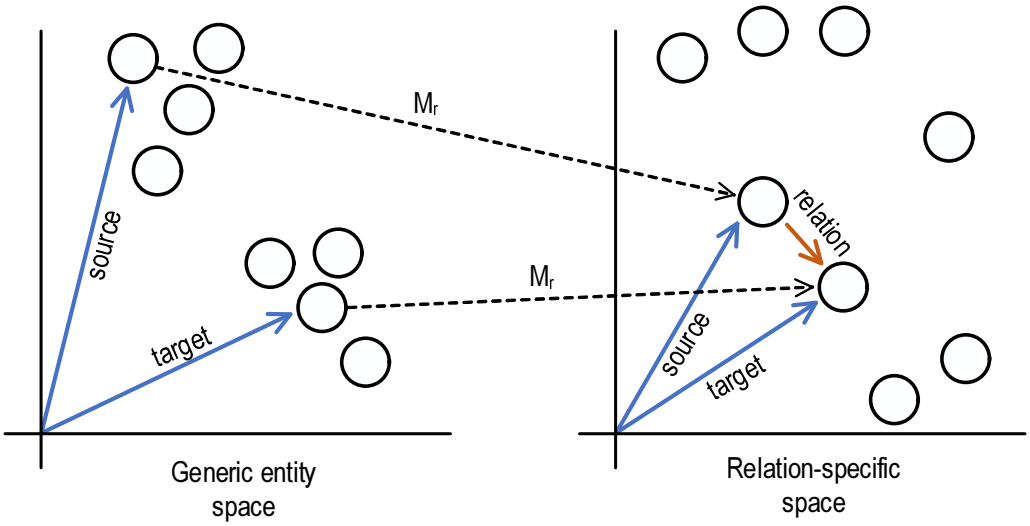


Figure 4.6: Visual representation of the TransR model in a 2D space

Ji et al. [68] introduced the TransD model, building upon TransH and improving it by constructing mapping matrices dynamically for every entity. Furthermore, it replaces the matrix multiplication operations used in the previous models with vector multiplications, which increases its operational speed. A similar approach is proposed by Do et al. [39] with TransF, which uses more lightweight matrices to perform the translation.

Fan et al. [41] devised TransM, addressing the fact that the original TransE proposal considers all triples to be equally important, and thus they all contribute equally towards generating the embedded space in such a way that it optimizes the transformation goal. TransM adds a weight parameter that can be assigned to every individual triple, which affects its ability to influence the embedded space as a whole and acts as an attention mechanism.

Xiao et al. [161] presented the TransA model, which uses an adaptive goal metric during the embedding generation process, overcoming the simple metrics used by other similar methods, making it more flexible when facing challenging relations.

Perhaps facing the prospect of running out of letters in the English alphabet to continue with the previous naming scheme, Sun et al. [138] proposed the RotatE model, which defines an embedded space that uses complex numbers. Contrary to the previous models, where a translation is defined as a straight movement through the space, RotatE defines each relation as a rotation on the whole embedded space.

4.4 Neural network-based models

In parallel to the development of research centered around Knowledge Graphs, the fields of neural networks (NNs) and deep neural networks (DNNs) have also seen significant improvements in the last decade [23, 117, 147].

Consequently, a number of NN and DNN architectures have been applied to the problem of completing a Knowledge Graph, by virtue of transforming a triple into a representation that can be consumed by these models, and training them so that they can learn what constitutes correct knowledge. Contrary to the previously discussed translation-based models, NN models apply non-linear transformations to the data they are provided. This can result in an increased ability to deal with more complex data, however, it also makes their reasoning harder to understand.

One of the first models that were proposed in this regard was NTN, by Socher et al. [134]. This model relies on a pre-computed set of word embeddings, in which semantically similar words are expected to be close to each other in the embedded space. NTN takes all the meaningful words in the name of an entity and averages the respective word embeddings together, resulting in a single embedding for every entity. In this manner, the embeddings of the two entities of a triple are provided as input to a bilinear neural network, which then outputs a confidence score for the entity pair. However, NTN is dependent on the availability and quality of the aforementioned word embeddings. Furthermore, these embeddings must be computed again if an entity with a novel word in its name is introduced in the Knowledge Graph.

A simplified version of NTN is used by Dong et al. [40] in their Multilayer Perceptron (MLP) proposal. MLP is used in the Google Knowledge Vault, a KG created by the same authors, to filter out possibly incorrect facts that are automatically extracted from plain text around the Web. To make it more lightweight in order to be applied to such a large KG, MLP changes the interaction function used by NTN to a multi-layer perceptron, which is much faster to train.

A further refinement is proposed by Liu et al. [84], who introduced the Neural Association Model (NAM). It uses deep neural networks to learn and infer the joint conditional probabilities between two facts. One of the main strengths of NAM is its increased explainability, since using conditional probabilities makes it easier to justify the correctness of a triple using other triples known to be correct as supporting evidence.

Guan et al. [52] proposed a Shared Embedding-based neural network (SENN) for the task of KG completion. SENN aims to achieve a higher level of specialization by using three separate neural sub-structures to predict a missing source entity, relation, or target entity in a triple. The results of these structures are combined together to assess the correctness of a whole triple. By virtue of being an ensemble of smaller

structures, SENN is also more efficient than some of the other pre-dating models.

The interconnected nature of Knowledge Graphs also calls for the application of convolutional neural networks (CNNs). These networks do not focus on only one entity at a time; they are also able to derive information from other entities close to it, having a larger picture of the KG as a whole. In this line, Dettmers et al. [38] introduced ConvE, a two-dimensional CNN that operates on the entire matrix that represents entity embeddings. ConvE uses a traditional 2D convolution to augment its predictive capabilities with respect to more classical NN architectures. It also has a more reduced number of parameters that must be trained, increasing its efficiency.

An evolution of ConvE is the InteractE model, proposed by Vashishth et al. [146]. InteractE uses a novel circular convolutional structure, which enables it to capture more meaningful interactions. Nguyen et al. [97] also build upon the main idea of ConvE to introduce CapsE, a method that works by using a capsule network [120] that converts the entities of a Knowledge Graph into images, and then performing more traditional 2D convolutions on said images.

In more recent years, graph convolutional networks, also called simply graph neural networks (GNNs) have emerged as a way to apply NNs to graph-like structures [24, 73, 160, 173]. It seems thus reasonable that Knowledge Graph completion could benefit from the application of GNNs [168].

In this regard, Schlichtkrull et al. [125] proposed R-GCN, a type of GNN that leverages graph neighborhoods in order to complete them. Shang et al. [127] have proposed SACN, an encoder-decoder method based on this type of NN. For encoding, SACN uses a composition of multiple weighted GNNs that is able to leverage information from the structure of a Knowledge Graph and the attributes contained in its entities. The decoder uses a modified version of the previously mentioned ConvE model. The addition of GNNs allows it to outperform the original ConvE method.

4.5 Summary

In this chapter, we have presented an overview of the existing methods for Knowledge Graph completion in the literature that rely on latent triple representations. We have introduced and described the models that use tensors to model a KG, and then factorize those tensors to obtain a predictive model. We have also presented the models that embed entities in a higher-order space and perform translations on them to find missing knowledge. Finally, we have enumerated the methods that use neural networks to perform Knowledge Graph completion, and we have presented the neural network architectures that are commonly used for this task.

Chapter 5

Path-based approaches

"Caminante, no hay camino, se hace camino al andar."
("Traveler, there is no path, you make your path as you walk.")

— Antonio Machado

Paths between entities are ubiquitous in any Knowledge Graph. For every entity pair, there are most likely a considerable number of distinct paths that connect them together. By analyzing these paths, many existing proposals are able to learn which pairs of entities should be linked together, and which ones have no direct connection whatsoever. Furthermore, paths can be used to navigate the neighborhood of an entity and extract information from it. In this chapter, we summarize the existing proposals for Knowledge Graph completion that leverage path information. It follows this structure: Section 5.1 introduces the reader to the basic concepts, Section 5.2 presents the proposals that are centered around using only information derived from paths, Section 5.3 introduces the methods that analyze entity neighborhoods, Section 5.4 provides an overview on the proposals that combine latent representations with path-based information and, finally, Section 5.5 concludes this chapter.

5.1 Introduction

A relational path is a sequence of relations that connect two entities together in a Knowledge Graph. There can be many such paths between any two entities, and they can vary in length and complexity. Given that these paths can provide an interesting insight on how related two entities are, many researchers have developed methods to analyze them to learn what constitutes correct knowledge.

Generally, these methods generate features that characterize the information contained in each path, and then train classifiers to predict the existence of missing relations based on these features. Due to the high number of possible paths that can exist between two entities, such methods usually use techniques like random walks to restrict the amount of paths that are analyzed, possibly constraining their effectiveness.

To solve this issue, other methods analyze graph neighborhoods, which refer to the entities and relationships that are directly connected to a given entity in a Knowledge Graph. This allows KG completion method to focus on a much more reduced amount of information that is more likely to accurately characterize an entity.

In this chapter, we present an overview on the methods that rely on these techniques to complete a Knowledge Graph. We first introduce the methods that rely purely on relational paths and their characterization. Next, we present the methods that analyze entity neighborhoods. Finally, we discuss more advanced methods that combine these approaches with techniques from the previous chapter.

5.2 Using relational paths

A relational path is simply a concatenation of triples that connect an entity to some other entity through a number of relations. Naturally, any Knowledge Graph has an abundance of possible paths to, from, or between entities. The idea of analyzing such paths to learn what constitutes valid knowledge has led to many research works.

One of the first approaches that exploited relational paths in a Knowledge Graph was the Path Ranking Algorithm (PRA), proposed by Lao et al. [78]. PRA uses random walks to generate a set of paths between a given pair of entities in a depth-first manner. The generated paths are then transformed into a series of features that aim to characterize the information contained in it. These features are ultimately used to train a binary log-linear classifier, which learns whether a direct relationship should exist between two entities according to the paths that are already present between them. However, PRA is not without limitations. The number of possible paths between a pair of entity can be very high, limiting its scalability. Additionally, the randomly guided paths that it uses may miss information just by mere chance.

Aiming to improve PRA, Gardner and Mitchell [47] introduced the Subgraph Feature Extraction (SFE) method. In opposition to PRA, SFE uses a breadth-first search that is not randomly guided, in order to better characterize the portion of the KG between two entities. SFE also requires the creation of a handmade "Alias" relation, which relates entities in the same KG that refer to the same element in the real world. It is able to achieve more expressive results than PRA, making it easier for the human user to understand what constitutes a good path that may be indicative of the existence of a direct relation between two entities.

Gardner et al. [48] also proposed an extension of PRA that addresses the issue of potentially having to consider a very high number of paths. It considers the semantic similarities between the relations of a Knowledge Graph, and uses them to merge very similar paths together, greatly reducing the number of paths that need to be analyzed afterwards. This addition increases the effectivity of PRA in the NELL Knowledge Graph, although the authors do not provide data for any other KGs.

Nastase and Kotnis [93] proposed the Abstract Path Model (APM), which produces an abstract graph from a KG, and then focuses on extracting relevant abstract paths from it. These abstract graphs provide a more general representation of the high-level relations between entities, and thus the paths that can be traced in it represent more high-level knowledge. Additionally, this abstract representation is, as a general rule, considerably smaller than the KG it is derived from and not very computationally taxing to obtain, which aids in its application to larger graphs.

Guu et al. [57] introduced the Trans-COMP method, which proposes a different strategy. Rather than analyzing a large number of paths between two entities to try to characterize the pair, it only uses a single path. Precisely, it aims to select the path that provides the highest predictive capabilities of the existence of a given relation. Toutanova et al. [144] further refine this idea with their All-Paths method by providing an explicit way to represent the intermediate entities that can be found in such a path and, additionally, enabling the method to consider more paths.

A number of other authors have also researched alternative ways to select and learn from the best paths in a Knowledge Graph. Jiang et al. [70] proposed the APCM model, which assigns different weights to the found paths and combines them based on these weights. Furthermore, the PRCTA model introduced by Lei et al. [80] employs an attention mechanism [147] to construct and select the paths, which allows it to work satisfactorily in more sparse KGs.

5.3 Using entity neighborhoods

Considering the possible paths between two entities is undoubtedly helpful, but this concept can be expanded further. Rather than a single path, we can consider the

entirety of the region of a Knowledge Graph around a certain entity, in other words, its neighborhood. The concept of entity neighborhood is visually shown in Figure 5.1, which depicts the closest and more extended neighborhoods of an entity shown in the center.

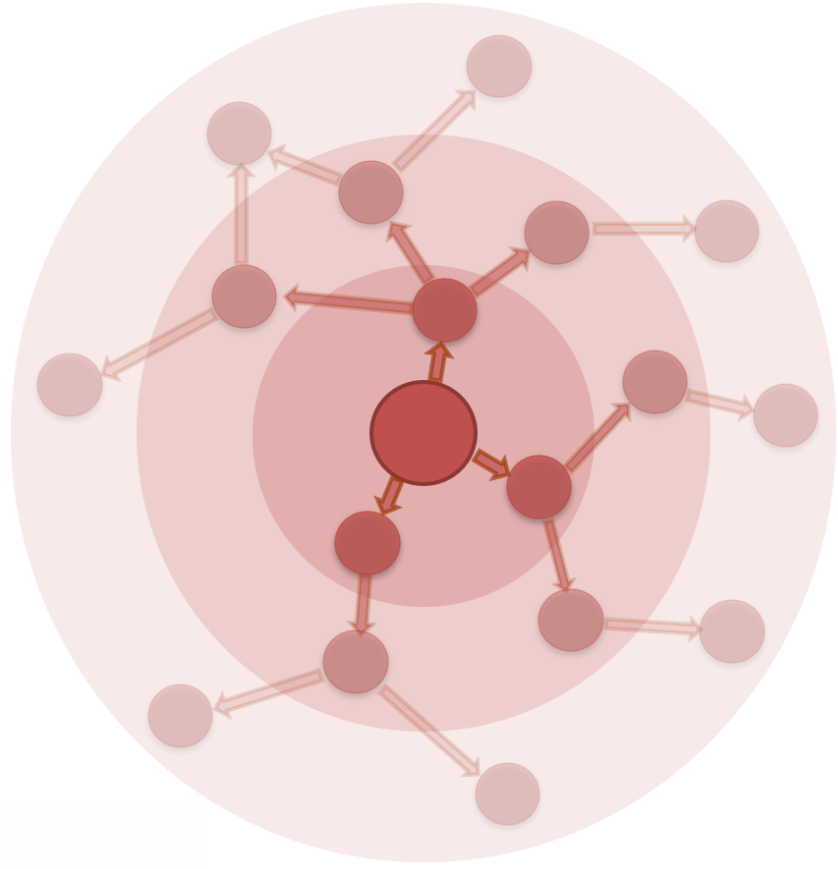


Figure 5.1: A visual representation of entity neighborhoods

Given a pair of entities, their neighborhoods can be analyzed to determine if a relation should be present between them. This idea has been put to practice by a number of authors.

Bansal et al. [12] proposed the A2N method, which aggregates the entities in a neighborhood together to obtain a more compact representation of said neighborhood. It uses an attention-based mechanism to focus on the most prominent entities, giving a trustworthy representation of the entities surrounding another one. Additionally, it is designed such that the computational cost of the method does not increase greatly with the size of the neighborhoods.

The use of attention-based mechanisms in this regard has not gone unnoticed by other authors. Wang et al. [151] have proposed LAN, another approach that aggregates the contents of an entity neighborhood together, however, the attention-based technique that they propose uses weights to give more importance to more relevant entities. Kong et al. [74] also proposes using attention to filter out possibly irrelevant relations in the graph, allowing the models to focus only on more meaningful information, which is

especially relevant in the case of heterogeneous Knowledge Graphs. Similarly, Nathani et al. [94] introduced the KBAT method, which is able to capture information from both the entities and the relations in a KG neighborhood.

Zhang et al. [172] further the previous idea by adding a weighted attention system to both entities and relations, acknowledging that not all knowledge in the KG is equally useful. Two different attention-based mechanisms are used in their proposal. First, the relation-level attention provides an intuition for which connections from an entity may provide more useful information. Then, the reached entities are considered according to their entity-level attention. This composite, hierarchical attention mechanism allows it to outperform other attention-based proposals.

Ferré [42] proposes finding missing information by finding similar entities in commonly occurring graph patterns, through the application of concepts of nearest neighbors [33]. It does not require any sort of pre-processing of the Knowledge Graph, making it very efficient. Furthermore, its reliance on graph patterns makes it a more interpretable method than other similar proposals.

5.4 Hybrid approaches

The previous chapter introduced the main ideas behind how embeddings and neural networks can be applied to Knowledge Graph completion. Some authors have proposed a series of approaches that combine path-based information with these techniques, in order to guide the path finding process towards the most significant path, avoiding having to explore a very large number of them.

An example of this is the PATH-RNN technique, which was proposed by Neelakantan et al. [95]. In addition to using path-based information, PATH-RNN uses the embedded representations of the relations in a path to further characterize it. It uses a recursive neural network (RNN) to combine these embeddings together, resulting in a single embedding that contains information about the entire path. For this reason, it can operate using paths of any length. Additionally, due to the fact that it operates on the embedded representations of the relations, which in turn capture semantic meanings, it can in theory predict relations that were not present in the KG at the time of training the model. Figure 5.2 graphically illustrates this idea.

Das et al. [30] improve the previously discussed PATH-RNN method with their Single-Model proposal. They point out that taking the embeddings of the entities in a path into account can provide useful information. To represent a path, they recursively combine the embeddings of both the entities and relations in it using RNNs. Furthermore, they propose using a number of different functions to select the best path: *Top-K*, *Average* and *LogSumExp*. They experimentally conclude that the *LogSumExp* function performs better when completing a Knowledge Graph. Figure 5.3 displays a

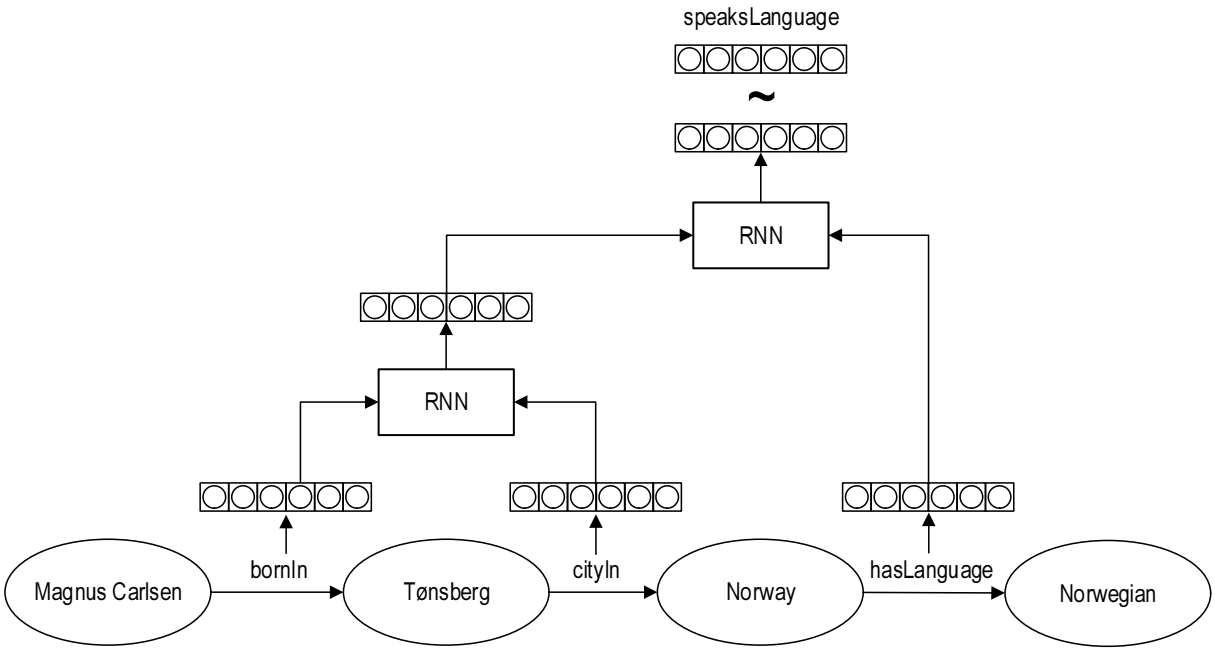


Figure 5.2: Overview of the PATH-RNN method

visual overview of Single-Model. In this Figure, entity embeddings are represented in blue, and relation embeddings in orange. The path that is being considered is the same as in Figure 5.2.

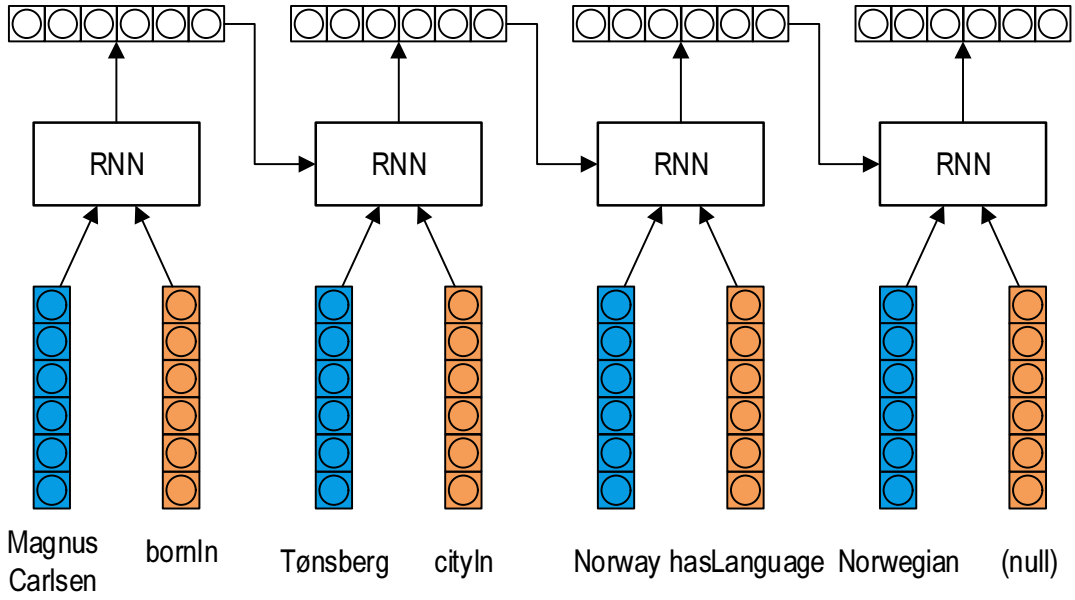
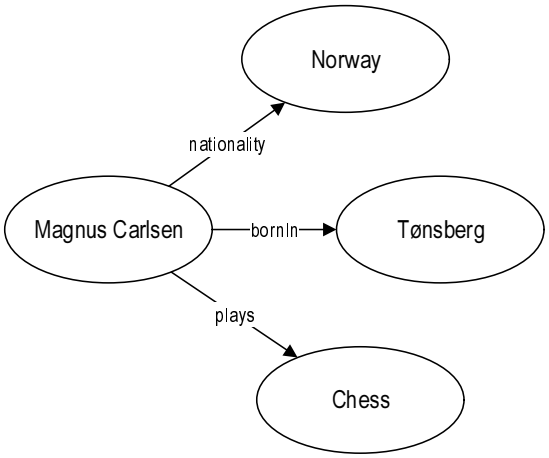


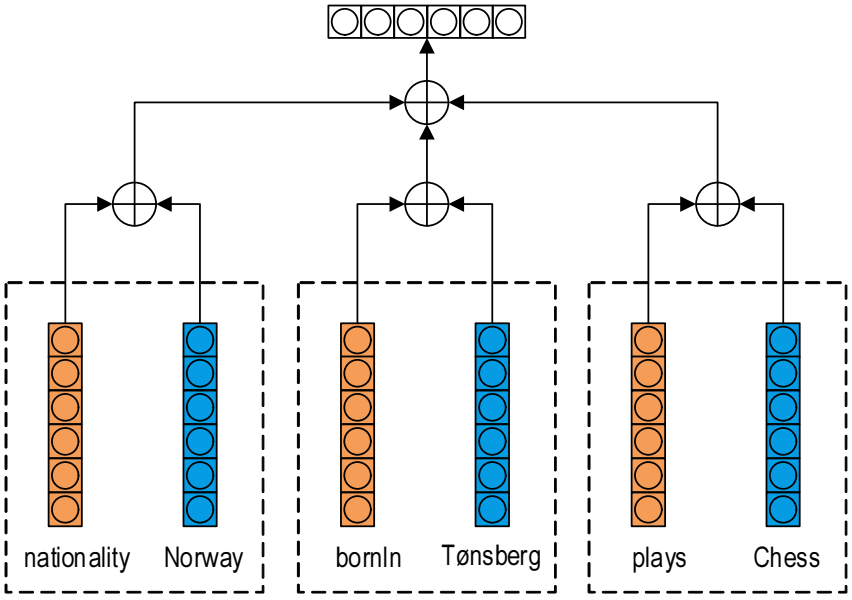
Figure 5.3: Overview of the Single-Model method

Xiong et al. [163] proposed GMatching, a technique that specializes in extracting information from KG neighborhoods using relatively infrequent relations, which are traditionally considered more challenging due to the reduced amount of information about them present in the graph. GMatching is comprised of two main components: a neighbor encoder, which creates an embedded representation for an entity in a

neighborhood; and a matching checker, which computes the similarity of two entity embeddings created by the first component. A visual representation of this proposal is provided in Figure 5.4. The meaning of the colors is the same as in the previous Figure.



(a) An example KG



(b) Encoding of the neighborhood of the entity *Magnus Carlsen*

Figure 5.4: Overview of the GMatching method

Shen et al. [129] proposed Implicit ReasoNets (IRNs), a neural network architecture that is able to reason about paths of different lengths in a KG. It is an encoder-decoder model that is governed by a central controller, which allows the whole process to be carried out with no human intervention. Also, it introduces the usage of a shared memory that implicitly stores relevant information about the graph, allowing it to be more efficient and have a smaller memory footprint.

Additionally, a number of improvements have been made to the baseline translational models to help them leverage path information in several ways. For

instance, Lin et al. [82] have proposed PTransE, an extension of TransE that uses path information in its confidence function. Its goal is to give a higher confidence score to those entities that are well-connected together by means of paths that are semantically similar to the relation in the triple that is being evaluated.

García-Durán et al. [45] proposed RTransE, which represents paths as a series of translations in the embedded space defined by the TransE model. For efficiency reasons, RTransE is limited to using only paths that contain two relations. Likewise, Xiong et al. [162] introduced PTransD, an enhancement of TransD that performs subsequent translations to model paths. However, PTransD uses two embeddings to represent each entity, to perform operations in parallel.

5.5 Summary

The contents of this chapter have provided an overview of the methods to perform Knowledge Graph completion that use relational paths. We have first listed the most prominent scientific proposals that extract paths from a KG and characterize them using features, to learn which paths can be predictive of correct knowledge. Afterwards, we have centered on the proposals that use entity neighborhood information in a number of ways. Finally, we have introduced the approaches that merge together path-based information with latent entity representations, entity and relation embeddings, and neural networks.

Chapter 6

Rule-based approaches

"Logic, like whiskey, loses its beneficial effect when taken in too large quantities."

— Edward J. M. D. Plunkett, Lord Dunsany

Logical rules are commonly used to perform Knowledge Graph completion. The proposals that employ these kinds of rules usually mine them first using the triples present in the graph, to generalize specific knowledge stored in it. Then, the extracted rules are used to materialize knowledge in the form of new triples, which can be added back to the KG. In this chapter, we provide an overview of the various ways in which this has been carried out by previous works. It is organized as follows: Section 6.1 lays out the foundational concepts, Section 6.2 presents the existing methods for mining logical rules from a Knowledge Graph, Section 6.3 introduces the proposals that aim to reduce a set of possible candidate triple using rules, Section 6.4 enumerates the approaches that combine rules with other popular ways to complete KGs, and Section 6.5 concludes the chapter.

6.1 Introduction

Knowledge Graphs are essentially large and incomplete collections of facts about a certain domain. One possible way to complete them is to observe which facts occur frequently together, and then express this relationship as a rule for those combinations that are observed very often. For example, if a person was born, studied and died in a city, it is very likely that they hold the nationality of the country in which that city is located. More formally, this can be expressed through the following logical rule, where p is a person, c is a city, and C is a country:

$$bornIn(p,c) \wedge studiedIn(p,c) \wedge diedIn(p,c) \wedge cityIn(c,C) \rightarrow hasNationality(p,C)$$

These rules are called first-order rules, and they represent explicit knowledge, easy for humans to understand and reason about, in opposition to most latent representation models. They are composed of two elements: the body of the rule (left-hand part) represents the logical condition that must be met, and the head (right-hand part) is the knowledge that is considered to be true if the condition is also true.

To complete a Knowledge Graph, one can first extract such rules from it, by observing common appearances of these kind of patterns. Then, the rules can be applied to materialize the head of a rule whenever its body exists, generating new explicit knowledge [136]. This process is visually depicted in Figure 6.1.

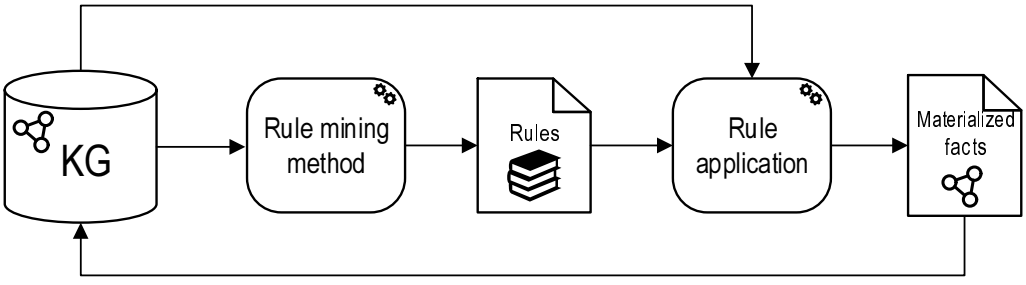


Figure 6.1: Extracting and applying rules on a Knowledge Graph

In this chapter, we present the existing methods in the literature for obtaining and applying first-order rules to complete a Knowledge Graph. First, we introduce the methods that focus solely on rule extraction. Then, we present some applications of first-order rules to the task of candidate filtering. Finally, we discuss some proposals that combine rules with other ideas presented in previous chapters.

6.2 Rule mining methods

There are a number of approaches to mine first-order rules in Knowledge Graphs. One of them is using Inductive Logic Programming (ILP) [92], a classical statistical relational learning method that can be used to extract such rules from a collection of facts.

In this regard, Jiang et al. [69] have proposed using ILP to perform Knowledge Graph completion on KGs that have a strong time constraint component. Such a KG may contain information on whether a person is the president of a country, whose correctness depends on the time period in which it is interpreted. In this proposal, the most common time periods for an assortment of different facts are inferred through ILP, and then used to assess the correctness of future facts. However, it relies on all facts having time annotations, which may not be commonplace.

Galárraga et al. [43] proposed the AMIE+ method, which generates similar rules using ILP. AMIE+ addresses the fact that, due to the Open World Assumption (OWA), a fact that is not present in a KG should not be considered false, but instead simply unknown. The OWA thus makes it very challenging to generate truly false examples to assess the overall validity of a rule. The authors use a bespoke confidence measure for their rules, known as the partial completeness assumption confidence. AMIE+ improves the efficiency of its predecessor method AMIE [44] and can be applied to larger Knowledge Graphs.

Wang and Li [154] refine this idea with their RDF2Rules method. Contrary to AMIE+, which is limited to only being able to mine one rule at a time, RDF2Rules speeds up the process by parallelizing rule extraction. It achieves this by detecting and extracting frequent relation cycles of a certain length in a KG, which are essentially loops that contain a given amount of relations. An example of such a loop can be found in Figure 6.2. Note that the directionality of the edges in a KG is relevant for the existence of a cycle. Once the most common cycles have been obtained, a number of rules can be extracted from them. This is done by iteratively selecting one relation as the head of the rule and the rest as the body, advancing on the loop, and repeating this process until the entire loop has been traversed.

The Never-Ending Language Learning (NELL) system that was proposed by Mitchell et al. [91] also learns knowledge rules from the data that it is constantly provided. These rules are manually screened to ensure a high level of quality, so as to not introduce incorrect facts into a Knowledge Graph. It then applies these rules to generate knowledge that was previously missing.

Markov Logic Networks (MLNs) [115] have also been used for the task of completing a Knowledge Graph. MLNs combine the previously discussed rules with probabilistic models, which allows them to derive generalized knowledge from a

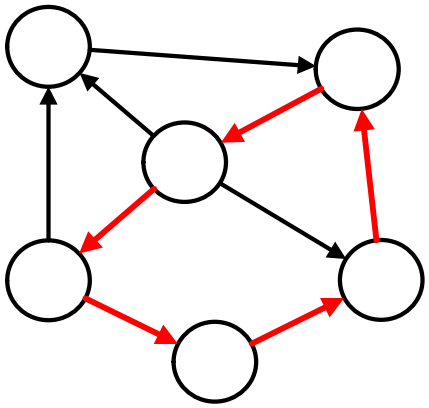


Figure 6.2: A cycle of 5 relations in a sample Knowledge Graph

smaller corpus of facts and to better handle complex and noisy information [167].

The use of MLNs for Knowledge Graph completion has been analyzed by Kuzelka and Davis [77]. The authors conclude that MLNs can provide a satisfactory performance on this task, assuming that the triples that are missing from the graph are independent from one another and have a roughly equal probability of being true, which is not always the case [19].

Yang et al. [167] presented Neural Logic Programming (NeuralLP), an approach that combines first-order rule mining with sparse matrix multiplication. In this approach, the authors propose using an attention mechanism to further refine the confidence value that is assigned to each individual triple. The main rule mining mechanism in NeuralLP is governed by a central neural controller. Additionally, it is able to learn rules of variable length with more ease than its predecessors.

Furthermore, Sadeghian et al. [121] introduced DRUM, which extends NeuralLP by analyzing the structure and confidence values of the rules that are being inferred, and then approximating these elements for other rules using tensors. It, however, is only limited to positive examples due to the OWA and is not able to infer negative rules.

Rocktäschel and Riedel [116] proposed NTP, a similar method to NeuralLP, which infers rules by using transitive relations between facts. Their approach requires that such relations be represented as a vector or a tensor, in order to leverage the semantic similarities that are commonly exploited in embedded spaces. It nonetheless suffers from a lesser scalability than the original NeuralLP method, due to the computational complexity that is required to carry out the process of rule inference.

To address the aforementioned scalability issues, Minervini et al. [90] presented an improved NTP2.0 method. This new version is able to focus only on the most promising rules during the mining process, by using a pooling method that is able to monitor the creation of multiple rules at once.

6.3 Candidate filtering

Some authors have proposed rule-based techniques for filtering candidate triples, instead of generating new knowledge. The process of generating and applying these rules is fundamentally different: due to the very large number of possible candidate triples, the rules must not be computationally expensive to apply. Additionally, it is not as important for them to be fully correct, since an incorrect fact will be evaluated in more detail further down the KG completion process. It is, however, desirable that the candidate filtering rules exclude as few correct candidates as possible, in order not to hinder the quality of the final set of generated triples.

Wei et al. [156] proposed the INS method, which uses the previously discussed TransE embedding model to filter out possibly incorrect knowledge. More specifically, they employ TransE to analyze the semantic similarity between the two entities in a triple, and discard the triple if the similarity does not exceed a certain threshold. This results in sets of candidate triples that are smaller than the original ones.

Shi and Weninger [130] also argued that it is generally not practical to apply any model to the whole set of possible candidate triples, and that it must be narrowed down in some way. In their work, they use a set of simple rules to determine that any given triple (s, r, t) is a valid candidate if another triple with the structure $(\_, r, t)$ is already present in the Knowledge Graph.

Zhang et al. [169] proposed IterE, an approach that prunes knowledge using graph traversing and random selection. It generates a set of plausible rules and monitors their performance as they are being built, removing those that are found not satisfactory and leaving only a smaller set of rules that can be generated quickly.

Some of the previously discussed works can also be used for filtering candidate triples. The authors of NTP2.0 [90] proved that a k-nearest neighbor search can provide satisfactory results to filter out wrong knowledge when inferring rules, which can be performed efficiently.

Other proposals incorporate parameters that can be fine-tuned to rapidly rule out rules that are not satisfactory. Omran et al. [103] proposed RLvLR, which allows the user to set values for the minimum required confidence for a rule. A similar approach is followed by the already discussed DRUM [121] method.

Additionally, the AMIE+ method [43] can also be used for this purpose. It includes a number of strategies that can be used to prune a set of candidate triples, by producing simpler rules with a high support. Additionally, AMIE+ can perform confidence approximations, which allows it to speed up the rule inference process, making it more appealing for its application to candidate filtering.

6.4 Hybrid approaches

Even though rule-based approaches excel in their explainability, they often have trouble scaling up to very large Knowledge Graphs [128]. To overcome this issue, many authors have proposed methods that combine more traditional rule mining with other approaches discussed in previous chapters, to try to guide the rule mining process towards more promising rules.

One of the first such proposals was made by Wang et al. [152], who introduced the r-KGE method. It combines the tensor-based model RESCAL, the embedding-based model TransE, and logical rules. These rules are then used to prune the embedded space using integer linear programming [126], which sees a significant reduction of its size. However, it is not properly equipped to handle N-to-N relations, and its reasoning process can still be quite computationally expensive. The overall model proposed by r-KGE is shown in Figure 6.3.

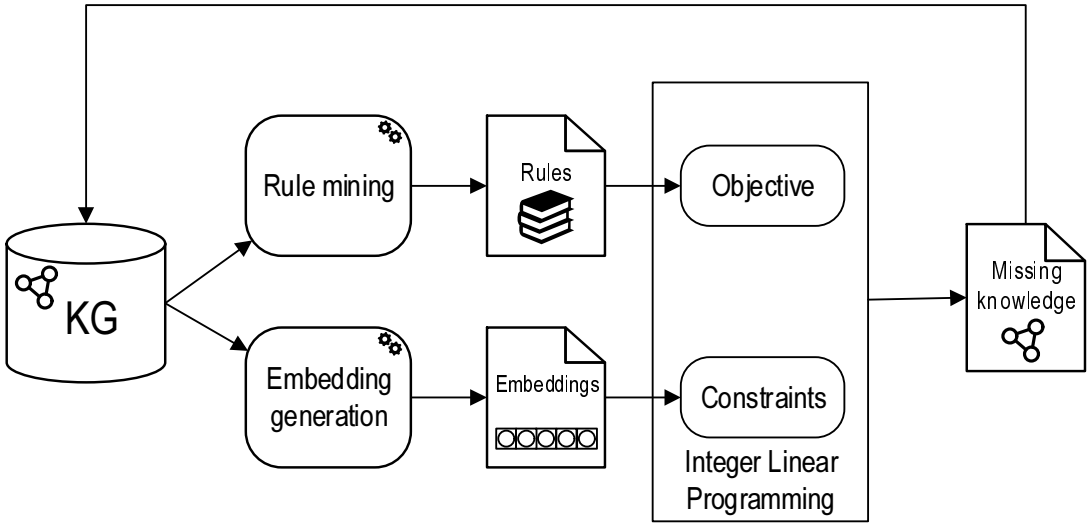


Figure 6.3: Overview of the r-KGE model

The previously discussed INS method [156] also incorporates TransE to quickly compute the degree of similarity between entities, and limits the rule reasoning process by taking only the top N most similar entities into consideration. Additionally, this similarity score can provide an approximation of the quality of a rule before it is completely built.

Guo et al. [55] introduced KALE, another model that combines logical rules and entity embeddings. KALE aims to provide a common ground in which rules and embeddings can directly interact, by representing triples as atomic formulae and rules as combination of these formulae. The semantic similarity information that is intrinsically present in the entity embeddings aids in expanding the predictive capabilities of the rules and their generality. A visual overview of the KALE architecture

is provided in Figure 6.4. In this Figure, entity embeddings are represented in blue, relation embeddings in orange, and scalar confidence values in green.

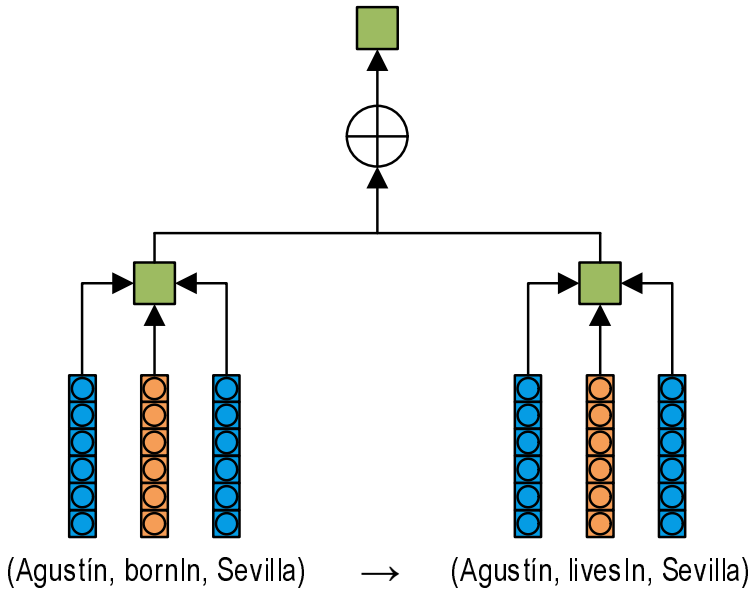


Figure 6.4: Overview of the KALE model

The same authors [56] also presented RUGE, a KG completion technique that combines the same elements in an iterative fashion. Rather than relying on pre-computed entity embeddings, RUGE generates its own embedded space with the aid of logical rules. Additionally, RUGE is able to operate on Knowledge Graphs that has both labeled and unlabeled triples. A series of logical rules are applied on the unlabeled triples to label them. Then, the labeled triples are used to rectify and improve the embedded space so that it better captures the relations between the entities. The improved embedded space provides feedback on the labels, and the rules can be updated accordingly. A diagram depicting this process can be found in Figure 6.5.

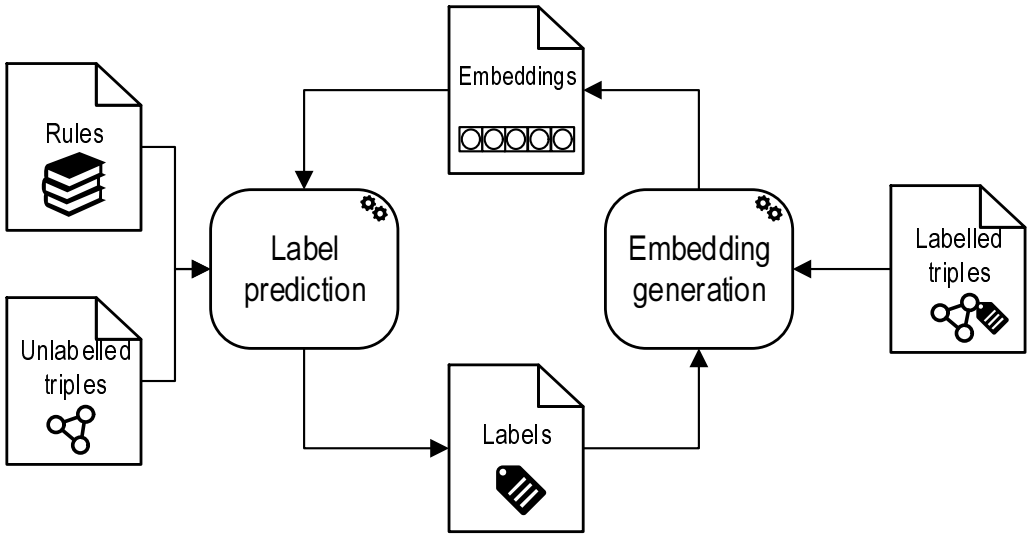


Figure 6.5: Overview of the RUGE model

Furthermore, another aforementioned method, IterE [169], proposes a similar approach. IterE generates an initial set of entity embeddings for the Knowledge Graph. These embeddings are used to generate a set of rules, whose quality is evaluated. The best-performing rules are then used to generate new triples that are introduced in the KG, and the process starts anew by generating new embeddings that take into account the newly generated knowledge. This is shown in a graphical manner in Figure 6.6.

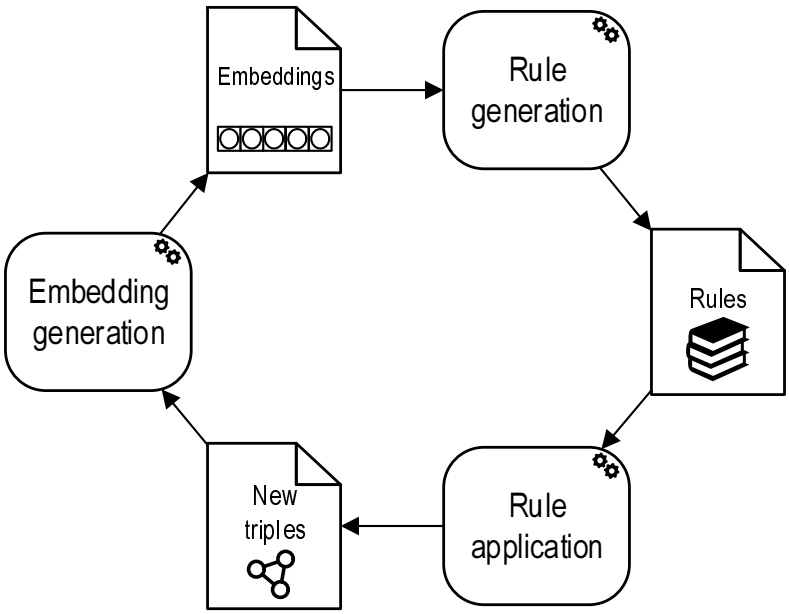


Figure 6.6: Overview of the IterE model

Meilicke et al. [88] proposed AnyBURL, a technique that can generate logic rules in a bottom-up manner and on-demand. AnyBURL works by deconstructing a Knowledge Graph into a set of labeled paths. Then, it uses path-based features to determine which paths contain more useful information to obtain rules from. AnyBURL is more lightweight than other related rule-based proposals and, due to the fact that it only considers the most promising paths inside a KG, can be applied to larger graphs.

Niu et al. [100] presented RPJE, a proposal that brings together path-based information and first-order rules. It first mines logical rules, and then uses those of length 2 to combine paths in the KG, and rules of length 1 to create a number of semantic relations.

Finally, Ma et al. [86] introduced ELPKG, a proposal that brings together all three main approaches that we have covered in the previous chapters. It uses entity embeddings to represent the relations between entities, and a breadth-first search to detect the paths between the two entities in a triple. The information gained from both the embeddings and the paths is combined together, and it then applies soft logic to obtain the final confidence value for the triple.

6.5 Summary

This chapter has provided an overview of the current approaches to Knowledge Graph completion that are based on logical rules. First, we have introduced the proposals that can be found in the literature that rely solely on obtaining and applying these rules. Then, we have focused on the methods for filtering candidate triples using first-order rules. At last, we covered the methods that combine rule mining with path-based information or latent representations, such as tensors or entity embeddings.

Part III

Our Proposal

Chapter 7

Conceptual framework

"Most problems can be solved using algebra, or violence."

— Bill Wurtz

Before we delve into the details of our proposal, it is important to establish a common and unambiguous vocabulary. For this reason, we have devised a conceptual framework that allows us to describe a number of relevant concepts in detail. The chapter is organized as follows: Section 7.1 introduces it, Section 7.2 models the concept of a triple, Section 7.3 presents the theoretical model of a Knowledge Graph, Section 7.4 introduces topology-based elements such as paths, distances and reachability, Section 7.5 illustrates the concept of neighborhood subgraphs, Section 7.6 presents the notions of candidate triples and candidate-filtering fitness, Section 7.7 describes candidate-filtering criteria and rules, and Section 7.8 introduces graph-based features and feature groups; finally, Section 7.9 summarizes the chapter.

7.1 Introduction

Throughout our proposal, we use a number of concepts, both established in this field and novel. In this chapter, we define their foundations, such as tuples of entities and relations known as triples, or the fields of a Knowledge Graph, which allow us to accurately describe further concepts. We also define paths inside a Knowledge Graph, which are the basis for many of the elements in our proposal. Building upon the notion of a path, we provide a formal definition of the distance between two entities in a Knowledge Graph, as well as of the concept of entity reachability.

Furthermore, in this chapter, we define neighborhood subgraphs, which are portions of a KG that contain the elements most closely related to a given entity. Then, we introduce candidate triples, which are combinations of entities and relations that have a high likelihood of representing correct knowledge. To measure the tentative aptness of a candidate triple, we present the idea of candidate fitness. In order to rule out those candidate triples with a low fitness, we define candidate filtering criteria and rules. Finally, we introduce a way to numerically model a triple in a KG through the use of graph-based feature groups.

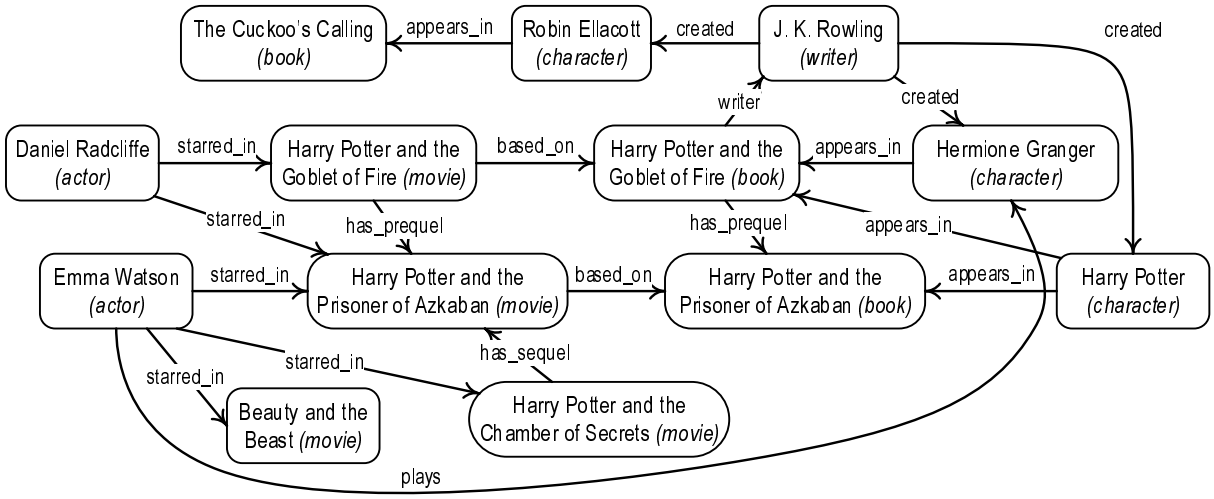


Figure 7.1: Sample KG describing works, actors, writers and characters

To help illustrate some concepts in this chapter, Figure 7.1 presents a KG that contains information about fictional works, actors, writers and characters.

7.2 Triple

The notion of a triple is pivotal to Knowledge Graphs, since it constitutes an atomical amount of structured information. A triple is a 3-tuple that contains two entities, commonly denoted *source* and *target*[1]; connected by means of a relation. This, in turn,

[1]Other literature sometimes refers to the two entities in a triple as *head* and *tail* [17, 34, 37], or *subject* and *object* [10, 99, 145].

represents a fact in a given domain.

Formally, a triple is defined as follows:

Definition 1. ***Triple:*** *Let $\mathcal{E}$ be a set of entities, and let $\mathcal{R}$ be a set of relations. We define a triple as a 3-tuple that represents the existence of a relation $r \in \mathcal{R}$ between a source entity $s \in \mathcal{E}$ and a target entity $t \in \mathcal{E}$. We denote triples as (s, r, t).*

In the sample KG depicted in Figure 7.1, a sample triple is *(Emma Watson, starred_in, Beauty and the Beast)*.

7.3 Knowledge Graph

A collection of triples forms a Knowledge Graph, which contains an assorted set of facts. Although, as discussed, triples have no inherent guarantees about the correctness of the knowledge they represent, it is in the best interests of both the curators and users of Knowledge Graphs to ensure that the triples contained in it are as trustworthy as possible.

A Knowledge Graph can thus be defined as:

Definition 2. ***Knowledge Graph:*** *Let $\mathcal{E}$ be a set of entities, let $\mathcal{R}$ be a set of relations, and let $\mathcal{T}$ be a set of triples of the form $\{(s, r, t) \mid s, t \in \mathcal{E}, r \in \mathcal{R}\}$. We define a Knowledge Graph as $\mathcal{KG} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$.*

It is important to note that, in addition to a set of triples, KGs also contain the sets of entities and relations, $\mathcal{E}$ and $\mathcal{R}$ respectively, that are considered to be included in the KG and thus allowed to take part in the triples that compose the KG. Although this may seem limiting, they are routinely expanded as new knowledge is added to a KG [40].

Figure 7.1 graphically represents a KG with 13 entities, 7 distinct relations and 19 triples, represented as edges that connect pairs of entities.

Note that a triple in a Knowledge Graph states a fact, but it may be one that is not considered to be true or correct. We thus define a correct triple as follows:

Definition 3. ***Correct triple:*** *Let $\mathcal{KG} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ be a Knowledge Graph, and let (s, r, t) $\in$ $\mathcal{T}$ be a triple in $\mathcal{KG}$. We consider that the triple is correct if the relation that it establishes between s and t holds true in the real world or in the domain of application of $\mathcal{KG}$.*

For example, the triple *(Barack Obama, born_in, Kenya)* is formally valid, but it is not correct. It is also noteworthy that the real-world correctness of a triple may depend on the time period in which it is interpreted, for example, *(Barack Obama, president_of, United States)*. A triple, on its own, does not have a mechanism to express a time constraint or any other restrictions about its correctness.

7.4 Topology-based concepts

Since a KG is a specific case of graph, we can leverage some concepts that arise from using such a structure.

7.4.1 Paths between entities

Given two entities in a Knowledge Graph, a relevant question is whether there exists a path in the KG that connects them together. To answer this question, we must first define the concept of path:

Definition 4. ***Path:*** *Let $\mathcal{KG} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ be a Knowledge Graph, and let $s, t \in \mathcal{E}$ be two entities in $\mathcal{KG}$. We define a path p between s and t as a sequence of triples of the form $p = \langle (e_i, r_i, e_{i+1}) \rangle$ for $i = 1..n$, where $e_1 = s$, $e_{n+1} = t$ and $(e_i, r_i, e_{i+1}) \in \mathcal{T}$ for $i = 1..n$. We denote a path p between s and t using the relations $r_1 \dots r_n$ as $path(s, t, r_1, r_2, \dots, r_n)$, or $path_n(s, t)$ for short.*

Building upon the previous definition, to further characterize a path, we define the length of a path as follows:

Definition 5. ***Path length:*** *Let s and t be two entities in $\mathcal{KG}$ with $s, t \in \mathcal{KG}$. Let p be a path between s and t of the form $p = \langle (e_i, r_i, e_{i+1}) \rangle$ for $i = 1..n$. We define the length of a path as the number of triples it contains, i.e., $|p|$.*

In the KG depicted in Figure 7.1, an possible example of a path of length 2 between the entities *J.K. Rowling* and *The Cuckoo's Calling* would be $\langle$ *(J.K. Rowling, created, Robin Ellacott), (Robin Ellacott, appears_in, The Cuckoo's Calling)* $\rangle$.

It is possible that there exists more than one possible path between a pair of entities. It is also a possibility that there are no possible paths between two entities. Thus, it can be useful to know how many distinct paths of a given length there exist connecting two given entities. For this purpose, we define a set of possible paths as follows:

Definition 6. ***Possible paths:*** *Let s and t be two entities in $\mathcal{KG}$ with $s, t \in \mathcal{E}$. We denote the set of all possible distinct paths of the form $path(s, t, r_1, r_2, \dots, r_n)$ as $\mathcal{P}(s, t, r_1, r_2, \dots, r_n)$.*

7.4.2 Distance between entities

In light of the previous definitions, it now seems reasonable to devise a measure of how close two entities are in a given Knowledge Graph. Given that KGs are not weighted graphs, we can define the distance between two entities as the minimum number of relations that we have to traverse to go from the first one to the second. Formally, the distance is defined as follows:

Definition 7. ***Distance between entities:*** *Let $\mathcal{KG} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ be a Knowledge Graph, and let $s, t \in \mathcal{E}$ be two entities in $\mathcal{KG}$. We define the distance between s and t as the length of the*

shortest path that exists between s and t in $\mathcal{KG}$, i.e., $|path_n(s,t)|$ such that $\nexists\ path_i(s,t) \mid i < n$. If no path of any length exists between s and t, then the distance between them is ∞. We denote the distance between s and t in $\mathcal{KG}$ as $dist(\mathcal{KG}, s, t)$.

In the KG depicted in Figure 7.1, the distance between the entities *Daniel Radcliffe* and *Harry Potter* is 4, since that is the length of the shortest path that exists between them.

The careful reader will note that, due to the fact that KGs are directed, distance is not symmetric, and thus the order of the entities is relevant: the distance between *Harry Potter* and *Daniel Radcliffe* in Figure 7.1 is ∞ because no path exists between them. The impossibility of such a path can be trivially verified by noting that *Daniel Radcliffe* has no inbound edges.

7.4.3 Reachability

Most graph algorithms rely on some notion of whether an entity is reachable from another one or not, and expand upon this notion to build assessments about the whole graph, for example, by determining its connected components. Knowledge Graphs are no exception, but given the semantical differences of the relations in them, it makes sense to restrict this notion of reachability to be able to answer a more precise question: "*Is an entity reachable from another one through a certain relation?*"

To formalize this idea, we define reachability in a KG as follows:

Definition 8. ***Reachability:*** *Let $\mathcal{KG} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ be a Knowledge Graph, let $s, t \in \mathcal{E}$ be two entities in $\mathcal{KG}$, let $r \in \mathcal{R}$ be a relation in $\mathcal{KG}$, and let $n \geq 1$ be a natural number. We define Reach as a predicate that determines whether there exists a path of length n between s and t in $\mathcal{KG}$ such that the relation r appears in the last triple of the path, i.e., $Reach(\mathcal{KG}, s, t, r, n) \iff \exists\ path_n(s,t) \wedge \exists\ a \in \mathcal{E} \mid last(path_n(s,t)) = (a, r, t)$.*

With a reachability predicate, we can proceed to find a subset of entities in a KG that are reachable from a given entity using a certain relation and distance:

Definition 9. ***Reachable entities:*** *Let $\mathcal{KG} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ be a Knowledge Graph, let $s, t \in \mathcal{E}$ be two entities in $\mathcal{KG}$, let $r \in \mathcal{R}$ be a relation in $\mathcal{KG}$, and let $n \geq 1$ be a natural number. We define the set of entities that can be reached from s through a relation r at distance n as the set of entities that match the predicate Reach under such circumstances, i.e., $\{t \in \mathcal{E} \mid Reach(\mathcal{KG}, s, t, r, n)\}$. We denote the previously defined set as $Reachable(s, r, n)$.*

In the example KG depicted in Figure 7.1, $Reachable(Hermione\ Granger,\ writer,\ 2) = \{J.K.\ Rowling\}$.

7.5 Neighborhood subgraphs

The previous sections have given us the necessary tools to accurately establish the concept of "neighborhood" in a KG by leveraging its topology. It seems appropriate to define that an entity e_1 should be in the neighborhood of e_2 if e_1 is reachable from e_2, given the previous definitions[2]. Since reachability is constrained by distance and a specific relation, we define the neighborhood subgraph of a given entity as follows:

Definition 10. ***Neighborhood subgraph:*** *Let $\mathcal{KG} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ be a Knowledge Graph, let $e \in \mathcal{E}$ be an entity in $\mathcal{KG}$, and let $n \geq 1$ be a natural number. We define the neighborhood subgraph of e of size n as a Knowledge Graph $\mathcal{KG}_n^n = (\mathcal{E}_e^n, \mathcal{R}, \mathcal{T}_e^n)$ that contains the triples whose target entities can be reached from e at a distance of at most n through any relation, and the entity set that can be derived from such triples, where $\mathcal{T}_e^n = \{(s', r', t') \in \mathcal{T} \mid Reach(\mathcal{KG}, e, t', r', i), i = 1..n\}$ and $\mathcal{E}_e^n = \bigcup\{\{s, t\} \subseteq \mathcal{E} \mid (s, r, t) \in \mathcal{T}_e^n\}$.*

To help illustrate this concept, Figure 7.2 showcases two possible neighborhood subgraphs for the KG shown in Figure 7.1.

7.6 Candidates

As discussed earlier, triples do not make inherent guarantees about the correctness of the knowledge they represent. To represent triples with a higher knowledge quality, this section introduces candidate triples and the notion of fitness in candidate filtering.

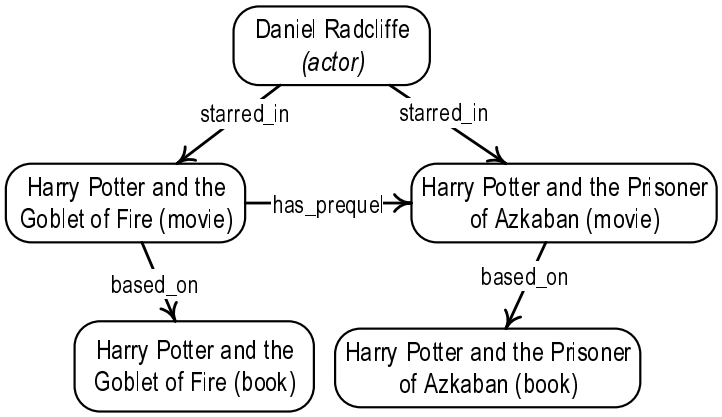
7.6.1 Candidate triples

One can form a syntactically correct triple by just combining two random entities with a random relation. However, chances are that the resulting fact is most likely not correct. Contrary to such a low-quality triple, a candidate triple, or just "candidate" for short, is a triple that has been created in such a way that its chance to represent correct knowledge is significantly higher than by mere randomness:

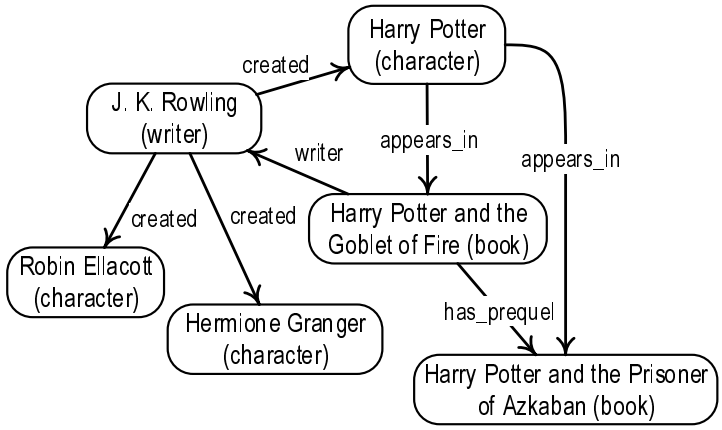
Definition 11. ***Candidate:*** *Let $\mathcal{KG} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ be a Knowledge Graph, let $s, t \in \mathcal{E}$ be two entities and let $r \in \mathcal{R}$ be a relation in $\mathcal{KG}$. We define a candidate as a triple (s, r, t) that has a significantly high chance to represent real-world knowledge, even if it does not exist in $\mathcal{T}$.*

For example, given the KG shown in Figure 7.1, a candidate triple could be *(Daniel Radcliffe, plays, Harry Potter)*. Note that this triple does not exist in the KG in its current state.

[2] Again, the non-symmetricality of distance may result in e_1 being in the neighborhood of e_2, but not vice-versa. Sadly, good neighbors are not always reciprocated.



(a) Neighborhood subgraph of size 2 for the entity *Daniel Radcliffe*



(b) Neighborhood subgraph of size 3 for the entity *Harry Potter*

Figure 7.2: Two neighborhood subgraphs for the KG shown in Figure 7.1

7.6.2 Fitness function

Candidate triples can be generated in a wide manner of ways, that we discuss in more detail in the following chapter. However, it is clear that the number of possible triples, in terms of combinations of entities and relations, is generally orders of magnitude greater than the amount of triples present in any given KG. Therefore, it is possible to generate very large sets of candidate triples that are not yet in a KG.

As a consequence, it is generally desirable to reduce these sets of candidate triples in a manner that maximizes the preservation of triples with a higher chance to be correct [19, 128]. To formalize this idea, we introduce the concept of fitness function, which assesses the quality of a set of candidate triples in terms of its size and how many correct triples it contains.

Definition 12. ***Fitness function:*** *Let $\mathcal{KG} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ be a Knowledge Graph, let $\mathcal{C}$ be a set of candidates and let $\mathcal{C}'$ be a set of filtered candidates, with $\mathcal{C}' \subseteq \mathcal{C}$. We define fitness as a function $fitness(\mathcal{KG}, \mathcal{C}, \mathcal{C}') \rightarrow \mathbb{R}$ that assigns a score to the filtered set of candidates, with respect to the original set of candidates and KG.*

Since there are many ways to achieve this, we introduce specific instances of fitness functions in the context of KG completion in Chapter 8.

7.7 Candidate filtering

As previously discussed, it is generally necessary to reduce the size of a set of candidate triples, in order to better assess the remaining candidates for their inclusion in a KG. To achieve this, this section introduces the concept of criterion, and then builds upon it to present the definition of a candidate-filtering rule.

7.7.1 Criterion

A criterion is an atomic element in candidate filtering. Given a candidate triple in the context of a particular Knowledge Graph, a criterion assigns a binary `False/True` label to the candidate, denoting whether it should be discarded immediately (`False`), or tentatively accepted and evaluated more carefully (`True`):

Definition 13. ***Criterion:*** *Let $\mathcal{KG} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ be a Knowledge Graph and let the triple $T = (s, r, t)$ be a candidate for $\mathcal{KG}$. We define a criterion as a function $cr(\mathcal{KG}, T) \rightarrow \{False, True\}$ that assigns a binary label to a candidate triple in the context of a KG.*

A given criterion defines a certain method to determine if a candidate triple is likely to be correct or not. For example, a possible criterion would be to accept all candidate triples in which the distance between its two entities is lower than a threshold.

7.7.2 Rule

In order to express more complex manners of filtering candidate triples, we combine several criteria into a rule. A candidate-filtering rule also produces a binary output for a candidate, in this case, by evaluating multiple criteria and combining their results using conjunctions and disjunctions:

Definition 14. ***Rule:*** *Let $\mathcal{KG} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ be a Knowledge Graph, let the triple $T = (s, r, t)$ be a candidate for $\mathcal{KG}$, and let cr_1, cr_2, ..., cr_n be a number of criteria. We define a rule as a function $rule(\mathcal{KG}, T) \rightarrow \{False, True\}$ resulting of the conjunction and/or disjunction of several criteria, i.e., $cr_1\ (\wedge|\vee)\ cr_2\ (\wedge|\vee)\ \ldots\ (\wedge|\vee)\ cr_n$.*

7.8 Graph-based features

To numerically characterize a triple, we propose a set of graph-based features that takes neighborhood subgraphs, reachable entities and paths into account. Due to the possibly large number of different features that can exist, we also introduce the concept

of feature groups. Each group can be parameterized to obtain a specific feature, which we call an instance of the feature group.

7.8.1 Feature

A feature is the simplest way to characterize a triple in the context of the Knowledge Graph it belongs to, by assigning a real number to it according to some operation:

Definition 15. ***Feature:*** *Let $\mathcal{KG} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ be a Knowledge Graph. We define a feature f as a function $f : \mathcal{T} \to \mathbb{R}$ that assigns a real number to a triple.*

For example, a feature f may convert a triple into the number of entities in the neighborhood subgraph of size 2 of the source entity, i.e., $f : (s, r, t) \mapsto |\mathcal{E}_s^2|$.

7.8.2 Feature group

Features can have an infinite number of small variations. In our previous example, we mentioned that a feature may leverage the neighborhood subgraph of size 2, but one may conceivably use any subgraph size that is considered appropriate. To be able to express these variations in a more concise way, we define a feature group as follows:

Definition 16. ***Feature group:*** *Let $\mathcal{KG} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ be a Knowledge Graph. We define a feature group f_n as a function $f_n : \mathcal{X} \to (\mathcal{T} \to \mathbb{R})$ that receives a set of parameters $\mathcal{X}$ and returns a feature.*

For example, a feature group f_0 may return a feature that converts a triple into the number of entities in the neighborhood subgraph of size n of the source entity, i.e., $f_0(n) = f : (s, r, t) \mapsto |\mathcal{E}_s^n|$, where n is a parameter of the feature group. Thus, $f_0(2) : (s, r, t) \mapsto |\mathcal{E}_s^2|$, which is the feature shown in the previous example. Consequently, feature groups allow us to represent a set of very similar features in a more compact way, where the only distinction between said features is a given set of parameters.

7.9 Summary

In this chapter, we have described the conceptual framework our proposal relies on. We have defined Knowledge Graphs, triples, paths, distance in a KG and reachability. Furthermore, we have introduced neighborhood subgraphs, candidate triples and fitness functions, as well as candidate-filtering criteria and rules. Finally, we have described graph-based features and their groups.

Chapter 8

CHAI: Our candidate filtering proposal

"I say let the world go to hell, but I should always have my tea."

— *Notes from Underground*, Fyodor Dostoyevsky

The first step towards completing a Knowledge Graph is narrowing down an initial set of theoretically potential candidates into a smaller subset that still retains most of the promising ones. This chapter introduces CHAI, our proposal for filtering candidate triples, and it is structured as follows: Section 8.1 introduces the chapter, Section 8.2 explains the criteria that CHAI uses, as well as the algorithm that it follows to create rules from them, Section 8.3 discusses its software architecture, Section 8.4 presents our experimental validation of CHAI and the conclusions we draw from it, Section 8.5 delves into the practical limitations of CHAI, and Section 8.6 concludes the chapter.

8.1 Introduction

In this chapter we introduce CHAI [19], our method for generating rules that are able to filter candidate triples in the context of a KG completion process by combining a number of criteria in such a way that it optimizes a given fitness function. CHAI works by producing rules that can be applied on the initial set of candidates and produce a reduced set that contains only the promising candidate triples. Then, this set can be passed on to any fact checking technique to check the correctness of each promising candidate and identify correct triples that complete the KG. The rules produced by CHAI are based on different criteria that take the internal features of the KG into account, such as the domains and ranges of every relation in the KG, in addition to the distances between its entities.

Additionally, we evaluate CHAI on a number of different Knowledge Graphs, and we show that it is able to achieve a good performance when dealing with all relationships in every KG under study, demonstrating that it is a generic and effective method, suitable for web-scale contexts.

Through this chapter, we continue to follow the running example of a Knowledge Graph that was introduced in Section 7.1.

8.2 Our proposal

When completing a Knowledge Graph, there is a very large number of possible triples that are not currently in said KG and that may represent correct knowledge. In practice, this number is generally large enough to prohibit an individual evaluation of every such triple.

For instance, let us consider a relatively small KG, with 10,000 entities and 20 possible relations. In the absence of further restrictions,[1] the theoretical maximum amount of different triples that could be present in this KG would be the Cartesian product of all possible pairs of entities and all relations, resulting in $2 \cdot 10^9$ combinations. This amount is hardly tractable through conventional means, and it only keeps scaling exponentially as the Knowledge Graph grows in size.

To overcome this issue, we devised CHAI, a technique for candidate filtering at scale. Given a Knowledge Graph, CHAI examines it and produces a set of candidate triples big enough to include most plausible knowledge, but small enough to allow other techniques further down the KG completion workflow to handle it. We achieve this by defining a reduced set of criteria, in which each criterion is responsible for

[1] Some Knowledge Graphs, such as CS-KG [37] or DBpedia [79], enforce type restrictions for the entities in a triple through an ontology. While this means that many Cartesian combinations of entities and relations are no longer valid, the result is usually still in the same order of magnitude.

filtering out implausible triples according to a heuristic. To achieve greater expressivity and to be able to represent more complex restrictions, CHAI progressively combines these criteria into rules of a fixed format, and assesses the quality of the rule after every step. These rules are produced on a per-relation basis, and thus CHAI is able to adapt them to the particularities of every relation in a Knowledge Graph.

In the following subsections, we describe these elements in detail, as well as the algorithm that is used to produce such rules.

8.2.1 Proposed criteria and rules

We propose a set of criteria for filtering candidates for a given Knowledge Graph $(\mathcal{E}, \mathcal{R}, \mathcal{T})$, where every criterion defines a heuristic to quickly reject a triple if it is considered implausible, doing so with a reduced computational cost.

Each criterion, thus, assigns a binary label to a candidate triple, denoting whether the candidate is rejected or accepted. A negative label means that, according to the criterion, the triple should be discarded immediately, while a positive label indicates that the triple may be interesting and should be allowed to continue further down the KG completion workflow. Each criterion is devised following a different approach, and therefore the sets of candidates that are allowed by each of them are relatively disjoint, although they might overlap to some extent. We further discuss the derived implications of this fact in Section 8.5.

These criteria, and their associated rationales, are as follow:

Criterion 1. Existing source entity and relation: Let (s, r, t) be a candidate triple. This criterion accepts the candidate if there exists a triple in $\mathcal{T}$ with the same source entity and relation as the candidate. We denote this criterion as:

$$exists_{\mathcal{KG}}((s, r, t)) \Leftrightarrow \exists\, e \in \mathcal{E} \mid (s, r, e) \in \mathcal{T}$$

This criterion was devised as a response to the observation that many Knowledge Graphs do not have an ontology that restricts which entity types are allowed to be combined through a given relation. As a consequence, it is highly likely that a random combination of a source entity and a relation will be nonsensical: in the running example shown in Figure 7.1, a random combination of source entity and relation may result in a real-life person having a movie prequel, or a book writing another book. It follows that allowing only candidate triples whose source and relation have already been observed before in correct triples will immediately discard many of such nonsensical instances.

Criterion 2. Target is in the domain of a relation $rel \in \mathcal{R}$: Let (s, r, t) be a candidate triple. This criterion accepts the candidate if its target entity appears at

least once as the source of an existing triple that has *rel* as its relation. We denote this criterion as:

$$dom_{\mathcal{KG},rel}((s,r,t)) \Leftrightarrow \exists\ e \in \mathcal{E} \mid (t,rel,e) \in \mathcal{T}$$

Contrary to the previous criterion, which operates solely on a candidate triple, this one also requires a relation to be specified. Given that most impossible combinations of source and relation will be rejected by the former criterion, this one focuses on restricting which entities are allowed to appear on the right side of a candidate triple. Generally, acceptable target entities will have a certain type or belong to a union of types. However, type information is not always readily available.

This criterion aims to provide a similar constraint even if the KG lacks type information, by only accepting candidates whose target entities appear as a source for a given relation somewhere in the KG. This allows CHAI to leverage the implicit type restrictions that will be in place by the already existing, correct knowledge: for example, it may deduce that entities that appear as the source of the relation *plays* are actors, even if this knowledge is not explicitly laid out.

Criterion 3. Target is in the range of a relation $rel \in \mathcal{R}$: Let (s,r,t) be a candidate triple. This criterion accepts the candidate if its target entity appears at least once as the target of an existing triple that has *rel* as its relation. We denote this criterion as:

$$ran_{\mathcal{KG},rel}((s,r,t)) \Leftrightarrow \exists\ e \in \mathcal{E} \mid (e,rel,t) \in \mathcal{T}$$

This criterion complements the previous one by only accepting candidates whose target entity is in the range of an existing relation in the KG. Again, this leverages implicit type information present in the KG to further remove non-plausible candidates.

Following the example introduced in Figure 7.1, thanks to this criterion, one could establish that any candidate triple for the relation *starred_in* should have a target entity that is also a target of the relation *has_prequel*, due to the fact that the range of both relations is generally comprised of movies. Thus, one can immediately discard the potential candidate *(Emma Watson, starred_in, Daniel Radcliffe)* because the entity *Daniel Radcliffe* never appears as a target for the relation *has_prequel*.

Entities are within distance i: Let (s,r,t) be a triple in $\mathcal{T}$. This criterion selects all candidates whose source and target entities have a distance between them that is at most i:

$$dist_{\mathcal{KG},i}((s,r,t)) \Leftrightarrow dist(\mathcal{KG},s,t) \leq i$$

Finally, this criterion covers the assumption that a good candidate triple should be such that its source and target entities are close each other in the Knowledge Graph, which has been repeatedly shown correct by related literature [11, 12, 20, 42, 74, 102].

While the previously introduced criteria could be useful on their own, it seems reasonable that a more complex combination of them could achieve both a higher expressivity and a more satisfactory candidate filtering performance. To achieve this, CHAI combines them into candidate filtering rules of the following format, where c_i are criteria other than $exists_{\mathcal{KG}}$:

$$exists_{\mathcal{KG}} \ \wedge (c_1 \vee c_2 \vee \ldots \vee c_n)$$

By enforcing the $exists_{\mathcal{KG}}$ criterion on all candidate triples, we can make sure that the resulting set of candidates has a lower number of incorrect or noisy candidates, as all of them have a combination of source entity and relation that already exists in the original KG while still allowing all possible target entities. In addition, the disjunction of criteria present in the rule allows for more flexibility: longer rules with more criteria are less strict, and thus produce more candidates by combining different criteria. The following subsection illustrates the process of creating such rules.

8.2.2 Algorithm

The algorithm that we propose for generating rules for candidate filtering is shown in Algorithm 1, and further described in the following. It receives the set of candidates to be filtered, the original KG in the form of a training and a testing split and a relation as input, and it outputs the generated rule for the relation.

First, the input set of candidates is narrowed down to only those that include the relation for which CHAI is being applied. This serves a dual purpose. On the one hand, since rules are produced and applied on a per-relation basis, this immediately discards any candidates that contain a different relation, which would not have been allowed anyway. On the other hand, this allows CHAI to only take into account the triples in which the desired relation is present, resulting in a more specialized rule.

Subsequently, a rule that contains only the $exists_{\mathcal{KG}}$ criterion is generated, and the set of candidates that results from applying it is obtained, which will be further refined by adding more criteria to the rule. By definition of $exists_{\mathcal{KG}}$, this will result in a set of triples whose combination of source and relation are already present somewhere in the KG, and all possible entities as targets.

Then, a set of criteria is instantiated, which contains the *dom* and *ran* criteria for every possible relation in the KG, as well as the *distance* criterion for up to a certain maximum distance. These criteria are the ones that will be used for building the rule, however, not every criterion in the set has necessarily to be added to the rule.

Following the previous step, a fitness value is computed for each criterion, by applying a certain fitness function on the set of candidates that are selected by that

Algorithm 1: CHAI

Input: $\mathcal{KG}_{trn} = (\mathcal{E}, \mathcal{R}, \mathcal{T}_{trn})$: Training split of the KG
$\mathcal{KG}_{tst} = (\mathcal{E}, \mathcal{R}, \mathcal{T}_{tst})$: Testing split of the KG
candidates : Set of potential candidates to be filtered
rel : Selected relation in $\mathcal{R}$
fitness : Fitness function
N : Maximum distance value for distance-based criteria
θ : Fitness threshold value

Output: *rule* : Generated rule

```
1  function CHAI(KG_trn, KG_tst, candidates, rel, fitness, N, θ)
2      // Select the candidates in which the relation rel appears
3      rc ← {(s, r, t) ∈ candidates | r = rel}
4      // Initialize the rule to initially contain only exists_{KG_trn}
5      rule ← exists_{KG_trn}
6      // Obtain a set of initially filtered candidates by applying
7      // exists_{KG_trn}
8      fc ← apply exists_{KG_trn} to rc
9      // Add all possible criteria to the set of available criteria
10     // using the training split of the KG
11     criteria ← ∅
12     forall r ∈ R, i ∈ [1..N] do
13         criteria ← criteria ∪ {dom_{KG_trn,r}, ran_{KG_trn,r}, dist_{KG_trn,i}}
14     // Sort all criteria by the fitness value obtained on the set of
15     // filtered candidates they generate, using the testing split
16     criteria ← sort criteria by fitness(KG_tst, fc, apply criteria to fc)
17     forall criterion ∈ criteria do
18         // Apply the rule to obtain a set of filtered candidates
19         selected_candidates ← apply rule to fc
20         // Compute the fitness value of the previous set
21         // using the testing split
22         if fitness(KG_tst, fc, selected_candidates) < θ then
23             // Add current criterion if the threshold is not met
24             rule ← add criterion to rule
25     return rule
```

criterion. The previous set of criteria is then sorted in descending order of the fitness value that is obtained in this manner.

Finally, these ordered criteria are added in an iterative fashion to the rule under generation, starting with the one that obtains the highest fitness value. Every time a criterion is added to the rule, the resulting set of filtered candidates produced by it is computed, and the fitness value associated with the rule is updated. This process is repeated until the fitness value exceeds a given threshold or the set of available criteria is depleted, at which point no more criteria will be added to the rule. Once

Rule	Fitness value	Meets threshold?
$exists_{\mathcal{KG}} \wedge$	-	-
$(\, ran_{\mathcal{KG},created} \vee$	0.80	No
$dom_{\mathcal{KG},appears\_in} \vee$	0.92	No
$dist_{\mathcal{KG},2}\,)$	0.96	Yes

Table 8.1: An example rule being built for the relation *plays*

this process ends, the generated rule is returned.

In Table 8.1, we present an example on the process of generating a rule for the relation *plays*, with a threshold $\theta = 0.95$. Through the process, different criteria are iteratively added to the rule, and the fitness value is included for every step. Once the fitness value meets or exceeds the threshold, the process ends and the rule is returned. In this case, selecting candidate triples whose target entities represent people would provide a good result.

Since an integral KG completion process involves every relation in the KG, CHAI should be applied once for each relation in the KG, in order to produce the complete set of rules and suitable candidates for KG Completion. This results in a total set whose size is significantly smaller than that of the input set of candidates, while still containing as many suitable candidates as possible.

8.3 Software Architecture

The classes that comprise the architecture of CHAI are shown in Figure 8.1, while its workflow is described in Figure 8.2. In the following, we further describe the architecture of CHAI:

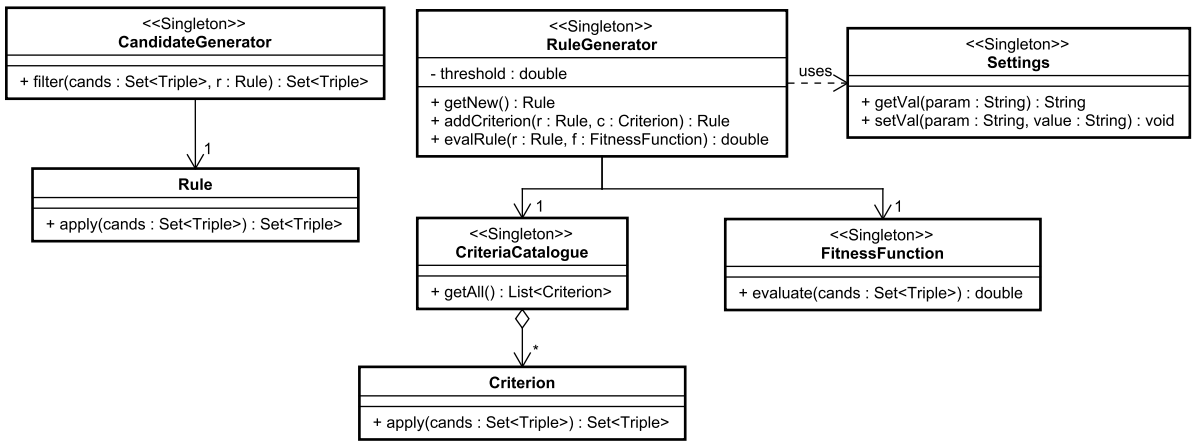


Figure 8.1: Architecture of CHAI

Class *CriteriaCatalogue* contains all the available criteria that can be used to build candidate filtering rules, which can be obtained using the method *getAll*. Criteria are

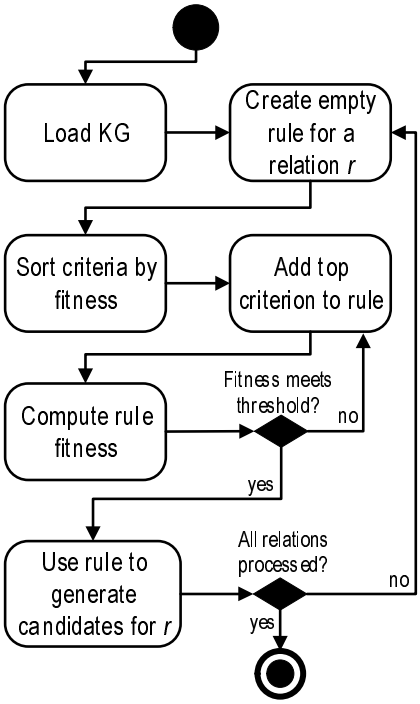


Figure 8.2: Workflow of CHAI

implemented using the class *Criterion*. The method *apply* of this class receives a set of candidate triples, and produces a smaller, filtered set of candidates that meet the criterion in question. For example, a criterion may determine that a triple is a valid candidate if its source and target entities are less than three hops apart in the KG.

Class *FitnessFunction* is used to implement a given fitness function, which gives a numerical score to a given set of candidate triples using the *evaluate* method. Some possible examples of fitness functions are the relative number of total candidate triples against the number of triples in the KG, or the percentage of candidate triples that are present in the validation split of a KG and thus more likely to be correct.

The *RuleGenerator* class is responsible for generating a candidate filtering rule for a given relation. A rule, as described in Section 8.2, is a conjunction of criteria that determine whether a given triple is a valid candidate or not.

First, a new rule is produced using the *getNew* method. Then, a series of criteria are iteratively added to it in order to progressively construct it. This is done by sorting all the available criteria according to the fitness function, and then using the *addCriterion* method to add the top criterion. The fitness of the resulting rule is subsequently evaluated using the *evalRule* method, and this process is repeated until *evalRule* returns a fitness value that meets a certain threshold.

Finally, the *Rule* instance produced by the *RuleGenerator* can be used to filter candidate triples in a KG, using its *apply* method. The *CandidateGenerator* class handles this process, applying a given rule to the desired set of candidates and returning the set of triples that are considered promising candidates according to the rule.

8.3.1 Design and performance considerations

The software architecture of CHAI has been devised using a number of common software patterns. First, given that only one instance of most classes is needed at runtime, all such classes have been designed as singletons, to statically guarantee a smaller memory footprint. However, there are two places where a singleton cannot be used. One of them is for class Rule, since the execution of CHAI will result in the generation of one rule per relation in the KG. Another similar case is class Criterion, because it is clear that multiple different criteria should be instantiated at runtime.

To facilitate the inclusion of new criteria in the future, they are accessed through a CriteriaCatalogue class, which takes care of detecting all available criteria in the system and loading them at runtime. This way, no further changes in the system are needed if additional criteria need to be added.

Finally, the way in which candidates are obtained from a rule enjoys a significant optimization. Rather than following a filtering approach, where the entire set of possible candidates needs to be instantiated and then pruned, we follow a generative approach. The CandidateGenerator class is able to inspect the conditions of a rule, and then generate exclusively those candidate triples that would have been allowed by it. While the final set of filtered candidates can be proved to be the same, this results in significant runtime and memory usage improvements.

8.4 Evaluation

In this section we present the experimental results that confirm that CHAI is effective in practice. First, we introduce the experimental setting. Then, we present the results of applying CHAI on several well-known Knowledge Graphs, comparing them against those of a state-of-the-art baseline technique by Shi and Weninger [130], which is, to the best of our knowledge, the only KG completion proposal that includes a well-defined way to filter candidate triples and experimental results on this regard. Finally, we discuss these results.

8.4.1 Setup and experimental data

We evaluated CHAI using a number of different Knowledge Graphs that are openly available and commonly used for the task of KG completion: FB13, WN11 [134], WN18 [18] (which are subsets of Freebase [14] and Wordnet [89], respectively), a subset of NELL introduced by Gardner and Mitchell [47], and EPSRC[2], which contains information about the grants provided by the Engineering and Physical Sciences Research Council of the United Kingdom. All of these Knowledge Graphs were

[2] `http://epsrc.rkbexplorer.com`

obtained from the publicly available AYNEC-DataGen tool [6], and an overview of their metadata can be found in Table 8.2. We used CHAI to generate rules for every relation in every KG, except for the case of NELL, in which we focused on the same subset of 10 relations as Gardner and Mitchell [47] due to the high number of total relations. All experiments were conducted on a computer with 32GB of RAM and an Intel Core i9-9900K CPU.

KG	Training triples	Test triples	Relations
FB13	285,208	78,490	13
WN18	117,160	58,564	18
NELL	201,870	13,491	519 (10)
EPSRC	341,372	85,337	20

Table 8.2: Overview of the KGs used for evaluating CHAI

The results of the approach followed by Shi and Weninger [130] were used as a baseline. Their proposal consists in generating candidate triples by altering the target entities of the triples already present in the KG, and replacing them by all entities that can be found in the range of the relation present in the triple. This is equivalent to applying only the $ran_{\mathcal{KG},r}$ criterion, where r is the relation for which CHAI is being applied.

8.4.2 Evaluation parameters

To conduct our experiments, we set the distance threshold N for the *distance* criterion to 4. This value was chosen empirically, aiming to allow for a threshold as high as possible while still being reasonable in terms of computation time. Additionally, these distances were computed on a partially undirected version of the KGs. This was done to fully exploit the highly relational nature of KGs, while still not allowing paths that would be connected by means of entities with a very high in-degree such as genders or nationalities.

The training and testing splits of the KGs were already provided by the KGs that we used, and thus we provided CHAI with these splits as is. We evaluated CHAI using a fitness function that combines reduction rate (rr) and coverage using their harmonic mean, as shown in Eq. 8.1. The θ threshold value required by the algorithm was set to 0.99, so as to allow CHAI to find highly satisfactory rules, and to study the evolution of the coverage and reduction rate of said rules if they keep growing in size without meeting the threshold. Formally, they are defined as follows:

Let $\mathcal{C}$ be a candidates set, and $\mathcal{C}' \subseteq \mathcal{C}$ a set of filtered candidates:

$$fitness(\mathcal{KG}, \mathcal{C}, \mathcal{C}') = \frac{2 \cdot rr(\mathcal{C}, \mathcal{C}') \cdot coverage(\mathcal{KG}, \mathcal{C}')}{rr(\mathcal{C}, \mathcal{C}') + coverage(\mathcal{KG}, \mathcal{C}')}, \text{ where} \tag{8.1}$$

$$rr(\mathcal{C}, \mathcal{C}') = 1 - \frac{|\mathcal{C}'|}{|\mathcal{C}|}$$

$$coverage(\mathcal{KG} = (\mathcal{E}, \mathcal{R}, \mathcal{T}), \mathcal{C}') = \frac{|\mathcal{C}' \cap \mathcal{T}|}{|\mathcal{T}|}$$

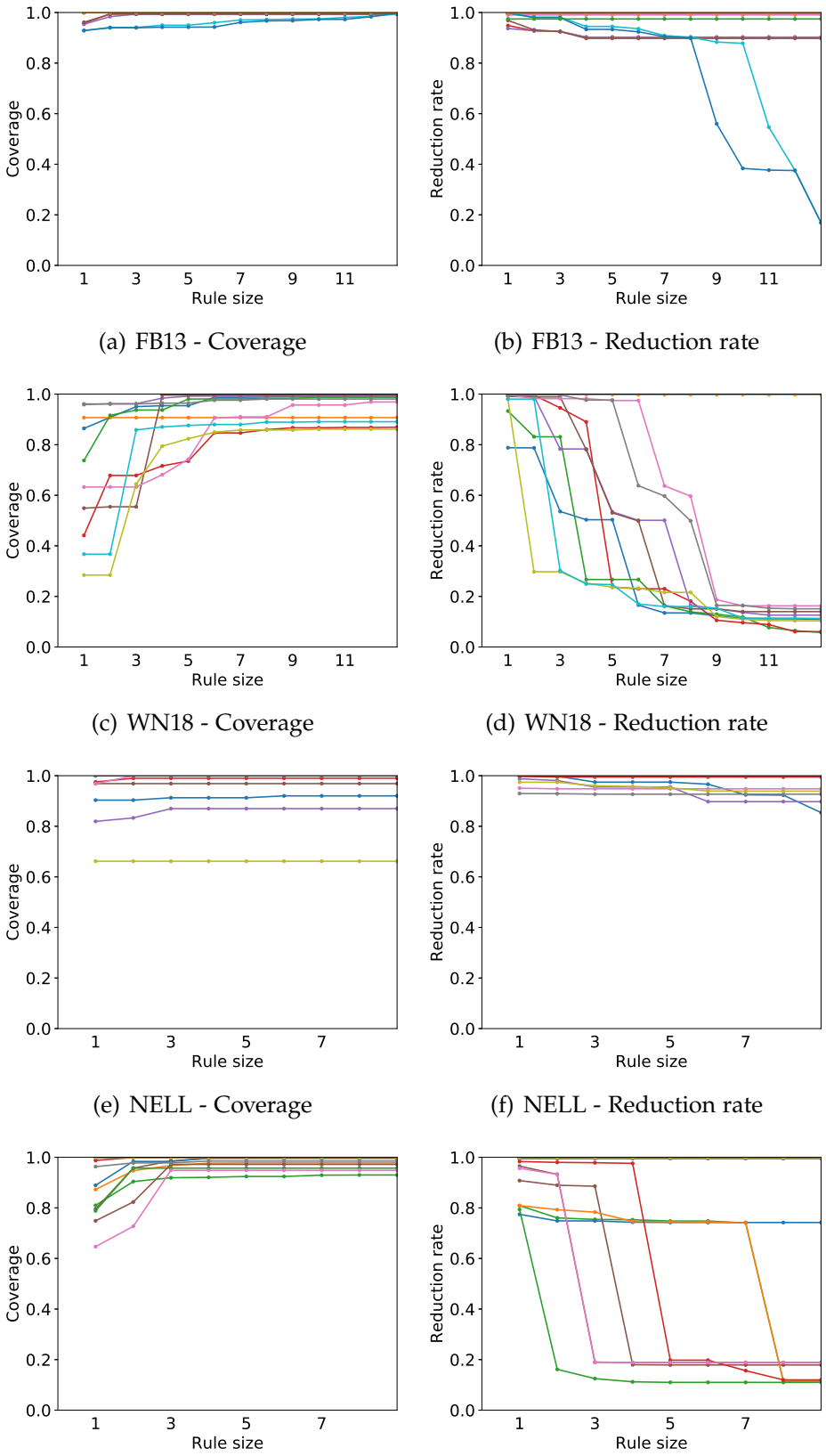
This fitness function was devised under the following rationale: focusing only on coverage would result in very long rules that allow as many candidates as possible, however, this would not be desirable as we aim to reduce the size of the set of candidates, to avoid having to evaluate low-quality candidates. Conversely, focusing only on reduction rate would yield very short (and thus more restrictive) rules. As a consequence, this fitness function achieves a compromise between reduction rate and coverage, and allows for more flexibility in the lengths of the rules in contrast to focusing only on one objective. To illustrate this difference, we have also tested CHAI using two alternative fitness functions: only coverage, and only reduction rate. The results achieved by every fitness function are shown in Figure 8.4.

8.4.3 Results and discussion

In the following, we present the results achieved by CHAI on the KGs under evaluation and the conclusions we draw from them.

Figure 8.3 reports on the evolution of the coverage and reduction rate for all KGs under study as rules grow in size, where each line represents a different relation; while Figure 8.4 display the values for the coverage and reduction rate for every iteration in all Knowledge Graphs as points in a 2-dimensional space. Finally, Table 8.3 provides an overview on the average maximum coverage and reduction rate that CHAI achieves for the relations in all KGs under study. This Table also includes the average coverage and reduction rate values achieved by the proposal of Shi and Weninger [130], which was denoted as "baseline" for brevity.

These results allow us to distinguish between two types of relations: those for which a high coverage value is obtained with a very short rule, and those for which the coverage starts at a lower value and increments as rules grow in size, as shown in Figure 8.3. We consider the former type of relations to be categorical, as they have a range of possible target entities that is relatively small: for example, the entities that are targets for the relation *location* are unlikely to appear as the target for any other relation, and thus using the entities that are targets for *location* as possible candidates for locations yields a very good result. On the other hand, relations that are non-categorical have a much wider range of possible candidates: in the case of the relation



(a) FB13 - Coverage

(b) FB13 - Reduction rate

(c) WN18 - Coverage

(d) WN18 - Reduction rate

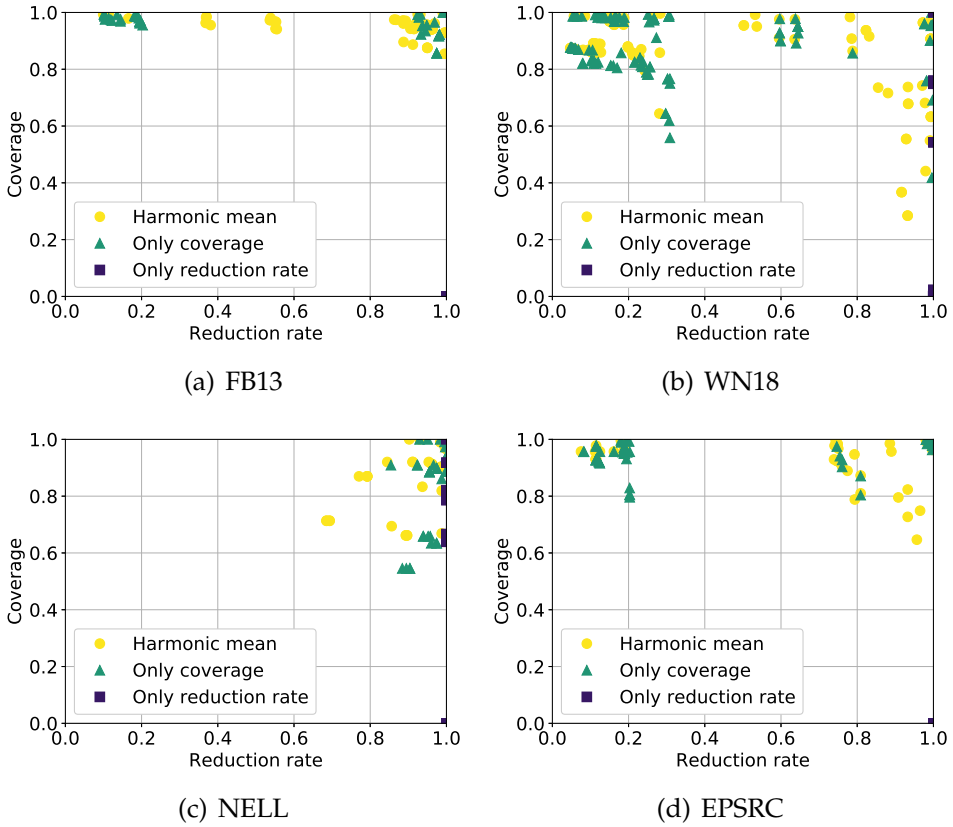
(e) NELL - Coverage

(f) NELL - Reduction rate

(g) EPSRC - Coverage

(h) EPSRC - Reduction rate

Figure 8.3: Evolution of the coverage (left) and reduction rate (right) values



(a) FB13 (b) WN18

(c) NELL (d) EPSRC

Figure 8.4: Reduction rate (x) and coverage (y) for different fitness functions

children, an entity may produce a good candidate even if it does not appear as the target of *children* (for example, they may appear in the relation *sibling*).

This conclusion is reinforced by the results shown in Figure 8.4, where there are groups of iterations in the top-right corner (the area of both high reduction rate and high coverage), which are obtained for categorical relations with a small subset of possible targets, while a different group of iterations show more scattered results in the top area, denoting that in order to achieve a high coverage, a bigger set of candidates must be used (non-categorical relations). Additionally, the results shown in this Figure lead us to the conclusion that using only reduction rate as the fitness function results in a very poor coverage, as the algorithm stops after having selected only one criterion that allows a very small number of candidates. On the contrary, using only coverage as the fitness function provides better results, but with a clear tendency towards prioritizing coverage at the expense of a lower reduction rate, while using the harmonic mean yields more balanced results.

In the case of non-categorical relations, there exists a trade-off between coverage and reduction rate. This is to be expected, since rules are disjunctions of criteria and thus rules that comprise more criteria are more likely to filter out less candidates, increasing coverage but decreasing the reduction rate. In these cases, it is up to the user to decide whether they are interested in achieving a very high coverage with a

KG	Avg. max. coverage (CHAI)	Avg. coverage (baseline)	Avg. max. RR (CHAI)	Avg. RR (baseline)
FB13	**0.92** (0.76-1.00)	**0.78** (0.58-0.99)	**0.91** (0.76-1.00)	**0.91** (0.76-1.00)
WN18	**0.94** (0.89-0.99)	**0.49** (0.26-0.72)	**0.97** (0.93-1.00)	**0.93** (0.85-1.00)
NELL	**0.89** (0.78-1.00)	**0.53** (0.26-0.80)	**0.97** (0.95-1.00)	**0.99** (0.99-1.00)
EPSRC	**0.99** (0.98-1.00)	**0.82** (0.68-0.97)	**0.95** (0.91-0.99)	**0.95** (0.92-0.99)

Table 8.3: Max. coverage and reduction rate values (avg and 95% conf.)

lower reduction rate, or a higher reduction rate with a usually lower coverage. For this kind of relations, distance-based criteria are generally useful: for example, one's parents or spouse are usually found within a short distance in the KG. In categorical relations, however, one is able to obtain both a high reduction rate and a high coverage, because the entities that are suitable targets for said relations can easily be found by analyzing the domains and ranges of the relation in question or other relations.

Regarding the reduction rate, it must be noted that it is computed with regard to the set of candidates that are selected by the *exists* criterion, and not with respect to the Cartesian product of all possible candidates. This results in reduction rates that are in all cases lower than those that would be obtained in the latter case, though more evenly spread between 0 and 1. Given that high reduction rates are still achieved, we do not consider this to be a problem, but rather an adequate way of measuring how many candidates are selected with respect to an initial set of filtered candidates, with many unlikely candidates already filtered out.

Finally, it is worth noting that CHAI consistently manages to achieve high coverage values for all Knowledge Graphs under study, as it can be seen in Figure 8.3. These values start to converge with rules composed of approximately five criteria, and thus we find that the criteria that are ranked higher by means of the proposed fitness function (Eq. 8.1) are indeed successful in allowing promising candidates to pass the filter. When comparing CHAI to the baseline approach proposed by Shi and Weninger [130], CHAI achieves much higher values of coverage while still being able to obtain similar reduction rates, as shown in Table 8.3. The values for CHAI shown in said Table refer to the average maximum values that can be achieved, and thus our proposal is more versatile as it allows the user to prioritize a higher coverage by using longer rules, or a higher reduction rate by using shorter rules. CHAI works well for all kinds of relations due to the versatility of the criteria it uses, while the proposal by Shi and Weninger [130] is not able to deal as effectively with non-categorical relations in terms of coverage, hindering its overall performance.

8.5 Limitations

While CHAI obtains satisfactory results, it is not without limitations. Perhaps the most important one would arise in the case of a KG with a very high number of total relations, because the amount of domain and range-based criteria would be equally high. In this case, the fitness function would have to be computed for every criterion in order to sort them by decreasing fitness value, resulting in a potentially high computational cost. Besides, CHAI may not work as well in very sparse KGs where all or most relations share the same entities in their domains and ranges, because the distance-based criterion would need a much higher threshold due to the sparsity of possible paths, and the domain and range-based ones would not prove useful.

8.6 Summary

In this chapter we have introduced CHAI, our proposal to filter candidate triples for Knowledge Graph completion. CHAI works by producing candidate filtering rules by combining a set of criteria in such a way that it optimizes a fitness function. Furthermore, CHAI produces a rule for every distinct relation in the KG, thus ensuring that the result of applying such rules is specifically tailored to the nature of every relation. Our experimental evaluation shows that CHAI is effective in practice, producing sets of candidate triples that are considerably smaller than other state-of-the-art approaches to this task, while still containing most of the candidates that could be considered promising.

Chapter 9

CAFE: Our triple classification proposal

"Coffee, the finest organic suspension ever devised."

— Star Trek: Voyager

When a set of candidate triples has been obtained, the next step in Knowledge Graph completion is to classify every triple in said set to determine if it represents correct knowledge or not and, if found to be correct, added back to the KG in order to enrich it. In this chapter we introduce CAFE, our proposal for candidate triple classification. This chapter is structured as follows: Section 9.1 introduces it, Section 9.2 presents the neighborhood-aware features that CAFE uses to discern between correct and incorrect triples, as well as the process it follows to do so, Section 9.3 discusses its internal architecture and the design choices behind it, Section 9.4 shows the experiments that we carried out to assess the effectivity of CAFE, as well as their results; Section 9.5 discusses its practical limitations; finally, Section 9.6 summarizes the chapter.

9.1 Introduction

This chapter presents CAFE [20], our proposal for triple classification. CAFE covers the second main step in Knowledge Graph completion: once an adequately small set of candidate triples has been obtained, every triple in it must be carefully evaluated to determine whether or not it represents correct knowledge and, if so, added to the KG to enrich and expand it.

CAFE works by transforming any possible triple into a numerical vector by using a novel set of neighborhood-aware features, which measure the correctness of a triple leveraging its context in a Knowledge Graph. By transforming all triples into vectors in this manner, CAFE trains and uses a binary classifier to discern between correct and incorrect triples.

We thoroughly evaluate CAFE using several well-known Knowledge Graphs, and our evaluation shows that it is able to achieve a high precision in challenging, real-world scenarios, thus allowing for a trustworthy KG completion process.

Over the course of this chapter, we continue to follow the running example introduced in Section 7.1.

9.2 Our proposal

As previously described, the second major step when completing a Knowledge Graph consists of analyzing the set of possible candidate triples, assessing which ones represent correct knowledge, and adding them back to the KG in order to augment it. In this step, it is important to achieve a high precision, in order to have a Knowledge Graph completion process that is as trustworthy as possible [128].

To carry out this process, we have devised CAFE, a technique that classifies candidate triples into correct or incorrect ones. Given a Knowledge Graph and a set of candidate triples, CAFE evaluates each one of them and assigns them a binary label, denoting whether it should be considered correct and added to the Knowledge Graph, or incorrect and discarded. CAFE does this by defining a set of neighborhood-aware features, which checks for shared neighborhoods at several distance levels and under certain conditions, under the assumption that the entities in correct triples usually have a higher degree of overlap in their neighborhoods. Then, using these features, each candidate triple is converted into a numerical vector. Finally, CAFE trains a number of neural classification models using these vectors to learn to separate correct triples from incorrect ones. In the following subsections, we describe the features and architecture of CAFE in detail.

9.2.1 Neighborhood-aware features

We propose a set of neighborhood-aware features that takes neighborhood subgraphs, reachable entities and paths into account. Due to the large number of possible variations of each feature, we present our feature set in terms of groups of features. Each group can be parameterized to obtain a specific feature, which we call an instance of the feature group.

For the sake of example, we illustrate a possible instance of every feature group and its value using the example triple *example* =*(Daniel Radcliffe, plays, Harry Potter)*, and the KG shown in Figure 7.1.

Feature group f_1: *Number of entities in the neighborhood subgraph of size n of the source entity in the triple.* Features in this group are computed as:

$$f_1(n) : (s, r, t) \mapsto |\mathcal{E}_s^n|$$

In the example shown in Figure 7.1, $f_1(2)$ applied to the *example* triple is $|\{$*Daniel Radcliffe, Harry Potter and the Goblet of Fire (movie), Harry Potter and the Prisoner of Azkaban (movie), Harry Potter and the Goblet of Fire (book), Harry Potter and the Prisoner of Azkaban (book)*$\}| = 5$.

Feature group f_2: *Number of entities in the neighborhood subgraph of size n of the target entity in the triple.* Features in this group are computed as:

$$f_2(n) : (s, r, t) \mapsto |\mathcal{E}_t^n|$$

In the example shown in Figure 7.1, $f_2(3)$ applied to the *example* triple is $|\{$*Harry Potter, J.K. Rowling, Harry Potter and the Goblet of Fire (book), Harry Potter and the Prisoner of Azkaban (book), Robin Ellacott, Hermione Granger*$\}| = 6$.

Feature group f_3: *Degree of N-path centrality of the source entity in the triple.* Features in this group are computed as:

$$f_3(n) : (s, r, t) \mapsto \frac{|\mathcal{E}_s^n|}{|\mathcal{E}| - 1}$$

In the example shown in Figure 7.1, $f_3(1)$ applied to the *example* triple is $|\{$*Harry Potter and the Goblet of Fire (movie), Harry Potter and the Prisoner of Azkaban (movie)*$\}|/(13-1) = 2/12 \approx 0.17$.

Feature group f_4: *Degree of N-path centrality of the target entity in the triple.* Features in this group are computed as:

$$f_4(n) : (s, r, t) \mapsto \frac{|\mathcal{E}_t^n|}{|\mathcal{E}| - 1}$$

In the example shown in Figure 7.1, $f_4(1)$ applied to the *example* triple is $|\{$*Harry Potter and the Goblet of Fire (book), Harry Potter and the Prisoner of Azkaban (book)*$\}|/(13-1) = 2/12 \approx 0.17$.

Feature group f_5: *Number of common entities between the neighborhood subgraph of size n of the source entity and the neighborhood subgraph of size m of the target entity in the triple.* Features in this group are computed as:

$$f_5(n,m) : (s,r,t) \mapsto |\mathcal{E}_s^n \cap \mathcal{E}_t^m|$$

In the example shown in Figure 7.1, $f_5(2, 3)$ applied to the *example* triple is $|\{$*Harry Potter and the Goblet of Fire (book), Harry Potter and the Prisoner of Azkaban (book)*$\}| = 2$.

Feature group f_6: *Jaccard index of similarity between the entities in the neighborhood subgraph of size n of the source entity and the neighborhood subgraph of size m of the target entity in the triple.* Features in this group are computed as:

$$f_6(n,m) : (s,r,t) \mapsto jaccard(\mathcal{E}_s^n, \mathcal{E}_t^m)$$

In the example shown in Figure 7.1, $f_6(2, 3)$ applied to the *example* triple is $|\{$*Harry Potter and the Goblet of Fire (book), Harry Potter and the Prisoner of Azkaban (book)*$\}|$ / $|\{$*Daniel Radcliffe, Harry Potter and the Goblet of Fire (movie), Harry Potter and the Prisoner of Azkaban (movie), Harry Potter and the Goblet of Fire (book), Harry Potter and the Prisoner of Azkaban (book), Harry Potter, J.K. Rowling, Robin Ellacott, Hermione Granger*$\}| = 2 \ / \ 9 = 0.22$.

Feature group f_7: *Adamic-Adar index of closeness between the neighborhood subgraphs of size n of the source and target entities in the triple.* Features in this group are computed as:

$$f_7(n) : (s,r,t) \mapsto \sum_{e \in \mathcal{E}_s^n \cap \mathcal{E}_t^n} \frac{1}{log|\mathcal{E}_e^n|}$$

In the example shown in Figure 7.1, $f_7(2)$ applied to the *example* triple is $\frac{1}{log|\mathcal{E}^2_{\textit{Harry Potter and the Goblet of Fire (book)}}|} = \frac{1}{log|3|} \approx 2.09$.

Feature group f_8: *Number of reachable entities through the relation r at distance n from the source entity in the triple.* Features in this group are computed as:

$$f_8(r,n) : (s,r,t) \mapsto |Reachable(s,r,n)|$$

In the example shown in Figure 7.1, $f_8(hasPrequel, 2)$ applied to the *example* triple is $|\{$*Daniel Radcliffe, Harry Potter and the Prisoner of Azkaban (movie)*$\}| = 2$.

Feature group f_9: *Number of reachable entities through the relation r at distance*

n ***from the target entity in the triple.*** Features in this group are computed as:

$$f_9(r,n) : (s,r,t) \mapsto |Reachable(t,r,n)|$$

In the example shown in Figure 7.1, $f_9(created,\ 2)$ applied to the *example* triple is $|\{Harry\ Potter,Hermione\ Granger,\ Robin\ Ellacott\}| = 3$.

Feature group f_{10}: *Number of common reachable entities through the relation r from the source entity at distance n and from the target entity at distance m.* Features in this group are computed as:

$$f_{10}(r,n,m) : (s,r,t) \mapsto$$
$$|Reachable(s,r,n) \cap Reachable(t,r,m)|$$

In the example shown in Figure 7.1, $f_{10}(created,\ 2,\ 3)$ applied to the *example* triple is $|\{Daniel\ Radcliffe\} \cap \{Harry\ Potter,\ Hermione\ Granger,\ Robin\ Ellacott\}| = 0$.

Feature group f_{11}: *Jaccard index of similarity between the reachable entities through the relation r from the source entity at distance n, and those reachable through the relation r from the target entity at distance m.* Features in this group are computed as:

$$f_{11}(r,n,m) : (s,r,t) \mapsto$$
$$jaccard(Reachable(s,r,n), Reachable(t,r,m))$$

In the example shown in Figure 7.1, $f_{11}(created,\ 2,\ 3)$ applied to the *example* triple is $|\emptyset|\ /\ |\{Harry\ Potter,\ Hermione\ Granger,\ Robin\ Ellacott\}| = 0\ /\ 3 = 0$.

Feature group f_{12}: *Number of distinct paths of length n between the source and the target entity in the triple, using relations $r_1, \ldots, r_n$.* Features in this group are computed as:

$$f_{12}(n, r_1, \ldots, r_n) : (s,r,t) \mapsto |\mathcal{P}(s,t,r_1,\ldots,r_n)|$$

In the example shown in Figure 7.1, $f_{12}(4,\ starred\_in,\ based\_on,\ writer,\ created)$ applied to the *example* triple is 1, as there is one path of length 4 between the entities *Daniel Radcliffe* and *Harry Potter* that matches the given relations.

The rationale behind this set of features is manifold. Regarding f_1 and f_2, knowing the size of the neighborhood of an entity can be helpful to determine whether said neighborhood encompasses relevant information. For instance, our hypothesis is that very large neighborhoods tend to contain a higher amount of unrelated information. The same idea is leveraged in f_3 and f_4, which provide normalized indices of centrality with respect to the total amount of entities in the KG. Following the previous reasoning, we hypothesize that entities with large indices of centrality (i.e. highly connected to other entities) will yield less useful information. Meanwhile, feature groups f_5, f_6 and f_7 measure the degree of overlap that exists in the neighborhoods of the two entities,

both in absolute and in relative terms, under the assumption that correct triples have a higher degree of overlap between the neighborhoods of their entities.

The previously discussed feature groups do not consider the specific relations involved. However, we deem it reasonable to assume that some relations may be more useful than others to determine whether a triple is correct, depending on their specific semantics. For example, having one or more children in common can be an indication of a marriage, while having the same nationality is not. In order to exploit this fine-grained information, feature groups f_8, f_9, f_{10} and f_{11} are similar to the previously discussed groups but restrict themselves to only one relation. These groups are computed for every relation in a KG.

Finally, feature group f_{12} allows CAFE to find the number of paths that exist between two entities for any given relations. It is our intuition that correct triples have more alternative paths between the two entities they contain than false ones.

9.2.2 Workflow

Our proposal, CAFE, receives a KG and a set of relations from that KG as input, and outputs a classification model for each of the provided relations. These models are able to determine if a given triple that represents an instance of the relation is correct and should belong to the KG. Its workflow is depicted in Figure 9.1 and, in the following, we describe each of its steps.

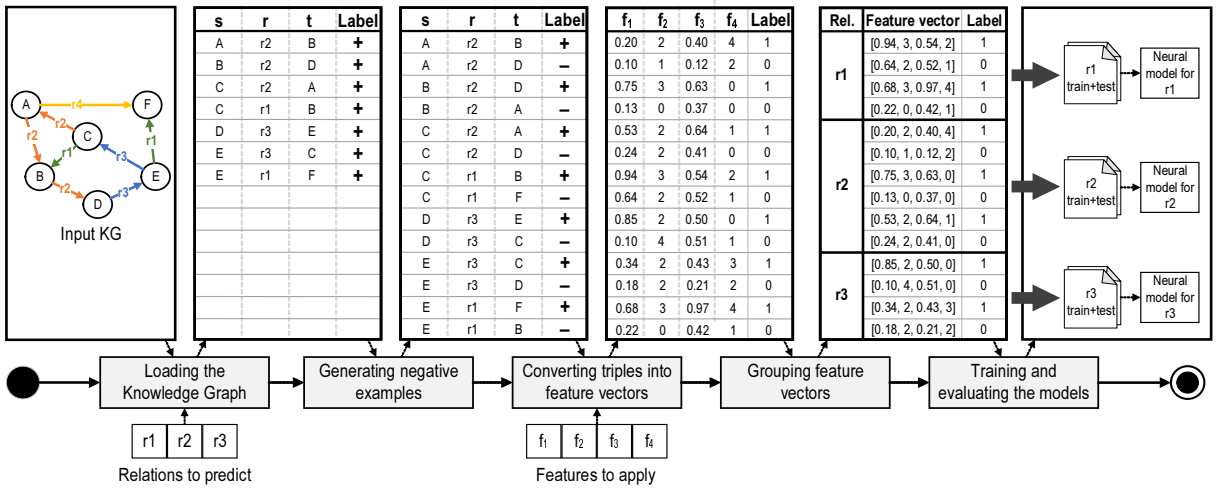


Figure 9.1: In-depth view of the CAFE workflow

- **Loading the Knowledge Graph:** CAFE internally stores the input KG in the form of (s, r, t) triples, using an efficient data structure based on hash tables, which is suitable for a high frequency of read operations due to its O(1) lookup times. The triples that contain relations for which a predictive model does not need to be generated are still taken into account when computing features, since they may provide valuable predictive information, but they are not transformed

into feature vectors in the following steps.

- **Generating negative examples:** A Knowledge Graph contains only positive information, i.e., it contains examples of the occurrence of a relation r between two entities. However, it does not contain explicit information about pairs of entities for which r does not hold. Our proposal relies on a classification model that requires negative examples for training, which means that a number of negatives for each positive triple must be produced. To accomplish this, we follow the type-constrained local closed world assumption [11], i.e., we generate negative examples from every triple (s, r, t) present in a KG by replacing their target entity t with a different one, t', such that the resulting triple (s, r, t') does not exist in the KG. Furthermore, to preserve the range of each relation, we randomly choose t' such that there exists some other triple in the KG where t' appears as the target entity for the relation r. This is known as type constraint.

  In the example depicted in Figure 7.1, a valid negative example is *(Hermione Granger, appears_in, The Cuckoo's Calling)*, since we know that *The Cuckoo's Calling* is a valid target for the relation *appears_in*. However, *(Hermione Granger, appears_in, Daniel Radcliffe)* would not be allowed as a negative example, because *Daniel Radcliffe* never appears as the target of the relation *appears_in*.

  It can be argued that generating negative evidence in this manner can produce false negatives by mere chance, i.e., statements that are deemed incorrect but that are true in the real world. While this is indeed plausible, it is generally accepted [68, 82, 134] that the chances of this happening are very low and, as a consequence, the possible effects on the final results are not significant.

- **Converting triples into feature vectors:** Once negative examples have been generated, our feature set is instantiated and applied to all triples. For all feature groups, we obtain all possible feature instances by applying all possible combinations of the values of their parameters. Each feature instance assigns a real number to each triple. Therefore, applying several features to a triple results in a feature vector. Each position of the feature vector represents that real number that the corresponding feature assigned to the triple.

  It is important to note that, to compute features on a positive training triple, we temporarily remove it from the KG, since not doing so would result in trivial prediction models such as "a person plays a character if there exists a triple in the KG stating that the person plays that character".

- **Grouping feature vectors:** The previous step computes feature vectors of triples. Since these triples can be either positive or negative, the feature vectors are accordingly labeled as positive or negative. Based on the labeled feature vectors, we train a classification model for each relation that predicts whether a triple should be added to the KG. We do this in order to allow the models to capture meaningful and distinctive information for every relation: even though the same set of features is applied to all triples, some features might have more predictive power for a relation, and other features may be more helpful for a different one.

- **Training and evaluating the models:** For every relation that we predict, we create one or more neural models, where each model focuses only on the features that are obtained from a certain neighborhood size. Thus, using only neighborhood subgraphs of size 1 results in one model, using neighborhood subgraphs of size of up to 2 results in two models, and so on. This allows each model to capture the specific information that every neighborhood size may yield. To combine two or more models, we use an additional combination layer to produce a single output.
  The neural models are trained using the labeled feature vectors in the training split for the desired relation, where each model receives only the features corresponding to its assigned neighborhood size, and the label or ground truth is shared among them. Prior to training our models, we first remove any individual features that have the exact same value in every feature vector and thus lack any predictive power. An example of this are path-based features (f_{12}), since only a small subset of all possible paths of fixed length occur between two given entities, and as a consequence most of them have a value of 0.
  It is important to note that we use neural classification models because they have been shown to consistently achieve satisfactory results in many different classification tasks [2, 8, 165], although other classification models that make use of our features could be used in this step.

9.3 Software Architecture

We show the internal class architecture of CAFE in Figure 9.2, and its data workflow is displayed in Figure 9.3. We further describe and discuss the architecture of CAFE in the following.

The main flow of CAFE is coordinated by the class *ParallelWorker*. This class can be configured to use a certain number of threads, and then internally spawns processes to parallelize the different tasks done by CAFE. This is done by splitting the total number of triples in the KG between the different threads, since the processing of triples can be done concurrently for the most part, as shown in Figure 9.3.

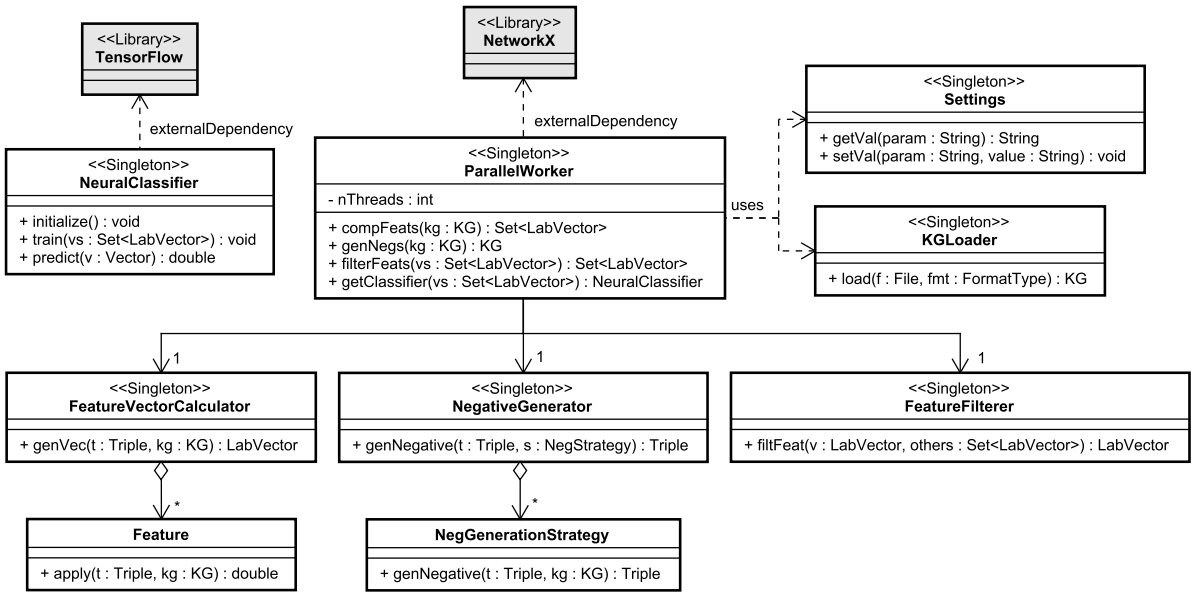


Figure 9.2: Architecture of CAFE

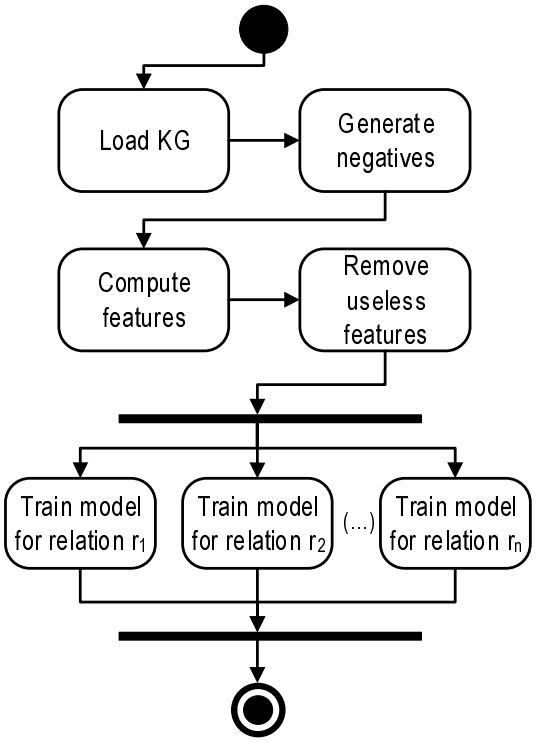


Figure 9.3: Workflow of CAFE

The method *genNegs* produces negative triples using the *NegativeGenerator* class and a generation strategy, since they are necessary to train the classification models.

Then, the method *compFeats* transforms every triple in the KG into a labeled feature vector, using a series of neighborhood-aware features that leverage the similarities of the neighborhoods of two entities in a KG. This is done by measuring the similarities of the entity neighborhoods of the source and target of a triple, using measures such as the Jaccard index of similarity, analyzing the size of such neighborhoods using

measures such as the Adamic-Adar index; as well as assessing the overall connectivity of the entities of a triple using the N-path centrality index.

The method *filtFeats* removes the features that have the same value in all labeled vectors, since they have no predictive power. Finally, method *getClassifier* trains and returns a neural classifier for a given set of labeled vectors, corresponding to a certain relation in the KG. Given that the work done by these methods is usually carried out in an independent manner for each triple, it is parallelized among different execution threads.

Class *ParallelWorker* uses other classes by means of composition to perform specialized tasks: Class *NegativeGenerator* provides a unified interface to easily generate a negative triple in a KG following one of the available strategies, which are implemented using *NegGenerationStrategy* classes.

Class *KGLoader*, through its *load* method, is used to read and write KGs in different formats, namely: N3, Turtle, and RDF/XML.

Class *FeatureVectorCalculator* is responsible for applying the previously discussed catalogue of features to any given triple using the method *getVec*, resulting in a labeled vector for that triple. The catalogue of available features can be easily expanded thanks to the *Feature* class, since each individual feature is implemented as an instance of that class.

Class *FeatureFilterer* specializes in removing useless features from a feature vector, i.e., those that share the same value in all vectors and thus have no predictive power. Method *filtFeat* receives the labeled vector to be processed and a set of all other labeled vectors, to allow for the multithreaded processing of different labeled vectors simultaneously.

Finally, the *NeuralClassifier* class produces, trains and evaluates the neural models that determine whether a triple is correct. These models receive as input the feature vector generated from the triple in question, and outputs a confidence value between 0 and 1. The models used by CAFE have three feed-forward layers with 1024, 512 and 256 neurons each. The method *initialize* produces a new model for a certain relation, to allow it to specialize in the specific details of said relation, which may be different to other relations present in the KG. The method *train* receives the training set of labeled vectors resulting from processing a KG with the *ParallelWorker*, and uses them to train the classification model. This method uses a learning rate of 0.001, a dropout of 0.1 for all layers, a batch size of 16 and 100 epochs, as these parameters have been proven to yield satisfactory results in some of our previous work [20]. Then, all vectors in the testing set are evaluated using the *predict* method.

Throughout the whole process, the *Settings* class is queried to retrieve the configuration parameters set by the user, e.g., the relative sizes of the training and

testing splits, or the negative generation strategy to use.

CAFE has two main dependencies with external libraries: NetworkX [58], which is used to internally store the KG and apply graph-based algorithms to compute the different features; and TensorFlow [1], which is used to create, train and apply the neural classification models.

9.3.1 Design and performance considerations

The architecture of CAFE shares some common patterns and design decisions with that of CHAI, described in the previous chapter. Most classes only need to be instantiated once throughout the execution of CAFE and, for this reason, we have decided to turn them into singletons for simplicity of use and to minimize the use of memory.

There are, nonetheless, a number of classes that clearly must allow multiple instances of them to exist, in order to represent variations of the same concept. A prime example of this is the Feature class: all features share the same interface, since they receive a triple and the KG that contains it and produces a numeric value. However, in order to accommodate all features defined in Section 9.2.1, each one of them exists as an instance of Feature.

Another example is the possible existence of multiple negative generation strategies, where one must be chosen at runtime according to the selection of the user. In both cases, they are accessed through auxiliary classes that act as catalogues, automatically detecting and loading all instances of the catalogued classes.

Given that CAFE is a very computationally intensive system, several optimizations have been made to make the best possible use of the resources of the system it runs on. Due to the fact that computing a feature vector for a triple can be done independently of all other triples in a KG, this task is parallelized and evenly distributed among all available cores in a CPU. This, however, comes at the cost of ensuring that all singletons in the system are thread-safe, given that they will be accessed by multiple processes at the same time.

Finally, it is important to note that all features defined by CAFE are deterministic, and will always return the same value for the same triple in a KG. Most features only analyze one of the entities in a triple at a time, and thus are re-computed very often for the same entity. For this reason, we have implemented a thread-shared caching strategy, in which the result of applying a feature to a triple is stored into a cache that is accessed by all threads. This allows CAFE to significantly reduce the number of calculations that must be performed when processing a KG.

9.4 Evaluation

In this section, we present the evaluation that we carried out to assess the performance and effectiveness of CAFE. First, we introduce the Knowledge Graphs on which CAFE was applied and an overview of their main characteristics. Next, we explain the methodology that we used to evaluate CAFE. Finally, we show and discuss the results of our evaluation.

9.4.1 Experimental data

We evaluated our proposal using four KGs provided by the freely available AYNEC-DataGen [6] tool: FB13-A, WN11-AR, WN18-AR and NELL-AR. These KGs are based on the well-known FB13, WN11 [134], WN18 [18], and a subset of NELL proposed in [47]. However, they have been processed to remove reciprocal relations detected by AYNEC, i.e., relations r and r' such that, if (s, r, t) exists, then (t, r', s) also exists very frequently. Additionally, relations that amount to less than 5% of the total number of triples in the graph have been removed.

These evaluation KGs originally contained one negative example per each positive triple in both their training and testing splits. In order to study how the KG completion techniques perform when presented with a much higher volume of negative evidence, we created versions of these graphs whose testing splits contained 10 negative examples per positive, using the AYNEC-DataGen tool. We believe that this is a more realistic scenario, since a much higher number of negative examples per positive triple is typically expected in real-world KG completion tasks [19]. To avoid confusion, we denote these versions as FB13-A-10, WN11-AR-10, WN18-AR-10 and NELL-AR-10.

For FB13-A-10, WN11-AR-10 and WN18-AR-10, we aimed to predict all possible relations, and for NELL-AR-10 we focused on the same subset of 10 relations that were used to evaluate SFE [47]. However, in the latter KG, one relation was removed by AYNEC for being the reciprocal of another relation, leaving 9 relations for evaluation. In the specific case of FB13-A-10, we transferred 25% of the training triples over to the testing set in order to provide testing examples for some relations, as they were not available in the original KG as introduced in [134]. Table 9.1 provides an overview of the aforementioned KGs. In the case of NELL-AR-10, we show in parentheses the amount of triples and relations that were considered for evaluation, although the entire graph was used for computing features.

9.4.2 Experimental setup

A neural prediction model was created for every relation of interest and trained using its corresponding training set. Then, the model was applied to all feature vectors in the test set, and we compared the expected label (which denotes whether it represents

KG	Training triples	Test triples	Entities	Relations
FB13-A-10	228,172	481,457	74,998	13
WN11-AR-10	77,948	198,231	38,195	9
WN18-AR-10	71,984	183,051	40,943	11
NELL-AR-10	86,971 (1,451)	219,374 (5,083)	53,934	148 (9)

Table 9.1: Overview of the KGs used for evaluating CAFE

a correct triple or not) against the label that was produced by our model. We report our results in terms of precision, recall and F1, in order to determine how effective our proposal is when determining the correctness of a given triple.

We evaluated three versions of CAFE, denoted CAFE_1 to CAFE_3, which were limited to using feature instances that exploited neighborhood subgraphs and paths of a maximum size of 1, 2, and 3, respectively. This was done in order to study how using larger neighborhoods affects the effectiveness of CAFE.

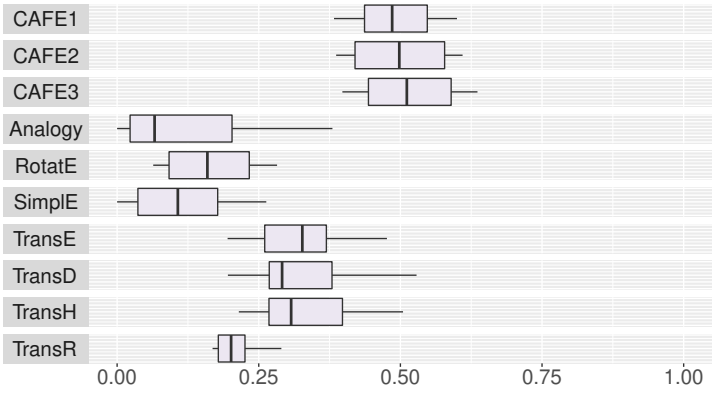
There exist many different KG completion proposals, and they often use different evaluation metrics [135]. Due to this, it is very difficult to perform a comparison across a large number of them in a manner that is fair and rigorous. For this reason, we used TransE [17], TransD [68], TransH [155], TransR [82], Analogy [83], SimplE [72] and RotatE [138] as baselines for our evaluation, since they are some of the most well-known state-of-the-art KG completion proposals. In order to provide a common evaluation environment for these different proposals, we used the OpenKE [59] framework to train and evaluate these proposals using the previously discussed Knowledge Graphs. Additionally, since these proposals usually report metrics like MRR and Precision@N, we used the utilities provided by OpenKE to obtain binary labels for the testing triples, by setting a likelihood threshold in a way that optimized the classification results.

We selected the following values for the hyperparameters of our neural models: 3 layers with 1024, 512 and 256 neurons each, learning rate of 0.001, batch size of 16, dropout of 0.1 for all layers, 100 epochs and validation ratio of 10%. When two or more models were to be combined, we joined their results using a hidden layer with 3 neurons and an output layer with a single neuron. These values for the hyperparameters were chosen using a hold-out or "dev" set for the FB13-A-10 KG, and all KGs were then evaluated using the same hyperparameters. We chose them because they provided satisfactory results in our empirical tests.

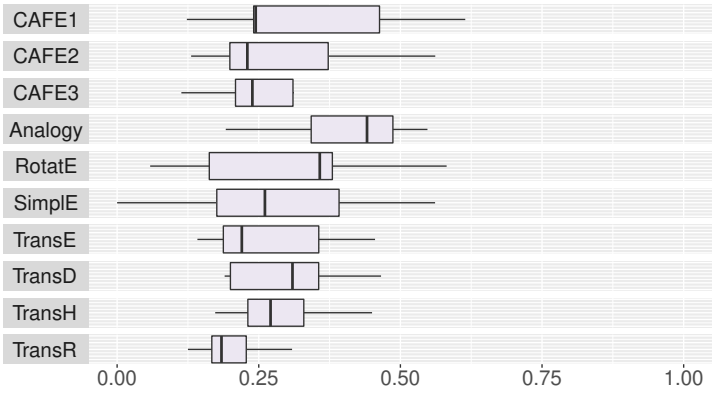
All our experiments were conducted on a computer equipped with an Intel Core i9-9900K CPU, 32GB of RAM and an Nvidia RTX 2080 Ti GPU.

9.4.3 Results and discussion

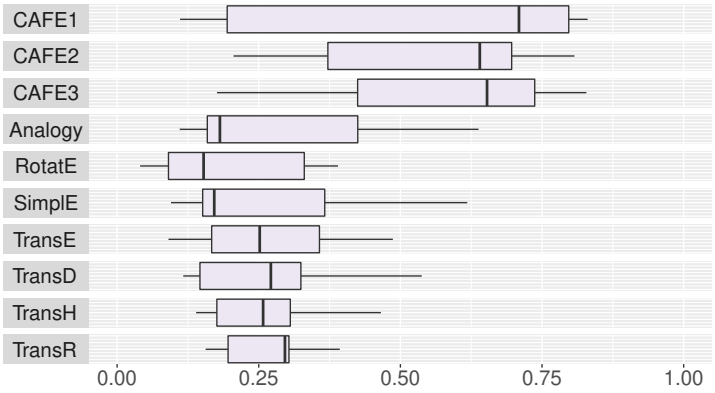
In Figure 9.4, we show the evaluation results for $CAFE_1$, $CAFE_2$, $CAFE_3$, and the related state-of-the-art proposals. For the sake of clarity, this Figure only displays the F1 values for each technique. Additionally, Table 9.2 shows the detailed results for all relations in every KG and for all metrics under evaluation.



(a) FB13-A-10



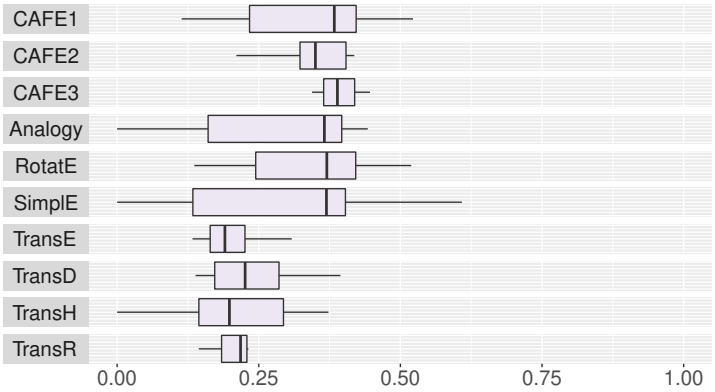
(b) NELL-AR-10



(c) WN11-AR-10

Figure 9.4: F1 comparison between CAFE and other proposals

Our results show that CAFE is able to match the performance of state-of-the-art proposals, and in many cases achieve higher values on the metrics under evaluation. In



(a) WN18-AR-10

Figure 9.4: F1 comparison between CAFE and other proposals (cont.)

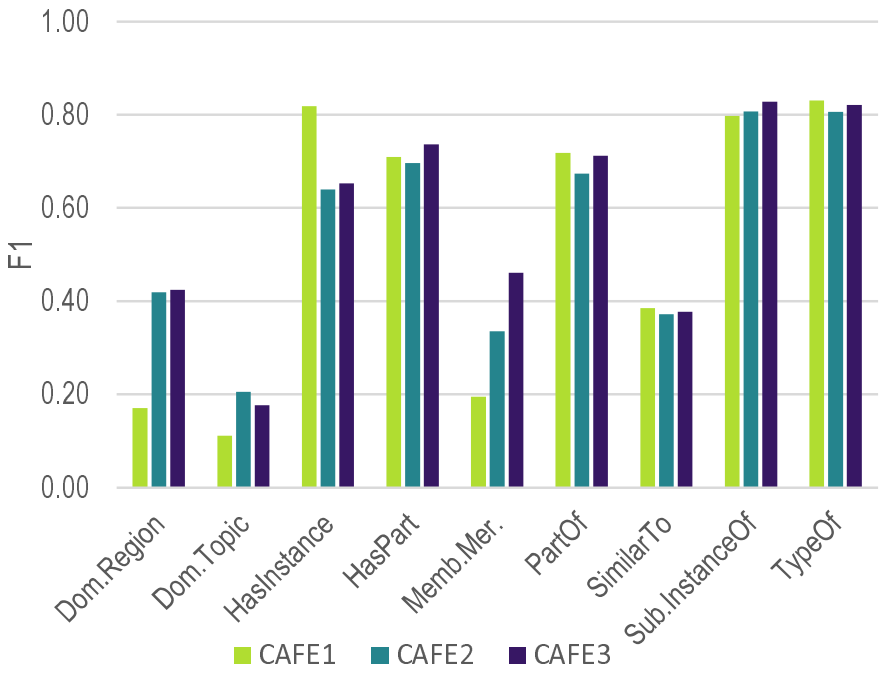


Figure 9.5: F1 scores in the WN11-AR-10 KG

the cases of FB13-A-10 (Figure 9.4(a)) and WN18-AR-10 (Figure 9.5(a)), CAFE can reach or surpass the F1 scores achieved by other proposals in a consistent manner. CAFE also provides better results in the WN11-AR-10 (Figure 9.4(c)) KG, although with a higher degree of variability, and matches the performance of the rest of the analyzed techniques in the NELL-AR-10 (Figure 9.4(b)) KG. These results show that CAFE can be more effective than other proposals in challenging classification scenarios.

		CAFE$_1$			CAFE$_2$			CAFE$_3$			Analogy		
		P	R	F1	P	R	F1	P	R	F1	P	R	F1
FB13-A-10	CauseOfDeath	**0.33**	0.73	**0.45**	0.29	0.79	0.42	0.31	0.76	0.44	0.04	0.02	0.02
	Children	0.40	0.78	0.53	0.44	**0.88**	0.59	**0.50**	**0.88**	**0.64**	0.17	0.14	0.16
	Ethnicity	**0.60**	0.60	**0.60**	0.53	0.63	0.57	0.48	0.56	0.52	0.03	0.01	0.02
	Gender	0.88	0.88	0.88	**0.89**	0.89	**0.89**	**0.89**	0.89	**0.89**	0.00	0.00	0.00
	Institution	0.26	0.70	0.38	**0.28**	0.67	0.39	**0.28**	0.67	**0.40**	0.10	0.05	0.07
	Location	**0.37**	0.51	**0.43**	0.28	0.63	0.39	0.34	0.57	0.42	0.36	0.38	0.37
	Nationality	0.45	0.90	0.60	**0.46**	0.89	**0.61**	**0.46**	0.89	0.60	0.02	0.01	0.02
	Parents	0.35	0.77	0.49	0.44	**0.85**	0.58	**0.45**	**0.85**	**0.59**	0.16	0.14	0.15
	PlaceOfBirth	**0.39**	0.49	**0.44**	0.29	0.58	0.39	0.31	0.55	0.40	0.37	0.39	0.38
	PlaceOfDeath	0.36	0.70	0.47	**0.39**	0.67	**0.50**	0.38	0.69	0.49	0.33	0.31	0.32
	Profession	0.40	0.85	**0.55**	0.40	0.85	**0.55**	**0.41**	0.85	**0.55**	0.03	0.02	0.02
	Religion	0.37	0.77	0.50	0.32	0.83	0.46	**0.38**	0.77	**0.51**	0.03	0.02	0.02
	Spouse	0.27	0.89	0.41	0.32	**0.90**	0.47	**0.33**	0.89	**0.48**	0.25	0.17	0.20
NELL-AR-10	ActorStarredInMovie	0.14	0.89	0.25	0.11	**0.94**	0.19	0.11	**0.94**	0.20	0.42	0.59	0.49
	AthletePlaysForTeam	0.33	0.80	**0.46**	0.24	0.83	0.37	0.19	**0.85**	0.31	**0.39**	0.51	0.44
	CityLocatedInState	0.15	0.61	0.24	0.15	0.66	0.24	0.15	0.62	0.25	**0.43**	0.57	**0.49**
	JournalistWritesForPublication	**0.47**	0.89	**0.61**	0.40	0.92	0.56	0.39	0.92	0.55	0.34	0.35	0.34
	RiverFlowsThroughCity	0.15	0.65	0.24	0.13	**0.81**	0.23	0.12	**0.81**	0.21	0.37	0.51	**0.43**
	SportsteamPositionAthlete	0.07	0.92	0.12	0.07	**1.00**	0.13	0.06	0.85	0.11	0.22	0.17	0.19
	StadiumLocatedInCity	0.15	0.77	0.24	0.11	0.91	0.20	0.12	**0.95**	0.21	**0.42**	0.56	**0.48**
	TeamPlaysInLeague	0.47	0.85	0.60	0.55	**0.88**	0.67	**0.56**	0.86	**0.68**	0.08	0.04	0.06
	WriterWroteBook	0.13	0.86	0.23	0.12	0.89	0.21	0.14	**0.93**	0.24	**0.45**	0.71	**0.55**
WN11-AR-10	DomainRegion	0.22	0.14	0.17	**0.42**	0.42	0.42	0.38	0.47	0.42	0.39	0.63	**0.48**
	DomainTopic	0.16	0.09	0.11	0.26	0.17	0.21	0.27	0.13	0.18	**0.49**	0.93	**0.64**
	HasInstance	**0.92**	0.74	**0.82**	0.52	**0.82**	0.64	0.54	**0.82**	0.65	0.17	0.20	0.18
	HasPart	0.63	0.81	0.71	0.59	**0.86**	0.70	**0.65**	**0.86**	**0.74**	0.13	0.15	0.14
	MemberMeronym	0.12	0.47	0.19	0.22	0.74	0.34	0.32	**0.81**	**0.46**	**0.35**	0.53	0.42
	PartOf	**0.67**	0.78	**0.72**	0.56	0.85	0.67	0.61	**0.86**	0.71	0.16	0.19	0.17
	SimilarTo	0.24	0.90	**0.39**	0.23	**0.91**	0.37	0.24	0.90	0.38	0.22	0.27	0.24
	SubordinateInstanceOf	0.72	0.89	0.80	0.72	0.92	0.81	**0.75**	**0.93**	**0.83**	0.11	0.12	0.11
	TypeOf	**0.84**	0.82	**0.83**	0.77	0.84	0.81	0.80	0.84	0.82	0.15	0.17	0.16
WN18-AR-10	AlsoSee	0.25	0.79	0.38	**0.26**	0.82	**0.39**	0.25	**0.83**	0.38	0.22	0.26	0.24
	DerivationallyRelatedForm	0.28	0.87	0.43	0.41	0.90	0.57	**0.43**	**0.92**	**0.59**	0.09	0.09	0.09
	HasPart	0.16	0.69	0.26	0.21	0.70	0.33	0.24	**0.73**	0.36	**0.36**	0.53	**0.43**
	Hypernym	0.31	0.50	0.38	0.23	0.60	0.33	0.27	**0.68**	0.39	**0.37**	0.54	**0.44**
	InstanceHypernym	**0.44**	0.50	**0.47**	0.29	0.77	0.42	0.31	0.82	0.45	0.32	0.45	0.37
	MemberHolonym	0.17	0.21	0.19	0.19	0.69	0.30	0.26	**0.81**	**0.39**	0.26	0.33	0.29
	MemberOfDomainRegion	0.27	0.17	0.21	0.17	0.28	0.21	0.19	0.26	0.22	0.33	0.41	0.37
	MemberOfDomainUsage	0.15	0.09	0.11	0.44	0.29	0.35	**0.48**	0.30	0.37	0.39	0.46	0.42
	SimilarTo	**0.26**	**1.00**	**0.42**	0.19	**1.00**	0.32	0.21	**1.00**	0.34	0.00	0.00	0.00
	SynsetDomainTopicOf	0.30	0.60	**0.40**	0.26	0.76	0.39	0.27	0.75	0.39	0.32	0.43	0.37
	VerbGroup	0.36	0.92	0.52	**0.40**	0.94	**0.56**	0.38	**0.96**	0.54	0.04	0.04	0.04

Table 9.2: Detailed CAFE results

RotatE			SimplE			TransE			TransD			TransH			TransR		
P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
0.09	0.06	0.07	0.05	0.03	0.04	0.23	0.57	0.33	0.18	0.78	0.29	0.19	0.73	0.31	0.13	**0.86**	0.23
0.20	0.24	0.22	0.17	0.18	0.18	0.15	0.59	0.24	0.19	0.46	0.27	0.17	0.52	0.25	0.14	0.54	0.22
0.11	0.08	0.09	0.11	0.07	0.09	0.25	0.67	0.37	0.26	0.71	0.38	0.35	0.64	0.45	0.11	**0.94**	0.20
0.18	0.14	0.16	0.00	0.00	0.00	0.55	**0.98**	0.70	0.71	0.81	0.76	0.82	0.88	0.85	0.74	0.84	0.79
0.11	0.09	0.10	0.14	0.09	0.11	0.20	0.52	0.29	0.18	0.61	0.28	0.16	0.72	0.27	0.10	**0.95**	0.18
0.20	0.19	0.20	0.18	0.14	0.16	0.23	0.55	0.33	0.18	0.65	0.28	0.19	0.64	0.29	0.10	**0.93**	0.18
0.11	0.08	0.09	0.03	0.01	0.02	0.32	0.91	0.48	0.38	0.86	0.53	0.35	0.88	0.50	0.09	**1.00**	0.17
0.21	0.26	0.23	0.18	0.18	0.18	0.17	0.55	0.26	0.19	0.48	0.27	0.19	0.51	0.27	0.16	0.59	0.25
0.27	0.30	0.28	0.22	0.17	0.19	0.16	0.54	0.24	0.18	0.50	0.27	0.16	0.56	0.25	0.10	**0.91**	0.18
0.24	0.25	0.24	0.18	0.13	0.15	0.21	0.67	0.32	0.24	0.62	0.34	0.25	0.62	0.35	0.09	**0.94**	0.17
0.09	0.06	0.07	0.03	0.02	0.02	0.27	0.79	0.40	0.26	0.83	0.39	0.26	0.80	0.40	0.10	**1.00**	0.18
0.08	0.05	0.06	0.08	0.04	0.06	0.24	0.69	0.35	0.25	0.73	0.38	0.24	0.73	0.36	0.12	**0.94**	0.21
0.24	0.30	0.27	0.26	0.27	0.26	0.12	0.59	0.20	0.12	0.65	0.20	0.14	0.49	0.21	0.19	0.67	0.29
0.47	0.76	**0.58**	0.45	0.73	0.56	0.08	0.50	0.14	0.20	0.72	0.31	0.16	0.72	0.26	0.10	0.78	0.18
0.14	0.07	0.09	0.22	0.15	0.18	0.18	0.67	0.28	0.30	0.54	0.38	0.24	0.52	0.33	0.13	0.80	0.23
0.21	0.13	0.16	0.09	0.05	0.06	0.13	0.75	0.22	0.11	0.56	0.19	0.14	0.62	0.23	0.11	**0.92**	0.20
0.28	0.22	0.25	0.26	0.22	0.24	0.24	0.68	0.36	0.25	0.60	0.36	0.22	0.82	0.35	0.18	**0.97**	0.30
0.39	0.37	0.38	**0.39**	0.40	0.39	0.12	0.23	0.16	0.12	0.42	0.19	0.10	0.79	0.17	0.10	0.47	0.17
0.36	0.39	**0.38**	0.32	0.22	0.26	0.12	0.46	0.19	0.12	0.69	0.20	0.11	0.46	0.18	0.06	0.08	0.06
0.37	0.35	0.36	0.29	0.30	0.30	0.35	0.41	0.38	0.21	0.68	0.32	0.18	0.59	0.27	0.20	0.09	0.13
0.09	0.04	0.06	0.00	0.00	0.00	0.32	0.77	0.46	0.33	0.79	0.47	0.35	0.62	0.45	0.19	0.86	0.31
0.45	0.68	0.54	0.44	0.69	0.54	0.13	0.34	0.19	0.15	0.58	0.23	0.21	0.46	0.28	0.10	0.77	0.18
0.30	0.40	0.34	0.39	0.62	**0.48**	0.09	0.12	0.10	0.09	0.42	0.15	0.09	0.38	0.14	0.09	**0.90**	0.16
0.30	0.37	0.33	0.47	0.89	0.62	0.11	0.33	0.17	0.10	0.19	0.13	0.11	0.45	0.18	0.11	**0.94**	0.20
0.14	0.16	0.15	0.16	0.18	0.17	0.15	**0.82**	0.25	0.20	0.71	0.31	0.20	0.64	0.30	0.20	0.65	0.30
0.06	0.07	0.06	0.12	0.13	0.12	0.17	0.75	0.28	0.17	0.70	0.27	0.16	0.71	0.26	0.13	0.79	0.23
0.19	0.23	0.21	0.31	0.44	0.37	0.08	0.10	0.09	0.07	0.32	0.12	0.08	0.50	0.14	0.09	0.77	0.16
0.09	0.09	0.09	0.14	0.16	0.15	0.27	0.66	0.38	0.21	0.74	0.32	0.20	0.71	0.31	0.19	0.68	0.30
0.33	0.48	**0.39**	0.22	0.28	0.25	0.15	0.38	0.22	0.12	0.59	0.21	0.14	0.51	0.22	0.23	0.82	0.35
0.04	0.04	0.04	0.09	0.10	0.10	0.38	0.68	0.49	0.36	0.75	0.49	0.34	0.76	0.47	0.18	0.90	0.30
0.13	0.15	0.14	0.14	0.16	0.15	0.22	**0.88**	0.36	0.41	0.79	0.54	0.38	0.78	0.51	0.26	0.80	0.39
0.23	0.30	0.26	0.17	0.21	0.19	0.11	0.56	0.19	0.14	0.56	0.23	0.12	0.56	0.20	0.13	0.76	0.22
0.15	0.17	0.16	0.08	0.08	0.08	0.20	0.81	0.32	0.27	0.75	0.39	0.25	0.74	0.37	0.25	0.87	0.38
0.32	0.44	0.37	0.32	0.45	0.37	0.08	0.39	0.13	0.08	0.49	0.14	0.08	0.45	0.13	0.08	0.59	0.14
0.36	0.52	0.42	0.36	0.50	0.42	0.14	0.54	0.22	0.17	0.45	0.24	0.16	0.47	0.24	0.13	0.39	0.20
0.35	0.52	0.42	0.33	0.48	0.39	0.20	0.69	0.31	0.20	0.69	0.31	0.19	0.72	0.29	0.12	**0.85**	0.22
0.28	0.39	0.33	0.23	0.29	0.26	0.12	0.40	0.19	0.12	0.53	0.19	0.13	0.38	0.20	0.11	0.45	0.18
0.43	0.66	0.52	**0.47**	**0.87**	**0.61**	0.12	0.25	0.17	0.11	0.77	0.20	0.12	0.22	0.16	0.14	0.53	0.23
0.40	0.57	**0.47**	0.39	0.57	0.46	0.10	0.53	0.16	0.09	0.47	0.15	0.08	0.23	0.11	0.11	**0.87**	0.19
0.21	0.25	0.23	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.33	0.45	0.38	0.32	0.44	0.37	0.13	**0.79**	0.23	0.16	0.71	0.26	0.20	0.53	0.29	0.14	0.69	0.23
0.13	0.14	0.14	0.04	0.04	0.04	0.14	0.61	0.23	0.26	0.44	0.33	0.21	0.57	0.30	0.36	0.61	0.46

Table 9.2: Detailed CAFE results (cont.)

Table 9.2 displays that, in general, both a satisfactory precision and recall can be achieved, and thus we consider that CAFE is generally effective. However, there exists a number of relations for which a very high precision value is obtained at the expense of a lower recall or vice-versa, resulting in a typical precision-recall trade-off. We also observe that the nature of every individual relation has a significant impact in the results, since some of them are harder to predict than others. Such is the case of the relation *cause_of_death*, since learning to predict the cause of the death of a person with a very high effectiveness would be a remarkable achievement that unfortunately falls out of the scope of this work. We further discuss this limitation in Section 9.5.

Regarding the question of how using different neighborhood sizes affects the effectiveness of CAFE, Figure 9.4 shows that the metrics under evaluation are generally higher for $CAFE_2$ than for $CAFE_1$, but the same cannot always be said for $CAFE_3$ and $CAFE_2$. Indeed, metrics appear to remain stagnant or even decrease when using larger neighborhoods in some cases. For example, in FB13-A-10 (Figure 9.4(a)), we do not observe a significant increase in effectiveness when using larger neighborhood subgraphs, which suggests that the most useful information is readily available in the immediate vicinities of the relevant entities. In the cases of WN11-AR-10 (Figure 9.4(c)) and WN18-AR-10 (Figure 9.5(a)), an improvement is observed when using neighborhoods of size 2, but further expanding the neighborhood size does not seem to have a significant impact. A possible conclusion for this is that, at a certain point, larger neighborhood subgraphs do not provide additional value or predictive power over smaller ones. In a worst-case scenario, the number of features with little to no predictive power would greatly increase by using larger neighborhood sizes, negatively affecting the results. This effect actually occurs in the case of the NELL-AR-10 KG (Figure 9.4(b)), where effectiveness decreases when increasing the size of the neighborhood subgraphs for which we compute features. A plausible explanation for this is that it has been noted that NELL contains noisier data [108], and thus taking larger neighborhoods into account significantly increases the amount of noise that our classification model has to deal with, reducing its effectiveness.

9.5 Limitations

Despite our triple classification proposal being generally effective, it has some limitations. It does not work well for relations for which useful predictive information cannot always be found in the neighborhoods of the entities in a triple. In this regard, we identify two types of relations: those that can be predicted using other information present in the KG, and those for which the entities and relations in their neighborhoods do not provide useful information, and thus are much harder to predict. This dichotomy can be observed in the results shown in Table 9.2, and it is especially notable in the WN11-AR-10 KG. For the sake of visualization, we display

the results of applying CAFE to this Knowledge Graph in Figure 9.5.

This difference is particularly visible in the *Similar to* and *Domain topic* relations, which have low F1 scores. Upon manual inspection, we found that *Similar to* tends to link words that are generally isolated within the KG and share very little common context. Also, *Domain topic* is extremely broad, causing the two entities in a triple to have few or no relevant common elements in their neighborhoods, e.g., (*Britain*, *Domain topic*, *Surgery*).

In other cases, even if some information is present in the neighborhoods under consideration, it may be not successfully captured by CAFE due to the large amount of irrelevant data surrounding it. In this regard, the NELL-AR-10 KG shows that larger neighborhood sizes are not always better for predictive purposes, since the amount of noise they may include can be detrimental for the performance of CAFE. This can be due to some source or target entities being present in many triples (for example, countries). Therefore, larger neighborhoods of these entities introduce many other entities that are not relevant to the triple under evaluation. In these cases, it is up to the users to decide which maximum neighborhood size best caters to their interests.

9.6 Summary

In this chapter we have introduced CAFE, our proposal to classify candidate triples for Knowledge Graph completion. CAFE defines a set of neighborhood-aware features which evaluate several aspects of a triple, by checking for shared neighborhoods at several distances with other triples in a KG, and then combines the values of all features into a feature vector. Once all candidate triples have been transformed into feature vectors, CAFE trains and applies a neural binary classifier to discern between correct and incorrect triples, and provides the user with a set of correct candidate triples to add back to the KG. Our experimental evaluation shows that CAFE is very effective, outperforming other state-of-the-art approaches in a number of well-known Knowledge Graphs of different sizes and domains.

Chapter 10

SciCheck: Completing scientific Knowledge Graphs

доверяй, но проверяй.
(Trust, but verify.)

— Russian proverb

Scientific knowledge is in constant expansion, and many efforts have been developed throughout the years to capture it in a structured format. Knowledge Graphs can support this task, by linking entities representing research concepts together. These scientific KGs have particular nuances that make completing them a particularly challenging task. In this chapter, we introduce SciCheck, our proposal to complete Knowledge Graphs representing research concepts. This chapter is structured in the following manner: Section 10.1 provides an introduction, Section 10.2 describes SciCheck in detail, including the features that it uses to characterize triples representing scientific knowledge, Section 10.3 presents the experimental evaluation that we have carried out to assess its efficacy and efficiency in practice, Section 10.4 shows a practical application of SciCheck on AI-KG, a large-scale Knowledge Graph of research concepts and, finally, Section 10.5 provides a summary of the chapter.

10.1 Introduction

In recent years, we have witnessed the appearance of a number of Knowledge Graphs that represent research and scientific knowledge. These graphs are generally either manually curated [13, 64, 76], or constructed automatically from academic metadata [34, 118, 123]. However, just like most KGs, they suffer from incompleteness, which means that some well-known scientific knowledge may not be present in them.

This chapter introduces SciCheck [22], our proposal to complete scientific Knowledge Graphs. SciCheck extends our generic triple classification technique, CAFE, by adding a number of features and heuristics specifically tailored for the scientific domain. SciCheck is able to compute a confidence score for every triple representing a scientific claim, and then derive a binary classification for it based on its confidence. Similarly to CAFE, SciCheck works by transforming a triple into numeric features, but includes additional heuristics to immediately discard incorrect scientific triples.

We evaluate the efficacy of SciCheck on seven different Knowledge Graphs and against nine alternative approaches to this task. Our evaluation shows that SciCheck is able to obtain a significantly better precision than its state-of-the-art counterparts, which is of the essence to reliably extend scientific Knowledge Graphs.

Additionally, we used SciCheck to extend the Artificial Intelligence Knowledge Graph (AI-KG) [34], a large-scale open KG that is constructed automatically using the 300K most cited papers in the field of Artificial Intelligence. Initially, AI-KG contained 1.2M statements about scientific claims in this field. By applying SciCheck to it, we have expanded it with 300K additional high-confidence statements. This application has resulted in the release of a new, expanded version of AI-KG, as well as in the creation of two high-quality Knowledge Graphs that can be used to benchmark KG completion.

10.2 Our proposal

SciCheck is a triple classification technique specifically designed to complete scientific statements in a Knowledge Graph. It is an extension of the CAFE approach [20] that incorporates a new set of features and heuristics that are specifically tailored to capture scientific knowledge.

SciCheck takes an entire KG in the form of triples as input, and produces one neural-based classifier for each relation in the KG as output. Specifically, given a relationship r, SciCheck generates a model $f_r : (s, r, t) \rightarrow c$, that assigns a confidence score c in the range $[0, 1]$ to any arbitrary triple (s, r, t) to solve a binary classification task ("is the triple correct or not?"). To feed the models, triples are converted into

a numerical vector representation using ad-hoc features and contextual embedding representations. SciCheck can operate on any KG and focuses on optimizing precision, to ensure that the knowledge deemed correct is trustworthy.

In the following, we describe and discuss the features that SciCheck uses to characterize a triple in the context of a scientific KG.

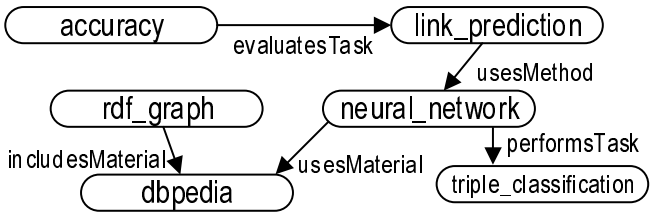


Figure 10.1: A small KG with research information about the Semantic Web

For the sake of illustration, Figure 10.1 displays a small KG that will be used to provide specific examples.

10.2.1 Extended feature set

SciCheck uses an extensible set of features specifically made to capture and process scientific information, which represent the neighborhoods of the two entities of a triple in a variety of ways. Each triple is evaluated by all features. Additionally, each feature can also depend on a number of parameters, such as a maximum neighborhood size.

Since, as described, SciCheck is an extension of CAFE, we enumerate only the additional features that it incorporates with respect to the original CAFE approach.

Feature f_1: Cosine similarity of the word embeddings of the source and target entities. This feature measures the semantic similarity of the two entities in a triple, using any entity embeddings. If we consider A and B to be the embeddings of the source and target entities of the triple respectively, it is defined as:

$$\cos(A, B) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

Feature f_2: Dot product of the word embeddings of the source and target entities. This feature complements the previous one by also taking into account the magnitudes of the embeddings of the entities. If we consider A and B to be the embeddings of the source and target entities of the triple respectively, it is defined as:

$$A \cdot B = \sum_{i=1}^{n} A_i B_i$$

Feature f_3: Types of the source and target entities according to the ontology of the Knowledge Graph. This feature encodes the known types of the entities according to the available ontology as two one-hot vectors. In Figure 10.1, the entity *dbpedia* is of type *Resource*, while *accuracy* is a *Metric*.

Regarding the rationales of the new features, f_1 incorporates information from the word embeddings of the two entities, which have been shown to be advantageous for triple classification [72, 138]. SciCheck uses by default the RoBERTa model [85] to generate the word embeddings, since is able to capture and represent semantic similarities across a wide range of domains. Similarly, f_2 provides a similar assessment, but uses the magnitudes of the embedding vectors to extract additional information from them.

Feature f_3 leverages the ontological schema of the KG. This allows SciCheck to include information regarding the types of the two entities in a triple into the feature vector for that triple. Furthermore, SciCheck can automatically classify a triple as incorrect if the triple does not respect the domains and ranges of the relation as defined in the ontological schema. For example, in the KG shown in Figure 10.1, the triple *(accuracy, evaluatesTask, rdf_graph)* would be considered incorrect without further evaluation, because the range of the relation *evaluatesTask* is the type *Task*, while *rdf_graph* is a *Material*.

This extension of the set of features allows SciCheck to better characterize scientific entities and predicates. In particular, the features based on word embeddings enable SciCheck to exploit the implicit contextual information from the training papers that may not be encoded in the KG. Additionally, the inclusion of ontology-based features allows SciCheck to take advantage of the available high-level knowledge about any specific domain. These improvements are particularly crucial for assessing scientific claims, which tend to use a specific jargon and to rely on a well-defined epistemological framework.

Furthermore, different types of relations in the graph may carry specific insight that should be captured separately. For this reason, SciCheck first computes all features in the input KG as-is, and then it computes them again in different versions of the KG where only relations of a single type are present. This is done for all the different relations in the KG. Additionally, in features that use the neighborhoods of the source and target entities, such as those originally defined by CAFE, these two neighborhoods are calculated using all possible combinations of relations. Finally, SciCheck concatenates all the resulting features in the final feature vector. The features which involve computing entity neighborhoods or paths use a maximum neighborhood size for their computations. Following the findings in [20], by default SciCheck computes them for a maximum size of 1, 2, and 3. The resulting set of features using different neighborhood sizes are eventually all added to the final feature vector.

10.3 Evaluation

In this section, we describe the experimental setup that we devised to test the effectiveness of SciCheck in practice. First, we introduce a number of similar state-of-the-art approaches to triple classification that serve as baselines for our evaluation. Next, we present the KGs that we used to validate our proposal. Finally, we show and discuss the results of our evaluation.

10.3.1 Baselines

We evaluated the performance of SciCheck against a number of alternative approaches. Five of the baselines are well-known embedding-based KG completion techniques: *TransE* [17], *TransD* [68], *TransH* [155], *SimplE* [72], and *ComplEx* [145]. To provide a common ground to train and test these techniques, we used the OpenKE [59] tool.

In order to better assess the contributions of the different components of SciCheck, we also considered five alternative versions of our approach:

- *CAFE Baseline*, which uses solely the context-aware features for KG completion such as neighborhood size, shared entities and connectivity from the original CAFE implementation [20].

- *CAFE+RoBERTa*, which extends CAFE by considering features based on the similarity of the embeddings of the source and target entities, using the RoBERTa model.

- *CAFE+SciBERT*, which extends CAFE by considering features based on the similarity of the embeddings of the source and target entities, using SciBERT, an alternative BERT-based text embedding model [112] specifically tailored to scientific documents.

- *CAFE+Ontology*, which extends CAFE by considering features that identify the types of the source and target entities according to the domain ontology and also filters triples whose entities are not consistent with the domain and range restrictions of the relation.

- *SciCheck*, the full version of our approach, which incorporates both features based on word embeddings and features based on the ontology of the Knowledge Graph.

To predict the correctness of a triple using SciCheck, we convert its confidence score in the interval $[0, 1]$ into a binary label by setting a confidence threshold for a correct triple of 0.5, as suggested in [20]. The thresholds of the other state-of-the-art

techniques under evaluation and their results were obtained using the OpenKE tool, allowing it to choose the optimal value for each one.

10.3.2 Evaluation data

The previously discussed baselines were evaluated on the following Knowledge Graphs, whose characteristics are summarized in Table 10.1:

KG	Training triples	Test triples	Entities	Relations
AIKG-1M	860,512	430,280	820,708	20
AIKG-500	860,512	500	228	7
FB13	228,172	105,509	74,998	13
WN11	77,948	36,042	38,195	9
WN18	71,984	33,282	40,943	11
WN18RR	86,835	3,134	40,943	11
NELL	86,971	40,104	53,934	148

Table 10.1: Overview of the KGs used for evaluating SciCheck

- *AIKG-1M*, a new KG that we created from AI-KG. We used a de-reified version of AI-KG, in order to consider only triples which involve tasks, methods, materials, metrics, and other scientific entities. As a result, 1,075,652 triples were directly generated from scientific literature, without considering facts that were materialized using the domain semantics defined in the AI-KG ontology (e.g., transitivity). Triples were split into a training and a testing set with a split ratio of 80%-20%, respectively. To generate negative triples in the testing split, each positive triple was corrupted once by randomly replacing the target entity with another one within the domain of the relation in the triple, i.e., if the range of the target entity is a Task, then it is substituted by another entity whose type is Task. We also make sure that the randomly generated negative triple is not already present in the KG, to prevent creating false negatives whenever possible. As an example, the triple *(dbpedia, usesOtherEntity, sparql_query)* is correct, while the corrupted version *(dbpedia, usesOtherEntity, cost_function)* is considered incorrect, where *sparql_query* and *cost_function* are both of type *OtherEntity*. However, negative examples were not generated for the training split, as specific KG completion techniques usually have a preferred way to generate them automatically [19]. In total, the training split comprised 860,512 positive triples and the testing split includes 430,280 triples (50% positive and 50% negative).

- *AIKG-500*, a new KG that we constructed by manually annotating triples in AI-KG about the Semantic Web. To construct it, we randomly selected 250 triples which had as their source entity one of the 24 sub-topics of the Semantic Web according to the CSO ontology [124] and were considered to be correct by at

least 2 methods among *TransE*, *TransD*, *TransH*, *SimplE*, *ComplEx*, and *SciCheck*. Another 250 triples were randomly selected out of those deemed incorrect by at least 2 of the previously mentioned techniques. The resulting 500 triples were manually annotated by five domain experts, with an inter-reviewer agreement of 0.61 (according to Cohen's kappa), which is typically considered a substantial agreement. A majority vote approach was used to determine that 221 triples were correct and 279 were incorrect. Since this Knowledge Graph was created for the purpose of providing a small but high-quality and manually-annotated testing split, in this evaluation we used AIKG-1M for the training split.

- *FB13* [134], a subset of FreeBase [14] that focuses on relevant people and their family relations, locations, professions, and other personal data.

- *WN11* [17], a subset of WordNet centered around different semantic relations between over 38K words.

- *WN18* [18], which expands WN11 with additional relations.

- *WN18RR* [38], which improves WN18 by removing reciprocal relations in the test set. This makes triple classification more challenging, since otherwise the model can predict that a triple *(a, hasChildren, b)* is true whenever the triple *(b, hasParent, a)* appears in the training set.

- *NELL* [47], a subset of the NELL KG [91] with information and relations about many different domains, e.g., actors which starred in movies, writers and their works, or athletes and their teams.

It is well-known [38] that these traditional KGs suffer from information leakage between the training and test sets, due to the presence of reciprocal relations. For this reason, we removed all reciprocal relations in all KGs except WN18, since we also include its previously discussed sanitized version, WN18RR.

10.3.3 Results and discussion

Table 10.2 and Table 10.3 report the precision and recall of the KG completion techniques on AIKG-1M. The results show that all CAFE-based variants outperform embedding-based techniques in precision, achieving notably higher values. Including features from the text embeddings also provides an important improvement over the base version of CAFE. Both SciCheck and the variants that improve the baseline using embedding-based features rank consistently among those with the highest precision for all relations, with the differences between them being very narrow.

Relation	# triples	SciCheck	Baseline	RoBERTa	SciBERT	Ontology	TransE	TransD	TransH	SimplE	ComplEx
usesMethod	460,723	**0.71**	0.67	0.70	0.70	0.70	0.38	0.49	0.36	0.56	0.56
usesOtherEntity	136,310	**0.72**	0.70	**0.72**	0.71	0.70	0.41	0.50	0.40	0.57	0.56
includesOtherEntity	113,678	**0.79**	0.77	**0.79**	0.78	0.76	0.43	0.49	0.41	0.55	0.55
narrower	107,811	**0.84**	0.80	0.83	0.81	0.76	0.49	0.49	0.48	0.57	0.56
usesMaterial	41,075	0.67	**0.68**	**0.68**	0.67	**0.68**	0.49	0.50	0.36	0.57	0.56
includesMethod	30,332	0.82	0.79	**0.83**	0.78	0.77	0.44	0.50	0.43	0.56	0.56
usesTask	22,341	**0.76**	0.73	0.75	0.68	0.73	0.49	0.49	0.48	0.55	0.54
evaluatesMethod	17,954	0.57	0.56	0.57	0.56	0.56	0.51	0.53	0.49	**0.60**	**0.60**
includesMaterial	10,190	0.70	0.67	**0.71**	0.67	0.65	0.49	0.49	0.40	0.57	0.56
usesMetric	7,749	0.67	0.64	**0.68**	0.60	0.66	0.41	0.50	0.37	0.56	0.56
includesTask	4,375	0.78	0.69	0.73	0.70	**0.81**	0.44	0.49	0.48	0.55	0.54
supportsTask	3,622	**0.87**	0.86	**0.87**	**0.87**	0.86	0.39	0.50	0.37	0.68	0.68
evaluatesOtherEntity	2,994	0.56	0.54	0.55	0.54	0.54	0.49	0.52	0.46	**0.60**	0.59
evaluatesTask	2,275	0.56	0.57	0.58	0.58	0.56	0.50	0.52	0.47	**0.59**	**0.59**
improvesMetric	1,860	0.84	0.81	0.84	0.81	**0.85**	0.54	0.52	0.53	0.67	0.67
supportsMethod	1,850	0.85	**0.86**	0.85	0.82	0.83	0.52	0.51	0.47	0.66	0.65
supportsOtherEntity	1,691	0.85	0.82	**0.87**	0.85	0.83	0.37	0.51	0.30	0.64	0.62
predictsOtherEntity	913	**0.84**	0.81	**0.84**	**0.84**	0.83	0.43	0.48	0.36	0.65	0.64
improvesMethod	814	0.67	0.68	**0.69**	0.68	**0.69**	0.39	0.52	0.44	0.67	0.66
improvesTask	639	**0.75**	0.72	**0.75**	0.73	0.71	0.43	0.48	0.42	0.66	0.66
Micro-average		**0.74**	0.70	0.73	0.72	0.71	0.42	0.49	0.39	0.56	0.56

Table 10.2: Precision values for SciCheck in AIKG-1M

Relation	# triples	SciCheck	Baseline	RoBERTa	SciBERT	Ontology	TransE	TransD	TransH	SimplE	ComplEx
usesMethod	460,723	0.27	0.22	0.28	0.21	0.19	0.20	0.69	0.16	0.71	**0.74**
usesOtherEntity	136,310	0.26	0.19	0.26	0.21	0.19	0.25	0.56	0.20	0.69	**0.73**
includesOtherEntity	113,678	0.26	0.16	0.25	0.20	0.16	0.27	0.67	0.22	0.75	**0.77**
narrower	107,811	0.39	0.17	0.32	0.20	0.28	0.67	0.51	0.59	0.70	**0.73**
usesMaterial	41,075	0.21	0.14	0.20	0.14	0.13	0.61	0.53	0.16	0.70	**0.72**
includesMethod	30,332	0.29	0.17	0.28	0.21	0.17	0.30	**0.75**	0.25	0.71	0.73
usesTask	22,341	0.28	0.15	0.28	0.20	0.15	0.65	0.65	0.61	0.77	**0.81**
evaluatesMethod	17,954	0.30	0.28	0.30	0.30	0.28	0.48	**0.76**	0.40	0.61	0.62
includesMaterial	10,190	0.25	0.16	0.24	0.16	0.17	0.60	0.67	0.20	0.73	**0.75**
usesMetric	7,749	0.18	0.10	0.17	0.12	0.10	0.25	0.63	0.18	**0.73**	**0.73**
includesTask	4,375	0.31	0.22	0.35	0.26	0.18	0.30	0.65	0.68	0.77	**0.80**
supportsTask	3,622	0.58	0.55	0.57	0.55	0.58	0.19	**1.00**	0.17	0.43	0.44
evaluatesOtherEntity	2,994	0.32	0.35	0.32	0.35	0.31	0.41	**0.77**	0.33	0.62	0.63
evaluatesTask	2,275	0.37	0.29	0.32	0.28	0.30	0.49	**0.86**	0.38	0.64	0.64
improvesMetric	1,860	0.41	0.41	0.42	0.40	0.37	0.67	**0.86**	0.49	0.44	0.45
supportsMethod	1,850	0.43	0.42	0.43	0.46	0.43	0.67	**0.77**	0.31	0.47	0.50
supportsOtherEntity	1,691	0.42	0.39	0.40	0.38	0.39	0.16	**0.63**	0.10	0.52	0.56
predictsOtherEntity	913	0.59	0.51	0.60	0.59	0.55	0.19	**0.66**	0.16	0.50	0.52
improvesMethod	814	0.33	0.29	0.34	0.32	0.33	0.19	**0.56**	0.34	0.44	0.48
improvesTask	639	0.70	0.65	0.70	0.69	0.65	0.29	**0.82**	0.25	0.49	0.48
Micro-average		0.28	0.20	0.28	0.21	0.20	0.31	0.65	0.24	0.71	**0.74**

Table 10.3: Recall values for SciCheck in AIKG-1M

The best performing method in terms of precision is the full version of SciCheck (0.74), followed by CAFE+RoBERTa (0.73), which can obtain better precision for some less common relationships. Interestingly, using text embeddings trained specifically on academic abstracts (SciBERT) yields a slightly worse performance than using the generic RoBERTa model. This may suggest that more general embeddings may sometimes produce better performance on KGs of research concepts, but this needs to be investigated further.

The *Ontology* variation, which includes one-hot type vectors and domain/range checking for the relation, only slightly improves the baseline. This is most likely due to the type-constrained way in which the negative triples were generated, since it already guarantees that the domain and range types of the relation are preserved.

The recall of SciCheck is naturally lower than that of the embedding-based approaches, in a typical precision-recall trade-off. However, this is acceptable since the main goal is to expand scientific Knowledge Graphs with correct triples, hence, a high precision is desirable. SciCheck also has a generally higher recall than all other CAFE variants. Consequently, the results suggest that SciCheck is the best performing technique for the task of reliably completing scientific Knowledge Graphs.

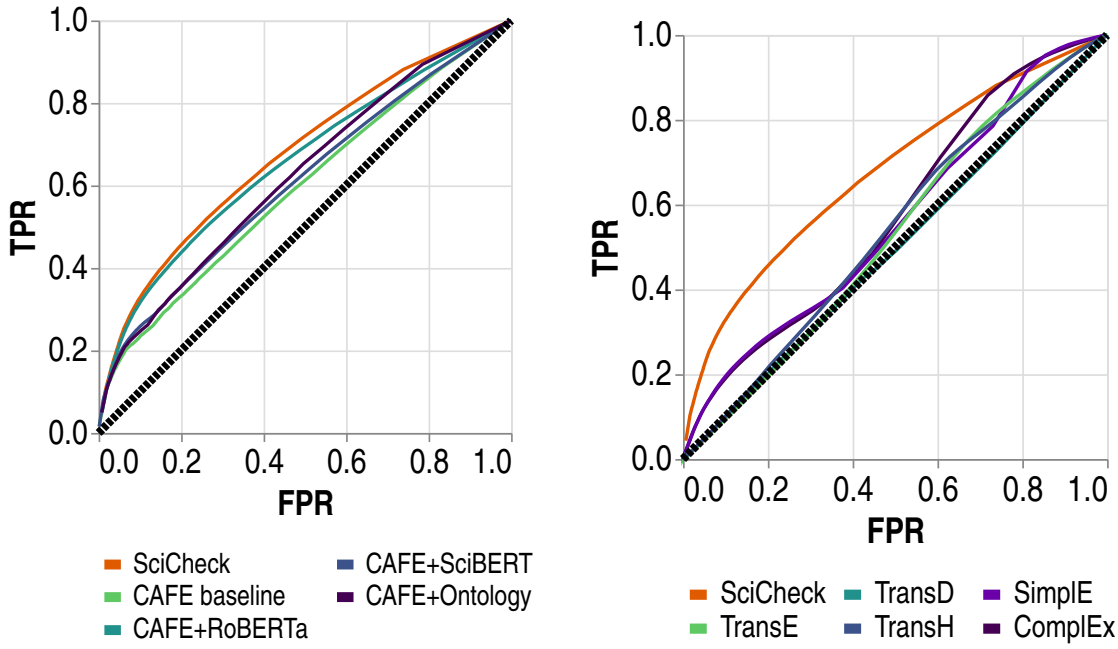
It is noteworthy that different relations can lead to very different performances. For instance, relations such as *narrower*, *supportsTask* and *supportsMethod* yield very good performance. Conversely, the methods under evaluation did not perform as well on relations such as *evaluatesTask* and *evaluatesOtherEntity*. This may depend on the number of relevant examples or the fact that some relations are inherently harder to predict.

In order to study the performance of the different techniques for all possible threshold values, we also report their corresponding ROC curves in Figure 10.2. This analysis confirms the previous findings:

- SciCheck outperforms all other methods under evaluation.
- Text embedding-based features significantly improve the baseline state-of-the-art methods.
- Ontology-based features provide slight further improvements.

In addition, Figure 10.2(b) confirms that SciCheck outperforms the standard state-of-the art methods regardless of the threshold.

To check whether the differences between the methods were statistically significant, we used DeLong's test [32] to compare the areas under two curves. The p-values obtained when comparing the ROC curve of SciCheck with the alternative methods in Figures 10.2(a) and 10.2(b) were all < 0.0001. This very high statistical confidence is due to the large number of observations, since the testing set of AIKG-1M includes more than 400,000 triples.



(a) SciCheck variants

(b) SciCheck and embedding techniques

Figure 10.2: ROC curves of the different methods on AIKG-1M

Table 10.4 shows the performance of the methods on AIKG-500, which are consistent with the previous findings. For the sake of brevity, here we do not report the results of all CAFE variants, which are in line with those obtained on AIKG-1M. Even in a smaller, manually annotated Knowledge Graph, SciCheck achieves a high precision, which confirms that it is suitable for completing scientific Knowledge Graphs.

		Precision						Recall					
Relation	# triples	SciCheck	TransE	TransD	TransH	SimplE	ComplEx	SciCheck	TransE	TransD	TransH	SimplE	ComplEx
includesMaterial	115	**0.69**	0.40	0.62	0.38	0.47	0.45	0.48	**1.00**	0.57	0.59	0.89	0.86
includesOtherEntity	93	**0.57**	0.50	0.55	0.38	0.49	0.45	**0.76**	0.24	0.74	0.42	0.74	0.61
usesMethod	88	**0.66**	0.39	0.46	0.42	0.53	0.59	**0.72**	0.44	0.53	0.53	0.50	0.61
usesMaterial	62	**0.86**	0.67	0.67	0.65	0.76	0.74	0.45	**0.95**	0.24	0.74	0.90	0.93
includesMethod	53	**0.61**	0.31	0.39	0.30	0.27	0.53	0.56	**0.72**	0.50	0.44	0.17	0.44
usesTask	46	**0.70**	0.48	0.55	0.48	0.57	0.51	0.73	**1.00**	0.77	**1.00**	0.95	0.86
usesOtherEntity	43	**0.78**	0.33	0.00	0.38	0.63	0.53	0.37	0.21	0.00	0.47	**0.53**	0.47
Micro-average		**0.68**	0.44	0.50	0.42	0.52	0.53	0.59	0.66	0.51	0.58	0.69	**0.70**

Table 10.4: Precision and recall values for AIKG-500

Table 10.5 reports the performance of all the techniques on five standard KGs for triple classification. The results show that SciCheck is able to outperform other techniques in almost all cases, thus being an effective triple classification tool for KGs of many different natures. They also confirm that completing scientific KGs is indeed a challenging task that requires specialized techniques, as the general-purpose embedding-based approaches yield worse results on KGs extracted from AI-KG in comparison to generic ones.

	Precision						Recall					
KG	SciCheck	TransE	TransD	TransH	SimplE	ComplEx	SciCheck	TransE	TransD	TransH	SimplE	ComplEx
FB13	**0.87**	0.60	0.64	0.67	0.31	0.41	**0.76**	0.66	0.67	0.67	0.10	0.16
WN11	**0.89**	0.45	0.48	0.47	0.57	0.44	**0.74**	0.53	0.58	0.60	0.33	0.22
WN18	**0.65**	0.40	0.39	0.19	0.51	0.49	0.81	0.56	0.60	0.12	**0.88**	0.87
WN18RR	**0.80**	0.31	0.36	0.35	0.62	0.73	**0.73**	0.51	0.53	0.44	0.36	0.40
NELL	0.66	0.47	0.50	0.45	0.68	**0.77**	**0.86**	0.53	0.62	0.62	0.31	0.34

Table 10.5: Micro-average precision and recall on four general KGs

In order to assess the scalability of our solution, Table 10.6 reports the seconds used by SciCheck to process the previously discussed graphs. To ensure statistical significance, we measured the runtime for each KG 10 times, and we report the average and the standard deviation for each one.

KG	Runtime
AIKG-1M	2,758.79 $\pm$ 37.27
AIKG-500	1,794.94 $\pm$ 12.58
FB13	9,400.10 $\pm$ 63.04
WN11	34.30 $\pm$ 0.28
WN18	55.59 $\pm$ 0.45
WN18RR	26.00 $\pm$ 0.14
NELL	4.33 $\pm$ 0.09

Table 10.6: SciCheck runtimes in seconds for all KGs (avg $\pm$ std)

Table 10.6 shows that the runtime ranges from a few seconds to over two hours according to the specific Knowledge Graph. These differences are caused by mainly two factors. First, the amount of distinct entities corresponds directly to the number of RoBERTa embeddings that have to be computed, which are typically quite time-consuming. Hence, a larger number of entities has a negative impact on runtime. Second, and most importantly, the specific topology of every KG affects the size of the neighborhoods of the entities, and thus also affects the time it takes to compute features on them. The case of FB13 is particularly noteworthy since, in contrast with the other KGs, it contains many entities with a very high cardinality. This causes the sizes of the entity neighborhoods to grow exponentially in size, resulting in longer runtimes.

Finally, in order to provide some insights on how these times compare to other state-of-the-art approaches, Table 10.7 reports their runtime in seconds compared to that of SciCheck for AIKG-1M.

Embedding-based KG completion approaches were run using $1,000$ iterations, as it is commonly done in related literature [17, 68, 72, 145]. SciCheck took considerably less time to run on the large AIKG-1M KG than its state-of-the-art counterparts. This suggests that it is able to complete scientific facts in a reasonable amount of time, thus making it efficient for its application to large-scale scientific Knowledge Graphs.

Technique	Runtime
SciCheck	2,758.79
TransE	7,147.52
TransD	13,871.79
TransH	10,134.41
SimplE	6,592.20
ComplEx	11,767.73

Table 10.7: Runtime comparison on AIKG-1M

10.4 Practical application: AI-KG

One practical application of SciCheck is the advancement and expansion of AI-KG [34], a comprehensive Knowledge Graph that encompasses information about research entities in the AI field. When it was introduced in late 2020, it contained roughly 14 million RDF triples and 1.2 million reified statements related to 800,000 entities extracted from 333,000 articles about AI. It defines 5 categories of entities (tasks, methods, materials, metrics, and others) connected by 27 relations (e.g., usesMaterial, evaluatesMethod, or supportsTask). The triples in AI-KG illustrate the associations between two entities based on their joint appearances in a group of scientific papers, for instance, *(sentiment_analysis, usesMaterial, twitter_data)*.

It should be emphasized that, in the context of AI-KG, the correctness of a triple depends on the content of its associated papers, i.e., a triple connected to a group of papers is considered to be true if said papers contain that statement. As a result, triples in AI-KG are constructed to represent particular statements made by researchers. For instance, the entity *sentiment_analysis* only captures the notion of sentiment analysis that existed in the original corpus of scientific articles on which AI-KG was built, and does not attempt to encompass all existing techniques for analyzing emotions and sentiments that are present, or under research, nowadays. A complete representation of such information would require elevating research entities into more abstract elements representing ontological knowledge, which is currently beyond the intended scope of AI-KG.

AI-KG was constructed by extracting entities and their relations from a corpus of scientific articles using Natural Language Processing (NLP) and Machine Learning (ML) techniques [35], using the following data pipeline:

1. Entity detection and extraction using transformer-based, domain-specific extractors [149].
2. Entity classification into the CSO ontology [122].
3. Relation detection and extraction through a number of NLP and ML techniques [4, 143, 149].

4. Fact identification and validation using AI-KG's own ontology.
5. Fact ranking using the number of research papers that contain the fact as a measurement of support.

The interested reader can find the fine-grained steps of this process in the relevant literature [34, 35]. At the time of the application of our technique, the entities contained within AI-KG were classified into one of the following types:

- *Task*: A specific challenge or piece of work to be completed as part of a research project.
- *Method*: A proposed approach or plan for accomplishing a research task.
- *Material*: Resources that are utilized for a research task, such as a dataset, image, or text corpus.
- *Metric*: Entities that can be measured and are used to evaluate the effectiveness of a research method.
- *OtherEntity*: A category that encompasses entities that do not fit into any of the previous terms.

The relations were created by clustering frequent verbs together, and asking human experts to define domain, range, and transitiveness restrictions. Some examples of object properties are *evaluatesMethod*, *includesMaterial*, or *usesMethod*.

AI-KG is currently in use by a number of organizations due to its ability to store structured information about the AI field, and has supported other related research, for example, entity extraction in the context of scientific articles [81], classifying such articles [63], and describing and managing competencies [61].

Despite AI-KG being a large-scale Knowledge Graph, extracting unstructured knowledge from natural language is a difficult and error-prone task. For this reason, AI-KG does not cover all well-known facts in the AI domain, either because they were not detected or not extracted correctly. As a consequence, AI-KG, as most automatically generated Knowledge Graphs, is incomplete. An example of this is the absence of the triple *(neural_network, usesMaterial, rdf_graph)* from AI-KG, even though RDF graphs are used as input for most of the neural network-based techniques that perform link prediction or triple classification, for example, CAFE.

Because of this, scientific Knowledge Graphs require specific methods for their completion [65]. Nevertheless, the most well-known methods that work on general-purpose Knowledge Graphs, like TransE, TransR, or RotatE, have not been successful in predicting triples with a high precision in AI-KG. As discussed in Section 10.3.3, even though these methods provide reasonable F1 values, they yield low precision values (usually between 40-60%). Applying these existing methods would lead to a considerable amount of inaccurate information being introduced in AI-KG. This, in turn, has been the motivation for this particular use case.

We applied SciCheck to AI-KG and, by setting a confidence threshold of 0.7, generated 303,760 new high-confidence facts. To do this, we used SciCheck to connect the most popular 500 entities that meet the domain and range restrictions imposed by the AI-KG ontology for every relation. As a result, many significant triples representing well-known facts that were missed by AI-KG's NLP extraction pipeline were materialized, such as:

(search_engine, includesMaterial, knowledge_base)
(f_measure, evaluatesMethod, neural_network)
(neural_network, usesMaterial, rdf_graph)
(recommender_system, usesMethod, predictive_model)

We collaborated with the original authors of AI-KG to make this improved version available online at `https://zenodo.org/record/7276434`.

10.5 Summary

In this chapter we introduced SciCheck, our proposal for scientific Knowledge Graph completion. SciCheck extends CAFE by adding a number of features specifically tailored for the scientific domain, namely, measuring the similarity of the embedded representations of two concepts in a certain way, as well as representing and applying ontological constraints on what is considered correct knowledge.

We have performed an extensive evaluation, comparing SciCheck with other state-of-the-art approaches and analyzing its individual components. The results show that SciCheck is able to classify triples representing scientific knowledge with a considerably higher precision than its counterparts, and that it is also more efficient in practice, requiring shorter runtimes than other proposals.

Finally, we have applied SciCheck to the large-scale AI-KG Knowledge Graph, which contains more than 1.2 million facts about the Artificial Intelligence domain. As a result of this process, we have expanded AI-KG with more than 300,000 additional high-confidence statements. In cooperation with the original authors of AI-KG, we have publicly released this improved version of it to the scientific community.

Part IV

Final Remarks

Chapter 11

Conclusions

"Be proud: you've come such a long way.
Be careful: there is so much further to go."

— *Letter to Marble 3*, Exurb1a

In this dissertation, we have presented a proposal to automatically complete large Knowledge Graphs. Our proposal takes as input the KG itself, and outputs a set of facts that it is missing and can be added back to it, enriching the graph with new knowledge. To achieve this, our proposal relies on a series of subsystems that address different problems of completing KGs.

First, it generates a set of candidate triples using CHAI, a technique that is able to efficiently generate rules to filter out incorrect knowledge. These rules combine aspects such as the distance between the entities of a triple, the domain and range restrictions of the relations in the KG, and the previous appearances of similar triples. Thanks to CHAI, our proposal is able to generate most of the missing facts in a KG while immediately discarding a large volume of incorrect knowledge.

Then, the candidate triples are evaluated using CAFE, our triple classification technique. CAFE defines a number of neighborhood-aware features that are able to accurately characterize the neighborhood of an entity, as well as the similarities between the neighborhoods of a pair of entities. These features are used to transform all triples in a KG, as well as possible candidate triples, into numeric vectors. CAFE then trains deep neural classification models using these vectors, learning to differentiate between correct and incorrect triples.

Both CAFE and CHAI have been evaluated using some of the most well-known Knowledge Graphs available today, and their theoretical performance has been shown

to be efficient and effective. However, to demonstrate the applicability in practice of our proposal, we have introduced SciCheck, a technique for completing scientific Knowledge Graphs that builds upon the previously discussed ones, extending them with capabilities specifically tailored for these KGs. We have applied SciCheck to AI-KG, a large-scale KG that contains more than 14 million triples representing 1.2 million statements about scientific facts regarding the Artificial Intelligence domain. This resulted in more than 300K additional facts being generated by SciCheck since they were missing in AI-KG, which were later included in a subsequent update of this graph.

As future work, we think that some shortcomings of the current proposals for KG completion deserve more attention: most of the existing proposals in the literature focus only on single-modal KGs, and they do not address the fact that many multi-modal and multi-media KGs currently exist and are still incomplete; more research should be carried out in the field of efficiently generating candidate triples, since there is still a very small body of work in this regard; and an emphasis should be put on developing methods with a high explainability that can scale up to very large KGs, since this is still an area of active research. Additionally, it would be interesting to further analyze the possibility of completing Knowledge Graphs using less studied or recently proposed approaches, such as reinforcement learning or graph neural networks.

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

[2] C. C. Aggarwal and C. Zhai. A survey of text classification algorithms. In *Mining Text Data*, pages 163–222. Springer, 2012. doi: 10.1007/978-1-4614-3223-4_6.

[3] I. Aliyu, A. Kana, and S. Aliyu. Development of knowledge graph for university courses management. *International Journal of Education and Management Engineering*, 10:1–10, 04 2020. doi: 10.5815/ijeme.2020.02.01.

[4] G. Angeli, M. J. J. Premkumar, and C. D. Manning. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the ACL and the 7th IJCNLP*, volume 1, pages 344–354, 2015.

[5] S. Angioni, A. A. Salatino, F. Osborne, D. R. Recupero, and E. Motta. AIDA: A knowledge graph about research dynamics in academia and industry. *Quant. Sci. Stud.*, 2(4):1356–1398, 2021. doi: 10.1162/qss_a_00162. URL `https://doi.org/10.1162/qss_a_00162`.

[6] D. Ayala, A. Borrego, I. Hernández, C. R. Rivero, and D. Ruiz. AYNEC: All you need for evaluating completion techniques in knowledge graphs. In *ESWC*, pages 397–411, 2019. doi: 10.1007/978-3-030-21348-0_26.

[7] D. Ayala, I. Hernández, D. Ruiz, and M. Toro. Tapon: A two-phase machine learning approach for semantic labelling. *Knowledge-Based Systems*, 163:931–943, 2019. doi: 10.1016/j.knosys.2018.10.017.

[8] D. Ayala, A. Borrego, I. Hernández, and D. Ruiz. A neural network for semantic labelling of structured information. *Expert Syst. Appl.*, 143, 2020. doi: 10.1016/j.eswa.2019.113053. URL `https://doi.org/10.1016/j.eswa.2019.113053`.

[9] D. Ayala Hernández. *On Data Engineering and Knowledge Graphs - A holistic, smarter approach to data enrichment*. PhD dissertation, 2020.

[10] I. Balazevic, C. Allen, and T. M. Hospedales. Hypernetwork knowledge graph

embeddings. In *Proceedings of the 28th International Conference on Artificial Neural Networks, ICANN 2019*, volume 11731, pages 553–565. Springer, 2019. doi: 10.1007/978-3-030-30493-5_52. URL https://doi.org/10.1007/978-3-030-30493-5_52.

[11] I. Bansal, S. Tiwari, and C. R. Rivero. The impact of negative triple generation strategies and anomalies on knowledge graph completion. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management, CIKM 2020*, pages 45–54. ACM, 2020. doi: 10.1145/3340531.3412023. URL https://doi.org/10.1145/3340531.3412023.

[12] T. Bansal, D. Juan, S. Ravi, and A. McCallum. A2N: attending to neighbors for knowledge graph inference. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019*, pages 4387–4392. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1431. URL https://doi.org/10.18653/v1/p19-1431.

[13] O. Bodenreider. The unified medical language system (UMLS): integrating biomedical terminology. *Nucleic Acids Res.*, 32(Database-Issue):267–270, 2004. doi: 10.1093/nar/gkh061. URL https://doi.org/10.1093/nar/gkh061.

[14] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *SIGMOD*, pages 1247–1250. ACM, 2008. doi: 10.1145/1376616.1376746.

[15] P. A. Bonatti, A. Hogan, A. Polleres, and L. Sauro. Robust and scalable linked data reasoning incorporating provenance and trust annotations. *J. Web Semant.*, 9(2):165–201, 2011. doi: 10.1016/j.websem.2011.06.003. URL https://doi.org/10.1016/j.websem.2011.06.003.

[16] A. Bordes and E. Gabrilovich. Constructing and mining web-scale knowledge graphs. In *SIGKDD*, pages 1967–1967. ACM, 2014. doi: 10.1145/2623330.2630803.

[17] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795, 2013.

[18] A. Bordes, X. Glorot, J. Weston, and Y. Bengio. A semantic matching energy function for learning with multi-relational data - application to word-sense disambiguation. *Machine Learning*, 94(2):233–259, 2014. doi: 10.1007/s10994-013-5363-6.

[19] A. Borrego, D. Ayala, I. Hernández, C. R. Rivero, and D. Ruiz. Generating rules to filter candidate triples for their correctness checking by knowledge graph completion techniques. In *Proceedings of the 10th International Conference*

on Knowledge Capture, K-CAP 2019, pages 115–122, 2019. doi: 10.1145/3360901.3364418.

[20] A. Borrego, D. Ayala, I. Hernández, C. R. Rivero, and D. Ruiz. CAFE: Knowledge graph completion using neighborhood-aware features. *Engineering Applications of Artificial Intelligence*, 103:104302, 2021. ISSN 0952-1976. doi: 10.1016/j.engappai.2021.104302. URL https://www.sciencedirect.com/science/article/pii/S0952197621001500.

[21] A. Borrego, M. Bermudo, F. Sola, D. Ayala, I. Hernández, and D. Ruiz. Silence — A web framework for an agile generation of RESTful APIs. *SoftwareX*, 20: 101260, 2022. doi: 10.1016/j.softx.2022.101260. URL https://doi.org/10.1016/j.softx.2022.101260.

[22] A. Borrego, D. Dessì, I. Hernández, F. Osborne, D. Reforgiato Recupero, D. Ruiz, D. Buscaldi, and E. Motta. Completing scientific facts in knowledge graphs of research concepts. *IEEE Access*, 10:125867–125880, 2022. doi: 10.1109/ACCESS.2022.3220241. URL https://doi.org/10.1109/ACCESS.2022.3220241.

[23] T. B. Brown, B. Mann, N. Ryder, et al. Language models are few-shot learners. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems, NeurIPS 2020*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

[24] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. In *Proceedings of the 2nd International Conference on Learning Representations, ICLR 2014*, 2014. URL http://arxiv.org/abs/1312.6203.

[25] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. Hruschka, and T. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 24, pages 1306–1313, 2010.

[26] D. Chakravarty, J. Gao, S. Phillips, et al. Oncokb: a precision oncology knowledge base. *JCO precision oncology*, 1:1–16, 2017.

[27] P. Chen, Y. Lu, V. W. Zheng, X. Chen, and B. Yang. KnowEdu: A system to construct knowledge graph for education. *IEEE Access*, 6:31553–31563, 2018. doi: 10.1109/ACCESS.2018.2839607. URL https://doi.org/10.1109/ACCESS.2018.2839607.

[28] B. Cheng, J. Zhu, and M. Guo. MultiJAF: Multi-modal joint entity alignment framework for multi-modal knowledge graph. *Neurocomputing*, 500:581–591, 2022. doi: 10.1016/j.neucom.2022.05.058. URL https://doi.org/10.1016/j.neucom.2022.05.058.

[29] E. F. Codd. A relational model of data for large shared data banks. *Commun.*

ACM, 13(6):377–387, 1970. doi: 10.1145/362384.362685. URL https://doi.org/10.1145/362384.362685.

[30] R. Das, A. Neelakantan, D. Belanger, and A. McCallum. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017*, pages 132–141. Association for Computational Linguistics, 2017. doi: 10.18653/v1/e17-1013. URL https://doi.org/10.18653/v1/e17-1013.

[31] S. Decker, S. Melnik, F. Van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann, and I. Horrocks. The semantic web: The roles of XML and RDF. *IEEE Internet computing*, 4(5):63–73, 2000.

[32] E. R. DeLong, D. M. DeLong, and D. L. Clarke-Pearson. Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. *Biometrics*, pages 837–845, 1988.

[33] T. Denoeux. A k-nearest neighbor classification rule based on dempster-shafer theory. *IEEE Trans. Syst. Man Cybern.*, 25(5):804–813, 1995. doi: 10.1109/21.376493. URL https://doi.org/10.1109/21.376493.

[34] D. Dessì, F. Osborne, D. R. Recupero, D. Buscaldi, E. Motta, and H. Sack. AI-KG: an automatically generated knowledge graph of artificial intelligence. In *Proceedings of the 19th International Semantic Web Conference, ISWC 2020*, volume 12507, pages 127–143. Springer, 2020. doi: 10.1007/978-3-030-62466-8_9. URL https://doi.org/10.1007/978-3-030-62466-8_9.

[35] D. Dessì, F. Osborne, D. R. Recupero, D. Buscaldi, and E. Motta. Generating knowledge graphs by employing natural language processing and machine learning techniques within the scholarly domain. *Future Generation Computer Systems*, 116:253–264, 2021. doi: 10.1016/j.future.2020.10.026.

[36] D. Dessì, D. R. Recupero, and H. Sack. An assessment of deep learning models and word embeddings for toxicity detection within online textual comments. *Electronics*, 10(7):779, 2021.

[37] D. Dessì, F. Osborne, D. Reforgiato Recupero, D. Buscaldi, and E. Motta. CS-KG: A large-scale knowledge graph of research entities and claims in computer science. In *ISWC 2022*, volume 13489, pages 678–696. Springer, 2022. doi: 10.1007/978-3-031-19433-7_39. URL https://doi.org/10.1007/978-3-031-19433-7_39.

[38] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel. Convolutional 2D knowledge graph embeddings. In *AAAI*, 2018.

[39] K. Do, T. Tran, and S. Venkatesh. Knowledge graph embedding with multiple relation projections. In *Proceedings of the 24th International Conference on Pattern*

Recognition, ICPR 2018, pages 332–337. IEEE Computer Society, 2018. doi: 10.1109/ICPR.2018.8545027. URL https://doi.org/10.1109/ICPR.2018.8545027.

[40] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *SIGKDD*, pages 601–610. ACM, 2014. doi: 10.1145/2623330.2623623.

[41] M. Fan, Q. Zhou, E. Chang, and T. F. Zheng. Transition-based knowledge graph embedding with relational mapping properties. In *Proceedings of the 28th Pacific Asia Conference on Language, Information and Computation, PACLIC 2014*, pages 328–337, 2014. URL https://aclanthology.org/Y14-1039/.

[42] S. Ferré. Link prediction in knowledge graphs with concepts of nearest neighbours. In *The Semantic Web - 16th International Conference, ESWC 2019*, volume 11503, pages 84–100. Springer, 2019. doi: 10.1007/978-3-030-21348-0_6. URL https://doi.org/10.1007/978-3-030-21348-0_6.

[43] L. Galárraga, C. Teflioudi, K. Hose, and F. M. Suchanek. Fast rule mining in ontological knowledge bases with AMIE+. *VLDB J.*, 24(6):707–730, 2015. doi: 10.1007/s00778-015-0394-1.

[44] L. A. Galárraga, C. Teflioudi, K. Hose, and F. M. Suchanek. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd International World Wide Web Conference, WWW 2013*, pages 413–422. International World Wide Web Conferences Steering Committee / ACM, 2013. doi: 10.1145/2488388.2488425. URL https://doi.org/10.1145/2488388.2488425.

[45] A. García-Durán, A. Bordes, and N. Usunier. Composing relationships with translations. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*, pages 286–290. The Association for Computational Linguistics, 2015. doi: 10.18653/v1/d15-1034. URL https://doi.org/10.18653/v1/d15-1034.

[46] A. García-Durán, A. Bordes, N. Usunier, and Y. Grandvalet. Combining two and three-way embeddings models for link prediction in knowledge bases. *CoRR*, abs/1506.00999, 2015. URL http://arxiv.org/abs/1506.00999.

[47] M. Gardner and T. Mitchell. Efficient and expressive knowledge base completion using subgraph feature extraction. In *EMNLP*, pages 1488–1498. The Association for Computational Linguistics, 2015. URL https://aclweb.org/anthology/D/D15/D15-1173.pdf.

[48] M. Gardner, P. P. Talukdar, B. Kisiel, and T. M. Mitchell. Improving learning and inference in a large knowledge-base using latent syntactic cues. In *Proceedings of*

the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, pages 833–838. ACL, 2013. URL https://aclanthology.org/D13-1080/.

[49] D. Gerber, D. Esteves, J. Lehmann, L. Bühmann, R. Usbeck, A. N. Ngomo, and R. Speck. Defacto - temporal and multilingual deep fact validation. *J. Web Semant.*, 35:85–101, 2015. doi: 10.1016/j.websem.2015.08.001. URL https://doi.org/10.1016/j.websem.2015.08.001.

[50] M. Glass and A. Gliozzo. A dataset for web-scale knowledge base population. In *ESWC*, pages 256–271. Springer, 2018. doi: 10.1007/978-3-319-93417-4_17.

[51] F. Gong, M. Wang, H. Wang, S. Wang, and M. Liu. SMR: medical knowledge graph embedding for safe medicine recommendation. *Big Data Res.*, 23: 100174, 2021. doi: 10.1016/j.bdr.2020.100174. URL https://doi.org/10.1016/j.bdr.2020.100174.

[52] S. Guan, X. Jin, Y. Wang, and X. Cheng. Shared embedding based neural networks for knowledge graph completion. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018*, pages 247–256. ACM, 2018. doi: 10.1145/3269206.3271704. URL https://doi.org/10.1145/3269206.3271704.

[53] H. Guo, J. Tang, W. Zeng, X. Zhao, and L. Liu. Multi-modal entity alignment in hyperbolic space. *Neurocomputing*, 461:598–607, 2021. doi: 10.1016/j.neucom.2021.03.132. URL https://doi.org/10.1016/j.neucom.2021.03.132.

[54] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, and Q. He. A survey on knowledge graph-based recommender systems. *IEEE Trans. Knowl. Data Eng.*, 34(8):3549–3568, 2022. doi: 10.1109/TKDE.2020.3028705. URL https://doi.org/10.1109/TKDE.2020.3028705.

[55] S. Guo, Q. Wang, L. Wang, B. Wang, and L. Guo. Jointly embedding knowledge graphs and logical rules. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016*, pages 192–202. The Association for Computational Linguistics, 2016. doi: 10.18653/v1/d16-1019. URL https://doi.org/10.18653/v1/d16-1019.

[56] S. Guo, Q. Wang, L. Wang, B. Wang, and L. Guo. Knowledge graph embedding with iterative guidance from soft rules. In *Proceedings of the 32ns Conference on Artificial Intelligence, AAAI 2018*, pages 4816–4823. AAAI Press, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16369.

[57] K. Guu, J. Miller, and P. Liang. Traversing knowledge graphs in vector space. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*, pages 318–327. The Association for Computational

Linguistics, 2015. doi: 10.18653/v1/d15-1038. URL https://doi.org/10.18653/v1/d15-1038.

[58] A. Hagberg, P. Swart, and D. S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[59] X. Han, S. Cao, X. Lv, Y. Lin, Z. Liu, M. Sun, and J. Li. Openke: An open toolkit for knowledge embedding. In *EMNLP*, pages 139–144, 2018.

[60] F. Hayes-Roth. *Building expert systems*, volume 1 of *Advanced book program*. Addison-Wesley, 1983. ISBN 0201106868. URL https://www.worldcat.org/oclc/09154010.

[61] N. Heist and P. Haase. Flexible and extensible competency management with knowledge graphs. In *ISWC (Posters/Demos/Industry)*, volume 2980. CEUR-WS.org, 2021. URL http://ceur-ws.org/Vol-2980/paper412.pdf.

[62] A. Hogan, E. Blomqvist, M. Cochez, et al. Knowledge graphs. *CoRR*, abs/2003.02320, 2020.

[63] F. Hoppe, D. Dessì, and H. Sack. Understanding class representations: An intrinsic evaluation of zero-shot text classification. In *Workshop on Deep Learning for Knowledge Graphs (DL4KG@ ISWC2021)*, volume 3034 of *CEUR Workshop Proceedings*, 2021.

[64] M. Y. Jaradeh, A. Oelen, K. E. Farfar, M. Prinz, J. D'Souza, G. Kismihók, M. Stocker, and S. Auer. Open research knowledge graph: Next generation infrastructure for semantic scholarly knowledge. In *Proceedings of the 10th International Conference on Knowledge Capture, K-CAP 2019*, pages 243–246. ACM, 2019. doi: 10.1145/3360901.3364435. URL https://doi.org/10.1145/3360901.3364435.

[65] M. Y. Jaradeh, K. Singh, M. Stocker, and S. Auer. Triple classification for scholarly knowledge graph completion. In *Proceedings of the 11th Conference on Knowledge Capture, K-CAP 2021*, pages 225–232, New York, NY, USA, 2021. doi: 10.1145/3460210.3493582. URL https://doi.org/10.1145/3460210.3493582.

[66] U. Javed, K. Shaukat, I. A. Hameed, F. Iqbal, T. M. Alam, and S. Luo. A review of content-based and context-based recommendation systems. *Int. J. Emerg. Technol. Learn.*, 16(3), 2021. doi: 10.3991/ijet.v16i03.18851. URL https://doi.org/10.3991/ijet.v16i03.18851.

[67] R. Jenatton, N. L. Roux, A. Bordes, and G. Obozinski. A latent factor model for highly multi-relational data. In *Proceedings of the 26th Annual Conference on Neural Information Processing Systems, NIPS 2012*, pages

3176–3184, 2012. URL https://proceedings.neurips.cc/paper/2012/hash/0a1bf96b7165e962e90cb14648c9462d-Abstract.html.

[68] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao. Knowledge graph embedding via dynamic mapping matrix. In *ACL (1)*, pages 687–696. The Association for Computer Linguistics, 2015.

[69] T. Jiang, T. Liu, T. Ge, L. Sha, B. Chang, S. Li, and Z. Sui. Towards time-aware knowledge graph completion. In *Proceedings of the 26th International Conference on Computational Linguistics, COLING 2016*, pages 1715–1724. ACL, 2016. URL https://aclanthology.org/C16-1161/.

[70] X. Jiang, Q. Wang, B. Qi, Y. Qiu, P. Li, and B. Wang. Attentive path combination for knowledge graph completion. In *Proceedings of The 9th Asian Conference on Machine Learning, ACML 2017*, volume 77 of *Proceedings of Machine Learning Research*, pages 590–605. PMLR, 2017. URL http://proceedings.mlr.press/v77/jiang17a.html.

[71] M. Kanehisa, S. Goto, M. Furumichi, M. Tanabe, and M. Hirakawa. KEGG for representation and analysis of molecular networks involving diseases and drugs. *Nucleic acids research*, 38(suppl_1):D355–D360, 2010.

[72] S. M. Kazemi and D. Poole. Simple embedding for link prediction in knowledge graphs. In *Advances in Neural Information Processing Systems*, 2018.

[73] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations, ICLR 2017*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

[74] F. Kong, R. Zhang, Y. Mao, and T. Deng. LENA: locality-expanded neural embedding for knowledge base completion. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*, pages 2895–2902. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33012895. URL https://doi.org/10.1609/aaai.v33i01.33012895.

[75] A. Krishnan. Making search easier: How amazon's product graph is helping customers find products more easily, 2018. URL https://blog.aboutamazon.com/innovation/making-search-easier.

[76] T. Kuhn, C. Chichester, M. Krauthammer, N. Queralt-Rosinach, R. Verborgh, G. Giannakopoulos, A. N. Ngomo, R. Viglianti, and M. Dumontier. Decentralized provenance-aware publishing with nanopublications. *PeerJ Comput. Sci.*, 2:e78, 2016. doi: 10.7717/peerj-cs.78. URL https://doi.org/10.7717/peerj-cs.78.

[77] O. Kuzelka and J. Davis. Markov logic networks for knowledge base completion: A theoretical analysis under the MCAR assumption. In *Proceedings of the*

35th Conference on Uncertainty in Artificial Intelligence, UAI 2019, volume 115 of *Proceedings of Machine Learning Research*, pages 1138–1148. AUAI Press, 2019. URL http://proceedings.mlr.press/v115/kuzelka20a.html.

[78] N. Lao, T. M. Mitchell, and W. 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, EMNLP 2011*, pages 529–539. ACL, 2011. URL https://aclanthology.org/D11-1049/.

[79] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6 (2):167–195, 2015. doi: 10.3233/SW-140134. URL https://doi.org/10.3233/SW-140134.

[80] K. Lei, J. Zhang, Y. Xie, D. Wen, D. Chen, M. Yang, and Y. Shen. Path-based reasoning with constrained type attention for knowledge graph completion. *Neural Comput. Appl.*, 32(11):6957–6966, 2020. doi: 10.1007/s00521-019-04181-1. URL https://doi.org/10.1007/s00521-019-04181-1.

[81] X. Li and M. Daoutis. Unsupervised key-phrase extraction and clustering for classification scheme in scientific publications. In *Proceedings of the Workshop on Scientific Document Understanding, SDU@AAAI 2021*, volume 2831 of *CEUR Workshop Proceedings*, 2021. URL https://ceur-ws.org/Vol-2831/paper2.pdf.

[82] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, volume 15, pages 2181–2187. AAAI Press, 2015.

[83] H. Liu, Y. Wu, and Y. Yang. Analogical inference for multi-relational embeddings. In *International conference on machine learning*, pages 2168–2178. PMLR, 2017.

[84] Q. Liu, H. Jiang, Z. Ling, S. Wei, and Y. Hu. Probabilistic reasoning via deep learning: Neural association models. *CoRR*, abs/1603.07704, 2016. URL http://arxiv.org/abs/1603.07704.

[85] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL http://arxiv.org/abs/1907.11692.

[86] J. Ma, Y. Qiao, G. Hu, Y. Wang, C. Zhang, Y. Huang, A. K. Sangaiah, H. Wu, H. Zhang, and K. Ren. ELPKG: A high-accuracy link prediction approach for knowledge graph completion. *Symmetry*, 11(9):1096, 2019. doi: 10.3390/sym11091096. URL https://doi.org/10.3390/sym11091096.

[87] F. Mahdisoltani, J. Biega, and F. Suchanek. Yago3: A knowledge base from

multilingual wikipedias. In *7th biennial conference on innovative data systems research*. CIDR Conference, 2014.

[88] C. Meilicke, M. W. Chekol, D. Ruffinelli, and H. Stuckenschmidt. Anytime bottom-up rule learning for knowledge graph completion. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI 2019*, pages 3137–3143. ijcai.org, 2019. doi: 10.24963/ijcai.2019/435. URL https://doi.org/10.24963/ijcai.2019/435.

[89] G. A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38(11): 39–41, 1995. doi: 10.1145/219717.219748.

[90] P. Minervini, M. Bosnjak, T. Rocktäschel, and S. Riedel. Towards neural theorem proving at scale. *CoRR*, abs/1807.08204, 2018. URL http://arxiv.org/abs/1807.08204.

[91] T. Mitchell, W. Cohen, E. Hruschka, et al. Never-ending learning. *Commun. ACM*, 61(5):103–115, 2018. doi: 10.1145/3191513.

[92] S. H. Muggleton and L. D. Raedt. Inductive logic programming: Theory and methods. *J. Log. Program.*, 19/20:629–679, 1994. doi: 10.1016/0743-1066(94)90035-3. URL https://doi.org/10.1016/0743-1066(94)90035-3.

[93] V. Nastase and B. Kotnis. Abstract graphs and abstract paths for knowledge graph completion. In *Proceedings of the 8th Joint Conference on Lexical and Computational Semantics, SEM@NAACL-HLT 2019*, pages 147–157. Association for Computational Linguistics, 2019. doi: 10.18653/v1/s19-1016. URL https://doi.org/10.18653/v1/s19-1016.

[94] D. Nathani, J. Chauhan, C. Sharma, and M. Kaul. Learning attention-based embeddings for relation prediction in knowledge graphs. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019*, pages 4710–4723. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1466. URL https://doi.org/10.18653/v1/p19-1466.

[95] A. Neelakantan, B. Roth, and A. McCallum. Compositional vector space models for knowledge base completion. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics, ACL 2015*, pages 156–166. The Association for Computer Linguistics, 2015. doi: 10.3115/v1/p15-1016. URL https://doi.org/10.3115/v1/p15-1016.

[96] S. Neumaier, J. Umbrich, J. X. Parreira, and A. Polleres. Multi-level semantic labelling of numerical values. In *ISWC*, volume 9981, pages 428–445, 2016. doi: 10.1007/978-3-319-46523-4_26.

[97] D. Q. Nguyen, T. Vu, T. D. Nguyen, D. Q. Nguyen, and D. Q. Phung. A capsule network-based embedding model for knowledge graph completion

and search personalization. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019*, pages 2180–2189. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1226. URL https://doi.org/10.18653/v1/n19-1226.

[98] M. Nickel, V. Tresp, and H. Kriegel. A three-way model for collective learning on multi-relational data. In L. Getoor and T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pages 809–816. Omnipress, 2011. URL https://icml.cc/2011/papers/438_icmlpaper.pdf.

[99] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich. A review of relational machine learning for knowledge graphs. *Proc. IEEE*, 104(1):11–33, 2016. doi: 10.1109/JPROC.2015.2483592. URL https://doi.org/10.1109/JPROC.2015.2483592.

[100] G. Niu, Y. Zhang, B. Li, P. Cui, S. Liu, J. Li, and X. Zhang. Rule-guided compositional representation learning on knowledge graphs. In *Proceedings of the 34th Conference on Artificial Intelligence, AAAI 2020*, pages 2950–2958. AAAI Press, 2020. URL https://ojs.aaai.org/index.php/AAAI/article/view/5687.

[101] N. F. Noy, Y. Gao, A. Jain, A. Narayanan, A. Patterson, and J. Taylor. Industry-scale knowledge graphs: lessons and challenges. *Commun. ACM*, 62(8):36–43, 2019. doi: 10.1145/3331166. URL https://doi.org/10.1145/3331166.

[102] B. Oh, S. Seo, and K. Lee. Knowledge graph completion by context-aware convolutional learning with multi-hop neighborhoods. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018*, pages 257–266. ACM, 2018. doi: 10.1145/3269206.3271769. URL https://doi.org/10.1145/3269206.3271769.

[103] P. G. Omran, K. Wang, and Z. Wang. Scalable rule learning via learning representation. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018*, pages 2149–2155. ijcai.org, 2018. doi: 10.24963/ijcai.2018/297. URL https://doi.org/10.24963/ijcai.2018/297.

[104] E. Palumbo, D. Monti, G. Rizzo, R. Troncy, and E. Baralis. entity2rec: Property-specific knowledge graph embeddings for item recommendation. *Expert Syst. Appl.*, 151:113235, 2020. doi: 10.1016/j.eswa.2020.113235. URL https://doi.org/10.1016/j.eswa.2020.113235.

[105] J. Pasternack and D. Roth. Knowing what to believe (when you already know something). In C. Huang and D. Jurafsky, editors, *Proceedings of the 23rd International Conference on Computational Linguistics, COLING 2010*, pages 877–885. Tsinghua University Press, 2010. URL https://aclanthology.org/C10-1099/.

[106] J. Pasternack and D. Roth. Making better informed trust decisions with generalized fact-finding. In T. Walsh, editor, *Proceedings of the 22nd International Joint Conference on Artificial Intelligence, IJCAI 2011*, pages 2324–2329. IJCAI/AAAI, 2011. doi: 10.5591/978-1-57735-516-8/IJCAI11-387. URL https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-387.

[107] H. Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, 8(3):489–508, 2017. doi: 10.3233/SW-160218.

[108] H. Paulheim and C. Bizer. Improving the quality of linked data using statistical distributions. *Int. J. Semantic Web Inf. Syst.*, 10(2):63–86, 2014. doi: 10.4018/ijswis.2014040104.

[109] C. Peng, F. Xia, M. Naseriparsa, and F. Osborne. Knowledge graphs: Opportunities and challenges. *Artif. Intell. Rev.*, 2023. doi: 10.1007/s10462-023-10465-9.

[110] R. J. Pittman, A. Srivastava, S. Hewavitharana, A. Kale, and S. Mansour. Making search easier: How amazon's product graph is helping customers find products more easily, 2017. URL https://www.ebayinc.com/stories/news/cracking-the-code-on-conversational-commerce/.

[111] T. Rebele, F. Suchanek, J. Hoffart, J. Biega, E. Kuzey, and G. Weikum. Yago: A multilingual knowledge base from wikipedia, wordnet, and geonames. In *Proceedings of the 15th International Semantic Web Conference, ISWC 2017*, pages 177–185. Springer, 2016.

[112] N. Reimers and I. Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL http://arxiv.org/abs/1908.10084.

[113] G. Rele, S. Adeshina, D. Bhargavi, K. Sindwani, V. S. Ravipati, , and M. Rhodes. Power recommendation and search using an imdb knowledge graph, 2022. URL https://aws.amazon.com/blogs/machine-learning/part-1-power-recommendation-and-search-using-an-imdb-knowledge-graph/.

[114] H. Ren, H. Dai, B. Dai, X. Chen, D. Zhou, J. Leskovec, and D. Schuurmans. SMORE: knowledge graph completion and multi-hop reasoning in massive knowledge graphs. In *Proceedings of the 28th Conference on Knowledge Discovery and Data Mining, SIGKDD 2022*, pages 1472–1482. ACM, 2022. doi: 10.1145/3534678.3539405. URL https://doi.org/10.1145/3534678.3539405.

[115] M. Richardson and P. M. Domingos. Markov logic networks. *Mach. Learn.*, 62 (1-2):107–136, 2006. doi: 10.1007/s10994-006-5833-1. URL https://doi.org/10.1007/s10994-006-5833-1.

[116] T. Rocktäschel and S. Riedel. End-to-end differentiable proving. In *Proceedings of the 30th Annual Conference on Neural Information Processing Systems, NIPS 2017*, pages 3788–3800, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/b2ab001909a8a6f04b51920306046ce5-Abstract.html.

[117] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022*, pages 10674–10685. IEEE, 2022. doi: 10.1109/CVPR52688.2022.01042. URL https://doi.org/10.1109/CVPR52688.2022.01042.

[118] A. Rossanez, J. C. dos Reis, and R. da Silva Torres. Representing scientific literature evolution via temporal knowledge graphs. In *Proceedings of the 6th Workshop on Managing the Evolution and Preservation of the Data Web (MEPDaW)*, volume 2821 of *CEUR Workshop Proceedings*, pages 33–42. CEUR-WS.org, 2020. URL https://ceur-ws.org/Vol-2821/paper5.pdf.

[119] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson, 2020. ISBN 9780134610993. URL http://aima.cs.berkeley.edu/.

[120] S. Sabour, N. Frosst, and G. E. Hinton. Dynamic routing between capsules. In *Proceedings of the 30th Conference on Neural Information Processing Systems, NIPS 2017*, pages 3856–3866, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/2cad8fa47bbef282badbb8de5374b894-Abstract.html.

[121] A. Sadeghian, M. Armandpour, P. Ding, and D. Z. Wang. DRUM: end-to-end differentiable rule mining on knowledge graphs. In *Proceedings of the 32nd Annual Conference on Neural Information Processing Systems, NeurIPS 2019*, pages 15321–15331, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/0c72cb7ee1512f800abe27823a792d03-Abstract.html.

[122] A. A. Salatino, F. Osborne, A. Birukou, and E. Motta. Improving editorial workflow and metadata quality at springer nature. In *The Semantic Web – ISWC 2019*, pages 507–525, Cham, 2019. Springer Int. Publishing. ISBN 978-3-030-30796-7.

[123] A. A. Salatino, F. Osborne, and E. Motta. Researchflow: Understanding the knowledge flow between academia and industry. In *Proceedings of the 22nd International Conference on Knowledge Engineering and Knowledge Management, EKAW 2020*, volume 12387, pages 219–236. Springer, 2020. doi: 10.1007/978-3-030-61244-3_16. URL https://doi.org/10.1007/978-3-030-61244-3_16.

[124] A. A. Salatino, T. Thanapalasingam, A. Mannocci, A. Birukou, F. Osborne, and E. Motta. The computer science ontology: A comprehensive automatically-generated taxonomy of research areas. *Data Intell.*, 2(3):379–416, 2020.

[125] M. S. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *Proceedings of the 15th Extended Semantic Web Conference, ESWC 2018*, volume 10843, pages 593–607. Springer, 2018. doi: 10.1007/978-3-319-93417-4_38. URL https://doi.org/10.1007/978-3-319-93417-4_38.

[126] A. Schrijver. *Theory of linear and integer programming*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1999. ISBN 978-0-471-98232-6.

[127] C. Shang, Y. Tang, J. Huang, J. Bi, X. He, and B. Zhou. End-to-end structure-aware convolutional networks for knowledge base completion. In *Proceedings of the 33rd Conference on Artificial Intelligence, AAAI 2019*, pages 3060–3067. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33013060. URL https://doi.org/10.1609/aaai.v33i01.33013060.

[128] T. Shen, F. Zhang, and J. Cheng. A comprehensive overview of knowledge graph completion. *Knowledge Based Systems*, 255:109597, 2022. doi: 10.1016/j.knosys.2022.109597. URL https://doi.org/10.1016/j.knosys.2022.109597.

[129] Y. Shen, P. Huang, M. Chang, and J. Gao. Modeling large-scale structured relationships with shared memory for knowledge base completion. In *Proceedings of the 2nd Workshop on Representation Learning for NLP, Rep4NLP@ACL 2017*, pages 57–68. Association for Computational Linguistics, 2017. doi: 10.18653/v1/w17-2608. URL https://doi.org/10.18653/v1/w17-2608.

[130] B. Shi and T. Weninger. Open-world knowledge graph completion. In *AAAI*, pages 1957–1964, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16055.

[131] S. Shrivastava. Bring rich knowledge of people, places, things and local businesses to your apps. Bing blogs., 2017. URL https://blogs.bing.com/search-quality-insights/2017-07/bring-rich-knowledge-of-people-places-things-and-local-businesses-to-your-apps.

[132] A. Singhal. Introducing the knowledge graph: things, not strings, 2012. URL https://www.blog.google/products/search/introducing-knowledge-graph-things-not/.

[133] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B. P. Hsu, and K. Wang. An overview of microsoft academic service (MAS) and applications. In *Proceedings of the 24th International World Wide Web Conference, WWW 2015*, pages 243–246. ACM, 2015. doi: 10.1145/2740908.2742839. URL https://doi.org/10.1145/2740908.2742839.

[134] R. Socher, D. Chen, C. D. Manning, and A. Ng. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*, pages 926–934, 2013.

[135] M. Speranskaya, M. Schmitt, and B. Roth. Ranking vs. classifying: Measuring knowledge base completion quality. In *AKBC*, 2020.

[136] D. Stepanova, M. H. Gad-Elrab, and V. T. Ho. Rule induction and reasoning over knowledge graphs. In *Reasoning Web. Learning, Uncertainty, Streaming, and Scalability, 2018*, volume 11078, pages 142–172. Springer, 2018. doi: 10.1007/978-3-030-00338-8_6. URL https://doi.org/10.1007/978-3-030-00338-8_6.

[137] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007*, pages 697–706. ACM, 2007. doi: 10.1145/1242572.1242667. URL https://doi.org/10.1145/1242572.1242667.

[138] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang. RotatE: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*, 2019.

[139] Z. H. Syed, M. Röder, and A. N. Ngomo. FactCheck: Validating RDF triples using textual evidence. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018*, pages 1599–1602. ACM, 2018. doi: 10.1145/3269206.3269308. URL https://doi.org/10.1145/3269206.3269308.

[140] Y. Tay, A. T. Luu, S. C. Hui, and F. Brauer. Random semantic tensor ensemble for scalable knowledge graph link prediction. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017*, pages 751–760. ACM, 2017. doi: 10.1145/3018661.3018695. URL https://doi.org/10.1145/3018661.3018695.

[141] C. F. Thorn, T. E. Klein, and R. B. Altman. Pharmgkb: the pharmacogenomics knowledge base. *Pharmacogenomics: Methods and Protocols*, pages 311–320, 2013.

[142] G. Töpper, M. Knuth, and H. Sack. DBpedia ontology enrichment for inconsistency detection. In V. Presutti and H. S. Pinto, editors, *I-SEMANTICS 2012 - 8th International Conference on Semantic Systems, I-SEMANTICS '12, Graz, Austria, September 5-7, 2012*, pages 33–40. ACM, 2012. doi: 10.1145/2362499.2362505. URL https://doi.org/10.1145/2362499.2362505.

[143] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 252–259, 2003.

[144] K. Toutanova, X. V. Lin, W. Yih, H. Poon, and C. Quirk. Compositional learning of embeddings for relation paths in knowledge base and text. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*.

The Association for Computer Linguistics, 2016. doi: 10.18653/v1/p16-1136. URL https://doi.org/10.18653/v1/p16-1136.

[145] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard. Complex embeddings for simple link prediction. In *ICML*, volume 48, pages 2071–2080, 2016.

[146] S. Vashishth, S. Sanyal, V. Nitin, N. Agrawal, and P. P. Talukdar. Interacte: Improving convolution-based knowledge graph embeddings by increasing feature interactions. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*, pages 3009–3016. AAAI Press, 2020. URL https://ojs.aaai.org/index.php/AAAI/article/view/5694.

[147] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Proceedings of the 30th Conference on Neural Information Processing Systems, NIPS*, pages 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

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

[149] D. Wadden, U. Wennberg, Y. Luan, and H. Hajishirzi. Entity, relation, and event extraction with contextualized span representations. 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 2019*, pages 5783–5788, 2019. doi: 10.18653/v1/D19-1585. URL https://doi.org/10.18653/v1/D19-1585.

[150] H. Wang, F. Zhang, M. Zhao, W. Li, X. Xie, and M. Guo. Multi-task feature learning for knowledge graph enhanced recommendation. In *Proceedings of the World Wide Web Conference, WWW 2019*, pages 2000–2010. ACM, 2019. doi: 10.1145/3308558.3313411. URL https://doi.org/10.1145/3308558.3313411.

[151] P. Wang, J. Han, C. Li, and R. Pan. Logic attention based neighborhood aggregation for inductive knowledge graph embedding. In *Proceedings of the 33rd Conference on Artificial Intelligence, AAAI 2019*, pages 7152–7159. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33017152. URL https://doi.org/10.1609/aaai.v33i01.33017152.

[152] Q. Wang, B. Wang, and L. Guo. Knowledge base completion using embeddings and rules. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence, IJCAI 2015*, pages 1859–1866. AAAI Press, 2015. URL http://ijcai.org/Abstract/15/264.

[153] R. Wang, Y. Yan, J. Wang, Y. Jia, Y. Zhang, W. Zhang, and X. Wang. AceKG:

A large-scale knowledge graph for academic data mining. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018*, pages 1487–1490. ACM, 2018. doi: 10.1145/3269206.3269252. URL https://doi.org/10.1145/3269206.3269252.

[154] Z. Wang and J. Li. Rdf2rules: Learning rules from RDF knowledge bases by mining frequent predicate cycles. *CoRR*, abs/1512.07734, 2015. URL http://arxiv.org/abs/1512.07734.

[155] Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, volume 14, pages 1112–1119. AAAI Press, 2014.

[156] Z. Wei, J. Zhao, K. Liu, Z. Qi, Z. Sun, and G. Tian. Large-scale knowledge base completion: Inferring via grounding network sampling over selected instances. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015*, pages 1331–1340. ACM, 2015. doi: 10.1145/2806416.2806513. URL https://doi.org/10.1145/2806416.2806513.

[157] C. Wise, V. N. Ioannidis, M. R. Calvo, X. Song, G. Price, N. Kulkarni, R. Brand, P. Bhatia, and G. Karypis. COVID-19 knowledge graph: Accelerating information retrieval and discovery for scientific literature. *CoRR*, abs/2007.12731, 2020. URL https://arxiv.org/abs/2007.12731.

[158] D. S. Wishart, C. Knox, A. C. Guo, D. Cheng, S. Shrivastava, D. Tzur, B. Gautam, and M. Hassanali. Drugbank: a knowledgebase for drugs, drug actions and drug targets. *Nucleic acids research*, 36(suppl_1):D901–D906, 2008.

[159] D. S. Wishart, C. Knox, A. C. Guo, et al. Hmdb: a knowledgebase for the human metabolome. *Nucleic acids research*, 37(suppl_1):D603–D610, 2009.

[160] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Networks Learn. Syst.*, 32(1): 4–24, 2021. doi: 10.1109/TNNLS.2020.2978386. URL https://doi.org/10.1109/TNNLS.2020.2978386.

[161] H. Xiao, M. Huang, Y. Hao, and X. Zhu. Transa: An adaptive approach for knowledge graph embedding. *CoRR*, abs/1509.05490, 2015. URL http://arxiv.org/abs/1509.05490.

[162] S. Xiong, W. Huang, and P. Duan. Knowledge graph embedding via relation paths and dynamic mapping matrix. In *Advances in Conceptual Modeling - ER 2018*, volume 11158, pages 106–118. Springer, 2018. doi: 10.1007/978-3-030-01391-2_18. URL https://doi.org/10.1007/978-3-030-01391-2_18.

[163] W. Xiong, M. Yu, S. Chang, X. Guo, and W. Y. Wang. One-shot relational learning for knowledge graphs. In *Proceedings of the 2018 Conference on Empirical Methods*

in Natural Language Processing, EMNLP 2018, pages 1980–1990. Association for Computational Linguistics, 2018. doi: 10.18653/v1/d18-1223. URL https://doi.org/10.18653/v1/d18-1223.

[164] K. Xu, L. Wang, M. Yu, Y. Feng, Y. Song, Z. Wang, and D. Yu. Cross-lingual knowledge graph alignment via graph matching neural network. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019*, pages 3156–3161. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1304. URL https://doi.org/10.18653/v1/p19-1304.

[165] V. Yadav and S. Bethard. A survey on recent advances in named entity recognition from deep learning models. *CoRR*, 2019.

[166] B. Yang, W. Yih, X. He, J. Gao, and L. Deng. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the 3rd International Conference on Learning Representations, ICLR*, 2015. URL http://arxiv.org/abs/1412.6575.

[167] F. Yang, Z. Yang, and W. W. Cohen. Differentiable learning of logical rules for knowledge base reasoning. In *Proceedings of the 30th Annual Conference on Neural Information Processing Systems, NIPS 2017*, pages 2319–2328, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/0e55666a4ad822e0e34299df3591d979-Abstract.html.

[168] Z. Ye, Y. J. Kumar, G. O. Sing, F. Song, and J. Wang. A comprehensive survey of graph neural networks for knowledge graphs. *IEEE Access*, 10:75729–75741, 2022. doi: 10.1109/ACCESS.2022.3191784. URL https://doi.org/10.1109/ACCESS.2022.3191784.

[169] W. Zhang, B. Paudel, L. Wang, J. Chen, H. Zhu, W. Zhang, A. Bernstein, and H. Chen. Iteratively learning embeddings and rules for knowledge graph reasoning. In *The World Wide Web Conference, WWW 2019*, pages 2366–2377. ACM, 2019. doi: 10.1145/3308558.3313612. URL https://doi.org/10.1145/3308558.3313612.

[170] W. Zhang, S. Deng, M. Chen, L. Wang, Q. Chen, F. Xiong, X. Liu, and H. Chen. Knowledge graph embedding in e-commerce applications: Attentive reasoning, explanations, and transferable rules. In *Proceedings of the 10th International Joint Conference on Knowledge Graphs IJCKG 2021*, pages 71–79. ACM, 2021. doi: 10.1145/3502223.3502232. URL https://doi.org/10.1145/3502223.3502232.

[171] Y. Zhang, M. Sheng, R. Zhou, Y. Wang, G. Han, H. Zhang, C. Xing, and J. Dong. HKGB: an inclusive, extensible, intelligent, semi-auto-constructed knowledge graph framework for healthcare with clinicians' expertise incorporated. *Inf. Process. Manag.*, 57(6):102324, 2020. doi: 10.1016/j.ipm.2020.102324. URL https://doi.org/10.1016/j.ipm.2020.102324.

[172] Z. Zhang, F. Zhuang, H. Zhu, Z. Shi, H. Xiong, and Q. He. Relational graph neural network with hierarchical attention for knowledge graph completion. In *Proceedings of the 34th Conference on Artificial Intelligence, AAAI 2020*, pages 9612–9619. AAAI Press, 2020. URL https://ojs.aaai.org/index.php/AAAI/article/view/6508.

[173] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1: 57–81, 2020. doi: 10.1016/j.aiopen.2021.01.001. URL https://doi.org/10.1016/j.aiopen.2021.01.001.