Expressive Rule Discovery for Knowledge Graph Refinement

Armita Khajeh Nassiri

université PARIS-SACLAY

Expressive Rule Discovery for Knowledge Graph Refinement

Découverte de règles expressives pour le raffinement de graphes de connaissances

Thèse de doctorat de l'Université Paris-Saclay

École doctorale n° 580 Sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat: Informatique
Graduate school: Informatique et sciences du numérique
Référent: Faculté des sciences d'Orsay

Thèse préparée dans l'unité de recherche Laboratoire interdisciplinaire des sciences du numérique LISN (Université Paris-Saclay, CNRS), sous la direction de **Fatiha SAIS** Professeure à l'Université Paris-Saclay, le co-encadrement de **Nathalie PERNELLE** Professeure à l'Université Sorbonne Paris Nord et le co-encadrement de **Gianluca QUERCINI** Professeur associé à CentraleSupélec.

Thèse présentée et soutenue à Paris Saclay, le 13 juillet 2023, par

Armita KHAJEH NASSIRI

Composition du Jury

Membres du jury avec voix délibérative

Maguelonne TEISSEIRE Directrice de recherche, INRAE Montpellier	Présidente
Jérôme DAVID Maître de conférences,HDR, INRIA Grenoble & Université de Grenoble	Rapporteur & Examinateur
Arnaud SOULET Maître de conférences, HDR, Université de Tours	Rapporteur & Examinateur
Sylvain CHEVALLIER Professeur, Université Paris Saclay	Examinateur
Luis GALARRAGA Chargé de recherche, INRIA Rennes	Examinateur
Hala SKAF MOLLI Professeure, Université de Nantes	Examinatrice

Thèse de doctorat
NNT : 2023UPASG045

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Découverte de règles expressives pour le raffinement de graphes de connaissances

Mots clés: Graphes de connaissances, Fouille de Règles, Raffinement des graphes de connaissances

Résumé: Les graphes de connaissances (KG) sont des structures de graphes hétérogènes représentant des faits dans un format lisible par une machine. Ils trouvent des applications dans des tâches telles que la réponse automatique aux questions, la désambiguïsation et liaison d'entités. Cependant, les graphes de connaissances sont intrinsèquement incomplets et il est essentiel de les raffiner pour améliorer leur qualité. Pour compléter le graphe de connaissances, il est possible de prédire les liens manquants dans un graphe de connaissances ou d'intégrer des sources externes. En extrayant des règles du graphe de connaissances, nous pouvons les exploiter pour compléter le graphe tout en fournissant des explications. Plusieurs approches ont été proposées pour extraire efficacement des règles. Or, la littérature manque de méthodes efficaces pour incorporer des prédicats numériques dans les règles. Pour répondre à cette lacune, nous proposons REGNUM, qui permet d'extraire des règles numériques avec des contraintes d'intervalle. REGNUM s'appuie sur les règles générées par un système d'extraction de règles existant et les enrichit en incorporant des prédicats numériques guidés par des mesures de qualité. En outre, la nature interconnectée des données web offre un potentiel significatif pour compléter et raffiner les KG, par exemple, par le liage des données, qui consiste à trouver des liens d'identité entre des entités de KG différents. Nous présentons RE-miner, une approche qui extrait des expressions référentielles (RE) pour une classe dans un graphe de connaissances. Les REs sont des règles qui ne s'appliquent qu'à une seule entité. Elles facilitent la découverte de connaissances et permettent de lier les données de manière explicable. De plus, nous visons à explorer les avantages et les opportunités de l'affinage des modèles linguistiques pour combler le fossé entre les KG et les données textuelles. Nous présentons GilBERT, qui exploite le fine-tuning sur des modèles linguistiques tels que BERT en optimisant une fonction de coût par triplet pour les tâches de prédiction de relation et de classification de triple. En prenant en compte ces défis et en proposant des approches novatrices, cette thèse contribue au raffinement des KG, en mettant particulièrement l'accent sur l'explicabilité et la découverte de connaissances. Les résultats de cette recherche ouvrent la voie à de nouvelles questions de recherche qui font progresser vers des KG de meilleure qualité.

Title: Expressive rule discovery for knowledge graph refinement

Keywords: Knowledge graph, Rule mining, Knowledge graph refinement, Knowledge discovery

Abstract:
Knowledge graphs (KGs) are heterogeneous graph structures representing facts in a machine-readable format. They find applications in tasks such as question answering, disambiguation, and entity linking. However, KGs are inherently incomplete, and refining them is crucial to improve their effectiveness in downstream tasks. It's possible to complete the KGs by predicting missing links within a knowledge graph or integrating external sources and KGs. By extracting rules from the KG, we can leverage them to complete the graph while providing explainability. Various approaches have been proposed to mine rules efficiently. Yet, the literature lacks effective methods for effectively incorporating numerical predicates in rules. To address this gap, we propose REGNUM, which mines numerical rules with interval constraints. REGNUM builds upon the rules generated by an existing rule mining system and enriches them by incorporating numerical predicates guided by quality measures. Additionally, the interconnected nature of web data offers significant potential for completing and refining KGs, for instance, by data linking, which is the task of finding sameAs links between entities of different KGs. We introduce RE-miner, an approach that mines referring expressions (REs) for a class in a knowledge graph and uses them for data linking. REs are rules that are only applied to one entity. They support knowledge discovery and serve as an explainable way to link data. We employ pruning strategies to explore the search space efficiently, and we define characteristics to generate REs that are more relevant for data linking. Furthermore, we aim to explore the advantages and opportunities of fine-tuning language models to bridge the gap between KGs and textual data. We propose GilBERT, which leverages fine-tuning techniques on language models like BERT using a triplet loss. GilBERT demonstrates promising results for refinement tasks of relation prediction and triple classification tasks. By considering these challenges and proposing novel approaches, this thesis contributes to KG refinement, particularly emphasizing explainability and knowledge discovery. The outcomes of this research open doors to more research questions and pave the way for advancing towards more accurate and comprehensive KGs.

To my parents, Anousheh and Sina

REMERCIEMENTS

What an incredible journey it has been! This ride wouldn't have been possible without Fatiha and Nathalie. They've granted me the opportunity to embark on a Ph.D., expand my knowledge and conduct research, grow, and also enrich my personal knowledge graph. And oh, the extraordinary entities I've crossed paths with! Whether at the lab, during conferences or summer schools, or teaching activities. There are too many to mention here, but let's just say they've added so much richness to my journey.

Fatiha, you've built a team where everyone feels included and heard. Thank you for the countless hours of working together, the freedom you gave me to explore different ideas and directions, and for creating a second family that holds a special place in my heart.

Nathalie, you're an inspiration, and your love for research is contagious. Thank you for the late nights and weekends that we worked together and for your dedication to every detail. We've shared lots of laughter and moments of doubt. Thanks for always having valued my thoughts (although oftentimes you were proven right).

Gianluca, thank you for being genuine and for giving me the opportunity to teach. It was a pleasure to experience life on the other side of the classroom. I also thank Bpifrance and IBM France for funding this Ph.D. (AIDA project).

To Arnaud and Jérôme, your feedback on my manuscript was invaluable. Thanks for generously taking the time to read my thesis. To all my jury members, thank you for joining me on this defense. It's truly an honor for me. And a double thanks to Luis, who provided insightful guidance during the mid-term evaluation of my Ph.D., and also to Fabian Suchanek, whose master's course sparked my interest and led me on this Ph.D. adventure.

To my incredible LaHDAK colleagues and friends, thank you for the years of friendship, idea exchanges, and coffee and tea pauses. Alexandra, Lucas,

and Taha – you're my loyal camarades on this journey. We've weathered the storms of the pandemic and (many) other crises together, shared the office, lots of laughs, all the while supporting and learning from one another. Cheers to our enduring friendship!

To my newer buddies in the team, Alan, Jonathan, and Thibaut, you've injected a fresh energy into the team. I'll miss our pool, baby foot, basketball games, and attempts at solving those random integral questions. Thank you for patiently explaining French slangs (and also French history) that even left you scratching your heads at times. As the younger generation (so says my KG), I'm deeply proud of you and eagerly await your defenses.

To the friends I've made over the years (and oh, years) at Saclay, I extend my gratitude. Pawel and Huong, from being interns at LIX to embarking on our Ph.D. journeys. Yuting, Aram, Kosar, Konrad, Olivier, Joe, Marc, Melan, Shima, Jindra, Martin, Anousheh joon, thank you for the shared moments here and there that have added sparks of joy to my life.

And to my dear friends a bit further distance-wise but never far in my heart – Parsia, Parham, Soheil, Shervin, Mahta, Khashi, and of course, the Akhavans, Nasim, Yassi, Farzaneh joon, and Ostad-Akhavan – your friendship is cherished. Even if we don't see each other often, the light you bring to my life shines bright.

And the rule I've mined with the highest confidence in my KG: Sina, your presence is my happiness. Thank you for sharing your life with me. I treasure every moment and appreciate your boundless patience and support and many hours of brainstorming during my Ph.D. years, before and hopefully after.

To my family, you are my rock. Mom and Dad (Fariba & Mehrdad), thank you for supporting me in everything I do and for instilling in me the values of independence and freedom. I love you both, and I'm proud to be your daughter. Anousheh, you're not just a sister in my knowledge graph; you're a part of me. I'm grateful for the many journeys you took between Paris and Brussels, using every conceivable mode of transportation.

And lastly, I thank the little girl I once was. We've come such a long way. Keep dreaming, exploring, and loving.

OVERVIEW

CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

0.1 MOTIVATION

We live in an age of rapid change and transformations, where data, technology, innovation, and artificial intelligence dominate, and large language models (LLMs) and generative AI have emerged as significant players. Conversations in our daily lives revolve around these topics, shaping our experiences. Witnessing constant evolution and revolutions, we find ourselves in an exciting era. However, as these advancements become deeply ingrained in our lives, ethical concerns and demand for accurate and explainable solutions remain essential.

For months now, central attention has been revolving around the implications of language models like GPT-3. How will they transform our lives? How close are we to achieving intelligent AI systems? At the same time, we acknowledge the limitations of these systems, such as hallucinations and their potential for providing incorrect information while sounding convincingly fluent. Moreover, the biases embedded within these models are a topic of concern. LLMs can also provide outdated information and lack precision in their query responses. They are overwhelmed by the amount of data they've been trained on and may be unable to memorize the complete set of items that should be obtained, as is done with databases. Additionally, they cannot offer explainability, meaning they cannot explain the reasoning behind the information they provide.

There can be potential to enhance the quality of answers provided by LLMs while achieving some level of explainability by coupling LLMs with knowledge graphs which can store factual information and offer reasoning capabilities. Knowledge graphs have extensive applications in question-answering, rule mining, disambiguation, and entity-linking tasks. For instance, when we query a search engine such as Google to ask, "Who is the CEO of OpenAI", it

directly replies, "Sam Altman". It may also provide an info box specifying his birth date, education, etc., all retrieved from a knowledge graph. ChatGPT, however, responds to the same question: "As of my knowledge cutoff in September 2021, the CEO of OpenAI is Sam Altman. However, please note that executive positions can change over time, so it's always a good idea to verify the latest information". Showing how the information provided by LLMs can be outdated.

Knowledge graphs are heterogeneous graph structures that contain information representing facts in a machine-readable format. The industry and academia have witnessed the popularity of large-scale knowledge graphs, such as those developed by Amazon, Google, Microsoft, YAGO, and Wikidata. These knowledge graphs serve as repositories for structured facts, represented as triples, for instance, $(GPT3, developedby, OpenAI)$. Despite containing millions of facts, knowledge graphs are incomplete and contain erroneous data. Thus, refining the KGs to improve their usefulness in various applications such as the ones mentioned above is essential. To complete the knowledge graph, research focuses on predicting missing links within a knowledge graph or integrating external sources and KGs.

In the knowledge graph, there are patterns that can be extracted as logical rules, supported by quality measures like confidence. For example, the rule "if someone is the CEO of a company, then they work in that company". This example also demonstrates the potential of mining such rules to complete the missing information in the knowledge graph with the advantage of explainibility. Various approaches have been proposed in the literature to mine such rules efficiently, considering different language biases and expressive levels and employing pruning and optimization strategies to cope with the exponential search space. However, existing approaches encounter limitations when it comes to handling numerical predicates. These limitations include being restricted to constants or only allowing simple comparisons between constant values. Currently, no approaches in the literature can effectively mine more complex and interesting numerical rules that involve constraints on intervals. Such constraints can be highly relevant in domains such as finance, public health, or life sciences. They can help uncover useful information, such as the increased likelihood of a patient with heart disease having taken mood

stabilizers for over five years. This gap in the literature motivates us to study this problem and the impact numerical rules can bring for KG completion.

Moreover, the data on the web is interconnected, giving rise to significant potential for completing and refining KGs. The linked open data (LOD) is a global network of interconnected knowledge graphs that continues to grow. LOD enables the integration of several knowledge graphs allowing for a connected and accessible web of data (Bizer et al., 2018). Within LOD, identity links (such as the $owl:sameAs$ relation) establish connections between entities, signifying that two IRIs represent the same real-world entity (Beek et al., 2018). Hence, data linking, which is the task of finding such sameAs links between entities, is a vital refinement task. Various data-linking techniques exist in the literature, with several methods relying on linkage rules such as keys. Some keys are valid within a class but do not involve constants (*e.g.*, for universities, their location accompanied by their year of establishment forms a key). Others are valid for class expressions that may involve constants (*e.g.*, for French universities, the number of graduate students is key). While efficient mining techniques have been developed to extract these rules from a knowledge graph, the use of such link rules often results in a low recall, meaning that many entities do not have corresponding links identified. When instantiated, these rules or keys provide unique references to individual entities and can be considered as referring expressions for them (*e.g.*, the university that is located in Cambridge, Massachusetts, and was established in 1636, is Harvard university). Hence each key can generate a set of referring expressions. However, the literature has not explored referring expressions as graph patterns that uniquely identify an instance in the KG for data linking that a key cannot express (*e.g.*, The University of Caen is the university where Pierre-Simon de Laplace studied). The question arises whether such referring expressions can be beneficial for data linking and contribute to an increase in recall. This question will be further investigated in this thesis.

Moreover, there are other refinement tasks such as relation prediction, which predicts the relation holding between two entities, or triple classification, beneficial for fact-checking, which is a binary classification problem to predict whether a triple is correct. Rules and other techniques have been extensively used for these downstream tasks; nevertheless, very few have tried to leverage the potential of language models (LMs). The motivation behind our research is

to investigate the effectiveness of using language models for these downstream tasks. We seek to explore the potential advantages and opportunities that arise from fine-tuning language models and adapting them to address various tasks that bridge the gap between knowledge graphs and textual data.

This thesis is dedicated to addressing the aforementioned research questions and overcoming the limitations identified.

0.2 CONTRIBUTION AND THESIS OUTLINE

This thesis aims to enhance the expressivity of existing approaches in rule mining on knowledge graphs, to generate more effective rules for knowledge graph completion and data linking. We also embark on an initial exploration of integrating information from the knowledge graph with language models.

Fundamentals. Chapter 1 provides an overview of the key concepts related to knowledge graphs and their representation. It briefly explores the integration of knowledge graphs through notions of ontology matching and data linking. The chapter also delves into rule mining on knowledge graphs, defining logical rules, and introducing some quality measures for evaluating them. Additionally, it explores the use of vector representations for entities and relations, *i.e.*, knowledge graph embeddings, highlighting the characteristics and procedures involved in obtaining such representations.

Related work. In Chapter 2, we consider the refinement tasks of knowledge graph completion and data linking. Knowledge graph completion can be achieved using logical rules mined from the knowledge graph, a symbolic approach that offers explainability and interpretability. We explore the current state-of-the-art rule-mining techniques, examining their language biases, pruning strategies, and individual characteristics. Then, we explore subsymbolic knowledge graph embedding techniques as a popular method for knowledge graph completion. We categorize KG embedding techniques based on their scoring functions and discuss the characteristics of each of the methods. Moreover, we take a close look at data linking as a refinement task, specifically examining key-based approaches in the literature. Additionally, we briefly discuss KG embedding-based methods for data linking.

Numerical Rule Discovery on Knowledge graphs. After an extensive review of the related work on rule mining in Chapter 2, a clear gap emerges in effectively incorporating numerical predicates, such as age or population, into the rule mining process. This arises from the large search space associated with numerical predicates as they take many different distinct values. To address this gap, we propose a novel approach called REGNUM in chapter 3. REGNUM builds upon the rules generated by an existing rule mining system and enriches them by incorporating numerical predicates that constrain the values to specific intervals. The process is guided by quality measures that assess the confidence and significance of the rules. REGNUM is the first approach to focus specifically on numerical rules in knowledge graphs, uncovering patterns related to membership or non-membership within an interval.

Referring Expression Mining on Knowledge graphs. Chapter 4 introduces RE-miner, an approach for mining referring expressions on knowledge graphs. Referring expressions can be regarded as rules that are only applied to a single entity. We employ pruning strategies to explore the search space efficiently. We also define characteristics such as minimality and diversity to help generate referring expressions that are more relevant for data-linking tasks. We conduct comprehensive experiments to demonstrate the advantages of using referring expressions in data linking.

Knowledge Graph Refinement based on Triplet BERT-Networks. In Chapter 5, we present a novel approach that involves fine-tuning LMs using knowledge graph information to generate clusters of desired entities, relations, and referring expressions (as future work). We leverage the embedding space and measure the proximity between embedded entities/relations in order to perform downstream tasks such as relation prediction, and triple classification. This approach can give rise to new possibilities for enhancing the performance and capabilities of knowledge graph-driven tasks such as implicit entity linking.

Conclusion. Chapter 5.4 summarizes the thesis and elaborates on future perspectives.

1

FUNDAMENTALS

1.1 INTRODUCTION

Knowledge graphs (KGs) are repositories of facts and model information about entities and the relations between them. They are usually, not always, accompanied by ontologies, which are also known as schemas. When an ontology is present, the term "Knowledge Graph" is often referred to as "Knowledge Base" (KB). An ontology provides domain knowledge that facilitates logical inference and reasoning, detects inconsistencies, and infers implicit knowledge.

KGs are a realization of the semantic web, which emerged from the need to make sense of the vast amounts of data available on the web, enabling machines to read and interpret it and ushering in the data and knowledge era. The Semantic Web addresses the challenge of "things, not strings," facilitating more advanced searches on the web, disambiguation, providing intelligent answers, and linking data. The World Wide Web Consortium (W3C)[1] is the main international standards organization for the World Wide Web. It is responsible for developing and maintaining the web's standards, including HTML, CSS, and JavaScript. The W3C also plays a key role in developing Semantic Web technologies, such as RDF, OWL, and SPARQL, which provide the foundation for building and sharing semantic data on the web[2].

This chapter recalls essential concepts and notations used throughout this thesis. In this section 1.2, we provide an overview of the RDF graph data model and explore how it can be accompanied by ontological knowledge using RDF Schema or more expressive languages. In section 1.3, emphasizing the importance of integrating data and ontology from diverse sources, we briefly

1 https://www.w3.org/
2 https://www.w3.org/standards/semanticweb/

touch on data linking and ontology matching notions. Moreover, we introduce logical rules that facilitate inductive reasoning on KGs and quality measures to assess their performance in section 1.4. Furthermore, in section 1.5, we delve into knowledge graph embedding, an alternative way to represent entities and relations as vectors, and the procedures involved in obtaining them.

1.2 KNOWLEDGE REPRESENTATION

The semantic web relies on the Resource Description Framework (RDF) data model, the World Wide Web Consortium (W3C) standard for representing web data, which allows making statements about entities. In the RDF graph, a triple (s, p, o) is a directed edge from the subject s to the object o labeled with the property p.

Definition 1. RDF Knowledge Graph. *In an RDF knowledge graph $\mathcal{G}$, a collection of facts $\mathcal{F}$ is represented as triples of the form $\{$(subject, predicate, object) $\mid$ subject $\in \mathcal{I} \cup \mathcal{B}$, predicate $\in \mathcal{P}$, object $\in \mathcal{I} \cup \mathcal{L} \cup \mathcal{B}$ $\}$ where the set of entities is denoted as $\mathcal{I}$, the set of predicates is denoted as $\mathcal{P}$. The set of literals is denoted as $\mathcal{L}$.*

Entities $\mathcal{I}$, such as `https://dbpedia.org/page/Rumi`, are uniquely identified by **Internationalized Resource Identifiers (IRIs)**. A namespace can be defined within the RDF graph to simplify these IRIs, with the prefix used as an abbreviation. For example, by defining the namespace *db* for `https://dbpedia.org/page/`, we can refer to Rumi as *db:Rumi*. Literals $\mathcal{L}$, such as "1212" and "Saadi Shīrāz "[3], represent primitive type constants like numbers or strings. Blank nodes $\mathcal{B}$ represent unknown IRIs or literals[4]; in other words, it denotes the existence of an entity.

Terminology-wise, in a triple or fact, the subject is also referred to as the *head*, the predicate as the *property* or *relation*, and the object as the *tail*. In this thesis, we use these terms interchangeably.

3 In this thesis, we distinguish literals using quotation marks (" ").
4 We do not consider blank nodes in this thesis.

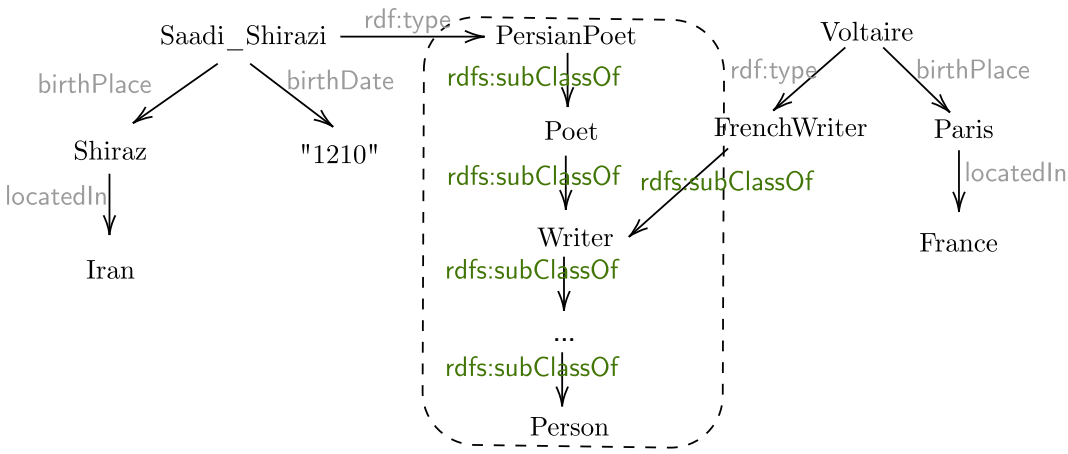

Figure 1.1.: Part of a knowledge graph, an example

RDF is a powerful tool for representing data on the web. However, it has its limitations when modeling more complex structures such as classes, hierarchies, or domain and range constraints. Ontologies, on the other hand, provide a more formal and structured way of representing knowledge. As described in (Gruber, 1995), an ontology is an explicit specification of a conceptualization. The conceptualization typically involves key concepts and relationships within a particular domain and organizing them into a hierarchical structure that reflects their dependencies. RDF Schema is commonly employed for this purpose.

RDF Schema (RDFS). RDFS is an extension of the RDF data model that adds semantic capabilities by introducing the concept of **class** and supporting ontological constraints. A class is a named collection of entities with particular characteristics or attributes. The predefined properties in RDFS include *rdf:type*, which is used to indicate that an entity is an instance of a class, *rdfs:subClassOf* and *rdfs:subPropertyOf*, which define hierarchies between classes and properties respectively, and *rdfs:domain*, which specifies that any entity having a certain property is an instance of one or more classes, and *rdfs:range*, which specifies that the values of a property are instances of one or more classes.

Taxonomy. In a knowledge graph, a taxonomy is a hierarchical arrangement of concepts or classes using *rdfs:subClassOf* relation that categorizes and organizes entities based on their characteristics.

Figure 1.1 illustrates a part of a knowledge graph featuring factual information about two entities, Saadi_Shirazi and Voltaire. The facts $(Saadi\_Shirazi, rdf : type, PersianPoet)$ and $(Voltaire, rdf : type, FrenchWriter)$ represent class constraints. The relations within the box in the graph denote *rdfs:subClassOf* relations, which provide a hierarchy of classes, *i.e.*, taxonomy.

RDF graph Saturation. While a knowledge graph contains explicit facts (triples), saturation operations can also reveal implicit facts that are not explicitly stated. This process involves inferring implicit RDF triples based on logical entailments that conform to RDF schema constraints. Forward chaining (Salvat and Mugnier, 1996) is one technique for graph saturation, which begins with the initial facts in the RDF graph and applies inference rules incrementally to derive new facts. This yields a more comprehensive data graph, but the process can be computationally expensive. The graph is considered fully saturated when no more new facts can be derived.

For example, in the graph depicted in Figure 1.1, the facts $(Saadi\_Shirazi, rdf : type, Poet)$, $(Saadi\_Shirazi, rdf : type, Writer)$, $\ldots$, $(Saadi\_Shirazi, rdf : type, Person)$, $\ldots$, and $(Voltaire, rdf : type, Person)$ can be included in the KG to create a fully saturated version of it.

OWL. RDFS provides some flexibility in expressing ontological constraints but has limitations in expressing certain properties like class disjointness and functional properties. To overcome these limitations, W3C endorsed the Web Ontology Language (OWL), a more expressive language based on description logics. OWL provides a rich set of logical axioms to define relationships between classes and individuals. These axioms are defined in TBox, which provides a vocabulary of concepts and properties. For instance, OWL properties include *owl:sameAs* for declaring that two resources represent the same thing, *owl:disjointWith* for stating that two classes are disjoint and have no instances in common, *owl:hasKey* allows keys to be defined for a given class, *owl:FunctionalProperty* states that a property can have only one (unique) value Y for each instance X. Moreover, OWL defines two types of properties: *owl:ObjectProperty* and *owl:DatatypeProperty*. An *owl:ObjectProperty* has an

entity in $\mathcal{I}$ as its object. For example, the *birthPlace* relation shown in Figure 1.1 is an instance of *owl:ObjectProperty*. Conversely, an *owl:DatatypeProperty* has a literal value in $\mathcal{L}$ as its object. OWL also allows axioms to assert facts associated with the TBox's conceptual model, called an ABox (Assertion Box). OWL has three sub-languages: OWL Lite, OWL DL, and OWL Full, which differ in expressiveness and complexity.

Unique Name Assumption (UNA). The unique name assumption is a simplifying assumption made in some ontology languages and description logics that states that a distinct name or identifier uniquely identifies each entity or concept. In other words, no two different names or identifiers refer to the same entity or concept.

Queries on RDF graphs. To retrieve data from RDF graphs, SPARQL queries[5] are used. SPARQL (a W3C recommendation) is a declarative query language similar to SQL. A SPARQL query includes a graph pattern that is utilized for data selection. It supports various query capabilities, including pattern matching, filtering, grouping, and aggregating data.

The following simple query retrieves all instances with a birth place with their respective locations. In Figure 1.1, this results would be $\{(Saadi\_Shirazi, Shiraz), (Voltaire, Paris)\}$.

```
SELECT ?instance ?location {
?instance birthPlace ?location .
}
```

A comprehensive study of knowledge graphs and models by which data can be structured, represented, and queried can be found in (Hogan et al., 2021).

1.3 ONTOLOGY MATCHING AND DATA LINKING

Ontology Matching (OM) or ontology alignment identifies correspondences between classes, properties, or individuals in two or more ontologies/KGs.

5 https://www.w3.org/TR/sparql11-query/

The resulting links can be used to integrate data from multiple knowledge graphs, creating a unified and coherent representation of knowledge within the graph. A comprehensive study of OM can be found in (Euzenat and Shvaiko, 2013).

Correspondence A correspondence or mapping is usually represented as a tuple $\langle e_1, e_2, r \rangle$ where e_1 and e_2 are entities (classes, properties, or individuals) of the two ontologies o_1 and o_2, r, is the semantic relation between them (such as equivalence ($\equiv$), more general ($\geq$), less general ($\leq$), or disjointness ($\perp$)[6]). Optionally, a confidence score c indicating the correspondence's certainty is attributed to the tuple.

Alignment The term used to refer to the set of correspondences between ontologies is "alignment". An alignment is not necessarily limited to a one-to-one (1:1) relationship; instead, it can take various forms, including one-to-many (1:m), many-to-one (m:1), or many-to-many (n:m) cardinalities.

The goal of ontology matching, achieved through a *matcher*, is to obtain an alignment for the ontologies o_1 and o_2.

Schema Matching and Instance matching. When alignment is at the TBox level (schema or ontological knowledge level), it is called schema matching, which involves matching classes or relations. On the other hand, aligning at the ABox level (assertion level or factual knowledge) is known as instance matching or data linking, which involves determining if two individuals refer to the same real-world entity.

1.4 RULE MINING

Ontology enables deductive reasoning to be applied to a knowledge graph, where new facts can be inferred by applying axioms and logical inferences in a top-down manner. For instance, given the knowledge that all poets are writers and that *db:Rumi* is a poet, we can deduce that *db:Rumi* is also a writer. On the other hand, inductive reasoning relies on observations and data, using a bottom-up approach to detect patterns that can lead to new facts. In this

6 In this thesis, the focus is on correspondences with equivalence relations.

context, rule mining mines patterns defined on the knowledge graphs that do not necessarily fit perfectly to the whole KG but allow exceptions. We define the notions in this context.

Definition 2. Atom. An atom is a basic well-formed first-order logic formula of the form $p(t_1, t_2, ..., t_n)$ where p is a predicate of arity n and $t_1, t_2, ..., t_n$ are its arguments that can be either variables or constants[7]. Any atom of arity $n > 2$ can be transformed to n binary relations $p(t_1, t_2, ..., t_n)$ rewritten as $p_2(t_1, t_2), p_3(t_1, t_3), p_4(t_1, t_4), ..., p_n(t_1, t_n)$ (Hernández et al., 2015). Also, the unary predicates, such as class membership, can be represented with a binary predicate, i.e., $type(x, y)$. If an atom's arguments are constants, the atom is said to be *grounded* and can be treated as a fact.

The atom $birthPlace(x, y)$ is an atom with two variables x and y. The atoms $birthPlace(Voltaire, Paris), hasPopulation(Paris, "2.1M")$ are grounded and are facts in the KG.

Definition 3. (Horn) Rule. A rule $r : \mathcal{B} \Rightarrow H$ is a first-order logic formula where the body $\mathcal{B}$ is a conjunction of atoms $B_1, ..., B_n$ and the head H is a single atom. A rule is *closed* if every variable appears at least twice in the rule. Two atoms are connected if they share at least one variable. A rule is connected if all atoms are transitively connected.

Figure 1.2 depicts a part of a knowledge graph. We can consider the rule $R_1 : worksIn(x, y) \Rightarrow bornIn(x, y)$ to be mined from this small KG. This rule is connected and closed.

Definition 4. Prediction of a Rule. Consider a rule $r : B_1, \ldots, B_n \Rightarrow H$, a substitution (a partial mapping from variables to constants) σ, and $\sigma(r)$ an instantiation of r which maps all variables in the body of r such that $\sigma(B_i) \in \mathcal{G}\ \forall i \in \{1, \ldots, n\}$. $\sigma(H)$ is a *prediction* of r where $\mathcal{G} \wedge r \models \sigma(H)$. A prediction is correct if $\sigma(H) \in \mathcal{G}$.

7 This manuscript represents variables using lowercase letters, whereas capitalized letters denote constants.

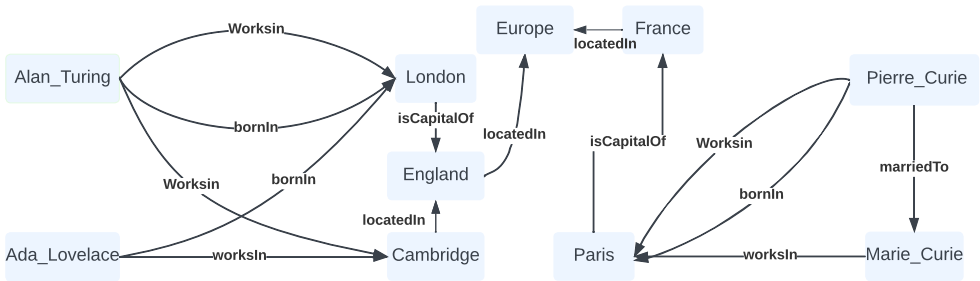

Figure 1.2.: Part of a knowledge graph, an example for rule mining

A prediction of the rule R_1 : would be $bornIn(Marie\_Curie, Paris)$.

For a rule $r : \mathcal{B} \Rightarrow H$, different measures to assess their qualities can be considered. We define the following quality measures as in (Lajus et al., 2020). In the absence of identity links (*i.e.*, *owl:sameAs*), we assume that the Unique Name Assumption (UNA) is fulfilled. If identity links exist, a pre-processing step is required to compute the quality measures and functionality score accurately.

Definition 5. Support. The support $supp(r) := |\{(x,y) : \mathcal{B} \wedge H(x,y)\}|$ measures the number of correct predictions made by the rule.

The support of rule R_1 on the small knowledge graph of 1.2 is 2 because we have $worksIn(Alan\_Turing, London), bornIn(Alan\_Turing, London)$ and $worksIn(Pierre\_Curie, Paris), bornIn(Pierre\_Curie, Paris)$. In other words, $bornIn(Alan\_Turing, London)$ and $bornIn(Pierre\_Curie, Paris)$ are correct predictions of R_1 since they exist in the knowledge graph.

Definition 6. Head coverage. Head coverage represents the proportion of instantiations of the head atom that are correctly predicted by the rule.

$$hc(r) = \frac{supp(r)}{|\{(x,y) : H(x,y) \in \mathcal{G}\}|}$$

In other words, the head coverage of a rule is the fraction of predicted head atoms that appear in the actual graph, i.e., the support divided by the total number of predicted head atoms (*i.e.*, head size).

For instance, the head coverage of rule R_1 mined on the graph of Figure 1.2 is $\frac{2}{3} = 0.66$ because the rule correctly predicts 2 out of the 3 facts in the head relation *bornIn*; since $bornIn(Ada\_Lovelace, London)$ is not among the correct predictions of the rule.

To calculate the confidence of a rule, counter-examples are necessary. Yet, we know that knowledge graphs are based on the *Open World Assumption (OWA)*, meaning they only contain positive examples, and missing facts are not necessarily false. Hence it is necessary to come up with strategies to consider counter-examples. The *Closed World Assumption (CWA)* is implemented in many contexts, such as in relational databases. This implies that what is currently not known to be true (not present in the database), is incorrect.

Definition 7. Standard confidence. The standard confidence of the rule r measures the precision of the rule under the CWA.

$$stnd\_conf(r) = \frac{supp(r)}{|\{(x,y) : \mathcal{B}\}|}$$

For instance, the standard confidence of rule R_1 mined on the graph of Figure 1.2 is $\frac{2}{5} = 0.4$ because we have correct predictions of $bornIn(Alan\_Turing, London), bornIn(Pierre\_Currie, Paris)$ and incorrect predictions $bornIn(Alan\_Turing, Cambridge), bornIn(Ada\_Lovelace, Cambridge)$, $bornIn(Marie\_Curie, Paris)$.

The CWA is not appropriate in the context of KGs as it assumes that all knowledge about a domain is already known and represented in the KG. Nevertheless we know that KGs can be incomplete or inaccurate due to various reasons such as limited data sources, noisy data, or errors in data integration, also new knowledge is constantly being discovered or added to the KGs. To find triples that can account for counter-examples, we follow the *Partial Completeness Assumption (PCA)* (Galárraga et al., 2013), which necessitates the introduction of a predicate's functionality score.

Definition 8. Functionality Score. The *functionality score* of a predicate is a value between 0 and 1 that measures the ratio of subjects that the property is related to in $\mathcal{G}$ to the total number of triples with that predicate. The inverse functionality score $ifun(p)$ is the functionality score for the inverse of the predicate p.

$$fun(p) := \frac{|\{x : \exists y : p(x,y) \in \mathcal{G}\}|}{|\{(x,y) : p(x,y) \in \mathcal{G}\}|}$$

Under PCA, if a fact $p(x,y) \in \mathcal{G}$ and if $fun(p) > ifun(p)$, then no other fact for x holding with the predicate p is correct and can be considered as a counter-example (*i.e.*, $p(x,y') \notin \mathcal{G}$). On the other hand, if $ifun(p) > fun(p)$, then all $p(x',y) \notin \mathcal{G}$.

Consider the KG in Figure 1.2. Since $fun(bornIn) > ifun(bornIn)$ and we have information that Alan Turing was born in London, then any other birthplace for Alan Turing would be a counter-example. In contrast, since we have no information about Marie Curie's birthplace, we cannot consider any counter-examples for her place of birth under the PCA.

Definition 9. PCA confidence. The PCA confidence of the rule r measures the precision of the rule under the PCA, *i.e.*, the ratio of correct predictions or support to the total number of predictions made by the rule. More precisely, if $fun(H) > ifun(H)$,

$$pca\_conf(r) = \frac{supp(r)}{|\{(x,y) : \exists y' : \mathcal{B} \wedge H(x,y')\}|}$$

Based on definition of counter-examples under PCA, if $ifun(H) > fun(H)$, then the denominator, namely PCA body size, becomes $|\{(x,y) : \exists x' : \mathcal{B} \wedge H(x',y)\}|$ in the above equation.

For instance, the PCA confidence of rule R_1 mined on the graph of Figure 1.2 is $\frac{2}{4} = 0.5$ because we have correct predictions of $bornIn(Alan\_Turing, London), bornIn(Pierre\_Currie, Paris)$ and incorrect predictions $bornIn(Alan\_Turing, Cambridge), bornIn(Ada\_Lovelace, Cambridge)$.

It can also be noted that using ontology matching as a pre-processing step for rule mining can help address the inherent incompleteness of knowledge graphs to some extent and also help in better counterfactual selections.

1.5 KNOWLEDGE GRAPH EMBEDDING

Symbolic knowledge representation, which encodes knowledge using logical expressions, rules, and axioms, can be useful for rule-based reasoning and interpretability. Yet, knowledge graphs often contain noise and are incomplete, and this representation is not well-suited for such settings. Therefore, subsymbolic knowledge representation, which encodes knowledge using numerical vectors, has gained popularity, and much research has been devoted to this area.

One common approach to subsymbolic knowledge representation is knowledge graph embeddings (KGE), which involves mapping entities and relations in a knowledge graph to a numerical vector space of a pre-defined dimension of embedding space d. The primary objective of KGE techniques is to preserve the underlying structure of the knowledge graph and capture the semantics of entities and relations between them. Different relation patterns KGE techniques try to capture as presented in (Sun et al., 2019), are as follows:

- A relation p is **symmetric** if $\forall x, y : (x, p, y) \implies (y, p, x)$. Let x =Emmanuel Macron and y =Brigitte Macron, and let p = married to. (x, r, y) denotes "Emmanuel Macron is married to Brigitte Macron", which implies (y, r, x) or "Brigitte Macron is married to Emmanuel Macron".
- A relation p is **antisymmetric** if $\forall x, y : (x, p, y) \implies \neg(y, p, x)$. Let x =Brigitte Macron and y = Tiphaine Auzière, and let p =has child. (x, p, y) denotes "Brigitte Macron has child Tiphaine Auzière", which implies $\neg(y, p, x)$ or "Tiphaine Auzière doesn't have child Brigitte Macron".
- A relation p_1 is **inverse** to relation p_2 if $\forall x, y : (x, p_1, y) \implies (y, p_2, x)$. Let x =Emmanuel Macron, y =Estelle Macron, p_1 =has sister, and

p_2 =has brother. (x, p_1, y) denotes "Emmanuel Macron has sister Estelle Macron," which implies (y, p_2, x) or "Estelle Macron has brother Emmanuel Macron".

- Relation p_1 is a **composition** (transitive) of relation p_2 and relation p_3 if $\forall x, y, z : (x, p_2, y) \wedge (y, p_3, z) \implies (x, p_1, z)$. Let x =Emmanuel Macron, y =Amiens, z =France, p_2 =is born in, p_3 =located in, and p_1 =has nationality. (x, p_2, y)= Emmanuel Macron is born in Amiens $\wedge (y, p_3, z)$= Amiens is located in France implies (x, p_1, z)= Emmanuel Macron has nationality France.

Different knowledge graph embedding techniques can capture several or all the above-mentioned relation patterns. This is highly dependent on the representation space and the scoring function used. A comprehensive survey by Wang et al. (Wang et al., 2017b) investigates some of the prominent options explored in the literature. Knowledge graph embedding techniques are generally distinguished by their representation space selection, scoring function, and loss function.

Representation space. Knowledge graph embedding involves mapping entities and relations in a knowledge graph to low-dimensional numerical vectors in a vector space. The choice of representation space is crucial in determining the patterns that KGE can capture and hence, the overall performance of knowledge graph embedding models. Various techniques for knowledge graph embedding use different representation spaces, such as Euclidean space, complex vector space, manifold, and Gaussian.

Scoring Function. The scoring function $f_r(h, t)$ in knowledge graph embedding is a function used to measure the plausibility of a triple or a fact (h, r, t). It is also called the energy function in the energy-based learning framework. It maps the triple (h, r, t) to a score that reflects the likelihood of the triple being true. The score must be high for correct triples and low for incorrect triples. The scoring function is key in training and evaluating knowledge graph embedding models. It will be discussed in more detail in the chapter 2.2.2.

Negative Sampling and Optimization. As previously mentioned, the scoring function is optimized during training to learn embeddings for entities and

relations, assigning higher scores to correct triples in the knowledge graph and lower scores to incorrect triples. As discussed in section 1.4, KGs implement the Open World Assumption and do not contain incorrect/negative triples. There are different ways to corrupt facts to obtain false triples to do the training; two commonly used negative sampling methods are introduced in (Wang et al., 2014a). For a golden triple (*i.e.*, a fact of the KG), *unif* method creates negative triples by randomly sampling a pair of entities (h', t') from the KG to construct the corrupt triple (h', r, t'). However, this method can introduce many false negative triples. To address this issue, the *bern* is introduced, which assigns Bernoulli distributed probabilities for replacing head or tail according to the type of property (1-to-1, 1-to-N, N-to-N).

To learn entity and relation embeddings, an optimization problem must be solved that maximizes the plausibility of correct facts and minimizes the probability of negative triples. This is achieved by minimizing the sum of the loss function over the training set of facts. Different loss functions can be used for this purpose. For instance, margin loss and logistic loss are two common loss functions used. Given the positive set of triples Δ and negative (*i.e.*, corrupted) set of triples Δ' constructed according to heuristics like those discussed above, the margin loss or pairwise ranking loss can be defined as

$$L = \sum_{(h,r,t)\in\Delta} \sum_{(h',r',t')\in\Delta'} max(0, \gamma + f_r(h,t) - f_{r'}(h',t'))$$

This function maximizes the score of correct facts by at least a margin $\gamma > 0$ higher than incorrect triples. The logistic loss minimizes the negative log-likelihood of logistic models

$$L = \sum_{(h,r,t)\in\Delta\cup\Delta'} log(1 + exp(-y_{hrt} \cdot f_{r(h,t)}))$$

where $y_{hrt} \pm 1$ is the triple instance's label (positive or negative).

The performance of a knowledge graph embedding model is influenced significantly by choice of the loss function (Nayyeri et al., 2019) and the negative sampling technique (Kotnis and Nastase, 2018; Bansal et al., 2020).

2

AN OVERVIEW ON SOME KNOWLEDGE GRAPH REFINEMENT APPROACHES

This chapter centers on overviewing some previous studies on refining knowledge graphs. We organize this exploration into two main categories: knowledge graph completion in section 2.2, which focuses on the link or relation prediction to refine a knowledge graph, and data linking in section 2.3, which refines the graph by identifying sameAs links to entities in other knowledge graphs. We review previous works on both tasks using symbolic and subsymbolic methods.

2.1 INTRODUCTION

Knowledge graphs are inherently incomplete; this may be due to the failure to capture certain information during the construction of the KG or the absence of certain facts in the available resources. Hence, in the KG, not all information about every entity is incorporated, and not all the facts contained within them are necessarily correct. The process of refining a knowledge graph

involves identifying and correcting erroneous facts, as well as adding missing information, and integrating new data sources.

A detailed survey on knowledge graph creation, curation, and refinement is available in (Weikum et al., 2021; Cimiano and Paulheim, 2017). This chapter focuses on existing work addressing knowledge graph completion and data linking. Usually, knowledge graph completion is the task of predicting missing statements: for the subject $(?, p, o)$, object $(s, p, ?)$, or both, as well as relation prediction $(s, ?, o)$. Data linking or ontology matching aims to uncover `owl:sameAs` relations between entities within various knowledge graphs. For each of these refinement tasks, we examine symbolic and subsymbolic approaches suggested in the literature.

2.2 KNOWLEDGE GRAPH COMPLETION

Around 71% of the roughly 3 million people in the Freebase knowledge graph have no known place of birth, 94% have no known parents, and 99% have no known ethnicity (West et al., 2014). By completing knowledge graphs, we can improve the accuracy and effectiveness of applications and downstream tasks that rely on them. For example, using a more comprehensive knowledge graph, a question-answering system can provide more accurate and complete answers to user queries.

The knowledge graph completion comprises predicting an entity with a specific relation with another given entity (link prediction) or predicting a relation between two existing KG entities (relation prediction). In this chapter, we focus on link prediction.

Developing an automated technique for curating the triples of the Knowledge Graph (KG) is crucial due to the high cost associated with manual curation (Paulheim, 2018). Knowledge graphs can be completed using logical rules on KGs or knowledge graph embeddings. Rules offer the benefits of being explainable, interpretable, and easily transferable to unseen entities. On the other hand, embeddings are less intuitive. Still, they have the advantage of being more scalable, more flexible (they can capture complex patterns and relationships between entities), and less sensitive to noise.

We can informally consider the following procedures for knowledge graph completion using rule-based and embedding-based approaches.

Rule-based Knowledge Graph Completion. Assuming we have a knowledge graph $\mathcal{G}$ consisting of facts $\mathcal{F}$ and a rule r of the form $\mathcal{B} \Rightarrow H$, the completion of $\mathcal{G}$ by r is a knowledge graph $\mathcal{G}_r$ such that $\mathcal{F}_r$ is extended by the predictions of the rule $\mathcal{F}_r = \mathcal{F} \cup \{H(a_1, b_1), \ldots, H(a_n, b_n) | \mathcal{G} \wedge r \models H(a_1, b_1), \ldots, H(a_n, b_n)\}$.

KGE-based Knowledge Graph Completion. Assuming we have a knowledge graph $\mathcal{G}$ and a scoring function $f_r(h, t)$ learned through a KG embedding technique, we can apply the following method to predict missing triples (h, r, t) in $\mathcal{G}$. First, we replace the head/tail entity of the triple with all entities in the set $\mathcal{I}$. We then use the scoring function $f_r(h, t)$ to rank each of these in descending order and select the one with the highest score.

Knowledge graph Completion Performance Indicators. Consider a KG embedding or rule-based model and a set of known true facts Q. For each fact (h, r, t) in Q, the predictions $(h, r, ?)$ are ranked using the scoring function for the KG embedding model; and for the rule-based model, a measure such as confidence of the rule can be used to rank the predictions. $rank_q$ denotes the position of the ground-truth tail t in the sorted rank of entities. Various metrics are employed to evaluate the effectiveness of a technique in predicting new facts:

- **Mean Rank (MR)** measures the average rank at which the correct entity is predicted.
$$MR = \frac{1}{|Q|} \sum_{q \in Q} rank_q$$

- **Mean Reciprocal Rank (MRR)** measures the average of the inverse ranks of the correct entity that is predicted.
$$MRR = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{rank_q}$$

- **Hits@K** measures if the correct entity appears within the top-k elements ranked.

$$Hits@K = \frac{1}{|Q|} \sum_{q \in Q} \mathbb{1}(rank_q \leq K)$$

Besides the commonly used evaluation methods for KG completion, there have been other proposed measures that are noteworthy. For instance, in a recommendation scenario, it may be crucial to identify the correct ground-truth entity and suggest related ones from a vast catalog. In a recent publication, (Hubert et al., 2022) argues that recommending semantically related entities to the correct one is valuable. To address this issue, they introduce Sem@K, a new semantic-oriented metric.

2.2.1 *Rule Mining Techniques on KGs.*

Several works have addressed finding logical rules in large knowledge graphs. They can be roughly categorized into two classes: top-down and bottom-up. Top-down rule mining starts with the head of the rule and constructs the body iteratively by refining it by analyzing the data. Bottom-up rule mining is a data-driven approach that starts with observing individual instances and then generalizing the observations (*e.g.*, paths) to extract common patterns or rules. In other words, bottom-up rule mining starts from the data and moves toward a higher level of abstraction. In contrast, top-down rule mining starts from general concepts or high-level rules and moves towards a more detailed understanding of the data.

These approaches employ specific language biases, pruning criteria, and optimization strategies to scale up the rule-mining process. The level of expressivity of the rules can vary depending on the language bias used, which allows for the search space restriction. For example, reflexive atoms of the form $r(x, x)$ are often excluded from most rule-mining approaches. Another common language bias is to limit the number of atoms in a rule. Additionally, these systems propose various optimization techniques and pruning or parallelization strategies to cope with the large size of RDF graphs. Examples

include pruning based on quality measures, optimization of calculation of quality measures, and approximation. A less restrictive language bias or pruning strategy can produce more expressive rules but a slower performance due to the large search space. We will provide an overview of the existing literature on the topic and elaborate on these different aspects.

AMIE introduced in (Lajus et al., 2020; Galárraga et al., 2013; Galárraga et al., 2015) is a state-of-the-art rule mining system for KGs that is fast and exhaustive. Its language bias allows it to discover all connected and closed rules, subject to predetermined thresholds for quality measures like confidence, head coverage, and a specified maximum number of atoms (*e.g.*, $minhc, minC$).

AMIE is a top-down approach. It initializes a queue of rules containing all possible head atoms with an empty body ($q = [\top \Rightarrow r_1(x,y), \ldots, \top \Rightarrow r_m(x,y)]$). The algorithm then explores the search space by iteratively extending rules using three different refinement operators. These include: adding a Dangling Atom, which introduces a new atom with one fresh variable (*e.g.*, $r(x,z)$); adding an Instantiated Atom, which introduces a new atom with one argument being instantiated and the other a variable of the rule(*e.g.*, $r(x,Z)$); and adding a Closing Atom, which introduces a new atom with both variables already present in the rule (*e.g.*, $r(x,y)$).

AMIE+ (Galárraga et al., 2015) is successor of AMIE (Galárraga et al., 2013). While it does not mine all rules satisfying its language bias, it significantly reduces computation time by implementing new pruning and approximation strategies such as approximate confidence calculations and query rewriting techniques. As a result, AMIE+ can achieve computation speeds that are orders of magnitude faster than the original version.

AMIE3 (Lajus et al., 2020), the newest version of AMIE, manages to speed up the rule mining process by a factor of 15, compared to the state-of-the-art methods, all while maintaining exhaustiveness. It accomplishes this by employing techniques such as lazy computation of the denominator of confidence until the rule is sure to be pruned based on the given threshold, using an in-memory database to store the KG, and optimizing query plans based on variable order.

RUDIK (Ortona et al., 2018) proposes a non-exhaustive approach to mine closed, connected logical rules with a maximum body size of *maxPathLen*. RuDiK is more expressive than AMIE as it can also mine negative rules (with negation in the head) that can be used to find erroneous triples in the KG; for instance, if a person is vegetarian, they cannot eat meat. RuDiK also allows us to perform comparisons beyond equality by using relationships from the set $rel \in \{<, \leq, \neq, \geq, >\}$. An example of such rules would be: $has\_spouse(x, y) \wedge date\_of\_birth(x, v_0) \wedge date\_of\_birth(y, v_1) \wedge v_0 > v_1 \Rightarrow has\_wife(y, x)$.

RuDiK also adopts a top-down approach to construct rules. RuDiK is designed to find a set of rules that cover the majority of positive and as few negative examples as possible. Specifically, it generates a *generation set* and a *validation set* under PCA for each predicate. The former contains positive examples (*e.g.*, all parent-child pairs), while the latter holds counter-examples (*e.g.*, pairs of individuals without the *hasChild* relation). To identify negative rules, RuDiK swaps the two sets' roles. For each pair of entity (x, y) where (x, rel, y) in the generation set (and hence in the KG), it applies a depth-first search to retrieve all nodes within a distance no greater than $maxPathLen$. This search allows the construction of the rule body, which can be viewed as a path in the undirected graph, and then computes the rule's coverage. RuDiK aims at generating robust rules (that do not overfit) that cover as many examples in the generation set and as few examples in the validation set. A weight $w(r)$ that captures the coverage over the generation and validation sets is computed. RuDiK maintains a rule queue, which selects the rule with the minimum estimated weight using a greedy approach. It removes facts that a rule covers to ensure that subsequent rules predict new facts. This also serves as a pruning strategy.

AnyBURL (Meilicke et al., 2019) is a bottom-up approach that initially samples paths and applies generalization techniques to expand them to obtain rules. AnyBURL's language bias focuses exclusively on rules based on graph paths, with the maximum rule length n as an input parameter. The rules generated by AnyBURL are not necessarily closed.

More precisely, the approach taken by AnyBURL involves sampling paths from the KG and constructing a bottom rule. A generalization lattice rooted

in the bottom rule is then built, and all useful rules that appear in the lattice are stored.

An extension of AnyBURL (Meilicke et al., 2020) uses reinforcement learning to sample better paths from the start. The advantage of these approaches is that they are anytime, meaning they can trade time for rule quality and quantity.

Ontological Path finding (Chen et al., 2016) generates paths that will be used to obtain candidate rules. Still, it exploits the information of the ontology, especially the domain and range axioms, to reduce the search space to predicates coherent with the domain and range of previously mentioned predicates and enumerates all conjunctions of atoms allowed by the schema. Ontological Path finding's language bias is also to find connected and closed rules.

To achieve scalable mining, Ontological Path finding employs a range of optimization and pruning strategies, such as partitioning candidate rules into smaller independent sets, parallelizing join queries through Spark, and eliminating untrustworthy rules that yield large intermediate results due to involving large-degree variables in the joining predicates.

NeuralLP. Another family of rule mining systems that are differentiable is based on TensorLog (Cohen, 2016), which can learn the confidence and structure of rules simultaneously. These approaches mostly operate under the Open World Assumption and are only trained using only positive examples. They map each entity to a one-hot encoded vector and each relation to an adjacency matrix, and rules are obtained through matrix multiplications. NeuralLP (Yang et al., 2017) defines a (differentiable) operator for each relation. These operators (relations) are chained to compute a score for each triple and learn rules and their confidence by maximizing this score using gradient-based optimization. However, this formulation is only able to generate rules with fixed lengths. To address this limitation, NeuralLP utilizes LSTM and attention mechanisms to learn variable-length rules. An extension of NeuralLP, NeuralLP-num (Wang et al., 2020b), can learn rules involving numerical features. Another extension is DRUM (Sadeghian et al., 2019). DRUM is able to learn a high-rank approximation of the rule confidences, whereas NeuralLP is limited to learning a rank-1 tensor approximation. Moreover, DRUM uses

bidirectional RNNs to share helpful information across the tasks of learning rules for different relations.

Because NeuralLP (and their extensions) is end-to-end differentiable, it can be easily parallelized on GPUs to learn rules quickly.

Recap and Conclusion.

We discussed some of the standard scalable techniques used for mining rules in knowledge graphs. These methods use refinement or generalization operators to discover rules and are influenced by various language biases and different pruning strategies or quality measures. There are other works in the literature with different twists in the rules they mine or quality measures they define. For instance, (Gad-Elrab et al., 2016) proposes post-processing/revision to the horn rules mined on a KG by adding exceptions (*i.e.*, negated atoms) into their bodies to improve the quality of the rules. (Simonne et al., 2021) presents a first approach to discovering differential causal rules in Knowledge Graphs. In (Tanon et al., 2018), the authors propose a novel rule ranking measure, namely the *completeness-aware confidence*. This measure takes advantage of the cardinality information related to the expected number of edges in the knowledge graph to evaluate the quality of learned rules more effectively.

Despite extensive research on rule mining, several challenges remain. One of these challenges is the absence of negative information in the knowledge graph (KG). Although this issue can be partially addressed by imposing assumptions like PCA, it can still lead to the inclusion of many correct facts as negatives (and many incorrect facts are not considered as negative). Some works have suggested using ontology and background knowledge to improve the negatives, but more research is needed to develop approaches that do not compromise scalability. Another challenge is ensuring scalability as KGs continue to grow; indeed, with the existing approaches, the number of atoms and their expressivity is limited to be able to scale. Additionally, there is room for enhancing the expressivity of the rules to capture more interesting information, as well as biases and trends in the data. Examples of possible enhancements include incorporating negated atoms, numerical rules with existing mathematical functions (*e.g.*, sigmoid function, greater than, less

than, etc.) or aggregations (*e.g.*, sum, average, etc.), and the definition of more interesting built-in predicates (*e.g.*, domain-related functions or temporal predicates extracted from existing data in the KG). Lastly, it is important to develop methods that enable the updating and enhancement of rules as the KG evolves.

2.2.2 *Knowledge Graph Embedding Techniques.*

Much research has been aimed at discovering embeddings representing entities and relations of a knowledge graph (KG) in a low-dimensional continuous vector space. Various scoring functions have been proposed to capture different types of relations such as symmetry, anti-symmetry, inversion, and composition, as discussed in Chapter 1.5. Surveys such as (Ji et al., 2021; Dai et al., 2020; Wang et al., 2014b) provide comprehensive reviews of recent developments in this area.

KG embedding techniques can be classified into three main families based on their scoring functions, and this section provides an overview of recent works in each of these categories. The novelty of each approach is briefly discussed, along with the types of relations they can capture.

Translational models

The first type of KG embedding techniques proposed are translational models, representing relations as translations between two entities. To put it simply, these models aim to satisfy the equation $\vec{h} + \vec{r} \approx \vec{t}$ for a given fact (h, r, t), where the embedding vector of the head entity added to the embedding vector of the relation should be approximately equal to the embedding vector of the tail entity. Various versions of this family of models will be explored.

TransE (Bordes et al., 2013) is the first work to view relations as translations from a head entity to a tail entity. This approach is motivated by hierarchical relationships being common in KGs and translations being natural transformations for representing them. Given a triple (h, r, t), TranE embeds

$\vec{h}, \vec{r}, \vec{t} \in \mathbb{R}^d$ such that $\vec{h} + \vec{r} \approx \vec{t}$. The scoring function of TransE measures the proximity of the embeddings and is defined as $f_r(h,t) = -\|\vec{h} + \vec{r} - \vec{t}\|$, where the L_1 or L_2 norm can be used to measure proximity. The embeddings are learned by minimizing a margin-based criterion, as described in 1.5.

TransE naturally captures antisymmetry, inversion, and composition (by adding the vectors of the relations), but it is not expressive enough to capture symmetric relationships. This is because the only way to capture symmetry $(h,r,t) \Rightarrow (t,r,h)$ is to have $\|\vec{h} + \vec{r} - \vec{t}\|_{L_1/L_2} = 0$, which means $\vec{r} = 0$. As a result, the embeddings of h and t become identical ($\vec{h} = \vec{t}$), despite being distinct entities. Moreover, TransE cannot model one-to-many (or many-to-one) relations (*e.g.*, (h,r,t_1) and (h,r,t_2)) because the scoring function will assign the same embedding to t_1 and t_2, even though they are different entities.

TransH, TransD, and TransR. To overcome the shortcoming of TransE rising from modeling translations of any relation in the same embedding space, TransH (Wang et al., 2014b) proposes to model a relation as a hyperplane with a translation operation. More specifically, for a triple (h,r,t), the embedding $\vec{h}$ and $\vec{t}$ are first projected to the relation-specific hyperplane w_r denoted by $(\vec{h}_\perp)$ and $(\vec{t}_\perp)$. The translation vector $\vec{d}_r$ on the hyperplane is used in the score function denoted by $f_r(h,t) = -\|(\vec{h}_\perp) + (\vec{d}_r) - (\vec{t}_\perp)\|_2$. One shortcoming of TransH is that translating on relation-specific hyperplanes results in $\vec{h}_\perp = \vec{t}_\perp$ for modeling one-to-many (and many-to-one) without the actual embeddings of the head and tail becoming identical. TransH still assumes that the relation and entity embeddings belong to the same embedding space, which fails to capture that an entity may have distinct aspects depending on the relation it appears within a fact.

Following a similar approach, TransR (Lin et al., 2015) represents entities and relations in separate embedding spaces and learns embeddings through translation between projected entities. TransR incorporates a translation relation vector $\vec{r} \in \mathbb{R}^d$ and a projection matrix $M_r \in \mathbb{R}^{k \times d}$ for each relation, while entities are modeled as vectors in $\mathbb{R}^d$. To obtain the projected entity vectors, TransR applies the matrix M_r to the original entity vectors resulting in $\vec{h_r} = M_r\vec{h}$ and $\vec{t_r} = M_r\vec{t}$. In this transformed space, TransE intuition is applied with the score function defined as $f_r(h,t) = -\|(\vec{h_r}) + (\vec{r}) - (\vec{t_r})\|_2$. CTransR

follows the same approach, but it clusters all entity pairs (h, t) concerning a specific relation r into different groups, and the pairs in the same group share the same relation vector r_c. It learns a matrix M_r for each relation. Due to the use of separate spaces for each relation, the compositional nature of relations cannot be captured.

TransD (Ji et al., 2015) addresses the issue of expensive matrix-vector multiplications and excessive parameters in TransR. TransR uses a shared mapping matrix M_r for all entities linked by a relation r, which may not be suitable for relations involving diverse types and attributes of entities. To overcome this limitation, TransD creates two vectors for each entity and relation, namely $\vec{h}, \vec{h_p}, \vec{t}, \vec{t_p} \in \mathbb{R}^n$ and $\vec{r}, \vec{r_p} \in \mathbb{R}^m$, respectively. Moreover, TransD dynamically constructs two mapping matrices $M_{rh}, M_{rt} \in \mathbb{R}^{m \times n}$ for each triple to project entities from entity space to relation space (*i.e.*, the mapping matrices are determined by both entities and relations).

RotatE, QuantE, and DualE. To address the limitation of TransR in modeling the compositional nature of relations and capturing symmetry/antisymmetry and inversion, researchers have proposed using complex vector embeddings instead of real vector embeddings. RotateE (Sun et al., 2019) maps the head and tail entities h, t to the complex embeddings $\vec{h}, \vec{t} \in \mathbb{C}^k$. Inspired by Euler's identity $e^{i\theta} = cos\theta + isin\theta$, RotateE defines each relation as a rotation from the head entity to the tail entity in the complex vector space.The score function is $f_r(h, t) = -\|\vec{h} \circ \vec{r} - \vec{t}\|_2$ where $\circ$ is the Hadamard product. RotateE can capture all patterns described in Chapter 1.5.

QuantE (Zhang et al., 2019) expands on using complex vector embeddings by exploring hypercomplex space for learning KG embeddings. While it is more closely related to semantic matching models, it draws inspiration from RotateE. In this approach, entities and relations are represented as quaternion embeddings, which are vectors in the hypercomplex space $\mathbb{H}$ with three imaginary parts i, j, and k, as opposed to $\mathbb{C}$, which has two components of imaginary and real, and has two planes of rotation, unlike RotateE's single plane.

To summarize, the translation family is insufficient for modeling all three fundamental patterns. The rotation family has limited effectiveness on hier-

archical and multiple relation patterns; DualE (Cao et al., 2021) offers a new approach that combines both rotation-based and translation-based models in 3D space. The embeddings in DualE are represented as vectors in the hypercomplex space H_d, a dual quaternion space of the form $a + \epsilon b$, where a and b are two quaternions representing the real and dual part of the vector, respectively.

ManifoldE, TorusE, and MobiusE. Various approaches propose alternative geometric or algebraic structures for embeddings in knowledge graph embedding models. One such approach is ManifoldE (Xiao et al., 2015), which extends the point-wise modeling used in translation-based models to manifold-wise modeling. In this model, a manifold function $M(h, r, t) = D_r^2$ is defined, where D_r is a relation-specific parameter and M is the manifold function. The score function is then calculated as $f_r(h, t) = -\|M(h, r, t) - D_r^2\|^2$. ManifoldE considers two types of embeddings, namely, Sphere and Hyperplane. A manifold is a topological space defined as a set of points with neighborhoods determined by set theory.

TorusE (Ebisu and Ichise, 2018) uses a special algebraic structure and embeds entities and relations on a torus $\vec{h}, \vec{r}, \vec{t} \in \mathbb{T}$ which is a compact Lie group. It fixes the problem of regularization in TransE. Inspired by TorusE, MobiusE (Chen et al., 2021) proposed to embed entities and relations to the surface of a Mobius ring.

Many other models have been proposed in the literature for translational knowledge graph embedding, which employs different representation spaces and score functions. For instance, KG2E (Wang et al., 2017a) represents each entity/relation as a multi-dimensional Gaussian distribution in probability space. This approach helps model the uncertainty present in the knowledge base. Or HAKE (Zhang et al., 2020b), which considers and preserves the semantic hierarchies of entities in KGs. HAKE achieves this by mapping entities into the polar coordinate system.

Semantic matching models

In contrast to the translational models discussed earlier, which use distance as their scoring function, another category of knowledge graph embedding methods is the *semantic matching* or *bilinear* models that utilize similarity-based scoring functions $f_r(h,t)$. Semantic matching models measure the similarity or relatedness between two embedded entities for a given relation (*e.g.*, cosine similarity).

RESCAL and DistMult. RESCAL is regarded as one of the initial knowledge graph embedding methods and the first and simplest bilinear model. It computes a three-way factorization of an adjacency tensor representing the knowledge graph. In the tensor $\mathcal{X}$, if a relation r_k exists between entities e_i and e_j, the entry $\mathcal{X}_{ijk} = 1$, whereas if the relation does not exist, $\mathcal{X}_{ijk} = 0$. RESCAL factorizes each side of $\mathcal{X}$ using a rank r factorization. A vector represents each entity, and each relation is represented by a matrix $M_r = \mathbb{R}^{d \times d}$. The model's scoring function is defined by $f_r(h,t) = h^T M_r t$. RESCAL has a large number of parameters and is computationally expensive.

DistMult (Yang et al., 2015) is a simplified version of RESCAL that uses bilinear diagonal matrices and fewer parameters, making it more efficient. In DistMult, entities are represented as vectors that capture their latent semantics, while relations are represented as diagonal matrices M_r that model pairwise interactions between the latent factors. DistMult can model symmetry but not anti-symmetry due to the commutativity of summation and product. Also, it is incapable of modeling inverse relations, such as r_1 and r_2, because $f_{r_1}(h,t) = \langle h, r_1, t \rangle = \langle h, r_2, t \rangle = f_{r_2}(h,t)$, which forces r_1 to be equal to r_2. Furthermore, DistMult is limited in its ability to model compositional relations, as it defines a hyperplane for each (h,t) and the union of these hyperplanes, such as for (r_1, r_2), cannot be expressed using a single hyperplane.

ComplEx (Trouillon et al., 2017) is a generalized version of DistMult that represents entities in the complex space $\mathbb{C}^d$ and relations as diagonal bilinear forms $M_r = diag(r), r \in \mathbb{C}^d$. This allows it to capture asymmetric and inversion patterns of relations. The scoring function is similar to DistMult, but it takes the real part of the Hermitian product: $f_r(h,t) = RE(< h, r, t >)$. However, ComplEx still lacks the ability to model compositional relations.

HoleE and ANALOGY. The aim of HoleE or Holographic Embeddings of Knowledge Graphs (Nickel et al., 2016) is to address the scalability issues of RESCAL caused by tensor products. To achieve this, it introduces circular correlation of embeddings, which is like a compressed form of the tensor product, to learn compositional representations. This approach is more efficient and scalable than RESCAL.

ANALOGY (Liu et al., 2017) also extends RESCAL to further model the analogical properties of entities and relations (*e.g.*, A is to B as C is to D) and address its scalability issues.

Deep models

Deep models are a popular category of knowledge graph embedding techniques. They enable the creation of more expressive representations of entities and relations by applying multiple layers of non-linear transformations and utilizing more complex score functions.

Typically, the training process for deep models consists of two phases. The first phase creates vectors for each entity and relation (encoding $\vec{h}, \vec{r}, \vec{t}$). The second phase involves evaluating the plausibility of these vectors in a layer-by-layer learning approach (scoring).

Many approaches are proposed in the literature, relying on different architectures. For instance, some approaches leverage Convolutional Neural Networks (CNNs) to learn deep expressive features. Others rely on Recurrent Neural Networks (RNNs/LSTMs) to capture long-term relational dependencies within knowledge graphs. Some approaches adopt transformers to excel in learning contextualized text representations. Graph Convolutional Networks (GCNs) use the topological structure of knowledge graphs and update node representations by aggregating and propagating node features within the graph.

Deep models also facilitate the integration of external information, such as types, literal attributes, and texts, which can further enrich the learned embeddings. We will briefly overview some of the proposed approaches in the literature.

ConvE, ConvKB, and HypER. ConvE (Dettmers et al., 2018) is the first model to use CNNs for knowledge graph completion. It uses 2D convolutions over the concatenated entity and relation embeddings. More specifically, ConvE reshapes and concatenates $\vec{h}$ and $\vec{r}$ into a 2-D matrix (M_h, M_r) fed to the convolution layer later. The resulting feature maps are reshaped into a vector. The dot product of this vector and the $\vec{t}$ is used to compute the score function $f_r(h,t) = \sigma(vec(\sigma([M_h, M_r] * \omega))W).\vec{t}$ where ω represents the convolutional filters.

ConvKB (Nguyen et al., 2018) extends ConvE by removing the reshaping operation in the encoding of representations in the convolution operation. ConvKB employs a 3-column matrix of $\vec{h}, \vec{r}$, and $\vec{t}$ to pass through the convolutional layer. Subsequently, the feature maps generated are concatenated to produce a vector concatenated with a weight vector to obtain a score for (h, r, t). The score function can be formulated as $f_r(h,t) = concat(\sigma([\vec{h}, \vec{r}, \vec{t}]) * \omega).w$ where w is the weight vector. ConvKB can capture global relationships and transitional characteristics between entities and relations in knowledge bases which is not the case for ConvE.

HypER (Balažević et al., 2019) extends ConvE by proposing relation-specific convolution filters. Unlike ConvE and ConvKB, it does not perform any reshaping or concatenating. It adopts hyper-networks based on ConvE, generates relation-specific convolutional filters, and applies them to subject entity embeddings. It needs fewer parameters than ConvE and is more efficient.

More recently, ConEx (Demir and Ngomo, 2021) proposed to apply the convolution operation on complex-valued embeddings of subjects and predicates.

DOLORES, KG-BERT, R-MeN, and CoKE. Knowledge graph embedding techniques based on RNNs/LSTMs typically aim to capture more than just a single triple by incorporating paths in the knowledge graph. These paths of varying lengths can be obtained by performing random walks (or similar techniques) on the graph. For example, DOLORES (Wang et al., 2020a) generates chains of entity-relations through random walks on the KG and uses a Bi-Directional LSTM architecture to learn entity and relation embeddings that can capture context dependence.

Transformer-based models have been recently employed to capture contextual information in knowledge graphs. KG-BERT (Yao et al., 2019) adopts the pre-trained BERT (Bidirectional Encoder Representations from Transformer) language model as an encoder for entities and relations and can easily incorporate external information (*e.g.*, entity types or descriptions). The model creates different inputs depending on the refinement task. For link prediction or triple classification tasks, the head, relation, and tail entities are concatenated and separated by the $[SEP]$ token, and the $[CLS]$ token is added at the beginning. The entities can be replaced with their textual descriptions. To fine-tune BERT, the final hidden state C corresponding to $[CLS]$ is used as the aggregate sequence representation for computing triple scores, which are then passed to a classification layer. The inputs are created for relation prediction using the head and relation entities. The final hidden state corresponding to $[CLS]$ is used to perform multi-class classification on the whole set of relations in the knowledge graph.

Similarly, R-MeN (Nguyen et al., 2020) passes triples of fact with their positional embedding as the input to a transformer-based architecture. Each input vector generates an encoded vector fed to a decoder based on a CNN. The decoder produces a score for the triple that is used for triple classification.

CoKE (Contextualized Knowledge Graph Embedding) (Wang et al., 2020c) generates paths in the form of $h \rightharpoonup r_1 \cdots \rightharpoonup r_n \rightharpoonup t$, and for the task of link prediction, it applies a mask with the token $[mask]$ to either h or t. The resulting sequence is then passed through Transformer encoding blocks, and the final hidden state, corresponding to the $[MASK]$ token, is used to predict the masked entity.

Recap and Conclusion.

We discussed various knowledge graph embedding techniques, primarily focusing on their scoring function and architectures. These techniques fall under three families: translational, semantic matching, and deep models. While we did not delve into their performance or adaptations to different downstream tasks, the approaches discussed are commonly used for knowledge graph completion tasks.

Many KG embedding techniques are still proposed in the literature, each with unique architectures designed to address specific shortcomings of previous approaches and with varying levels of expressivity and integration of external information. For instance, Relational Graph Convolutional Networks (RGCN) (Schlichtkrull et al., 2018) is an extension of Graph Convolutional Networks that can operate on large-scale relational data by incorporating local graph neighborhoods. Another example is OTKGE (Cao et al., 2022), which proposes embeddings for multi-modal knowledge graphs where text, image, and video information coexist. This approach is based on the Optimal Transport problem.

While KGE techniques have seen a lot of advancements, there are still several challenges that need to be addressed. One such challenge is their limited effectiveness in zero-shot or few-shot settings for link prediction or relation prediction tasks. The entities or relations are either unseen or have very few occurrences in the training data. Some recent works have been proposed to address this issue (Xiong et al., 2018; Zhang et al., 2020a). Another challenge is the flawed evaluation of KGE techniques, as some studies in the literature have pointed out. The calibration studies are not well-defined, and the datasets often involve semantically inverse relations, which are unsuitable for proper evaluation (Pezeshkpour et al., 2020). Some efforts have been made to address these evaluation challenges by introducing new benchmark datasets designed explicitly for knowledge graph completion. For instance, the CoDEx dataset (Safavi and Koutra, 2020) has been developed to overcome the challenges faced with previous datasets. Similarly, there is the WD-Known dataset (Veseli et al., 2023), designed to assess the potential of knowledge graph completion for language models. These new datasets help researchers better understand and evaluate the performance of knowledge graph completion techniques. In addition, studies such as the one in (Teach et al., 2020), re-implement prior KGE approaches and scrutinize their performance, revealing that training parameters play an important role. In addition, knowledge graph embedding (KGE) techniques lack interpretability, and their use in downstream tasks, such as knowledge graph completion, does not explain the results obtained.

2.3 DATA LINKING TECHNIQUES

In Section 2.2, we explored both symbolic and subsymbolic approaches proposed in the literature for knowledge discovery and enhancing the comprehensiveness of knowledge graphs. In this section, we will shift our focus to data linking, which involves determining whether entities from different datasets refer to the same real-world entity.

In Section 1.3, we introduced the concept of ontology matching, and a comprehensive survey of different approaches can be found in (Rahm and Bernstein, 2001; Euzenat and Shvaiko, 2013). Ontology matching encompasses two main tasks: schema matching and instance matching. Schema matching involves establishing alignments between classes and relations in different knowledge graphs. On the other hand, instance matching, also known as data linking, focuses on determining, with a certain degree of confidence, whether two individuals refer to the same real-world object (Ferrara et al., 2011).

Numerous approaches have been proposed for both schema matching and instance matching. These link discovery approaches can be categorized into three different groups based on how they incorporate schema and instance matching into their workflow.

Some systems solely concentrate on instance matching. Among these systems, some depend on declared linkage rules that can be used to logically infer identity links (Al-Bakri et al., 2015; Fan et al., 2015) while others compute a similarity score thanks to complex rules that can involve simple similarity measures and aggregation functions, like LIMES or Silk (Ngonga Ngomo and Auer, 2011; Volz et al., 2009a). These rules are typically based on sets of discriminative properties or more complex graph patterns, such as OWL2 keys and schema mappings. However, specifying such properties can be challenging. To address this, certain approaches aim to discover discriminative properties using one or multiple knowledge graphs, assuming that the mappings are known (Symeonidou et al., 2011, 2014, 2017), or by allowing the discovery of keys involving property mappings (Atencia et al., 2019).

Some systems only focus on schema matching. These systems typically leverage terminological similarities, structural similarities, instance similarities,

external resources, or logical axioms to identify more or less complex schema mappings (Doan et al., 2000; Aumueller et al., 2005).

On the other hand, certain systems perform both instance matching and schema matching in their ontology matching processes. Examples include PARIS (Suchanek et al., 2011), and ILIADS (Udrea et al., 2007), which employ interleaved schema matching and instance matching in iterative iterations, with mappings from one task assisting in refining the mappings of the other task.

In this section, our focus will be on instance-matching techniques. We will delve into key-based and rule-based approaches, which belong to the symbolic paradigm. We will also explore data-linking methods that leverage embeddings and machine learning-based models.

Key-based and Rule-based approaches

KD2R and SAKey. KD2R (Pernelle et al., 2013) is the first key discovery approach specifically designed for knowledge graphs. These approaches assume that UNA is fulfilled. It first finds a set of maximal non-keys and then derives the minimal keys from them. However, one limitation of KD2R is its scalability to large knowledge graphs. SAKey (Symeonidou et al., 2014) extends KD2R to make it more scalable and introduces the concept of *almost keys*, which relaxes the traditional key constraint by allowing a specified number n of exceptions. The value of n is user-defined, and in the strictest scenario, o-almost keys do not permit any exceptions to the key constraint. For a class $\mathcal{C}$ and a set of properties $P = p_1, \ldots, p_n$, the exception set E_P is defined as:

$$E_P = \{X | \exists Y (X \neq Y) \wedge C(X) \wedge C(Y) \wedge \bigwedge_{p \in P} \exists U (p(X, U) \wedge p(Y, U))\}$$

A set of properties is an n-almost key if there exist at most n instances that share values for this set of properties. More formally, for a class $\mathcal{C}$, a set of properties $P = \{p_1, \ldots, p_n\}$, and an integer n, P is an n-almost key for $\mathcal{C}$ if $|E_P| \leq n$ and it's an n-non key if $|E_p| \geq n$. SAKey finds first the maximal

$(n+1)$ non-keys and then derives the minimal n-almost-key. A maximal (n+1) non-key has at least (n + 1) exceptions. The discovery of maximal n-non keys follows a depth-first exploration principle, incorporating mappings and pruning strategies while traversing the exception sets. Since as soon as n exceptions are found for a set of properties P, it's sure that $E_p \geq n$ and the exploration shall stop. Once SAKey has identified and obtained all non-keys, it deduces that all remaining combinations of properties are keys, thereby generating the output of the SAKey algorithm. The instantiation of n-almost keys is used for data linking. We highlight that SAKey focuses on the discovery of S-keys (OWL2 semantics of a key), meaning that in the context of multi-valued properties within a key, SAKey considers it sufficient for them to share at least one common value for each property in the key.

ROCKER (Soru et al., 2015a) presents a refinement operator for Key Discovery. The refinement operator ρ is used to refine candidate keys based on a scoring function that compares sets of properties. The scoring function calculates the rate of distinguishable instances in a class given a set of properties representing a candidate key. The scoring function induces a quasi-ordering $\preccurlyeq$ over the set of all candidates. In other words, if $P \preccurlyeq Q$, it means that $\min_{p \in P} \text{score}(p) \preccurlyeq \min_{q \in Q} \text{score}(q)$ and the other way around. The algorithm employed by ROCKER computes the score for all properties and sorts them based on their scores. This heuristic approach aims to discover keys earlier in the refinement process, allowing for the pruning of descendant keys in the refinement tree and reducing the number of score calculations. To iteratively search for keys, a priority queue is used, where the priority is determined by the scores. The algorithm also utilizes key monotonicity to prune branches during the search process. It is worth mentioning that, unlike SAKey, ROCKER specializes in discovering F-keys. This means that in the case of multi-valued properties, all values of each property in the key must be shared for it to be considered a valid key. In their experiments, it has been demonstrated that ROCKER exhibits faster execution times on larger datasets in comparison to SAKey. Moreover, the researchers also reported lower memory consumption.

VICKEY (Symeonidou et al., 2017) considers situations where there may be a scarcity or absence of valid keys throughout the knowledge graph. They define the concept of "conditional keys" – keys that are only applicable

to specific subsets of data. Examples of conditional keys include the date of creation as a key for museums located in Paris. Or the restriction of each doctoral student to having only one advisor professor within German universities where $universityLocation(x, Germany)$ is a conditional property. More formally, a conditional key for a knowledge graph $\mathcal{G}$ is a non-empty set of conditional properties $\{cd_1, \ldots, cd_n\}$, and a non-empty set of key properties $\{p_1, \ldots, p_m\}$ disjoint from those in conditions, such that:

$$\forall x \forall y \forall z_1 ... z_m (\bigwedge_{i=1}^{n} (cd_i(x) \wedge cd_i(y)) \wedge \bigwedge_{i=1}^{m} (p_i(x, z_i) \wedge p_i(y, z_i)) \Rightarrow \ x = y)$$

To ensure scalability, VICKEY adopts a strategy of exploring minimal conditional keys by focusing on the maximal non-keys identified by SAKey. This approach limits the consideration of property combinations to those derived from the maximal non-keys. The rationale behind this restriction is that adding conditional properties to a key is not relevant, whereas adding conditions to non-key properties can result in relevant conditional keys.

VICKEY proceeds by constructing a collection of conditional key graphs, considering all possible conditions $p = a$ that combine a property p from the maximal non-keys with an instance or literal a from the dataset. However, these conditions are bound by a threshold θ, ensuring that they have a support exceeding the specified threshold. In a subsequent step, these conditional key graphs are systematically mined to identify the minimal conditional keys. This mining process involves traversing the nodes of the graph level by level (breadth-first strategy). Initially, nodes with a single property are considered, followed by nodes with two properties, and so on. During each traversal, the validity of the nodes is assessed, taking into account the definition of a conditional key and ensuring that the identified keys are minimal compared to the keys discovered thus far.

Link Key (Atencia et al., 2014b) aims to discover sets of property pairs that simultaneously characterize equivalence between two knowledge graphs. A link key between a pair of classes $\langle C, D \rangle$ is defined by pairs of corresponding properties $\{\langle p_1, q_1 \rangle, \ldots, \langle p_n, q_n \rangle\}$ that collectively identify unique entities. For example, consider the classes $\langle$Album, MusicAlbum$\rangle$.

One possible link key for the pair of classes $\langle \text{Album}, \text{MusicAlbum} \rangle$ is $\{\langle \text{name}, \text{albumName} \rangle, \langle \text{publishdate}, \text{releaseDate} \rangle\}$. It is important to highlight that link keys can be categorized into different types based on their characteristics, such as weak and strong. For instance, if $\{p_1, \ldots, p_n\}$ and $\{q_1, \ldots, q_n\}$ are keys for classes C and D, respectively, then they are considered strong link keys. To gain a deeper understanding of link keys and their relationship with keys and different types of link keys, extensive research has been conducted, as discussed in (Atencia et al., 2021).

The Link Key approach follows a two-step process. First, it extracts all candidate link keys, which are maximal sets of property pairs where at least two instances share a common value. In (Atencia et al., 2020), Formal Concept Analysis and Relational Concept Analysis are employed to generate these candidate link keys. In the second step, measures such as discriminability and coverage are applied to select the most promising link keys. The resulting link keys can then be used for generating links between the entities of the two knowledge graphs.

Unlike key-based approaches discussed earlier, which require schema alignment (matching classes and properties) beforehand, the Link Key approach does not rely on pre-aligned schemas.

LIMES and Silk. In the literature, some approaches require manual rules and user-specified configurations. These approaches take a link specification (LS) or linkage rule as input, defining the conditions data items must meet to be linked.

One such approach is LIMES (Link Discovery Framework for metric spaces) (Ngonga Ngomo and Auer, 2011) (Ngonga Ngomo et al., 2021), which takes a configuration file provided by the user as input. LIMES employs triangle inequality to estimate entity similarity. It approximates the similarity of entity pairs and computes the actual similarity for pairs with high approximated similarity to choose the pair with the highest similarity. LIMES leverages various similarity metrics to compute the relatedness between entities' attributes based on user-specified thresholds.

The Silk Link Discovery Framework (Volz et al., 2009b), also offers a declarative language for defining link discovery tasks. Users can specify the sources,

attributes, similarity metrics, and rules for matching entities. Silk supports a range of similarity metrics, providing flexibility in the link discovery process.

Machine learning and Knowledge Graph Embedding-based approaches.

EAGLE and GenLink. Discovering useful link specifications and rules is a challenging task, prompting the development of new approaches in the literature that focus on learning rules through supervised learning.

One such approach is EAGLE presented in (Ngonga Ngomo and Lyko, 2012), which extends the LIMES framework. EAGLE introduces a machine learning-based approach for link specification of varying complexity, utilizing genetic programming techniques.

Another approach, GenLink, builds upon the SILK framework. It leverages a supervised learning algorithm employing genetic programming to learn linkage rules from a set of existing reference links.

AttrE (Trisedya et al., 2019);Entity Alignment between Knowledge Graphs Using Attribute Embeddings, is an approach that introduces a novel embedding model that combines entity structure embedding and attribute character embedding for aligning entities between knowledge graphs. In this approach, the embedding of relations and entities of the two KGs (to be matched) are in the same vector space. By jointly learning structure embedding and attribute character embedding, the proposed approach ensures that similar entities in $\mathcal{G}1$ and $\mathcal{G}2$ have similar embeddings. This enables the computation of similarities between entities in the two KGs. A similarity threshold β is then used to filter out entity pairs that are not similar enough to be inked.

KERMIT (Hertling et al., 2022), KnowlEdge gRaph MatchIng with Transformers, is a matching technique that builds upon sentence BERT (SBERT) (Reimers and Gurevych, 2019). The use of transformer-based language models allows for semantical textual comparison rather than relying on pure string sequence matches. The process involves fine-tuning a cross-encoder model using S-BERT by generating positive and negative correspondences through two methods: (1) sampling from the reference and (2) utilizing a high-precision matcher. The generated inputs are then passed to SBERT for fine-tuning.

Subsequently, the fine-tuned model is applied within the matching pipeline. The obtained results benefit from a logic-based alignment repair step, which enhances precision.

Recap and Conclusion.

We explored different data-linking methods, with a particular emphasis on key-based approaches, and briefly touched upon embedding-based techniques. Despite various works addressing ontology matching and data linking, particular challenges persist.

One challenge pertains to the evaluation process and the scarcity of ground truth in many scenarios. Additionally, knowledge graphs are dynamic and continuously expanding as the Linked Open Data (LOD) expands. This highlights the significance of scalable approaches that can effectively handle the evolving nature of knowledge graphs without excessive effort.

3

NUMERICAL RULES DISCOVERY ON KNOWLEDGE GRAPHS

The work presented in this chapter has been done with Nathalie Pernelle and Fatiha Saïs and has been published and presented at:

Armita Khajeh Nassiri, Nathalie Pernelle, Fatiha Saïs. "REGNUM: Generating Logical Rules with Numerical Predicates in Knowledge Graphs." Full paper at Extended Semantic Web Conference (ESWC) 2023. (Khajeh Nassiri et al., 2023)

Armita Khajeh Nassiri, Nathalie Pernelle, Fatiha Saïs. "Enrichissement de règles de Horn par des predicats numériques", Short Paper at La conférence Extraction et Gestion des Connaissances (EGC) 2023. (Nassiri et al., 2023a)

3.1 INTRODUCTION

Context and Motivation

Chapter 2 examined how knowledge graphs can provide valuable insights by extracting logical rules. These rules can be used for predicting facts (Galárraga et al., 2017), curating the KG (Loster et al., 2021), identifying trends or biases, ontology alignment (Galárraga et al., 2013), or fact-checking (Qudus et al., 2022; Ahmadi et al., 2019). In section 2.2.1, we investigated various scalable rule-mining techniques with different language biases. However, rule mining systems face challenges when dealing with numerical data due to the many values that numerical predicates can take, resulting in a significantly large search space (Galárraga and Suchanek, 2014).

In this chapter, we first provide an overview of the related work in the literature that focuses on rule mining with numerical predicates/attributes

before presenting our own contributions. We have observed that the current numerical rule mining techniques are either unsuitable for large knowledge graphs or are not expressive enough. As a result, we have been motivated to address this gap in the literature through the work presented in this chapter. To categorize our literature review, we have divided it into three groups.

Association Rule mining. Association rule mining (ARM) (Agrawal et al., 1993) is a widely used data mining technique that identifies frequent patterns among items and transactions based on a minimum number of observations. It typically generates if-then patterns, represented by association rules $X \rightarrow Y$, indicating that the presence of X suggests the presence of Y in the same transaction. However, ARM faces challenges when dealing with numerical attributes since the values of these attributes rarely repeat themselves. To address this, a special type of association rule called *quantitative association rules* has been developed, which involves at least one numerical attribute in the rule, such as $(25 < age < 40) \wedge (3K < salary < 5K) \rightarrow (120K < loan < 200K)$. Quantitative Association Rule Mining (QARM) can be achieved through different strategies, including discretization-based approaches such as pre-processing steps to partition numerical data (Srikant and Agrawal, 1996) or statistical analysis of variables and distribution of the numerical variables (Aumann and Lindell, 1999), and optimization-based approaches where numeric attributes are optimized during the mining process, for instance with the use of genetic algorithms (Salleb-Aouissi et al., 2007), (Jaramillo et al., 2021), (Minaei-Bidgoli et al., 2013). Nevertheless, these patterns or dependencies are restricted to single variables and differ from the logical rules relevant to complex relationships in knowledge graphs.

Inductive Logic Programming (ILP) is a method for automatically deriving rules from positive and negative examples. For instance, WARMR (Dehaspe and Toironen, 2001) is an ILP system that extends the Apriori algorithm to mine association rules across multiple relations. Other ILP-based techniques, such as DL-Learner (Bühmann et al., 2016), concentrate on learning complex concept definitions, including numerical range restrictions. To discover rules with numerical attribute intervals, (Melo et al., 2014) proposes an ILP extension that involves a pre-processing step to compute correlation lattices for numerical and categorical attributes, thereby reducing the search space. Nonetheless, ILP is not well-suited for the open world assumption (OWA) in

large knowledge graphs, where counter-examples are not explicitly stated, and missing information cannot be considered negative but rather unknown.

First-Order Logic (FOL) Rules in Knowledge Graphs. In Section 2.2.1, we provided a comprehensive overview of FOL rule mining in knowledge graphs. However, when it comes to numerical rules, existing methods such as AMIE (Galárraga et al., 2013), AnyBURL (Meilicke et al., 2019), Ontological Path finding (Chen et al., 2016), and their extensions can only extract rules that involve constants (*e.g.*, `age(x,53)`). These rules can be overly specific and uninteresting when it comes to numerical predicates. RuDik (Ortona et al., 2018), on the other hand, can perform comparisons beyond equality by utilizing relationships from $rel \in \{<, \leq, \neq, \geq, >\}$. For instance, a rule generated by RuDik may appear as follows: $p_1(x, v_0) \wedge p_2(y, v_1) \wedge v_0 > v_1 \Rightarrow p_3(x, y)$, where v_0 and v_1 are values from the knowledge graph, not thresholds. In the category of differentiable rule-based methods, NeuralLP-num (Wang et al., 2020b) can learn rules involving numerical features. Like RuDik, these rules may make pair-wise comparisons between the numerical values of different atoms in the rules. Additionally, the rules produced by NeuralLP-num may include classification operators, which are sigmoid functions over numerical values of atoms with numerical predicates in the rule. For instance, a rule with a classification operator could appear as follows: $f\{y_1, y_2 : p_1(X, y_1), p_2(X, y_2)\} > 0.5 \wedge p_3(X, Z) \Rightarrow p_4(X, Z)$ where f is the sigmoid function and p_1 and p_2 are numerical predicates. To the best of our knowledge, RuDik and NeuralLP-num are the only techniques in the literature that can extract interesting rules with numerical predicates on large knowledge graphs. However, both approaches are limited to using numerical values from the knowledge graph and applying functions or comparisons between them. They cannot discover numerical intervals or thresholds as constraints to improve the quality of rules and derive additional knowledge.

Contributions

This chapter presents our approach *REGNUM* to address the gap in incorporating numerical predicates into logical rules mined on KGs. Our approach involves two main steps. First, we obtain the FOL rules using efficient rule

mining tools (like those presented in section 1.4). Second, we enrich the rules with numerical predicates and interval constraints. We focus on constraints that express the membership or non-membership to a value interval (*e.g.*, the country's population between 1M to 5M or the country's GDP not less than 1 trillion dollars). One way to incorporate these constraints is by discretizing the numerical predicates' values in a pre-processing step. Unsupervised techniques like *EqualWidth* and *EqualFrequency* can be used for this purpose, as they do not rely on instance labels unavailable in this setting. However, this approach may not provide relevant constraints for each rule and could result in information loss. Moreover, calculating constraints while the rule mining technique explores the search space (as it generalizes or refines the rule) results in having to re-calculate the interval at each step, making the approach time-consuming over large graphs. Hence, we propose a novel approach that, for this second step, we consider the problem as a classification problem to obtain the intervals based on the correct and incorrect predictions of the rule, guided by the quality of the rules. This way, we can use supervised discretization techniques while restricting the search space. Various techniques can be employed to discretize numerical values, including supervised discretization methods (Kohavi and Sahami, 1996; García et al., 2013). One family is the Chi-square-based supervised discretization (Kerber, 1992), a statistical technique involving a bottom-up merging process of intervals after an initialization step. This technique is univariate and supervised. Another family is the entropy-based supervised discretization, such as the Minimum Description Length-based discretized (MDLP) (Fayyad and Irani, 1993). This approach uses the entropy of class information to determine the discretization boundaries and employs the MDL principle as a stopping criterion. However, most discretization methods are univariate and consider only a single attribute at a time. Another option would be to consider sequential covering approaches, *e.g.*, RIPPER (Cohen, 1995) or FURIA (Hühn and Hüllermeier, 2009). These approaches use the separate-and-conquer strategy, learning one rule at a time and gradually repeating the process to cover the complete set of positive examples. They tend to have high precision and avoid creating numerous rules that overlap in the instances they cover. Alternatively, QARM techniques introduced in section 3.1 can also be used. In this work, the search strategy to obtain interval constraints on the numerical predicates relies on

tree-based algorithms that can consider membership and non-membership to an interval.

The main contributions of the work presented in this chapter are:

- REGNUM is a novel approach that enhances the expressiveness of the rules generated by a rule mining system by incorporating numerical constraints expressed through value intervals. To our knowledge, REGNUM is the first approach to utilize intervals in rules mined from large RDF graphs.
- REGNUM efficiently selects intervals that increase the rule's confidence using existing supervised discretization techniques that best distinguish the correct and incorrect predictions made by the rule.
- Since some value intervals can be too specific to lead to relevant rules, REGNUM considers both the membership and the non-membership of a value to an interval to offer more possibilities of generating a rule with high quality.
- The experimental evaluation shows that the numerical rules generated by REGNUM using the rules provided by two state-of-the-art rule mining systems, AMIE and AnyBURL, have a higher overall quality score and can potentially improve prediction results.

3.2 REGNUM: GENERATING LOGICAL RULES WITH NUMERICAL PREDICATES

In this section, we describe REGNUM, a system that automatically enriches connected and closed rules mined on a given knowledge graph, regardless of the method used to mine them, with numerical predicates by constraining the introduced numerical values to specified intervals. REGNUM aims to enhance the PCA confidence defined in Definition 9 in the considered rules while ensuring that the rules do not become overly specific.

3.2.1 *Problem statement*

REGNUM aims to mine numerical rules that are defined as follows:

Definition 10. Numerical Rule. A numerical rule is a first-order logic formula of the form: $\mathcal{B} \wedge C \Rightarrow H$, where $\mathcal{B}$ is a conjunction of atoms of the KG. The range values of the numerical atoms of $\mathcal{B}$ can be constrained using C, which are conjunction (resp. disjunction) of atoms that express their membership (resp. their non-membership) to an interval $[inf, sup]$.

Example. Here are two examples of numerical rules with constraints defined on numerical predicates:

r_1 : $\text{worksIn}(x, y) \wedge \text{hasPopulation}(y, w) \wedge w \in [1000, 5500] \wedge \text{hasHusband}(x, z) \Rightarrow \text{worksIn}(z, y)$

r_2 : $\text{worksIn}(x, y) \wedge \text{hasHusband}(x, z) \wedge \text{age}(z, a) \wedge \text{hasPopulation}(y, w) \wedge (w \notin [1000, 5500] \vee a \notin [50, \infty)) \Rightarrow \text{worksIn}(z, y)$

Generating all numerical rules that fulfill quality measure thresholds (*e.g.*, $minHC$ and $minconf$ in (Lajus et al., 2020)) can be very time-consuming. This is because the intervals used to constrain the range of numerical predicates must be recalculated each time a rule is generalized or refined as the search space is explored. This ensures that the constraints applied to the rule are appropriate for the updated rule.

To overcome this issue, we propose REGNUM, an approach that builds on the shoulders of the rules mined by an existing rule-mining technique, namely *parent rules*. It expands the body of these rules through an enrichment process to generate numerical rules. More precisely,

- A numerical rule is considered relevant if it improves the PCA confidence of its parent rule by at least a $marginC$ without its head coverage decreasing more than $marginHC$. This criterion guarantees that the rule has higher PCA confidence than its parent rule while preventing over-fitting the KG.

- The enrichment process of a parent rule is driven by considering *diverse* sets of numerical predicates. If a numerical rule involving a specific numerical predicate p is deemed relevant in a constraint containing a number of numerical atoms, the method will not generate any further numerical rules by considering additional numerical predicates that contain p. In other words, *diversity* is an intuitive heuristic to prune the search space and avoid searching for more relevant rules with additional numerical predicates once a rule meeting the desired quality criteria is identified.

3.2.2 *Rule Enrichment with Numerical Predicates*

Given a knowledge graph $\mathcal{G}$, a set of closed rules $\mathcal{R}$ mined from $\mathcal{G}$, called *parent rules*, and thresholds *marginC* and *marginHC* as introduced in 3.2.1, our approach REGNUM is able to enrich the parent rules in $\mathcal{R}$ to obtain relevant and diverse numerical rules. The algorithm, described in Algorithm 1, performs the following steps.

(1) Pre-processing step

[line 1 in Algorithm 1]. As a pre-processing step, we first identify the set of numerical predicates, denoted as $\mathcal{P}_{num}$ in $\mathcal{G}$. We use the domain and range definition axioms if they are available in the ontology, and if not, we discover them by considering the Datatypes of values they take as their range (*e.g.*, numbers, time and date). We also compute the functionality score, as defined in Definition 8, for all predicates in the head of the parent rules $\mathcal{R}$.

Then, for each parent rule $r \in \mathcal{R}$, we proceed with the following steps.

Algorithm 1: REGNUM

Input:

- $\mathcal{G}$: knowledge graph
- $\mathcal{R}$: set of parent rules mined on $\mathcal{G}$
- $marginHC$, $marginC$: margins on head coverage and PCA confidence

Output: $\mathcal{E}$: set of enriched rules

Identify P_{num} and compute functionality degree of predicates P
$\mathcal{E} = \emptyset$
foreach $r : \mathcal{B} \Rightarrow H$ *in* $\mathcal{R}$ **do**
    compute quality measure $hc(r)$ and $pca\_conf(r)$; compute $minHC$ and $minC$; create an empty queue q_{atoms};
    for $p_{num} \in P_{num}$ **do**
        if $hc(p_{num}(x_i, x_{new}) \wedge \mathcal{B} \Rightarrow H) > minHC$ **then**
            Enqueue $p_{num}(x_i, x_{new})$ in q_{atoms}
    end
    $ln = 1$ `// number of numerical atoms`
    while $|q_{atoms}| > ln$ **do**
        for B_{num} *created by combining* ln *atoms from* q_{atoms} **do**
            $r_s : \mathcal{B}_{num} \wedge \mathcal{B} \Rightarrow H$
            if $hc(r_s) > minHC$ **then**
                $< X, Y > \leftarrow construct\_prediction\_classes(\mathcal{G}, r_s)$;
                $r_{nodes} \leftarrow discretize(< X, Y >, minHC, minC)$;
                foreach r_{node} *in* r_{nodes} **do**
                    if $hc(r_{node}) > minHC$ *and* $pca\_conf(r_{node}) > minC$ **then**
                        add $r_{node} \Rightarrow H$ to $\mathcal{E}$;
                end
        end
        remove from q_{atoms} atoms that resulted in a numerical rule;
        $ln += 1$
    end
end
return $\mathcal{E}$;

(2) Computation of minHC and minC

[line 4 in Algorithm 1]. A numerical rule obtained by enriching a parent rule r is considered relevant if its PCA confidence increases by at least $marginC$ and if its head coverage does not decrease by more than $marginHC$. We first query the KG to compute $pca\_conf(r)$ and $hc(r)$ if the rule mining system does not provide them, and we calculate the $minHC : (1 - marginHC) * hc(r)$ and $minC : (1 + marginC) * pca\_conf(r)$ that the enriched rule must satisfy to be considered relevant. For instance, if the parent rule has a PCA confidence of 0.8 and a head coverage of 0.3, and the $marginC = 10\%, marginHC = 20\%$, the numerical rule's confidence should be at least $minC = 0.88$ and its head coverage at least $minHC = 0.24$ to be considered relevant.

(3) Enqueue Numerical Atoms

[lines 5–8 in Algorithm 1.] In this step, we find all possible numerical atoms that can enrich the parent rule and store them in q_{atoms}. We consider the set of all variables $vars = \{x_1, \ldots, x_n\}$ that appear in the rule r, and enqueue all the atoms $p_{num}(x_i, x_{new})$ with $x_i \in vars$, x_{new} is a new variable and such that the head coverage of $p_{num}(x_i, x_{new}) \wedge \mathcal{B} \Rightarrow H$ is greater than $minHC$. Otherwise, the atom is discarded since its conjunction with other atoms will only lead to lower head coverage due to monotonicity.

Example. Let r_1 : workPlace$(x_1, x_2) \Rightarrow$ birthPlace(x_1, x_2), be a parent rule with head coverage of 0.4 and the $maginHC = 10\%$. The atom involving the numerical predicate hasPopulation with variables x_1 will be pruned as the rule hasPopulation$(x_1, x_3) \wedge$ workPlace$(x_1, x_2) \Rightarrow$ birthPlace(x_1, x_2) will have a head coverage of 0.01, which does not satisfy the $minHC = 0.36$.

(4) Selection of numerical atoms and generation of the best intervals

[lines 11–20 in Algorithm 1]. We aim to identify relevant numerical rules that meet the quality measure requirements by utilizing the fewest numerical predicates. To construct these numerical rules, we iteratively search through

the space of possible conjunctions of atoms in q_{atoms}. We begin with a single numerical atom ($ln = 1$) and continue until the queue q_{atoms} does not contain ln atoms or more. At iteration ln, we apply the following steps:

(4.1) Enriching r with ln numerical atoms

[lines 11–12 in Algorithm 1] We retrieve ln atoms from q_{atoms} and consider their conjunctions $\mathcal{B}_{num} : p_{num_1}(x_i, x_{n+1}) \wedge \ldots \wedge p_{num_l}(x_j, x_{n+l})$ to construct:

$$r_s : \mathcal{B}_{num} \wedge \mathcal{B} \Rightarrow H$$

We query the knowledge graph and proceed with the enrichment process only if r_s satisfies the $minHC$ as constraining the values of these numerical predicates will not satisfy $minHC$ either.

Example. At iteration 3, we can have r_s : worksIn$(x, y) \wedge$ hasHusband$(x, z) \wedge$ hasPopulation$(y, w) \wedge$ age$(x, v) \wedge$ hasRevenue$(x, u) \Rightarrow$ worksIn(z, y) involving three different numerical predicates hasPopulation, age and hasRevenue that satisfies the $minHC$.

(4.2) Classification problem based on rule predictions

[line 14 in Algorithm 1]. The rules r_s created in the previous step are used to classify the instantiations of $\mathcal{B}_{num} : p_{num_1}(x_i, x_{n+1}) \wedge \ldots \wedge p_{num_l}(x_j, x_{n+l})$ as correct or incorrect examples and define a binary classification problem.

In this classification step, we build a class A to represent the set of instantiations $(x_{n+1}, ..., x_{n+l})$ of the numerical values of B_{num} that lead to a correct prediction $H(x_a, x_b)$ for the rule r_s. The examples of A are defined as follows:

$$\{(x_{n+1}, \ldots, x_{n+l}) \mid \mathcal{B}(x_1, \ldots, x_n) \wedge \mathcal{B}_{num}(x_i, \ldots, x_j, x_{n+1}, \ldots, x_{n+l}) \wedge H(x_a, x_b)\}$$

Moreover, we build a class B to represent the set of instantiations $(x_{n+1}, ..., x_{n+l})$ representing the numerical values of B_{num} that result in an incorrect prediction for the rule r_s under PCA, provided that $fun(H) > ifun(H)$. In other words, the predictions $H(x_a, x_b)$ such that $H(x_a, x_b) \notin \mathcal{G} \wedge \exists x'_b \; H(x_a, x'_b) \in \mathcal{G}$.

$$\{(x_{n+1}, \ldots, x_{n+l}) \mid \mathcal{B}(x_1, \ldots, x_n) \wedge \mathcal{B}_{num}(x_i, \ldots, x_j, x_{n+1}, \ldots, x_{n+l}) \wedge \exists \, x_{b'} H(x_a, x'_b)\}$$

If $ifun(H) > fun(H)$, we classify the instantiation as incorrect if a fact does not exist in the KG for the target object and if there exists at least one subject for this object (*i.e.*, $H(x_a, x_b) \notin \mathcal{G} \wedge \exists x'_a \; H(x'_a, x_b) \in \mathcal{G}$.).

We generate a data structure $< X, Y >$ that represents for each correct and incorrect prediction $H(x_a, x_b)$, the set of numerical values per numerical predicate of B_{num} (since the numerical predicates can be multi-valued), and the label Y.

The correct and incorrect example values are retrieved from the KG through the queries defined for the label A and B.

Example. Let r_2 : worksIn$(x, y) \wedge$ hasHusband$(x, z) \Rightarrow$ worksIn(z, y) be the considered parent rule. One possible refinement r_s of this rule to consider in step 2 would be: hasPopulation$(y, w) \wedge$ worksIn$(x, y) \wedge$ hasHusband$(x, z) \wedge$ hasRevenue$(z, r) \Rightarrow$ worksIn(z, y) In this rule, the target variable is z since $ifun$(worksIn) $<$ fun(worksIn). Consider the following facts in $\mathcal{G}$: {worksIn$(Marie, Lyon)$, worksIn$(Marie, Gordes)$, worksIn$(Joe, Lyon)$, hasPopulation$(Lyon, 513\ 000)$, hasPopulation$(Gordes, 2\ 000)$, hasHusband$(Marie, Joe)$, hasRevenue$(Joe, 1\ 500)$, hasRevenue$(Joe, 800)$}. The triple worksIn$(Joe, Lyon)$ is a correct prediction of r_s, and for the introduced numerical features $<$ hasPopulation$_y$, hasRevenue$_z$ $>$ the two sets of numerical values $(513\ 000, 800)$, and $(513\ 000, 1\ 500)$ are examples that belong to class A. worksIn$(Joe, Gordes)$ is an incorrect prediction. The numerical values $(2\ 000, 800)$ and $(2\ 000, 1\ 500)$ are examples that belongs to class B. If we

do not know where Joe works, the possible instantiations of the numerical values do not belong to any class.

(4.3) Constraining the numerical rule to intervals

[lines 16–19 in Algorithm 1]. To obtain a set of rules for classes A and B defined by a rule r_s, we can discretize the values of the numerical predicates.

We aim to find the purest intervals that can effectively differentiate between examples in class A and class B. However, if we limit ourselves to constraints that only express interval membership for correct groundings of class A, the resulting rule may have low head coverage if the interval is too specific. On the other hand, if we exclude intervals that lead to incorrect predictions, we may overlook rules with high confidence that can enhance the accuracy of predictions in KG completion tasks.

Therefore, we consider both candidate rules. For instance, it is common for people to work in the city where they were born if that city has between 50,000 and 500,000 inhabitants. Moreover, this rule is usually not true for megacities, so we can also consider another rule that excludes cities with over 1,000,000 inhabitants.

Hence, we decided to employ a supervised method to discretize the continuous values of numerical predicates in $\mathcal{B}_{num}$ and keep track of the number of correct and incorrect predictions falling in each interval to consider both membership and non-membership constraints. This method involves constructing a univariate CART Decision Tree (DT), where the numerical predicates serve as features. The DT is binary and built using impurity-based criteria, specifically entropy, as the splitting criteria.

The root of the tree corresponds to the numerical $r_s : \mathcal{B}_{num} \wedge C \ \wedge \ \mathcal{B} \Rightarrow H$ where C is initially empty. At each split, the instance space is divided into two subspaces by constraining the range values of one of the atoms in $\mathcal{B}_{num}$ to the split threshold and hence updating C. The rules at each child node r_{node} are created according to the rule of the parent node and the split made at that node.

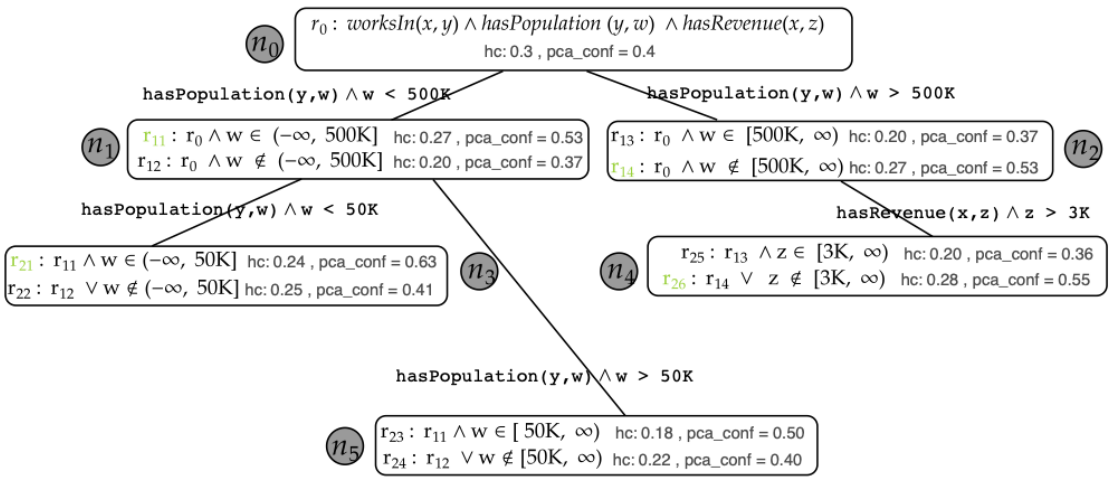

Figure 3.1.: Example of a part of the DT and the considered rules at each node

More specifically, if a split is made using a numerical atom $p(x,y)$ and a threshold α at node i, a membership constraint creates a rule for the left child by updating C with $\wedge\ y \in (-\infty, \alpha]$ and $\wedge\ y \in [\alpha, \infty)$ for the right child. A non-membership constraint, however, creates the rule for its left child by updating C with $\vee\ y \notin (-\infty, \alpha]$ for the left child and $\vee\ y \notin [\alpha, \infty)$ for the right child.

Example. Consider the parent rule r : worksIn$(x,y) \Rightarrow$ livesIn(x,y) and enriching the body of r with two predicates r_0 : worksIn$(x,y) \wedge$ hasPopulation$(y,w) \wedge$ hasRevenue(x,z). Figure 3.1 depicts the construction of the rules at each node for a part of the tree to constrain the values of w and z with membership or non-membership. The $minHC = 0.23$ and the $minC = 0.5$. Node n_1 shows the inclusion and exclusion rules as well as their respective head coverage and PCA confidence constructed using the constraint hasPopulation$(y,w) \wedge w < 500K$.

Furthermore, we ensure that each node only contains the most concise rule. This means that if an atom $p(x,y)$ has already been selected for a split in the path from the root to the child nodes, the constraint already exists in the body

of the parent node. Therefore, instead of adding the constraint, we update the constraint on y (*i.e.*, the range values of the atom $p(x, y)$).

Example. At node n_5, the rule r_{23} is expressed as worksIn$(x, y) \wedge$ hasPopulation$(y, w) \wedge w \in [50K, 500K] \wedge$ hasRevenue(x, z). At node r_{21}, the rule is worksIn$(x, y) \wedge$ hasPopulation$(y, w) \wedge w \notin (-\infty, 50K] \wedge$ hasRevenue(x, z). At node n_4, the rule r_{26} is expressed as worksIn$(x, y) \wedge$ hasPopulation$(y, w) \wedge$ hasRevenue$(x, z) \wedge (w \notin [500K, \infty) \vee z \notin [3k, \infty))$.

We use the stopping criteria based on *minHC* and *minC* of each node's inclusion and exclusion rules to decide when to stop splitting further. The sizes of class A and B are determined using $< X, Y >$ from the previous step. First, we check head instantiations belonging to class A are less than $minHC * headsize(r)$. This means that the support for the membership rule is insufficient to meet the *minHC* requirement, and since support is monotonic, we can safely prune the splitting process for inclusion rules.

Next, we propose a heuristic for excluding rules based on their performance. The heuristic suggests that if the number of incorrect predictions (*i.e.*, instances classified as class B) falls below a certain threshold at a particular node, the minimum coverage requirement *minC* cannot be satisfied. The ideal scenario for satisfying *minC* while using non-membership constraints is to retain all correct instances ($supp(r_s)$ unchanged). In this case, the maximum size PCA body size would be $\frac{supp(r_s)}{minC}$. To achieve this, in the best-case scenario of only excluding incorrect instances, we must remove at least $pca\_bodysize(r_s) - \frac{supp(r_s)}{minC}$ instances. Therefore, if the number of head instantiations for class B falls below $pca\_bodysize(r_s) - \frac{supp(r_s)}{minC}$, we can be sure that by splitting further (and hence reducing $pca\_bodysize(r_s)$), the *marginC* can never be obtained and we can prune the splitting process.

In other words, if none of the rules can fulfill the expected *marginC* and *marginHC*, we stop splitting.

Example. In Figure **??**, rule r_{24} will not appear in node n_5 because generalizing r_{12} with the constraint of the split would not exclude enough incorrect predictions to satisfy the *minC*.

We could select all nodes that meet the requirements of *minHC* and *minC* (note that the root node with an empty C and free variables for numerical predicates can also be kept, so our language bias involves rules that are not closed). However, to limit the number of generated numerical rules and avoid redundant rules covering the same instances, we have implemented a strategy that selects the most general rules along each path from the root to the leaves. Specifically, we choose the rules with the highest PCA body size.

Example. In Figure 3.1, rules corresponding to nodes n_1, n_3, and n_4, namely r_{11}, r_{21}, and r_{25} respectively, meet the requirements of *minHC* and *minC*. We include r_{11} and r_{25} in $\mathcal{E}$ because nodes r_{11} and r_{21} are along the same path from the root, and r_{11} is the more general rule.

(5) Rule Diversity

[line 21 in Algorithm 1]. To maintain diversity, as explained in Section 3.2.1, after each iteration, we remove any atoms that have led to a numerical rule with parent rule r that meets the conditions of *minHC* and *minC* from q_{atoms} This also serves as a pruning strategy.

3.3 EXPERIMENTAL EVALUATION

We have conducted two groups of experiments. First, we evaluate the quality of the set of enriched rules vs. their parent rules. The parent rules have been obtained by running rule mining techniques of AMIE (Lajus et al., 2020) and AnyBURL (Meilicke et al., 2020). Secondly, we have evaluated the performance of KG completion task using these enriched rule sets.

Datasets. We consider three different benchmark datasets that involve numerical values. *FB15K-237-num* and *DB15K-num* are variants of Freebase and DBPedia knowledge graphs involving numerical predicates and values proposed in (García-Durán and Niepert, 2018). *LitWD19K* is one of the three datasets proposed in LiterallyWikidata (Gesese et al., 2021), a recent dataset

Dataset	$\|\mathcal{I}\|$	$\|\mathcal{P}\|$	$\|\mathcal{P}_{num}\|$	$\|\mathcal{G}\|$	$\|\mathcal{G}_t\|$
DB15K-num	12,867	278	251	79,345	9,789
FB15K-237-Num	14,541	237	116	272,115	1,215
LitWD19K	18,986	182	151	260,039	14,447

Table 3.1.: Statistics of the benchmark datasets.$|\mathcal{G}_t|$ denotes the size of test set.

gathered from Wikidata and Wikipedia with a special focus on literals. Table 3.1 shows the statistics for these datasets.

Experimental Setup. All experiments are run on a single machine with a processor 2.7GHz, 8 cores, and 16GB of RAM that runs Mac OS X 10.13. REGNUM is written in Python, and we have used Stardog[1] RDF data management system. The source code and the datasets used in our experiments are publicly available[2]. The time taken for rule generation ranges from 20 minutes to 15 hours, depending on the number of parent rules, their quality, and the KG.

3.3.1 *Rules Quality Assessment*

In this first set of experiments, we compare the quality of the parent rules that could be enriched with the set of enriched rules. Specifically, we compare the percentage of gain regarding PCA confidence and head coverage. To measure the overall quality of the rules, we rely on $F_r = 2 * \frac{pca\_conf(r) * hc(r)}{pca\_conf(r) + hc(r)}$, which is a harmonic mean between the pca_conf and hc. This is because just a high pca_conf or a high hc is not a good indicator of the overall quality of a rule (*i.e.*, the rule can be too specific or not yield good predictions).

Setup. We have run AMIE with default values $minHC = 0.01$ and $minC = 0.1$, and maximum rule length of 3, on the *LitWD19K* and *DBPedia15K* datasets. To limit the number of parent rules, we used an increased $minHC = 0.1$ when running AMIE on *FB15K-237-num*. As AMIE mines only closed rules, no post-processing of the rules obtained was needed. We have run AnyBURL with default parameters except for the maximum rule length being set to 3,

1 https://www.stardog.com/
2 https://github.com/armitakhn/REGNUM

and the rules are learned for 100s with the $min\_conf = 0.03$. We performed post-processing on the obtained rules to retain only closed rules. REGNUM enriches the parent rules with $marginC = 20\%$, $marginHC = 10\%$.

Dataset	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{R}_{enriched}\rvert$	$\lvert\mathcal{E}\rvert$	level 1	level 2	level 3	g_{conf}	g_{hc}	g_F
DB15K-num	4,163	402	2,783	2,747	36	0	+38.3%	-1.2%	+9.9%
FB15K-237-num	9,591	1,187	5,434	4,640	789	5	+28.6%	-4.2%	+9.8%
LitWD19K	2,481	859	9,068	7,764	1,272	12	+31.2%	-2.5%	+3.5%

Table 3.2.: Statistic of rules mined by AMIE, compared to numerical rules in terms of the quality measure.

Table 3.2 and Table 3.3 detail the number of parent rules mined $\mathcal{R}$ by AMIE and AnyBURL, respectively. The number of parent rules that could be enriched with numerical predicates $\mathcal{R}_{enriched}$ and the number of numerical rules obtained by REGNUM $\mathcal{E}$ are also presented. On the three datasets, we compute the average of the rules' pca_conf, hc, and F measure of $\mathcal{R}_{enriched}$ and on the numerical rules $\mathcal{E}$. In the tables, 3.2 and 3.3, we provide the percentage of improvement of PCA confidence, head coverage, and F measure of $\mathcal{E}$ over parent rules $\mathcal{R}_{enriched}$, denoted by g_{conf}, g_{hc}, and g_F, respectively. The results indicate that the pca_conf of the numerical rules increased significantly across all benchmark datasets, irrespective of the rule mining technique used for obtaining parent rules. This improvement has been achieved without sacrificing much head coverage, and the overall quality of the rules (F measure) increased.

When we set a more relaxed value for $marginHC$, we noticed a decrease in the overall quality of rules. However, we were able to obtain more numerical rules. For instance, setting $marginHC$ to 20% on FB15K-237-num in table 3.2 reduces the g_F from 9.8% to 6.22%, but the number of enriched rules increases to 10,141 with 1,744 parent rules that could be enriched.

Approximately 25% of the rules on across all datasets incorporate membership constraints that include intervals. For example, the LitWD19K dataset mines 2,585 rules with membership constraints and 6,483 rules with non-membership constraints using parent rules from AMIE. Similarly, the AnyBURL dataset comprises 2,157 numerical rules with membership and 5,564 numerical rules with non-membership. As elaborated in Section 3.2.2, we expect that member-

Dataset	$\|\mathcal{R}\|$	$\|\mathcal{R}_{enriched}\|$	$\|\mathcal{E}\|$	level 1	level 2	level 3	g_{conf}	g_{hc}	g_F
DB15K-num	1,539	515	2,184	2,052	132	0	+30.4%	-3.5%	+3.3%
FB15K-237-num	7,959	1,688	7,252	6,597	654	1	+29.1%	-3.5%	+8.1%
LitWD19K	1,758	787	7,721	6,407	1,266	48	+29.0%	-3.8%	+4.5%

Table 3.3.: Statistic of rules mined by AnyBURL, compared to numerical rules regarding the quality measure.

ship rules will generally have lower head coverage and higher PCA confidence compared to non-membership rules, which exclude incorrect predictions. This is demonstrated to be true when we limit the rules to only inclusion or only exclusion rules. For instance, for the LitWD19K dataset, the membership rules obtained from AMIE parent rules show g_{conf} of 36.8%, and g_{hc} of -3.0%, whereas for non-membership rules g_{conf} is 30.1%, and g_{hc} is -2.2%. We observe the same trend in all datasets.

We have also explored the use of the Minimum description length principle (MDLP) (Fayyad and Irani, 1993) and Optimal Binning (Navas-Palencia, 2020) as supervised discretization techniques. However, these methods can only discretize a single numerical predicate at a time and cannot handle combinations of numerical predicates. To ensure a fair comparison, we have limited the rules of REGNUM to level 1. We have found that DT can enrich more parent rules by providing more relevant intervals. For example, on the FB15K-237-num dataset in Table 3.2, using MDLP results in the enrichment of 940 parent rules ($|\mathcal{R}_{enriched}|$), leading to a total of 7,005 numerical rules ($|\mathcal{E}|$) with a g_F of 11.7%. On the same dataset, the results of Optimal Binning are $|\mathcal{R}_{enriched}|$ = 874, $|\mathcal{E}|$ = 3,832, and g_F = 12.0%. Finally, using only REGNUM rules with one numerical predicate (level 1) enriches 1,042 parent rules, resulting in 4640 numerical rules and a g_F of 10.13%. Using DT can enrich more parent rules and this shows the relevance of the intervals it proposes.

3.3.2 *KG Completion*

In this second set of experiments, we focus on the task of knowledge graph completion, where we aim to evaluate the effectiveness of integrating the numerical rules obtained via REGNUM with the parent rules for knowledge

graph completion. KG completion aims to predict a missing object o in a fact $(s, p, o) \notin \mathcal{G}$.

For each test data (s, p, o), we examine all the rules mined with predicate p in the head of the rule, *i.e.*, $p(x, y)$. For each such rule, we execute a SPARQL query by substituting x with the subject of the test data s and obtain a set of predictions generated by the rule. Each candidate c can be given by a set of rules $C = \{r_1, ..., r_n\}$. We use four different aggregation strategies to assign a score to each candidate based on the rules that predicted them. The aggregation methods we considered are:

1. The democracy aggregation where the score depends on the number of rules that fired a candidate $\mathcal{S}_c = |C|$.
2. The max-aggregation $\mathcal{S}_c = max\{pca\_conf(r_1), ..., pca\_conf(r_n)\}$ where the rule with the highest PCA confidence defines the score.
3. The noisy-or aggregation $\mathcal{S}_c = 1 - \prod_{i=1}^{n}(1 - pca\_conf(r_i))$.
4. The weighted-F aggregation $\mathcal{S}_c = \Sigma_{i=1}^{n} \frac{1}{\#Prediction(r_i)} \times F(r_i)$ which penalizes the rules that result in many predictions (candidates).

The rules with statistics reported in Table 3.2 are used to find the candidates. To assess the performance of the rules, we report the Hits@10 result, which is the number of correct head terms predicted out of the top 10 predictions. Table 3.4 shows the results of KG completion on three datasets using the rules mined by AMIE vs. the numerical rules of REGNUM added to the set of rules of AMIE. The four different aggregations are used to score the candidates and report the hits@10 results in the filtered setting (*i.e.*, a prediction that already exists in $\mathcal{G}$ or $\mathcal{G}_t$ will not be ranked).

	FB15K-237-num		DBPedia15K		LitWD19K	
	AMIE	AMIE+REGNUM	AMIE	AMIE+REGNUM	AMIE	AMIE+REGNUM
Democ	61.6	61.0	33.8	35.8	31.9	31.6
Max	70.5	71.7	34.5	36.9	32.4	32.6
Noisy-or	68.1	66.9	34.7	37.0	32.5	32.4
Weighted-F	69.1	68.3	34.7	37.0	32.9	32.8

Table 3.4.: Hits@10 results of KG completion with rules of AMIE ($\mathcal{R}$) and numerical rules of REGNUM with the rules of AMIE ($\mathcal{R} \cup \mathcal{E}$)

	AMIE		AMIE+REGNUM	
	Hits@1	Hits@10	Hits@1	Hits@10
DBPedia15K	4.6	7.7	6.4	10.2
FB15K-237-num	5.5	14.7	6.3	15.3
LitWD19K	12.6	22.5	13.9	23.6

Table 3.5.: Hits@1 and Hits@10 results of KG completion with $\mathcal{R}_{enriched}$ and $\mathcal{R}_{enriched} \cup \mathcal{E}$

On all three datasets, we found that adding the rules of REGNUM to the set of rules from AMIE improved the performance of knowledge graph completion when using the Max aggregation method. This suggests that numerical rules can improve predictions. With the Max aggregation method, we know that whenever a candidate is selected, it is because a numerical rule of REGNUM with higher confidence than its parent rule has been chosen. If no numerical rule exists, the parent rule will be chosen.

The marginal benefit of numerical rules on these benchmark datasets can be attributed to the small number of rules that could be enriched, as well as the generic nature of the datasets that do not heavily rely on numerical predicates for accurate predictions. Hence, to better understand the impact of the enriched rules, we focus only on the rules that could be enriched, $R_{enriched}$, and use them for knowledge graph completion. Table 3.5 shows the results using the Max aggregation method, indicating improvements in the accuracy of knowledge graph completion when enriched rules are combined with their respective parent rules.

3.4 CONCLUSION

This chapter presented REGNUM, a new approach that enhances rule mining systems by adding numerical predicates to parent rules. These numerical predicates include constraints for membership or non-membership to intervals obtained through supervised discretization. The enriched rules are considered relevant if they have a higher PCA confidence than the parent confidence without significantly reducing the head coverage. We demonstrated that the

enriched rules have higher average quality and can improve the accuracy of rule mining systems for the knowledge graph completion task.

Future research will explore alternative methods of obtaining constraints, such as using sequential covering approaches. Also applying numerical rules to other domains where numerical values play a critical role in predictions. We also plan to investigate more complex aggregation techniques, such as latent-based Aggregation (Betz et al., 2022), and consider using an in-memory database to improve query run-time, as proposed in AMIE3 (Lajus et al., 2020). Ultimately, we intend to compare our approach's run-time and optimality with an approach that mines numerical rules while finding optimal intervals. We expect our approach to be faster but less accurate.

Another potential avenue for future research is to take a more strategic approach to selecting which numerical atoms to add to a rule. Instead of evaluating all possible combinations, it may be beneficial to consider factors such as correlation and semantics when assessing the independence of numerical predicates. For instance, one could avoid adding both $hasPopulation(x, y)$ and $hasResidents(x, z)$ due to their high correlation or avoid adding both $hasPopulation(x, y)$ and $hasAltitudeFromSea(x, z)$ due to their dissimilar semantics. Such a strategic approach could enhance the discovered rules' quality while reducing the search space.

4

MINING REFERRING EXPRESSIONS ON KNOWLEDGE GRAPHS

The work presented in this chapter has been done with Nathalie Pernelle, Fatiha Saïs, and Gianluca Quercini and has been published and presented at:

Armita Khajeh Nassiri, Nathalie Pernelle, Fatiha Saïs, Gianluca Quercini. "Generating Referring Expressions from RDF Knowledge Graphs for Data Linking". Full paper at International Semantic Web Conference (ISWC) 2020 (Khajeh Nassiri et al., 2020).

Armita Khajeh Nassiri, Nathalie Pernelle, Fatiha Saïs, Gianluca Quercini. "RE-miner for data linking results for OAEI 2020", OAEI Paper at Ontology Matching (OM) Workshop 2020 - co-located with ISWC (Khajeh Nassiri et al., 2020).

4.1 INTRODUCTION AND CONTRIBUTIONS

In Chapter 2, we explored two tasks for refining KGs: knowledge graph completion and data linking. In Chapter 3, we proposed a novel approach that leverages logical rules with numerical predicates to complete the KG. While logical rules (as defined in Definition 3) can be used for knowledge graph completion, they are not inherently suitable for the data linking task. To adapt them for data linking, rules should be designed to conclude on head atom $sameAs(x, y)$ or $\mathcal{B} \Rightarrow x = y$, resembling keys used for data linking introduced in Chapter 2.3. However, previous works have shown that such rules are not always efficient and do not necessarily yield better results for data linking (Symeonidou et al., 2017). To address this, we propose a new approach in this chapter that focuses on mining rules that are valid for only one instance in the knowledge graph: referring expressions. If a description uniquely identifies

an entity in a source KG, applying the same rule in a target knowledge graph should also identify the same entity.

More precisely, a *referring expression* (*RE*) is a description in natural language or a logical formula that can uniquely identify an entity. For instance, the statement "president of the United States who was born in Hawaii" is a referring expression that unambiguously characterizes Barack Obama. It is possible that multiple logical expressions can be used to identify an entity uniquely. This chapter embarks on automatically discovering REs for each entity within a class of a knowledge graph that possesses specific characteristics making them more suitable for data linking.

Contribution

These referring expressions explored in this chapter are conjunctions of atoms, *e.g.*, $isPresident(x) \wedge isPoliticianOf(x, \texttt{USA}) \wedge bornIn(x, \texttt{Hawaii})$[1]. They can also contain existentially quantified variables, *e.g.*, $isPresident(x) \wedge marriedTo(x, y) \wedge hasName(y, \text{"}Michelle\text{"})$. Such conjunctions of atoms are not all relevant when exploited in a data-linking task. As knowledge graphs are built independently and autonomously, individual IRIs are rarely re-used in different knowledge graphs. This is the reason why a referring expression that involves a specified IRI may not be useful for the task of linking instances. Additionally, since data are usually incomplete and generally several referring expressions can be associated with an individual, to foster the utility of REs, it is preferable to diversify the properties involved in the referring expressions of a given individual.

In order to reduce the enormous search space of referring expressions, our approach relies on defining types of graph patterns and quality measures that focus on REs that are more suitable in a data-linking task. Moreover, we direct our attention to REs that cannot be found by instantiating the keys. As a reminder, the keys of a class are sets of properties whose values can uniquely identify one entity of that class. Hence, if the properties of the keys are instantiated, they can each be considered a referring expression. For

1 USA and Hawaii are IRIs (Internationalized Resource Identifier) that refer to the country USA and to the state of Hawaii, respectively.

instance, take the class "book" and imagine that ISBN is key to this class. If we instantiate the books with their corresponding ISBNs, we can be sure to find them each uniquely. Recent approaches in the literature can efficiently discover keys in knowledge graphs (Symeonidou et al., 2014, 2017; Soru et al., 2015b; Atencia et al., 2021), some of which do so by first finding the maximal non-keys (Symeonidou et al., 2014, 2017). Hence, our proposed RE-miner algorithm is based on the search space defined by these non-keys.

More precisely, our contributions are as follows:

- We define graph patterns and quality criteria that specify REs relevant for data linking. This is the first approach in the literature to consider referring expressions for data linking.
- We propose an algorithm, called RE-miner, that computes complementary REs to those that correspond to instantiated OWL2 keys.
- We conduct an extensive set of experiments to evaluate our approach. Our results demonstrate that our algorithm scales well to large datasets and that the discovered REs significantly improve recall in data linking tasks.

4.2 RELATED WORK ON REFERRING EXPRESSIONS

In section 2.3, we discussed the related work for data-linking. This section focuses on the related work of generating referring expressions. Various algorithms have been proposed to automatically discover referring expressions with different objectives, depending on the expressivity of the logical formulas they can generate. For example, some approaches, such as (Dale, 1992; Krahmer et al., 2003), create referring expressions as conjunctions of atoms, while others, such as (Ren et al., 2010b), discover more complex expressions represented in description logics that may involve the universal quantifier. To improve efficiency and reduce the search space, some methods prioritize the minimality of the expression they discover, while others focus on predicate preferences (Galárraga et al., 2020). We take a closer look at the literature on this topic.

The problem of identifying the properties required to refer to an entity was first introduced in (Dale, 1989), where the *Full Brevity* algorithm was proposed to generate the shortest possible referring expression by testing all combinations of properties. However, finding the shortest RE is NP-hard, and a more efficient algorithm was later proposed in (Dale, 1992). The greedy algorithm iteratively adds properties with the most discriminative power to an empty expression; although it does not necessarily produce the shortest RE, it is much more efficient than Full Brevity.

While both algorithms prioritize the number of properties, the length of the RE is not the only criterion for generating a good RE. Sometimes, a slightly longer and more informative RE may be preferable. The *Incremental Algorithm* adds properties to an expression based on a preference order and does not necessarily produce the shortest RE (Dale and Reiter, 1995). (Deemter, 2002) has extended this algorithm to include the generation of (Boolean) expressions referring to sets of target entities (instead of just one) and using negations and disjunctions.

Although logic optimization techniques are used to shorten the resulting REs, in some cases, this algorithm might produce overly lengthy REs; a problem addressed in further research (Gardent, 2002; Horacek, 2004). Incremental algorithms do not lend themselves well to generating *relational REs* (Krahmer and Van Deemter, 2012), that identifies a target through a relation to another entity (*e.g.*, "the dog near the house"). Relational REs are best modeled with a graph (the *scene graph*), where relations between entities are represented as edges that link the corresponding nodes; the generation of referring expressions reduces to searching a subgraph (the *description graph*) that uniquely identifies the target (Krahmer et al., 2003).

(Croitoru and Van Deemter, 2007) takes this graph approach a step further and propose the use of *conceptual graphs*, a logic-based knowledge representation model that enriches the *factual knowledge* with ontological or background knowledge(*e.g.*, "a cup is a vessel"). Ontological knowledge is particularly useful for performing automatic inference. Other approaches turn to description logic as an alternative knowledge representation model (Areces et al., 2008; Ren et al., 2010a).

Similar to conceptual graphs, description logic can model background knowledge and apply reasoning. However, these methods have difficulties scaling to today's large knowledge graphs.

REMI (Galárraga et al., 2020) is an algorithm that mines intuitive REs from a knowledge base by computing the intuitiveness of an expression with the Kolmogorov distance. It operates a trade-off between its length and the use of properties that people are familiar with (*e.g.*, "Paris is better described as the capital of France than the birthplace of Voltaire"). REMI represents expressions in a tree structure representing conjunctions of atoms (*e.g.*, cityIn(x,y) $\wedge$ officialLanguage(y, z) $\wedge$ langFamily(z, Romance)) and identifies the RE with the least cost in terms of intuitiveness through a depth-first search with backtracking.

Our proposed approach, goes beyond discovering a single referring expression and uncovers all the REs that complement those obtained by instantiating key properties. As far as we know, this is the first work to mine referring expressions that are beneficial for linking instances across two knowledge graphs.

4.3 RE-MINER

In this section, we begin by defining the referring expressions in section 4.3.1 and their characteristics, namely minimality and diversity, which are relevant for our mining purposes.

Subsequently, we present our proposed method for automatically discovering REs that are both minimal and diverse for each instance within a given class of a knowledge graph. Our approach to generating such REs involves two distinct steps, as outlined in section 4.3.2. First, we explain generating a set of minimal and diverse REs for each instance, which is accomplished using the *RE-miner* algorithm. We concentrate on discovering REs that complement instantiated keys, as several recent techniques have been developed for discovering keys. Second, we describe how to expand each RE, leading to a more detailed expression.

In section 4.3.3, we demonstrate how these REs can be applied to data linking.

4.3.1 *Problem Statement*

This section will define the referring expressions relevant to our study. We focus on referring expressions that pertain to an individual u within a class $\mathcal{C}$ in a given knowledge graph G. First, to reduce the search space, we focus on referring expressions that can not be obtained by instantiating an OWL2 key can be defined as follows:

Definition 11. (Key). A key $\{p_1, \ldots, p_n\}$ for a class C expresses that:

$$\forall x \forall y \forall z_1 ... z_n (C(x) \wedge C(y) \wedge \bigwedge_{i=1}^{n} (p_i(x, z_i) \wedge p_i(y, z_i)) \rightarrow \ x = y)$$

By definition, each instantiation of the key properties for a class $\mathcal{C}$ will uniquely identify an individual present in $\mathcal{C}$. This instantiation will potentially yield many REs. Nevertheless, this does not represent the complete set of possible minimal REs that can be discovered. Thus, we are interested in enriching this set with REs that only involve non-key properties. To this end, we will only exploit sets of properties included in one of the maximal non-keys of class $\mathcal{C}$ to construct complementary REs.

Definition 12. (Maximal Non-Key). A maximal non-key for a class $\mathcal{C}$ in a knowledge graph $\mathcal{G}$ is a set of properties P such that P is not a key, but the addition of any property to P makes it a key for that class.

Specifically, we define these expressions as follows:

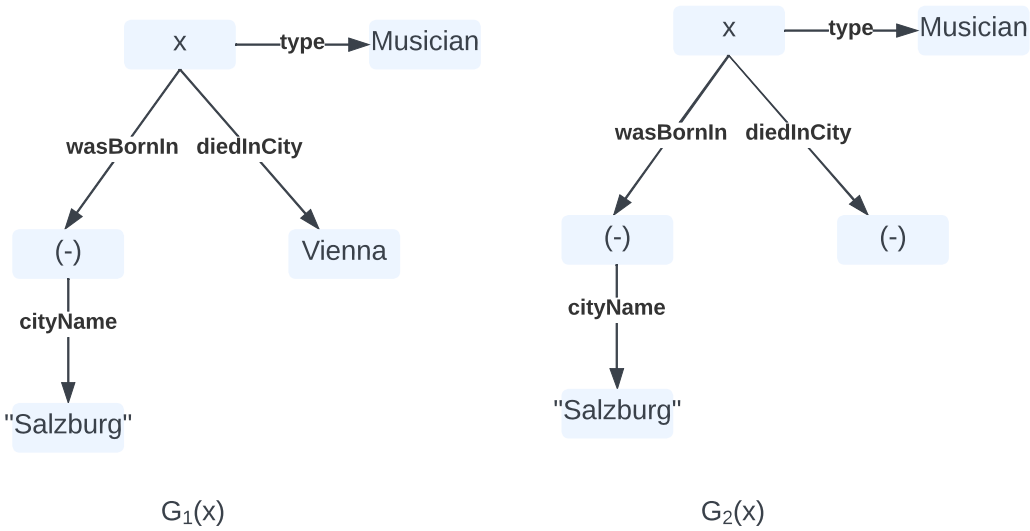

Figure 4.1.: Two graph patterns. $G_1(x)$ is compliant with definition 14, and $G_2(x)$ is not.

Definition 13. (Referring Expression). Given a knowledge graph $\mathcal{G}$, a referring expression, denoted by $RE_k(u)$ for the k^{th} RE of a given individual u of a class C can be expressed by the following first-order logic formula:

$$C(x) \wedge \bigwedge_{p_i \in \mathcal{OP} \cup \mathcal{DP}} p_i(w, y)$$

Such that the formula, existentially closed, is restricted to those conjunctions of atoms (with properties that can be *owl:ObjectProperty* or *owl:DatatypeProperty*) that form a connected graph pattern rooted at x with the leaves being either an individual in $\mathcal{I}$ or a literal in $\mathcal{L}$ and the other nodes being variables.

Definition 14. (Referring Expression Validity). A referring expression $RE_k(u)$ is valid in a graph G if it holds when x is instantiated by u and does not hold for any other individual $v \neq u$ of $\mathcal{C}$ in G.

Example. Figure 4.1 shows two graph patterns, $G_1(x)$ and $G_2(x)$. $G_1(x)$ is a valid referring expression for Mozart, who was a musician born in Salzburg and died in Vienna and is compliant with Definition 13. It can be expressed as:

$$Musician(x) \wedge wasBornIn(x, c) \wedge cityName(c, \text{“}Salzburg\text{”}) \wedge diedInCity(x, Vienna)$$

However, $G_2(x)$ is not compliant with Definition 13 as it includes variables in the leaves of the graph pattern. Therefore, it cannot be considered as a referring expression:

$$Musician(x) \wedge wasBornIn(x, c) \wedge cityName(c, \text{“}Salzburg\text{”}) \wedge diedInCity(x, z)$$

Our goal, driven by data linking, is to discover **minimal** referring expressions. These expressions represent the simplest graph patterns that distinguish one individual from all the others.

Definition 15. (Referring Expression Minimality). A referring expression $RE_k(u)$ is minimal iff:

$$\nexists\, RE_j(u)\ s.t.\ (RE_k(u) \cup \mathcal{F} \cup \mathcal{A}) \models RE_j(u)$$

where $\mathcal{F}$ is the collection of triples and $\mathcal{A}$ is a set of axioms.

To address the challenge of linking incomplete datasets, we narrow our focus to REs that are particularly useful for this task. We utilize various properties to achieve this while keeping the number of REs and their complexity in check. Hence, we do not construct REs that involve different instantiations of the same property, and we only consider *diversified* REs, simply meaning that when a valued property appears in a RE for an individual, it cannot reappear in another RE for the same individual having more atoms. This should not be confused with the notion of minimality.

Example. Take the valid $RE_1(u)$ for the film *Ocean's Eleven*: $Film(x) \wedge hasActor(x, George\_Clooney) \wedge wasCreatedOnYear(x, \text{“2001”})$. Then the following $RE_2(u)$, although valid for this movie and minimal, will not be discovered. $Film(x) \wedge hasActor(x, Julia\_Roberts) \wedge wasCreatedOnYear(x, \text{“2001”}) \wedge editedBy(x, Stephen\_Mirrione)$. Because $RE_2(u)$ is not diversified; since it has more atoms than $RE_1(u)$ while sharing the subgraph pattern $wasCreatedOnYear(x, \text{“2001”})$ with it.

Definition 16. (Diversified Referring Expression) A referring expression $RE_i(u)$ is **diversified** if there is no $RE_j(u)$ with fewer number of atoms that contains a subgraph $p1(x, t_1) \wedge \ldots \wedge p_i(t_{i-1}, t_i) \wedge p_m(t_{m-1}, v_m)$ of $RE_i(u)$, where $v_m \in \mathcal{L} \cup \mathcal{I}$.

Additionally, in the data linking task, one might argue that graph patterns that involve mostly IRIs of individuals are not relevant. Indeed, individuals that are described in two knowledge graphs are rarely represented with the same IRI. This is why we also consider *Expanded REs* that are not minimal but where the individuals' IRIs in a RE are replaced by a description constructed from instantiated key properties.

Definition 17. (Referring Expression Expansion). The expansion $exp(RE_k(u))$ of a referring expression $RE_k(u)$ is a set of referring expressions in which, each leaf node n_i of $RE_k(u)$ that represents an individual i is replaced by an existential variable x_j. These variables are recursively expanded by a subgraph G rooted by n_i representing one possible instantiation of a key K for the class typing i, such that $exp(RE_k(u))$ leads to a graph pattern whose leaves are only literals.

Example. Consider the following referring expression for Marie Curie: $Scientist(x) \wedge wasBornOnYear(x, \text{“1867”}) \wedge isCitizenOf(x, Poland)$. We observe that Poland is a leaf node representing an individual, thus we can expand it by creating keys for the class country (range of the property $isCitizenOf$). Suppose it has two sets of keys, namely $\{hasName\}$ and $\{hasArea, isLocatedIn\}$. We obtain the following RE when *Poland* is replaced by an instantiation of the first key set: $Scientist(x) \wedge wasBornOnYear(x, \text{“1867”}) \wedge$

$isCitizenOf(x, y) \wedge hasName(y, \text{“}Poland\text{”})$. And the following RE, when using the second set of keys for country, and subsequently expanding *Europe* by considering $\{hasName\}$ as a key for the class location (range of the property $isLocatedIn$): $Scientist(x) \wedge wasBornOnYear(x, \text{“}1867\text{”}) \wedge isCitizenOf(x, y) \wedge hasArea(y, \text{“}312,696 < km2 > \text{”}) \wedge isLocatedIn(y, z) \wedge hasName(z, \text{“}Europe\text{”})$.

4.3.2 *Referring Expression Generation*

We outline the procedure of mining the complete set of complementary, minimal, and diversified REs in Algorithm 2. To retain a reasonable search space and to prevent the REs from becoming too complex for data linking, the depth of the aimed REs is restricted to 2. Nevertheless, this restriction can be dropped by applying a recursive adaptation of the function $existentialRE(\mathcal{G}, RE_{new})$ (see line 10 of algorithm 1). The algorithm takes as input a knowledge graph $\mathcal{G}$, a class $\mathcal{C}$, and a Boolean E which is set to True if we aim to mine REs at depth 2 (*i.e.*, REs that contain at least an existential quantifier). If the E is False, the REs will not contain any existential quantifiers.

To generate the set of REs for a given class $\mathcal{C}$ in a knowledge graph $\mathcal{G}$, we start by creating a dataset for that class (line 1). The dataset $\mathcal{D}$, which serves as the search space, consists of all the facts (s, p, o) in $\mathcal{G}$ where the subjects s belongs to $\mathcal{C}$.

Next, we use SAKey (Symeonidou et al., 2014), a key discovery approach, to create the maximal non-key sets NK from the dataset $\mathcal{D}$ (line 2). We then construct the powerset of each set in NK, excluding the empty set, and group them according to their cardinality (line 3). To illustrate, suppose $\text{NK} = \{\{p_1, p_2\}, \{p_3, p_4, p_5\}\}$. At level 1, we have subsets of cardinality 1, namely $\{\{p_1\}, \{p_2\}, \{p_3\}, \{p_4\}, \{p_5\}\}$. Level 2 consists of subsets of cardinality 2 $\{\{p_1, p_2\}, \{p_3, p_4\}, \{p_3, p_5\}, \{p_4, p_5\}\}$, and level 3 contains subsets of cardinality 3, namely $\{\{p_3, p_4, p_5\}\}$.

Since we aim to find minimal REs, the algorithm proceeds level by level (line 4), starting from level 1, which results in REs containing only one atom. To mine REs at level l, we take one set of properties P within that level at a time

(line 6). The algorithm generates subgraph patterns from the search space with instantiated properties P (*e.g.*, $p_1(x, v)$ for level 1 and $p_1(x, y) \wedge p_2(x, z)$ for level 2) as candidate expressions (line 7). We keep the valid REs, among the candidates in RE_{new} (line 8); these expressions are compliant with definition 14 and hence uniquely identify an individual of the class $\mathcal{C}$. If we also aim to find REs of depth 2, *i.e.*, if E is True, the *existentialRE* algorithm detailed in 3 is called on the knowledge graph $\mathcal{G}$ and REs found at level l with properties P (line 10). We add all the recently discovered REs composed of properties P to RE_{level} (line 12) and reiterate until all sets of properties at this level are covered. Then RE_{level} is added to the resulting REs (line 14), and all the facts (s, p, o) involved in these referring expressions are removed from the search space; to ensure both minimality and diversity (line 15).

Algorithm 2: RE-miner

Input: A knowledge graph: $\mathcal{G}$, a class: $\mathcal{C}$, a Boolean:E
Output: The set of minimal REs for instances of type $\mathcal{C}$: RE_{set}

1 $\mathcal{D} \leftarrow$ createData ($\mathcal{G}, \mathcal{C}$) `// serves as the search space`
2 NK $\leftarrow$ generateNK($\mathcal{D}$), $RE_{set} \leftarrow \emptyset$
3 createRankedPowerset(NK) `// Dictionary` *level* `to` *props*
4 **for** *level = 1 to* $|longestNonKey|$ **do**
5 $RE_{level} \leftarrow \emptyset$
6 **foreach** $P \in props_{level}$ **do**
7 $RE_{candidates} \leftarrow constructSubgraphs\ (\mathcal{D}, P)$
8 $RE_{new} \leftarrow validSubgraphs(RE_{candidates})$
9 **if** E = *True* **then**
10 $RE_{new}.add(existentialRE(\mathcal{G}), RE_{new})$
11 **end**
12 $RE_{level}.add(RE_{new})$
13 **end**
14 $RE_{set}.add(RE_{level})$
15 $\mathcal{D} \leftarrow suppressFacts\ (\mathcal{D}, RE_{level})$ `// reduce the search space to preserve minimality and diversity`
16 **end**
17 return RE_{set}

As described in algorithm 2, we can mine more complex REs containing the existential quantifier where the depth of the subgraph patterns will be 2. To do so, all the REs at a level l composed of properties P, are passed to the *existentialRE* algorithm. The output of this algorithm will be a set of referring expressions, each containing one or more existential quantifiers. The details are sketched in algorithm 3.

Algorithm 3: existentialRE

Input: A knowledge graph: $\mathcal{G}$, a set of REs having properties P: RE_{new}
Output: The set of REs with existential variable : $RE_{existential}$

1 $RE_{existentialCands} \leftarrow RE_{new}$.copy()
2 **foreach** $p \in P$ **do**
3 $\quad \mathcal{C}' \leftarrow$ getRange($\mathcal{G}$, p)
4 $\quad$ **if** *exists an instance of* $\mathcal{C}' \in \mathcal{I}$ **then**
5 $\quad\quad$ inducedSubgraphs $\leftarrow$ RE-miner($\mathcal{G}$, $\mathcal{C}'$, E= False)
6 $\quad\quad$ **foreach** $RE \in RE_{existentialCands}$ **do**
7 $\quad\quad\quad$ $RE_{existentialCands}$.add(replaceNodeSubgraph(RE, p, inducedSubgraphs))
8 $\quad\quad$ **end**
9 $\quad$ **end**
10 $\quad$ $RE_{existentialCands}$.remove(RE_{new})
11 $\quad$ $RE_{existential} \leftarrow$ validSubgraphs($RE_{existentialCands}$)
12 **end**
13 return $RE_{existential}$

The *existentialRE* algorithm maintains a copy of RE_{new} in the RE existential candidate set (line 1), which will grow as the algorithm progresses. For each property p in P (line 2), the algorithm retrieves the range of p using $\mathcal{G}$'s schema (line 3). Let the range of p be the class $\mathcal{C}'$. If not all instances of the range class $\mathcal{C}'$ are literals, a subgraph pattern is added to the leaf node to expand it (line 4). These subgraph patterns must be selected to ensure that the resulting pattern is a valid RE when replaced. To this end, using RE-miner, algorithm 2, we construct REs of depth one by setting E to False for the class $\mathcal{C}'$ (line 5). It should be noted that to have the complete and minimal resulting $RE_{existential}$ when creating the dataset for $\mathcal{C}'$ (line 1 of algorithm 2), we only keep those instances $o \in \mathcal{C}'$ that are involved in such facts $(s, p, o)|s \in \mathcal{C}$. The resulting

REs from RE-miner are kept in *inducedSubgraphs* (line 5). Then for each referring expression in the candidate set (line 6), we replace the applicable node, based on p, with the appropriate subgraph in *inducedSubgraphs* (line 7). In the end, we remove the REs of depth one RE_{new} we had added initially to the existential candidate set (line 10) and return the unique subgraphs, which are valid referring expressions having depth 2 (line 11).

We have implemented a post-processing step to obtain the expansion of referring expressions discovered by *RE-miner*, as defined in Definition 17. This step involves using the set of minimal keys of the class to which each IRI u in the leaves of a given RE belongs. We exploit the set of minimal keys of the class u belongs to and expand the RE by instantiating properties of every minimal key. If the keys involve object properties, this step is re-performed recursively on the generated IRIs until either a maximum depth d specified beforehand is reached, or the leaves only correspond to literal values. To avoid regenerating the minimal keys each time, we store them on disk and generate them only when a new class is considered. Finally, the resulting graphs from the expansion of each IRI are combined with other IRI expansions to construct the expanded REs.

4.3.3 *Referring Expressions for Data Linking*

Each $RE(u)$ declared for an individual u in the source knowledge graph $\mathcal{G}_1$ can be expressed by a linking rule as follows :

$$\forall x RE(x) \rightarrow sameAs(x, u)$$

where $RE(x)$ can be rewritten using the classes and properties of a target KG $\mathcal{G}_2$ at the linking step. These rules can be represented in SWRL[2]. Hence, to discover identity links between individuals described in two given KGs, we focus on referring expressions that only involve mapped properties and individuals belonging to classes that have been aligned. Such mappings can be obtained using schema-matching techniques, some of which are discussed in section 2.3. It should be noted that in data linking approaches that rely on

2 https://www.w3.org/Submission/SWRL/

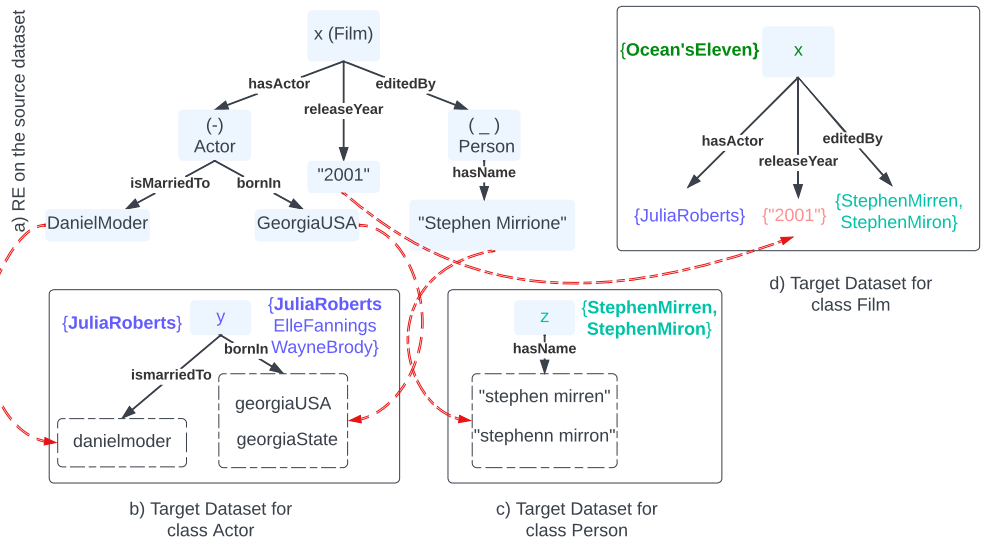

Figure 4.2.: Bottom-up linking with $RE_k(u)$. (a) is a valid RE for the Film *Ocean's Eleven* in the source dataset. We adopt a bottom-up approach where the traversal begins from leaf nodes. (b,c) For each of the RE's leaf nodes n_i in (x, p, n_i), verifies the matches (s, p, o) in the target dataset for the class of x; such that o has a high similarity with n_i. (d) The matches are intersected at the internal nodes until those intersections at the root are reported as links.

schema matching, having a complete set of mappings between properties and classes is not always necessary, and very simple approaches can suffice. For instance, in the OAEI 2019 Knowledge Graph track (Algergawy et al., 2019), a baseline solution that adopted a terminological approach and used only schema labels achieved an F-measure of 0.79 for mapping properties.

The linkage rules introduced above can be used either logically to deduce identity links or by linking tools where simple similarity measures and aggregation functions can be introduced. Since available existing linking tools like (Ngonga Ngomo and Auer, 2011; Volz et al., 2009a; Ma et al., 2019) do not consider such intricate graph patterns (*i.e.*, not just paths of properties), we have developed a simple bottom-up approach explained in Figure 4.2, where

normalizations or classical similarity measures can be declared and applied to datatype properties.

We consider a data linking problem between a source KG in which the REs are discovered and other target KGs/datasets with a non-empty set of properties mapped to the source dataset (that can be obtained using ontology alignment tools (Euzenat and Shvaiko, 2013)). The linking process is comprised of exploiting, for every individual u in the source KG, the set of distinct $RE(u)$s to find all the individuals x in the target datasets that check $RE(u)$. When a RE is discovered in the source dataset, it cannot necessarily be assumed valid for other target datasets. Indeed, even if the source dataset is voluminous, several distinct individuals that can instantiate a RE may exist in the other dataset. Theoretically, when the unique name assumption (UNA) is fulfilled, only one $sameAs(u, x)$ link can be found for a given $RE(u)$ in the target dataset. If this is the case, the quality of $RE(u)$ has to be weakened. Therefore, we assign to every $RE(u)$ a confidence degree inverse proportional to the number of distinct links the $RE(u)$ finds.

To select the best identity link(s), we adopt a voting strategy that assigns weight to each link based on the sum of the confidence degrees of the REs that can instantiate the link. The instance(s) associated with the link(s) having the highest score is selected as the best identity link(s). These confidence degrees can be stored and updated for future data-linking tasks on the source KG.

4.4 EXPERIMENTAL EVALUATION

We conduct two experiments to evaluate *RE-miner*. The first set of experiments is quantitative and investigates the average number of REs discovered for each entity in a class $\mathcal{C}$ of a knowledge graph, the average number of nodes in the graph representation of REs, and the algorithm's scalability. The second set of experiments examines the potential of the discovered REs in enhancing the data-linking task.

All experiments run on a machine with a processor 2.7GHz, 8 cores, and 16GB of RAM that runs Mac OS X 10.13. The source code and the datasets used in our experiments are publicly available[3].

4.4.1 *Datasets*

Below, we summarize the key characteristics of the four datasets used in our experiments.

DBpedia-YAGO.[4] We use 10 different classes of YAGO and DBpedia knowledge graphs. The data for these 10 classes are the same data used in VICKEY (Symeonidou et al., 2017), where the properties of the two knowledge graphs have been aligned manually. Moreover, the properties of YAGO have been rewritten using their DBpedia counterparts. This dataset contains 206,736 ground truth entity pairs.

IM@OAEI2019.[5] We use the Sandbox SPIMBENCH dataset of the instance matching track at OAEI 2019, its gold standard is available and consists of 300 entity pairs. This dataset is composed of a Tbox and an Abox for each of the source and target ontologies. The goal of the task is to match instances describing the same *Creative Work*, which can be a news item, blog post, or program.

IM@OAEI2020.[6] We participated in the OAEI campaign of 2020, and hence we could use both Sandbox and the MAINBOX of SPIMBENCH dataset of the instance matching track at OAEI 2020. The SANDBOX dataset has about 380 instances and 10000 triplets, and the MAINBOX dataset with about 1800 instances and 50000 triplets.

IM@OAEI2011.[7] We use IIMB (ISLab Instance Matching Benchmark) dataset which consists of a set of interlinking tasks used by the instance matching track of OAEI 2011. The source dataset (File 000) describes movies, locations,

3 https://github.com/armitakhn/RE-miner
4 https://github.com/lgalarra/vickey
5 https://project-hobbit.eu/challenges/om2019/
6 https://project-hobbit.eu/challenges/om2020/
7 http://oaei.ontologymatching.org/2011/instance/index.html

actors, etc. Files 001 to 080 are generated by applying several transformations to the source dataset. A gold standard containing around 12.3k identity links has been provided for each of these files.

4.4.2 *Quantitative Results*

Here, we study the scalability of our approach and will report the number of REs found on average for each individual in the considered knowledge graph, as well as the average number of nodes in the graphs representing the discovered REs; first at depth 1 and then at depth 2. Initially, we run *RE-miner* on the 10 classes of YAGO at depth 1, *i.e.*, without allowing for any existential variables. Table 4.1 details the characteristics of each of these 10 classes. Furthermore, this table shows the number of discovered referring expressions for each class and their run time. We can observe that the process takes less than 2 minutes for all classes except for organization, which took more than 3 hours to complete[8]. To compute how many REs each instance of $\mathcal{C}$ has on average, we divide the number of REs to the number of instances for class $\mathcal{C}$. On average, there are less than 7 REs per individual for all classes, except for the most voluminous class organization, with almost 158 REs for each individual. Without having limited the number of discovered REs' atoms, these expressions do not tend to be complex; the maximum number of atoms among all 10 classes is 4, and on average, each RE has two atoms or less[9].

Similar results have been obtained on OAEI2011 and OAEI2019 datasets with 1.19 and 3.75 average atoms and a maximum of 4 and 8 atoms, respectively (see Table 4.4).

Example. The following examples translated to natural language, have been chosen among the REs of depth 1: (i) Yellow Submarine is an album created by the Beatles on date 1966-05-26. (ii) MIT university's motto is *mind and hand*. (iii) Charles Louis Alphonse Laveran is a scientist who was born in year 1845

8 Note that the non-key sets had been computed beforehand in all experiments.
9 Note that the $rdf : type$ properties are not being counted in the number of atoms.

Class	#triples	#instances	#properties	#NKs	#REs	runtime	max#atoms	avg#atoms
Museum	81.6k	21.1k	7	5	53.5k	2.6s	3	1.23
Mountain	116.7k	32.9k	6	4	59.2k	1.4s	3	1.28
Book	123.6k	41.8k	7	6	66.3k	3.5s	3	1.27
University	131.8k	23.3k	9	9	161.8k	17.7s	3	1.62
Scientist	335.6k	93.1k	18	92	309.9k	64.0s	4	1.58
Album	381.1k	137.1k	5	2	212.1k	14.7s	3	1.30
Actor	514.7k	108.4k	16	69	725.6k	95.1s	3	1.74
Film	533.5k	123.9k	9	7	690.9k	102.3s	4	1.77
City	1.1M	83.5k	17	29	1.2M	109.7s	3	1.23
Organization	2.2M	430.3k	17	43	68.3M	3.48h	4	2.05

Table 4.1.: Class statistics, number of non-keys (#NKs), number of discovered REs at depth 1 (#REs), runtime, and size of the REs

in Paris, graduated from university of Strasbourg and has won the Nobel prize in Physiology or Medicine.

To show how many more REs can be found at depth 2 (*i.e.*, REs that contain at least an existential quantifier), we run *RE-miner* with the Boolean E set to True on the three classes of YAGO having the least number of referring expressions at depth 1. As described in section 4.3.2, the algorithm should create the dataset for the class the variable belongs to. To this end, we use instances of this class and all its sub-classes in the non-saturated (*i.e.*, no OWL2 entailment rule has been applied) YAGO version 3.1[10] to ensure having a dataset with at least 1000 instances whenever possible. Table 4.2 shows that, as expected, many additional REs can be generated at depth 2. However, the proportion of REs with only literals values at leaf nodes is rather small, and we can use those REs for data linking. On average, these referring expressions have 2 atoms more than REs of depth 1.

4.4.3 *Data Linking*

Here, we evaluate data linking on the 3 datasets, each time comparing the results with the previous works in the literature that used the same datasets.

10 http://resources.mpi-inf.mpg.de/yago-naga/yago3.1/yago3.1_entire_tsv.7z

Class	#REs	runtime	%AllLiterals	max#atoms	avg#atoms
Mountain	150.8k	3006s	13.4%	6	3.03
Museum	1.4M	3143s	5.2%	5	3.57
Book	1.3M	1.2h	2.50%	6	3.47

Table 4.2.: The number of additional REs detected at depth 2 (#REs), runtime, percentage of REs having only literals in leaf nodes, the maximum and average number of atoms.

We study the advantage of using REs of depth 1 and 2, REs plus keys (*i.e.*, the complete set of referring expressions), and expanded REs. For each of the datasets, we compare the results with a baseline approach that picks random subgraph patterns (*i.e.*, random expressions) and uses them for linking just as it is done with referring expressions. To be fair when comparing the random baseline results to that of REs for data linking, for each dataset, a) the number of generated random expressions is the same as that of discovered REs. b) the size of the random baseline's subgraphs comes from the same distribution as the discovered referring expressions. In other words, the same number of random subgraphs and referring expressions with n atoms exist. c) the results for the random baseline are averaged over three runs.

DBpedia-YAGO. We report our linking results on those 8 classes of this dataset for which other approaches had previously published results. We have used all REs of depth 1 whose statistics were delineated in Table 4.1, with strict string equality. The quality of linking results is reported in terms of precision, recall and F-measure, and is compared to the results of linking with keys (Ks), keys and conditional keys (Ks + CKs) reported in (Symeonidou et al., 2017), ontological graph keys (OGK) reported in (Ma et al., 2019), and the random baseline (RBL). A conditional key is a valid key for a specified part of a class's instances (Symeonidou et al., 2017). Ontological graph keys defined in (Ma et al., 2019) are a variant of keys defined by a graph pattern extended by ontological pattern matching. Table 4.4.3 shows that RE-miner outperforms the other approaches regarding recall and F-measure on all classes except book. More precisely, only using REs of depth 1, we can detect much more correct links without significantly changing the precision. We also observe that the baseline solution – taking thousands to millions of random subgraphs,

depending on the dataset, and using them for linking – results in much lower scores than REs; showcasing that using the referring expressions discovered through RE-miner are indeed effective for linking.

IM@OAEI2011. We first evaluate our data linking results on the entire IIMB dataset. IIMB is made of 13 different classes (*e.g.*, person, actor, location, etc.); 5 of which are at the top of the ontology according to the schema. We create the saturated dataset for these five classes and discover all minimal and diversified REs of depth 1 and 2 on the source file 000. We use these REs to find identity links in the files 001 to 010. The accuracy measures reported in Table 4.4 are averaged over these ten files to compare to the results of the Combinatorial Optimization for Data Integration (CODI) system (Huber et al., 2011), which reformulates the alignment problem as a maximum-a-posteriori optimization problem. We can observe that *RE-miner* outperforms CODI by a large margin of 12% to 7% in recall and F-measure. Also, the baseline solution, which generates 1.3 M random expressions regardless of being RE or not, exhibits poor results. Similar to the previous datasets, this performance reassures us that the RE-miner algorithm is beneficial for linking. We also investigated the effects of using expanded RE and observed that it helps increase the recall by 5.1% on average over REs of depth 1.

Moreover, we compare data linking results using the discovered REs, on the class Film, against data linking with keys reported in (Atencia et al., 2014a). We obtained a high F-measure of 99% and gained about a 70% increase in the recall.

IM@OAEI2019. We report the linking results using the discovered REs of depth 1 for the *Creative Work* class on the source dataset of SPIMBENCH; as the datasets were not saturated, we could not mine and use REs of depth 2 for linking. We compare our results to the three systems with the best performances in the competition[11]: Lily (Wu et al., 2019), AML (Faria et al., 2019), and FTRLIM (Wang et al., 2019), as well as the baseline solution. Looking at Table 4.4, we observe that the random baseline approach is the least effective and that REs alone have comparable performance to the other three systems. However, when combined with the instantiation of keys, resulting in the full set of REs, they outperform all other systems achieving an

11 http://ceur-ws.org/Vol-2536/oaei19_paper0.pdf

Class	Recall					Precision					F1				
	Ks	Ks+CKs	OGK	RBL	REs	Ks	Ks+CKs	OGK	RBL	REs	Ks	Ks+CKs	OGK	RBL	REs
Actor	0.27	0.60	0.66	0.19	**0.69**	0.99	0.99	1.00	0.37	0.99	0.43	0.75	0.79	0.25	**0.81**
Album	0.00	0.15	-	0.35	**0.65**	1.00	0.99	-	0.22	0.98	0.00	0.26	-	0.27	**0.78**
Book	0.03	0.13	**0.85**	0.12	0.80	1.00	0.99	0.97	0.38	0.98	0.06	0.23	**0.90**	0.18	0.88
Film	0.04	0.39	-	0.30	**0.73**	0.99	0.98	-	0.73	0.94	0.08	0.55	-	0.43	**0.82**
Mountain	0.00	0.29	-	0.05	**0.78**	1.00	0.99	-	0.08	0.99	0.00	0.45	-	0.06	**0.87**
Museum	0.00	0.29	0.42	0.20	**0.85**	1.00	0.99	0.99	0.34	0.99	0.00	0.45	0.58	0.25	**0.91**
Scientist	0.00	0.29	0.67	0.24	**0.70**	1.00	0.99	0.99	0.14	0.99	0.00	0.45	0.80	0.18	**0.82**
University	0.09	0.25	0.50	0.29	**0.68**	0.99	0.99	0.96	0.64	0.98	0.16	0.40	0.66	0.40	**0.80**

Table 4.3.: Linking results with keys (Ks), conditional keys and keys (Ks+CKs), ontological graph keys (OGK), random baseline (RBL), and REs of depth 1.

Dataset	#classes	#triples	#props	#NKs	#REs	System	Precision	Recall	F-measure
IIMB OAEI 2011	13	87.3k	23	17	1.3M	REs	0.92	0.87	0.90
						REs+Ks	0.93	**0.88**	**0.91**
						CODI	**0.94**	0.76	0.84
						RBL	0.69	0.29	0.41
Film OAEI 2011	1	11.8k	13	4	1.2M	REs	0.99	**0.98**	**0.99**
						Ks	**1.00**	0.27	0.43
						REs +Ks	0.99	**0.98**	**0.99**
						RBL	0.89	0.03	0.06
SPIMBENCH OAEI 2019	1	6.2k	18	3	1.6k	REs	0.98	0.84	0.91
						REs + Ks	**0.99**	0.99	**0.99**
						Lily	0.84	**1.00**	0.91
						AML	0.83	0.89	0.86
						FTRLIM	0.85	**1.00**	0.92
						RBL	0.78	0.87	0.82

Table 4.4.: Class statistics and linking results of REs + keys, random baseline (RBL), and other systems compared to results with REs of depth 1 and 2 on IM@OAEI2011 dataset and the class Film of IM@OAEI2011, and with REs of depth 1 on IM@OAEI2019 dataset.

F-measure of 99%. The average confidence of the discovered links is 85.5%, whereas this number increases to 97.9% among the links picked through the voting strategy described in section 4.3.3.

IM@OAEI2020. We report the linking results using the full set of REs of depth 1 for the *Creative Work* class on the source dataset of SPIMBENCH. The MELT (Hertling et al., 2019) framework is used for evaluation. Matching EvaLuation Toolkit (MELT) is a framework optimized for OAEI campaigns, facilitating submissions to the SEALS and HOBBIT evaluation platforms. The SPIMBENCH track, on which we evaluate our performance, is available on the HOBBIT, Holistic Benchmarking of Big Linked Data, platform[12]. We used MELT to wrap it as a HOBBIT package, and as our implementation is in Python, we used MELT's External Matching. We evaluated our performance against AML (Faria et al., 2019), Lily (Wu et al., 2019), FTRL_IM (Wang et al., 2019), and LogMap (Jiménez-Ruiz, 2019), who also participated in this track in 2020, and compared the results in Table 4.4.3. For the Sandbox dataset, we created 6920 REs, while for the Mainbox dataset, there were 39892 REs, including 14085 from key instantiation. Our system outperformed the other systems regarding precision and F-measure on both datasets, exhibiting

12 http://project-hobbit.eu/

slightly better performance than Lily. However, our system's time performance was slower due to the necessity of computing the keys and non-keys of a given class using a Java-based application before finding the REs. Further optimization can be done to decrease the runtime.

		Precision	Recall	F-measure	Time
SANDBOX	AML	0.8348	0.8963	0.8645	6446
	Lily	0.9835	**1.0**	0.9917	2050
	FTRL-IM	0.8542	**1.0**	0.9214	**1525**
	LogMap	0.9382	0.7625	0.8413	7483
	RE-miner	**1.0**	0.9966	**0.9983**	7284
MAINBOX	AML	0.8385	0.8835	0.8604	38772
	Lily	0.9908	**1.0**	0.9953	3899
	FTRL-IM	0.8558	0.9980	0.9214	**2247**
	LogMap	0.8801	0.7094	0.7856	26782
	RE-miner	**0.9986**	0.9966	**0.9976**	33966

Table 4.5.: Comparison of Performance in SPIMBENCH track of OAEI 2020 on Sandbox and Mainbox datasets. The time performance is reported in ms.

Relevancy of diversity. We also performed another set of experiments to observe the effects discovering diversified REs brings to the data-linking task. By modifying the RE-miner algorithm, we discovered all minimal REs on the same three classes of the DBpedia-YAGO dataset presented in table 4.2. We observed a considerable increase in the number of discovered REs (*e.g.*, it almost doubled for the class book); whereas the recall and F-measure of the linking task either remained the same or slightly decreased (*e.g.*, for the class book, it dropped by 2%). These results support that using diversity as a quality criterion for referring expressions is beneficial in limiting the number of REs while preserving the quality of data linking.

To sum up, we showed that using REs improves data linking results compared to previous works and the random baseline. The results were verified on different datasets containing classes with 5 to 23 properties and 300 to 137k instances.

4.5 CONCLUSION

In this chapter, we proposed an approach that efficiently discovers referring expressions by reducing the search space by leveraging non-keys. The generated REs are adapted to a data-linking task through the notions of minimality and diversification. We showed that *RE-Miner* can scale to classes consisting of millions of triples. The defined REs can significantly improve the performance of instance matching and boost the recall of rule-based data linking methods.

In future work, we plan to extend RE-miner to incorporate various similarity measures (*e.g.*, Jaccard, Levenstein, Jaro, etc.) for comparing Literal values based on their nature (*e.g.*, time, date, etc.).

Another area of future work is to explore the use of REs for other downstream tasks. One intriguing task is implicit entity linking, where the goal is identifying the entity to which a given text refers. This can also be helpful for disambiguation of a text.

5 KNOWLEDGE GRAPH REFINEMENT BASED ON TRIPLET BERT-NETWORKS

The work presented in this chapter has been done with Nathalie Pernelle, Fatiha Saïs, and Gianluca Quercini and has been presented at:

Armita Khajeh Nassiri, Nathalie Pernelle, Fatiha Saïs, Gianluca Quercini. "Knowledge Graph Refinement based on Triplet BERT-Networks". Full paper at Deep Onto NLP workshop 2022- co-located with ESWC (Nassiri et al., 2022).

5.1 INTRODUCTION AND MOTIVATION

In this chapter, we propose GilBERT, an approach for Knowledge Graph Refinement based on Triplet BERT-Networks.

Motivation

In the previous chapter 4, we introduced RE-miner, a technique for mining referring expressions on knowledge graphs, which we used for data linking and demonstrated its efficacy for this refinement task. In the future work of chapter 4.5, we discussed the potential of using referring expressions for other tasks, such as implicit entity linking. Implicit entity linking pertains to identifying and linking an entity from a text to a corresponding entity in a knowledge graph without explicitly mentioning or annotating the target entity. While there is limited research on this topic, (Perera et al., 2015) highlights the prevalence of implicit mentions in clinical documents, underscoring the need to address them for disambiguation, for instance, *acute inflammation of appendix* is used for *appendicitis*. In (Perera et al., 2016), the ubiquity of

implicit mentions of entities in tweets is discussed and it's mentioned that these references can be very much time and trend dependent. The problem they aim to address is when given a tweet with an implicit entity mention of a particular type (*e.g.*, Movie, Book), the output should be the entity mentioned by the tweet for a given knowledge base. For instance, *Ellar Coltrane on his 12-year movie role* refers to the movie *Boyhood*. Implicit entity linking usually relies on analyzing the context and leveraging other information available in the text and can be helpful for the disambiguation of a text.

With this in mind, we conceive the idea of constructing an embedding space in which the embeddings of textual sequences, formed from referring expressions of entities, are positioned close to one another. For example, let us consider these 3 REs for Marie Curie: $Scientist(x) \wedge isMarriedTo(x, PierreCurie)$, $Scientist(x) \wedge wasBornIn(x, Warsaw) \wedge hasAcademicAdvisor(x, GabrielLippmann)$, $Scientist(x) \wedge wasBornOnYear(x, "1867") \wedge isCitizenOf(x, Poland)$. By creating textual sequences from these REs and embedding them in close proximity to each other in the embedding space, when a sentence like "the female scientist born in 1867 in Poland and living in Paris" is embedded, we can potentially associate the implicit entity with Marie Curie by examining its neighbors in the embedding space.

The factors mentioned above, coupled with the remarkable achievements of transformers and language models, serve as strong motivation for this chapter. We propose a method based on fine-tuning language models designed for two important KG refinement tasks. The absence of a benchmark dataset for implicit linking and the importance of relation prediction and triple classification as refinement tasks set the motivation for the proposed approach. Relation prediction (*i.e.*, predicting the missing relation $(h, ?, t)$) assists in completing the knowledge graph and retrieving more comprehensive query answers, including the identification of hyperlinks and sameAs relations. Triple classification (*i.e.*, binary classification of whether a fact (h, r, t) is true or not) plays a crucial role in fact-checking. Therefore, our approach involves generating textual sequences and clustering facts related to entities or relations.

As discussed in Chapter 2, knowledge graph embedding techniques are commonly employed for addressing these downstream tasks. They learn a

scoring function that assigns high scores to true triples and low scores to false ones. However, these techniques often have computational overhead during evaluation. For example, in link prediction tasks, the tail entity is replaced with all available entities in the KG, and the scores of the resulting triples are computed and ranked to choose the tail entity. Similarly, in relation prediction tasks, the relation is substituted with all possible relations in the KG, and the relation belonging to the triple with the highest score is chosen. To mitigate this overhead, we propose an approach that minimizes the need for substitutions during evaluation in the relation prediction tasks. During model training, we only consider the head and tail entities (and possibly related external information for them). Consequently, there is no longer a need to consider all relations in the KG, rendering the substitutions, as mentioned before, unnecessary. More precisely, we focus on generating textual sequences from each relation's head and tail entities and creating clusters using a triplet network. This aligns perfectly with our idea of using referring expressions for implicit entity linking in the future. For triple classification, we examine the one-hops of each entity to create the textual sequence from the facts of that entity and create clusters for it. Traditional knowledge graph embedding techniques employ a threshold value, denoted as σ, where triples with scores below this threshold are labeled as incorrect, and those above it are labeled as correct. In this chapter, we will follow a similar approach for evaluation, as elaborated further in Section 5.3.

Contribution.

This chapter proposes a novel approach that utilizes a triplet architecture of a pre-trained language model like BERT to cluster facts related to a specific entity or relation in an embedding space. By fine-tuning BERT with a triplet loss that takes textual sequences created from facts as input, GilBERT addresses some of the limitations of previous methods. The evaluation protocol utilizes FAISS (Johnson et al., 2017), a spatial semantic search technique, to compute the proximity of a test example's embedding to the points in the embedding space, reducing the overhead caused by candidate substitutions. Experiments show that GilBERT is more suitable for few-shot learning settings in relation prediction. Additionally, unlike translational and semantic matching-based

models presented in 1.5 that require additional efforts to incorporate supplementary information like entity descriptions, types, or paths, our approach easily integrates such information. Finally, we propose techniques to generate more challenging negative information for the triplet network, capturing some of the KG's semantics.

5.2 GILBERT

We introduce GilBERT, an approach that involves fine-tuning BERT to embed information about entities or relations closer together in clusters, depending on the refinement task. This is achieved by using textual sequences obtained from different parts of a triple as the "information". However, for some downstream tasks, not all triple parts may be known during evaluation, such as when the relation is unknown in relation prediction. Therefore, to create the embedding space, we only use the available parts of the facts during evaluation to ensure that the test data is embedded in the same way as during training.

Let us define a partial fact PF as a sequence of text generated from a triple (h, r, t) by concatenating different parts of either the entities or entities and relations. For instance, consider the triple $(Barack\_Obama, isMarriedTo, Michel\_Obama)$. In this case, PF_{rt} would be "isMarriedTo Michel_Obama", and PF_{ht} would be "Barack_Obama Michel_Obama". We can also replace the entity with its corresponding description wherever possible to enrich the model with additional background knowledge.

We employ a triplet network architecture of BERT, inspired by Sentence-BERT (Reimers and Gurevych, 2019), to generate the clusters in the training data. The network maps partial facts of an entity (or relation) closer to each other, while partial facts of other entities (or relation) are mapped farther apart. The network takes three partial facts PF as input, one as the anchor, one as positive (similar to the anchor), and one as negative (dissimilar to the anchor). To obtain a fixed-sized embedding vector for the input PF, we apply a mean pooling layer to BERT's output layer instead of using the output of the [CLS]

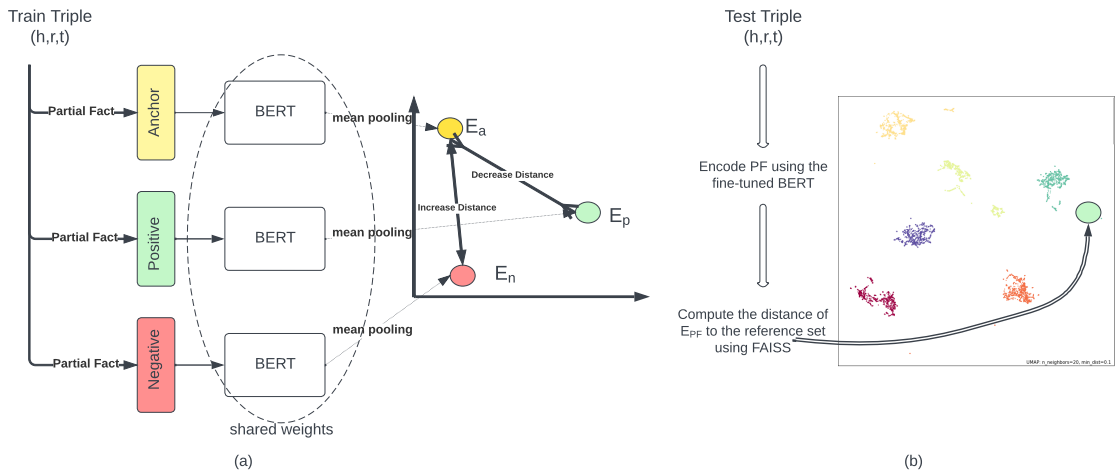

Figure 5.1.: (a) Overview of fine-tuning BERT using the partial facts of triples in $\mathcal{G}$. (b) Overview of evaluation, clusters obtained on FB15K dataset for the relation prediction task, illustration with UMAP.

token produced in the last layer of BERT, as done in KG-BERT. Experiments and (Reimers and Gurevych, 2019) show that the pooling layer output provides better embeddings for the given input than the $[CLS]$ token embedding. This architecture clusters all the information about an entity or relation.

Utilizing PFs as model inputs enables us to use only the fact parts available during evaluation, such as the head and tail, for a relation prediction task. This eliminates the evaluation time overhead of previous models that required generating all possible substitutions and ranking their plausibility using the learned scoring function.

Figure 5.1 illustrates the fine-tuning process of BERT through the triplet network. The network consists of three identical networks with shared parameters and takes in three input textual sequences: the anchor, positive, and negative partial facts. The objective is to place the anchor and positive in the same cluster while separating the anchor and negative into different ones. Specifically, the embedding vectors of the anchor E_a and positive E_p are pushed closer together, while the distance between the anchor E_a and negative E_n is increased. The triplet loss function is defined as follows:

$$\mathcal{L} = max(0, dist(E_a, E_p) - dist(E_a, E_n) + \gamma),$$

where γ is the safety margin ensuring that the embedding of the negative is at least γ further from the anchor than the positive.

For each gold triple (h, r, t) in $\mathcal{G}$, we create n sets of input partial facts (PFs), including anchor, positive, and negative PFs, to train the model. As discussed in Chapter 1.5, creating negative examples can significantly impact the performance of knowledge graph embedding (KGE) techniques. Therefore, it is crucial to generate hard negative examples to prevent the loss function $\mathcal{L}$ from being too easy. In the literature, negative examples are typically created by corrupting a gold triple by replacing its h or t. Other methods, such as (d'Amato et al., 2021), using background knowledge, such as the KG schema, to select better negatives.

Our approach assumes a schema is unavailable and aims to capture the KG's semantics when generating input data. The data creation process varies depending on whether the clustering is based on entities or relations.

Clustering based on Relation:

We use the partial fact PF_{ht} to create the anchor for relation-based clustering. For each positive sample, $PF_{h_i t_i}$ is generated by randomly selecting a fact from the set $\{(h_i, r, t_i) \mid (h_i, r, t_i) \in \mathcal{G}\}$, while the negative samples $PF_{h_j t_j}$ are created from the set $\{(h_j, r', t_j) \mid (h_j, r', t_j) \in \mathcal{G} \wedge r' \in close_r\}$. We aim to create hard negative samples to avoid trivial solutions. We achieve this by creating PFs from triples whose relation is semantically similar to that of the anchor and positive.

We define the set F_r as the set of all triples in $\mathcal{G}$ having the relation r. We use two criteria to select the set $close_r$: (1) the percentage of tail entities shared with F_r among all relations in $\mathcal{R}$, and (2) the percentage of head entities shared with F_r among all relations in $\mathcal{R}$. These percentages reflect the domain and range of the relation, and we consider the set $close_r$ as the set of all relations whose computed sharing percentage is greater than a given threshold. For example, if $close_{BornIn} = \{worksIn, diedIn, locatedIn, studiedIn\}$, we choose

relations that have a high semantic similarity to "BornIn" based on the shared entities in their corresponding triples.

For instance, given a triple $(Maryam\_Mirzakhani, bornIn, Tehran)$ the anchor is *Maryam_Mirzakhani Tehran*, a positive sample can be *Albert_Einstein Ulm* constructed from the triple $(Albert\_Einstein, bornIn, Ulm)$ and a negative sample *Paris France* from $(Paris, locatedIn, France)$.

Clustering Based on Entity:

The clusters are based on the head entities, and they can be readily expanded to include the tail entities by generating reverse relations. For a triple (h, r, t), the anchor is either the partial fact PF_{rt} or PF_{hrt} or the combination of both. The positive sample is a partial fact created from a triple with the same head as the anchor, given by $(h, r_i, t_i)|(h, r_i, t_i) \in \mathcal{G}$. To generate more difficult negative samples, we follow the idea that entities with the same type are semantically related to each other. Therefore, we select an entity that is likely to have the same type as the head entity h and create PF_{rt_j} or $PF_{h_jrt_j}$ from the set $(h_j, r, t_j)|(h_j, r, t_i) \in \mathcal{G} \wedge (h, r, t_j) \notin \mathcal{G}$. This approach avoids separating a city from a person, which would be too easy.

To create a cluster for an entity, such as Albert Einstein in the example $(Albert\_Einstein, bornIn, Ulm)$, we generate a positive sample by selecting another fact about Einstein, such as $(Albert\_Einstein, studiedIn, ETH)$, and create the textual sequence $studiedIn\ ETH$ or $Albert\_Einstein\ studiedInm\ ETH$. To generate a negative sample, we select a fact with the same relation as the anchor, in this case, $bornIn$, and ensure that it is not true for Albert Einstein due to multi-valuation. For instance, we select $(Pierre\_Currie, bornIn, Paris)$ and create the negative sample $bornIn\ Paris$ or $Pierre\_Currie\ bornIn\ Paris$.

The evaluation pipeline for the relation prediction task is summarized in Figure 5.1. To begin, a partial fact of the test triple (h, r, t) is generated in the same manner as the training inputs. The fine-tuned model is then used to obtain the vectorized embedding E_{test}. Euclidean distance is used to determine the similarity between E_{test} and the embeddings of the training inputs for evaluation purposes. Let Δ and Δ_e denote the set of embeddings for all training anchors and those corresponding to entity e, respectively.

For triple classification, the distance of E_{test} is calculated with the reference set of its head entity Δ_h. Meanwhile, the K closest points to E_{test} in Δ are used to determine the most plausible relation for relation prediction. To find the nearest points to E_{test} in the corresponding reference sets, we rely on FAISS (Johnson et al., 2017), a library for efficient similarity search that prioritizes speed over precision. Further details regarding the evaluation process for each refinement task will be provided in Section 5.3.

5.3 EXPERIMENTAL RESULTS

Dataset

In this study, we utilize three benchmark datasets that are widely used in the literature: WN11 (Socher et al., 2013), FB13 (Socher et al., 2013), and FB15K (Bordes et al., 2013). These datasets are subsets of real-world KGs, namely Wordnet and Freebase. We adopt the same training/validation/test splits as (Socher et al., 2013) and (Bordes et al., 2013). The test sets of WN11 and FB13 consist of both positive and negative triples, while FB15K only includes correct triples. The table of the datasets' statistics is presented in Table 5.1.

In our experiments, we fine-tuned the pre-trained RoBERTa-Base model for four epochs using a batch size of 64, Adam optimizer with a learning rate of 2e-5, and a margin γ of 5. It is worth noting that we did not use any external information, such as entity descriptions or paths, during the training process, which could potentially improve the results. All experiments were conducted on a single machine with a 2.6GHz processor, 12 cores, 64 GB of RAM, and an Nvidia TITAN V GPU. The source code and the datasets used in our experiments are publicly available[1].

Triple Classification

This task assesses whether a given triplet (h, r, t) is correct or not, *i.e.*, binary classification on a triplet. We evaluate our approach on two benchmark

1 https://github.com/armitakhn/gilbert/

Dataset	#R	#E	#Facts (Train/Valid/Test)		
WN11	11	38,696	112,581	2,609	10,544
FB13	13	75,043	316,232	5,908	23,733
FB15K	1345	14,951	483,142	50,000	59,071

Table 5.1.: Statistics of datasets

datasets: WN11 and FB13. To form clusters for the entities appearing as the head in the graph $\mathcal{G}$, we follow the method outlined in Section 5.2. The partial facts PF_{hrt} are randomly used with a 30% probability. For WN11 dataset, we add the reverse versions of relations (t, r^{-1}, h) to the KG to overcome the cold start problem caused by some test data's heads not appearing as the head to any fact in the training. We create five positive and five negative samples for each anchor in all datasets.

To evaluate a triple (h, r, t), we calculate the distance between the embedding of PF_{hrt} and the embeddings of all facts in the cluster of the head entity h, *i.e.*, the reference set Δ_h. We use three different decision strategies, MEAN, MAX, and MIN, to aggregate the computed distances to the reference set. We tune the threshold σ on the validation data and report the best results obtained by the MIN strategy.

We have not used any external information such as entity descriptions or paths while training the model. The results of triple classification on WN11 and FB13 datasets are presented in Table 5.2. Our approach, GilBERT, achieves the best performance on average and for FB13 dataset and is second best after KG-BERT for the WN11 dataset. We compare our approach's results with the state-of-the-art methods and take their reported results from the cited papers, except for TransE and DistMult, whose results are taken from the papers (Lin et al., 2015) and (Yao et al., 2019), respectively.

Relation Prediction

The objective of the relation prediction task is to predict the relation between two entities given their head and tail, *i.e.*, $(h, ?, t)$. To accomplish this task, clusters are created based on relations. The network is trained to learn

Method	WN11	FB13	Avg.
TransE (Bordes et al., 2013)	75.9	81.5	79.2
TransH (Wang et al., 2014b)	78.8	83.3	87.7
TransR (Lin et al., 2015)	85.9	82.5	83.9
DistMult (Yang et al., 2015)	87.1	86.2	86.7
DOLORES + ConvKB (Wang et al., 2020a)	87.5	89.3	88.4
KG-BERT (Yao et al., 2019)	**93.5**	90.4	91.9
R-MeN (Nguyen et al., 2020)	90.5	88.9	89.7
GilBERT	92.7	**92.5**	**92.6**

Table 5.2.: Triples classification Accuracy results (in %) reported on the WN11, FB13 test sets.

embeddings that yield small distances for partial facts belonging to the same relation and large distances for different ones. We generate five positive and five negative samples for each anchor during experiments.

For the evaluation, the correct relation r is determined given its head and tail. For this purpose, the PF_{ht}'s embedding E_{test} is derived. The k closest points in the embedding space (using all training points in the reference set Δ) to E_{test} are computed. The true relations of the closest points are known from the training data. We propose two different strategies for assigning a relation to our test point of interest. The strategy MIN assigns the relation of the closest point to E_{test}. And the strategy K-MODE chooses the relation that has most frequently appeared among the k-closest points of E_{test}.

Method	MR	Hits@1
TransE (Bordes et al., 2013)	2.5	84.3
TransR (Lin et al., 2015)	2.1	91.6
ProjE (listwise) (Shi and Weninger, 2017)	1.2	95.7
TKRL (Xie et al., 2016b)	1.7	92.8
DKRL(CNN) (Xie et al., 2016a)	2.5	89.0
DKRL (CNN) + TransE (Xie et al., 2016a)	2.0	90.8
KG-BERT (Yao et al., 2019)	1.2	**96.0**
GilBERT	1.3	92.0

Table 5.3.: Relation prediction MR and HITs@1 results on FB15K dataset.

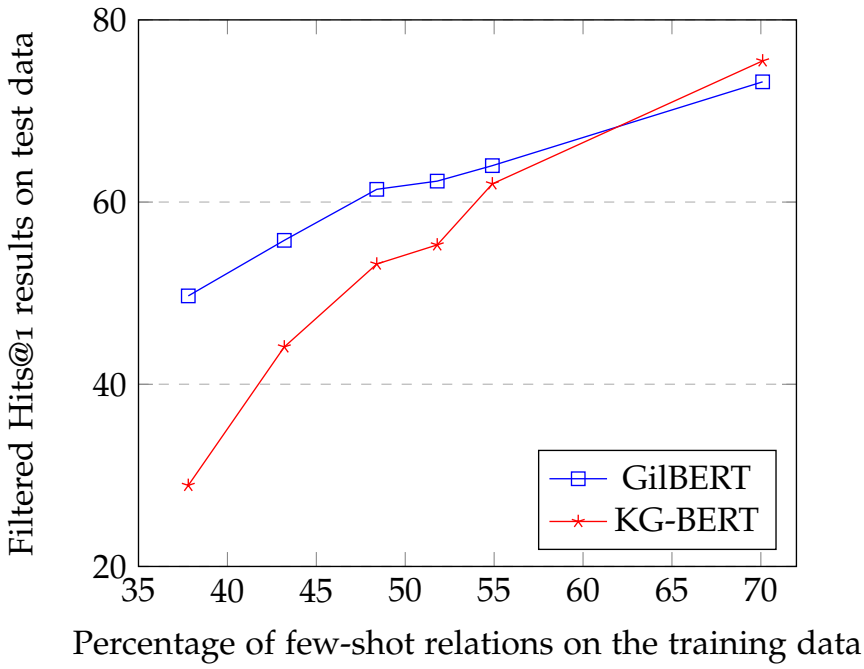

Figure 5.2.: Comparison of KG-BERT and GilBERT hits@1 results on long-tail relations of the training data.

The results of the relation prediction task on the FB15K dataset are reported in Table 5.3. All results are taken from their respective paper, except TransE and TransR whose results are taken from (Xie et al., 2016b). The evaluation metrics used are Hits@1 (proportion of correct relations in top 1 ranking) and Mean rank *MR*. Note that the reported results are after filtering. If a suggested triplet is already known to be true, *i.e.*, if it already exists in the train, validation, or test data of the knowledge graph, it is not correct to rank it among the suggestions. The best results with GilBERT, tuned on the validation data, were obtained using the K-MODE strategy with K being set to 10. The Hits@1 results obtained by our approach are comparable to the state-of-the-art, without integrating any external information, with a very competitive Mean Rank.

In addition, as mentioned in the introduction, GilBERT is more suited to a few-shot learning scenario than KG-BERT, which treats relation prediction as a classification task. To investigate this, we compared the Hits@1 results of the

two approaches on long-tail relations, defined as those with less than 10, 15, 20, 25, and 30 facts in the training data, covering 37.8%, 43.2%, 48.4%, 51.8%, and 54.9% of the relations in the training data, respectively. The Hits@1 results on these proportions of relations in the test data are plotted in Figure 5.2 for KG-BERT and GilBERT. The plot shows that GilBERT is more robust in predicting relations that have been observed only a few times during training

To elaborate more, we look at 664 relations (almost half of the relations) with less than 20 facts, comprising 8.4% of FB15K instances. Our Hits@1 result on the test data involving these properties is 64% (best results obtained with MIN strategy) compared to KG-BERT, which gives a Hits@1 of 53%.

5.4 CONCLUSION

This chapter proposed GilBERT, a new approach that fine-tunes a triplet network of pre-trained transformer-based language models by passing textual sequences created from facts as input.

As discussed in the motivation section of this chapter (Section 5.1), our goal is to modify a network like GilBERT to group referring expressions (REs) of the same entity together in clusters. By doing so, we can potentially use our network for implicit entity linking.

In addition, we plan to apply our approach to other KG refinement tasks, such as link prediction. Moreover, considering the issues of data leakage in widely used benchmark datasets like FB15K, we will explore other benchmark datasets like CoDEx (Safavi and Koutra, 2020) and investigate the impact of adding external information to our training process.

5.4.1 *Conclusion*

In this thesis, we have proposed novel techniques for refining KGs. We summarize our main contributions and highlight their originality.

REGNUM took a step forward in introducing numerical rules with membership or non-membership constraints on intervals. The originality of our approach lies in building upon existing rule mining techniques such as AMIE and AnyBURL, eliminating the need for updating and maintaining optimal intervals while exploring the search space. REGNUM is guided by quality measures of PCA confidence and head coverage to determine which numerical atoms should be added to a parent rule. The numerical predicates integrated into rules serve as features for a classification task that distinguishes correct and incorrect predictions of the parent rule. This approach allows REGNUM to discover relevant intervals for each parent rule, promoting knowledge discovery rather than relying on a pre-processing step that generates intervals for numerical predicates independent of the rules. We conducted experiments on three benchmark datasets; we demonstrated that the quality of the generated set of numerical rules is increased on average compared to that of the parent rules. We also observed improvements in knowledge graph completion using the numerical rules.

We introduced RE-miner that mines referring expressions for a class in a knowledge graph. The referring expressions are subgraph patterns that are valid for one entity in the KG, they are existentially closed, minimal, and diverse (for each entity). The mining process is guided by exploring combinations of maximal non-keys, ensuring that the obtained referring expressions are specific to each entity rather than being instantiations keys, allowing for more entity-specific expressions. For each entity in the class, the approach

first identifies referring expressions involving a single property and then traverses the search space to discover more complex patterns. We demonstrated the effectiveness of mining referring expressions on large knowledge graphs through quantitative and qualitative experimental evaluations on 15 benchmark datasets. These expressions proved valuable in identifying sameAs links between different knowledge graphs, significantly improving recall without compromising precision.

We also introduced GilBERT, a system that employs the fine-tuning of language models such as BERT using a triplet loss. The objective is to create an embedding space where facts about a relation/entity are embedded close to each other. We also explored techniques for obtaining more challenging negative examples to improve the learning process. GilBERT was applied to two downstream tasks: relation prediction and triple classification. The evaluation strategy employed relies on the distance between the embeddings of test data and the embeddings of the training data. The chapter demonstrated the effectiveness of GilBERT in these refinement tasks on 3 benchmark datasets, highlighting its suitability for few-shot settings for relation precision compared to a system that treats this task as multi-class classification.

We structure our future perspectives into short-term and long-term perspectives with regard to the work carried out during this thesis.

5.4.2 *Short-Term Perspectives*

Evaluation In order to fully explore the capabilities and potential limitations of the presented works in this thesis, it is essential to conduct more comprehensive evaluations and experiments.

Regarding REGNUM discussed in Chapter 3, it is important to run REGNUM on more domain-related datasets where numerical data plays a crucial role, such as financial or health data rather than general benchmark datasets. By doing so, we think that REGNUM can outperform other rule-mining techniques that may not effectively capture interesting numerical patterns. Conducting evaluations on such datasets will provide deeper insights into the capabilities of REGNUM. Moreover, in our experiments, we did not consider

parent rules that involved constants (*e.g.*, $livesIn(x, France) \wedge worksIn(x, y) \Rightarrow bornIn(x, y)$) that can be discovered by some existing approaches such as AMIE and AnyBURL. Adding such rules can lead to many more parent rules, yet considering them may have a positive impact on the performance of KG completion task. This is to be verified in future experiments. Furthermore, in the context of KG completion, we explored various aggregation techniques, including democracy, exaggeration, noisy-or, and weighted F-score aggregation. However, it is worth noting that other aggregation methods exist that have demonstrated better results and are better at scaling, such as the approach presented by Betz et al. (2022). This method introduces end-to-end trainable aggregation functions that combine rules into a confidence score to answer queries. In this approach, the rules are encoded with latent feature vectors and a sparse aggregator is learned through supervised learning.

In Chapter 4, we explored a post-processing technique aimed at expanding Referring Expressions (REs) to enhance their usefulness for data linking purposes by reducing IRIs involved in the REs. However, the resulting expanded REs can become quite complex, making tracking and using them for data linking challenging. To address this issue, we plan to study the effect of implementing a heuristic that can replace IRIs (that cannot be matched with high confidence by applying similarity measures) with a subgraph consisting of predicates that have literal values as their range for that node, even if the combination of these predicates is not a key. The resulting subgraph is without IRIs and can be used for data linking without becoming too complex. It should be noted that this subgraph will not necessarily be a referring expression.

To extend the capabilities of GilBERT of Chapter 5, we plan to make the necessary adaptations to enable link prediction as well. To achieve this, we propose clustering all entities (*i.e.*, all entities that appear as the head and tail of facts in the knowledge graph (KG)), similar to the approach used for triple classification tasks. Based on the specific prediction task, whether it is $(h, r, ?)$ or $(?, r, t)$, we create partial facts denoted as P_{hr} and P_{rt} and generate their respective embeddings. We examine the nearest embeddings during the evaluation and determine the associated entities to them. To obtain a score for each candidate entity, we can calculate the sum of distances to test partial fact's embedding, which serves as a sorting mechanism to select the best candidate. Furthermore, to address the concerns regarding data leakage in commonly

used benchmark datasets such as FB15K, we will broaden our exploration to include alternative benchmark datasets like CoDEx Safavi and Koutra (2020). We will specifically examine the effects of incorporating external information, such as entity type and textual data like entity descriptions, into our training process.

Alternative strategies for Generation of Intervals for Numerical rules. In Chapter 3, the numerical rules discussed involve interval constraints. These constraints are derived by constructing a decision tree that effectively differentiates between correct and incorrect predictions (under PCA) made by the parent rules. Quality measures such as head coverage and confidence guide the decision tree construction process. However, there exist alternative strategies for obtaining interval constraints as elaborated in Chapter 3.1. One such alternative is the use of RIPPER or its extensions. These rule induction algorithms can be employed to generate rules with interval constraints based on a given dataset that can be directly used to enrich the parent rule. Additionally, we will explore the application of Quantitative Association Rule Mining techniques, which offer another way to obtain interval constraints by relying on genetic algorithms or discretization techniques. These strategies are completely different from DT as we have used in REGNUM. Hence completely different intervals can be obtained, and it's interesting to see how their resulting rules differ and how they impact the KG completion.

Scalability and Pruning strategies In order to enhance the efficiency and scalability of REGNUM (Chapter 3) and RE-miner (Chapter 4), several potential improvements can be implemented. Firstly, integrating an in-memory database, as done in AMIE3, to store the data can significantly improve performance. Using an in-memory database, facts can be indexed based on subject, object, relation, and pairs of relation/subject and relation/object. This indexing allows for faster data retrieval and more efficient query execution. Additionally, more pruning strategies can be employed to reduce the search space and improve run-time. In the case of REGNUM, one strategy is to calculate the correlation between numerical predicates and prune combinations of predicates that are highly correlated for the same variable. This pruning technique helps to eliminate redundant and uninteresting combinations of predicates and even potentially yields more interesting numerical rules.

Implicit Linking In chapters 4 and 5, we have discussed the potential application of referring expressions (REs) for implicit linking. We have proposed an approach demonstrating promising results in relation prediction and triple classification on a knowledge graph, namely GilBERT. This approach involves creating textual sequences from facts in the knowledge graph and clustering them based on relations or entities. Additionally, RE-miner allows the generation of multiple REs for each entity, and these REs can be easily converted into textual sequences. We think that GilBERT can be rather directly used for the implicit entity linking using REs. More precisely, we can cluster entities based on their referring expressions by leveraging the proposed triplet network architecture of GilBERT. In this process, each RE serves as an anchor, with another RE for the same entity as a positive example and a RE for a different entity as a negative example. Once the chosen LM is fine-tuned, we can embed a given text (*e.g.*, "The famous scientist who was born in Ulm and made significant contributions to the theory of relativity") using the trained network. By examining the neighboring embeddings (corresponding to some referring expressions) and the entities they refer to, we can, in this case, Albert Einstein.

5.4.3 *Long-term Perspectives*

Expressivity. There are various avenues to explore in order to enhance the expressiveness of the numerical rules discussed in Chapter 3. As suggested by Galárraga and Suchanek (2014), an interesting future work would be considering numerical constraints on the head atom instead of only on the atoms in the body. For example, a rule like $isLocatedIn(x, Europe) \wedge hasGDP(x, y) \Rightarrow y \in [5K, 4M]$ can be considered. However, it is not straightforward to adapt REGNUM to discover such rules. Nevertheless, if parent rules concluding on numerical predicates are available, it is possible to discover rules where the head atom is enriched. To this end, in REGNUM, the features to be considered for the decision tree for classification are the numerical predicate of the head atom.The challenge lies not in finding intervals for the numerical predicate of the head atom but rather in identifying parent rules that conclude on a numerical predicate. Another possible way to enhance the expressivity of

rules is to use aggregates such as sum or average calculations in the body or head atoms. However, strategies are needed to explore the search space using such aggregations and to use such rules for data completion, when the numerical predicate appears in the head of the rule.

Evolving KGs and Integrating KGs. Another potential challenge arises when knowledge graphs evolve. As the KG evolves, the referring expressions may no longer be minimal or valid. In future, it would be interesting to deploy strategies for updating the referring expressions as the KG evolves. For instance, if a triple is removed from the KG, the obtained REs remain referring expressions, but they might not be minimal anymore. On the other hand, if triples are added to the KG, some referring expressions may become invalid. Studying the effects of changes in the KG bring to referring expressions and what are the most efficient ways of updating them remains as a future work.

Furthermore, investigating the updating and fusion of referring expressions (REs) when merging two knowledge graphs is an interesting area of study. Additionally, considering the linking step, when the UNA is satisfied and a single RE of an entity links to multiple entities in a target KG, it suggests that there may be numerous missing pieces of information in the source KG. Therefore, proposing approaches that simultaneously complete the KG and perform linking represents another envisioned future direction of research.

Reduction of Rule Redundancies. In Chapter 3, REGNUM can generate multiple numerical rules based on a single parent rule. These numerical rules can vary regarding numerical properties and constraints related to membership or non-membership. We implemented a simple strategy to limit rules with a semantic overlap. Once the decision tree is constructed, we select the most general rules (with the highest PCA body size) for membership or non-membership predicates. However, it is still possible for multiple rules with different constraints and intervals to have a semantic overlap. Such redundant rules are not ideal for two reasons. Firstly, if the rules are intended for use by an expert, they would likely prefer fewer rules with high quality and minimal overlap between the instances they cover. Secondly, various strategies were introduced for selecting an entity for link prediction among candidate entities when using the rules. Some of these strategies consider all the rules pointing to an entity (*e.g.*, democracy), which may result in counting

the same rule multiple times, thereby reducing the robustness of the approach. In our experiments, we used the max-aggregation that is immune to this setting. However, it is important to acknowledge that other strategies may treat redundant rules as independent, leading to an overestimation of the final score.

Global Optimization of Numerical Rule. As mentioned in Chapter 3, our current approach involves finding numerical rules based on parent rules mined from the KG using an existing rule mining technique. This is mainly because calculating constraints while the rule mining technique explores the search space (as it generalizes or refines the rule) results in having to re-calculate the interval at each step, making the approach time-consuming over large graphs. Therefore, REGNUM may lose some interesting numerical rules, and the obtained numerical rules may not be optimal wrt. quality measures. As future work, it'd be very interesting to develop a system that optimizes both rule structure and interval constraints simultaneously and compare this approach with REGNUM to evaluate their respective behaviors. While REGNUM is expected to exhibit higher efficiency, it may occasionally produce sub-optimal solutions.

5.4.4 *Final Note.*

Knowledge graphs are able to capture real-world knowledge in a machine-readable format. They serve as a solid foundation for capturing domain-specific knowledge and facilitating complex reasoning tasks and it's important to refine them to have more accurate and complete results. By incorporating knowledge into data, a multitude of advantages can be gained. Nevertheless, the governance of humans and adopting human-centered AI seems crucial in ensuring explainability and minimizing the potential for risks and incorrect information. In this context, it raises the question: Are we in an era of "data and knowledge" or an era of "data, knowledge, and people"? Data provides scalability, knowledge provides semantic understanding, and humans provide trust. The possibility of large language models replacing knowledge graphs remains an open question. Are knowledge graphs the beacon of hope for the future, offering advantages that LLMs cannot replicate? or will KGs become

a relic of the past? The answer to these questions and many more has yet to unfold and shape the future of data, knowledge, and human interactions.

RÉSUMÉ EN FRANÇAIS

Cette thèse vise à améliorer l'expressivité des approches existantes d'extraction de règles dans les graphes de connaissances afin de découvrir des règles plus efficaces pour l'enrichissement des graphe de connaissances et le liage de données. Nous nous intéressons également à l'intégration d'informations des graphes de connaissances dans des modèles linguistiques. Le plan de la thèse et les contributions sont les suivants.

Fondamentaux. Le chapitre 1 présente les concepts essentiels liés aux graphes de connaissances et à leur représentation. Il présente brièvement les graphes de connaissances à travers les notions d'alignement d'ontologies et de liage de données. Le chapitre traite également de la fouille de règles dans les graphes de connaissances, de la définition des règles en logique du premier ordre et de l'introduction de certaines mesures de qualité pour les évaluer. En outre, il explore l'utilisation de représentations vectorielles pour les entités et les relations, les plongements (embeddings) de graphes de connaissances, en soulignant les caractéristiques et les procédures impliquées dans l'obtention de telles représentations.

État de l'art. Dans le chapitre 2, nous passons en revue les différentes approches pour le raffinement des graphes de connaissances, notamment la complétion de graphe de connaissances et le liage de données. Nous discutons des approches symboliques basées sur des règles ainsi que des approches subsymboliques basées sur des embeddings de graphes de connaissances. Nous présentons les techniques d'extraction de règles les plus récentes, en examinant leurs biais linguistiques, leurs stratégies de réduction de l'espace de recherche et leurs caractéristiques propres. Ensuite, nous étudions les techniques d'embedding de graphes de connaissances subsymboliques en tant que méthode fréquemment utilisée pour compléter les graphes de connaissances. Nous classons ces techniques en nous basant sur leurs fonctions de score et discutons des caractéristiques de chacune des méthodes. En outre, nous présentons le liaige de données en tant que tâche de raffinement, en

particulier les approches basées sur les clés de la littérature. De plus, nous abordons brièvement les méthodes basées sur de embeddings de graphes de connaissances pour le liaige des données.

Enrichissement de règles de Horn par des predicats numériques. Après une revue approfondie de l'état de l'art sur l'extraction de règles dans le chapitre 2, un manque évident se dégage dans l'incorporation efficace de prédicats numériques tels que l'âge ou la population dans les règles extraites par les systèmes existants. Ce manque est dû à l'ampleur de l'espace de recherche associé aux prédicats numériques, car ils prennent de nombreuses valeurs distinctes. À cette fin nous proposons une approche novatrice appelée REGNUM - Règles numériques pour le raffinement des graphes de connaissances dans le chapitre 3. REGNUM permet d'incorporer des prédicats numériques dans le processus d'extraction de règles. Nous décrivons en détail le fonctionnement de REGNUM, en mettant l'accent sur l'utilisation de mesures de qualité pour guider le processus d'enrichissement des règles. REGNUM s'appuie sur les règles générées par un système existant d'extraction de règles et les enrichit en incorporant des prédicats numériques qui contraignent les valeurs à des intervalles spécifiques. Le processus est guidé par des mesures de qualité qui évaluent la confiance et la signification des règles. REGNUM est la première approche à se concentrer spécifiquement sur les règles numériques dans les graphes de connaissances, révélant des motifs liés à l'appartenance ou à la non-appartenance à un intervalle.

Découverte d'expressions référentielles dans les graphes de connaissances. Le chapitre 4 présente RE-miner, une approche pour la découverte d'expressions référentielles dans les graphes de connaissances. Dans un graphe de connaissances, une expression référentielle est une formule logique qui permet d'identifier de façon unique une entité. De telles expressions peuvent être exploitées pour répondre à des requêtes, lier des données, annoter des ressources textuelles, ou encore anonymiser des données. Il peut potentiellement exister de nombreuses expressions logiques pour identifier de manière unique une entité. Nous proposons une approche permettant de découvrir efficacement certaines expressions référentielles en nous concentrant sur celles qui ne peuvent être trouvées en instanciant des clés. Les expérimentations montrent que cette approche passe à l'échelle de jeux de données de plusieurs millions de triplets RDF et que ces expressions perme-

ttent de lier efficacement les instances de classes de différents graphes de connaissances.

Raffinement de graphes de connaissances au travers de grands modèles de langage. Chapitre 5, présente une nouvelle approche qui consiste à affiner (fine-tuning) les modèles de langages en utilisant les informations du graphe de connaissances pour générer des clusters d'entités, de relations et d'expressions de référence souhaitées. Nous exploitons l'espace d'embedding et mesurons la proximité entre les entités/relations intégrées afin d'effectuer des tâches telles que la prédiction des relations et la complétion du graphe de connaissances. Cette approche peut donner lieu à meilleures performances pour des tâches basées sur les graphes de connaissances, telles que l'établissement de liens implicites entre les entités.

Conclusion. Le chapitre 5.4 résume la thèse et développe les perspectives futures.

A

LIST OF PUBLICATIONS

International Conferences

Khajeh Nassiri, A., Pernelle, N., Saïs, F. (2023). **REGNUM: Generating Logical Rules with Numerical Predicates in Knowledge Graphs**. In: , et al. The Semantic Web. ESWC 2023. Lecture Notes in Computer Science, vol 13870. Springer, Cham.

A. Khajeh Nassiri, N. Pernelle, F. Saïs, and G. Quercini. **Generating referring expressions from rdf knowledge graphs for data linking.** In J. Z. Pan, E. Jimenez-Ruiz, J. McCusker, D. Faria, M. Dragoni, A. Dimou, R. Troncy, and S. Hertling, editors, The Semantic Web, pages 139–155, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-33455-9.

International Workshops

Armita Khajeh Nassiri, Nathalie Pernelle, Fatiha Saïs, Gianluca Quercini **Reminer for data linking results for OAEI 2020.** In P. Shvaiko, J. Euzenat, E. Jiménez- Ruiz, O. Hassanzadeh, and C. Trojahn, editors, Proceedings of the 15th International Workshop on Ontology Matching co-located with the 19th In- ternational Semantic Web Conference (ISWC 2020), Virtual conference, November 2, 2020, volume 2788 of CEUR Workshop Proceedings, pages 211–215.

Armita Khajeh Nassiri, Nathalie Pernelle, Fatiha Saïs, Gianluca Quercini **Knowledge Graph Refinement based on Triplet BERT-Networks.** (2022) 2020

National Conferences

Armita Khajeh Nassiri, Nathalie Pernelle, Fatiha Saïs, **Enrichissement de règles de horn par des predicats numériques.**In C. Faron and S. Loudcher, editors, Extraction et Gestion des Connaissances, EGC 2023, Lyon, France, 16 - 20 janvier 2023, volume E-39 of RNTI, pages 425–432. Editions RNTI, 2023a.

BIBLIOGRAPHY

R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, SIGMOD '93, page 207–216, New York, NY, USA, 1993. Association for Computing Machinery. ISBN 0897915925. doi: 10.1145/170035.170072. URL https://doi.org/10.1145/170035.170072.

N. Ahmadi, J. Lee, P. Papotti, and M. Saeed. Explainable fact checking with probabilistic answer set programming. *CoRR*, abs/1906.09198, 2019.

M. Al-Bakri, M. Atencia, S. Lalande, and M. Rousset. Inferring same-as facts from linked data: An iterative import-by-query approach. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Texas, USA*, pages 9–15. AAAI Press, 2015.

A. Algergawy, D. Faria, A. Ferrara, I. Fundulaki, I. Harrow, S. Hertling, E. Jiménez-Ruiz, N. Karam, A. Khiat, P. Lambrix, H. Li, S. Montanelli, H. Paulheim, C. Pesquita, T. Saveta, P. Shvaiko, A. Splendiani, E. Thiéblin, C. Trojahn, and L. Zhou. Results of the ontology alignment evaluation initiative 2019. 10 2019.

C. Areces, A. Koller, and K. Striegnitz. Referring expressions as formulas of description logic. In *fifth international natural language generation conference*, pages 42–49. ACL, 2008.

M. Atencia, M. Chein, M. Croitoru, J. David, M. Leclère, N. Pernelle, F. Saïs, F. Scharffe, and D. Symeonidou. Defining Key Semantics for the RDF Datasets: Experiments and Evaluations. In *ICCS: International Conference on Conceptual Structures*, Graph-Based Representation and Reasoning, pages 65–78, Romania, 2014a. Springer.

M. Atencia, J. David, and J. Euzenat. Data interlinking through robust linkkey extraction. In *Proceedings of the Twenty-First European Conference on Artificial Intelligence*, ECAI'14, page 15–20, NLD, 2014b. IOS Press. ISBN 9781614994183.

M. Atencia, J. David, J. Euzenat, A. Napoli, and J. Vizzini. Link key candidate extraction with relational concept analysis. *Discrete Applied Mathematics*, 03 2019.

M. Atencia, J. David, J. Euzenat, A. Napoli, and J. Vizzini. Link key candidate extraction with relational concept analysis. *Discrete Applied Mathematics*, 273:2–20, 2020. ISSN 0166-218X. doi: https://doi.org/10.1016/j.dam.2019.02.012. URL https://www.sciencedirect.com/science/article/pii/S0166218X19300952. Advances in Formal Concept Analysis: Traces of CLA 2016.

M. Atencia, J. David, and J. Euzenat. On the relation between keys and link keys for data interlinking. *Semantic Web – Interoperability, Usability, Applicability*, 12(4):547–567, 2021. doi: 10.3233/SW-200414. URL https://hal.science/hal-03426150.

Y. Aumann and Y. Lindell. A statistical theory for quantitative association rules. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, page 261–270, New York, NY, USA, 1999. Association for Computing Machinery. ISBN 1581131437. doi: 10.1145/312129.312243. URL https://doi.org/10.1145/312129.312243.

D. Aumueller, H.-H. Do, S. Massmann, and E. Rahm. Schema and ontology matching with coma++. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, page 906–908. Association for Computing Machinery, 2005. ISBN 1595930604.

I. Balažević, C. Allen, and T. M. Hospedales. Hypernetwork knowledge graph embeddings. In I. V. Tetko, V. Kůrková, P. Karpov, and F. Theis, editors, *Artificial Neural Networks and Machine Learning – ICANN 2019: Workshop and Special Sessions*, pages 553–565, Cham, 2019. Springer International Publishing. ISBN 978-3-030-30493-5.

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 amp; Knowledge Management*, CIKM '20, page 45–54, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450368599. doi: 10.1145/3340531.3412023. URL https://doi.org/10.1145/3340531.3412023.

W. Beek, J. Raad, J. Wielemaker, and F. van Harmelen. sameas.cc: The closure of 500m owl:sameas statements. In A. Gangemi, R. Navigli, M.-E. Vidal, P. Hitzler, R. Troncy, L. Hollink, A. Tordai, and M. Alam, editors, *The Semantic Web*, pages 65–80, Cham, 2018. Springer International Publishing. ISBN 978-3-319-93417-4.

P. Betz, C. Meilicke, and H. Stuckenschmidt. Supervised knowledge aggregation for knowledge graph completion. In *European Semantic Web Conference*, pages 74–92. Springer, 2022.

C. Bizer, M.-E. Vidal, and H. Skaf-Molli. *Linked Open Data*, pages 2096–2101. Springer New York, New York, NY, 2018. ISBN 978-1-4614-8265-9. doi: 10.1007/978-1-4614-8265-9_80603. URL https://doi.org/10.1007/978-1-4614-8265-9_80603.

A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper/2013/file/1cecc7a77928ca8133fa24680a88d2f9-Paper.pdf.

L. Bühmann, J. Lehmann, and P. Westphal. Dl-learner—a framework for inductive learning on the semantic web. *Journal of Web Semantics*, 39:15–24, 2016.

Z. Cao, Q. Xu, Z. Yang, X. Cao, and Q. Huang. Dual quaternion knowledge graph embeddings. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):6894–6902, May 2021. doi: 10.1609/aaai.v35i8.16850. URL https://ojs.aaai.org/index.php/AAAI/article/view/16850.

Z. Cao, Q. Xu, Z. Yang, Y. He, X. Cao, and Q. Huang. Otkge: Multi-modal knowledge graph embeddings via optimal transport. In S. Koyejo,

S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 39090–39102. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/ffdb280e7c7b4c4af30e04daf5a84b98-Paper-Conference.pdf.

Y. Chen, S. Goldberg, D. Z. Wang, and S. S. Johri. Ontological pathfinding. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, page 835–846, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450335317. doi: 10.1145/2882903.2882954. URL https://doi.org/10.1145/2882903.2882954.

Y. Chen, J. Liu, Z. Zhang, S. Wen, and W. Xiong. Möbiuse: Knowledge graph embedding on möbius ring. *Know.-Based Syst.*, 227(C), sep 2021. ISSN 0950-7051. doi: 10.1016/j.knosys.2021.107181. URL https://doi.org/10.1016/j.knosys.2021.107181.

P. Cimiano and H. Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semant. Web*, 8(3):489–508, jan 2017. ISSN 1570-0844. doi: 10.3233/SW-160218. URL https://doi.org/10.3233/SW-160218.

W. W. Cohen. Fast Effective Rule Induction. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.

W. W. Cohen. Tensorlog: A differentiable deductive database, 2016.

M. Croitoru and K. Van Deemter. A conceptual graph approach for the generation of referring expressions. In *IJCAI*, pages 2456–2461, 2007.

Y. Dai, S. Wang, N. N. Xiong, and W. Guo. A survey on knowledge graph embedding: Approaches, applications and benchmarks. *Electronics*, 9(5), 2020. ISSN 2079-9292. doi: 10.3390/electronics9050750. URL https://www.mdpi.com/2079-9292/9/5/750.

R. Dale. Cooking up referring expressions. In *27th Annual Meeting of the association for Computational Linguistics*, pages 68–75, 1989.

R. Dale. *Generating referring expressions: Constructing descriptions in a domain of objects and processes.* The MIT Press, 1992.

R. Dale and E. Reiter. Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive science*, 19(2):233–263, 1995.

C. d'Amato, N. F. Quatraro, and N. Fanizzi. Injecting background knowledge into embedding models for predictive tasks on knowledge graphs. In *The Semantic Web - 18th International Conference, ESWC 2021, Virtual Event, June 6-10, 2021, Proceedings*, volume 12731 of *Lecture Notes in Computer Science*, pages 441–457. Springer, 2021. doi: 10.1007/978-3-030-77385-4_26. URL https://doi.org/10.1007/978-3-030-77385-4_26.

K. v. Deemter. Generating referring expressions: Boolean extensions of the incremental algorithm. *Computational Linguistics*, 28(1):37–52, 2002.

L. Dehaspe and H. Toironen. *Discovery of Relational Association Rules*, page 189–208. Springer-Verlag, Berlin, Heidelberg, 2001. ISBN 3540422897.

C. Demir and A.-C. N. Ngomo. Convolutional complex knowledge graph embeddings. In R. Verborgh, K. Hose, H. Paulheim, P.-A. Champin, M. Maleshkova, O. Corcho, P. Ristoski, and M. Alam, editors, *The Semantic Web*, pages 409–424, Cham, 2021. Springer International Publishing. ISBN 978-3-030-77385-4.

T. Dettmers, M. Pasquale, S. Pontus, and S. Riedel. Convolutional 2d knowledge graph embeddings. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, pages 1811–1818, February 2018. URL https://arxiv.org/abs/1707.01476.

A. Doan, P. Domingos, and A. Levy. Learning source description for data integration. In *Proceedings of the Third International Workshop on the Web and Databases, in conjunction with ACM PODS/SIGMOD 2000.*, pages 81–86, 01 2000.

T. Ebisu and R. Ichise. Toruse: Knowledge graph embedding on a lie group. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and*

Eighth AAAI Symposium on Educational Advances in Artificial Intelligence, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018. ISBN 978-1-57735-800-8.

J. Euzenat and P. Shvaiko. *Ontology Matching, Second Edition*. Springer, 2013. ISBN 978-3-642-38720-3.

W. Fan, Z. Fan, C. Tian, and X. L. Dong. Keys for graphs. *PVLDB*, 8(12): 1590–1601, 2015.

D. Faria, C. Pesquita, T. Tervo, F. M. Couto, and I. F. Cruz. AML and AMLC results for OAEI 2019. In P. Shvaiko, J. Euzenat, E. Jiménez-Ruiz, O. Hassanzadeh, and C. Trojahn, editors, *Proceedings of the 14th International Workshop on Ontology Matching co-located with the 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 26, 2019*, volume 2536 of *CEUR Workshop Proceedings*, pages 101–106. CEUR-WS.org, 2019. URL http://ceur-ws.org/Vol-2536/oaei19_paper3.pdf.

U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *IJCAI*, 1993.

A. Ferrara, A. Nikolov, and F. Scharffe. Data linking for the semantic web. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 7(3): 46–76, 2011.

M. H. Gad-Elrab, D. Stepanova, J. Urbani, and G. Weikum. Exception-enriched rule learning from knowledge graphs. In *International Workshop on the Semantic Web*, 2016.

L. Galárraga and F. M. Suchanek. Towards a Numerical Rule Mining Language. In *AKBC workshop*, Montreal, Canada, Dec. 2014. URL https://hal-imt.archives-ouvertes.fr/hal-01699877.

L. Galárraga, C. Teflioudi, K. Hose, and F. M. Suchanek. Amie: association rule mining under incomplete evidence in ontological knowledge bases. *Proceedings of the 22nd international conference on World Wide Web*, 2013.

L. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek. Fast Rule Mining in Ontological Knowledge Bases with AMIE+. *The VLDB Journal*, 2015. URL https://hal-imt.archives-ouvertes.fr/hal-01699866.

L. Galárraga, S. Razniewski, A. Amarilli, and F. M. Suchanek. Predicting completeness in knowledge bases. In M. de Rijke, M. Shokouhi, A. Tomkins, and M. Zhang, editors, *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*, pages 375–383. ACM, 2017.

L. Galárraga, J. Delaunay, and J. Dessalles. REMI: mining intuitive referring expressions on knowledge bases. In *Proceedings of the 23nd International Conference on Extending Database Technology, EDBT 2020, Denmark*, pages 387–390. OpenProceedings.org, 2020.

L. A. Galárraga, N. Preda, and F. M. Suchanek. Mining rules to align knowledge bases. In *Proceedings of the 2013 workshop on Automated knowledge base construction, AKBC@CIKM 13, San Francisco, California, USA, October 27-28, 2013*, pages 43–48. ACM, 2013.

L. A. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek. Amie: Association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, page 413–422, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450320351.

S. García, J. Luengo, J. A. Sáez, V. López, and F. Herrera. A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning. *IEEE Transactions on Knowledge and Data Engineering*, 25(4):734–750, 2013. doi: 10.1109/TKDE.2012.35.

A. García-Durán and M. Niepert. Kblrn: End-to-end learning of knowledge base representations with latent, relational, and numerical features. In *Proc. of the 34th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018.

C. Gardent. Generating minimal definite descriptions. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 96–103. ACL, 2002.

G. A. Gesese, M. Alam, and H. Sack. Literallywikidata - a benchmark for knowledge graph completion using literals. In *The Semantic Web – ISWC 2021*, pages 511–527, Cham, 2021. Springer International Publishing.

T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing? *Int. J. Hum. Comput. Stud.*, 43:907–928, 1995.

D. Hernández, A. Hogan, and M. Krötzsch. Reifying rdf: What works well with wikidata? In *SSWS@ISWC*, 2015.

S. Hertling, J. Portisch, and H. Paulheim. MELT - matching evaluation toolkit. In *Semantic Systems. The Power of AI and Knowledge Graphs - 15th International Conference, SEMANTiCS 2019, Karlsruhe, Germany, September 9-12, 2019, Proceedings*, pages 231–245, 2019.

S. Hertling, J. Portisch, and H. Paulheim. Kermit – a transformer-based approach for knowledge graph matching, 2022.

A. Hogan, E. Blomqvist, M. Cochez, C. D'amato, G. D. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, A.-C. N. Ngomo, A. Polleres, S. M. Rashid, A. Rula, L. Schmelzeisen, J. Sequeda, S. Staab, and A. Zimmermann. Knowledge graphs. *ACM Comput. Surv.*, 54(4), jul 2021. ISSN 0360-0300. doi: 10.1145/3447772. URL https://doi.org/10.1145/3447772.

H. Horacek. On referring to sets of objects naturally. In *International Conference on Natural Language Generation*, pages 70–79. Springer, 2004.

J. Huber, T. Sztyler, J. Noessner, and C. Meilicke. Codi: Combinatorial optimization for data integration - results for oaei 2011. In *Proceedings of the 6th International Conference on Ontology Matching - Volume 814*, page 134–141. CEUR-WS.org, 2011.

N. Hubert, P. Monnin, A. Brun, and D. Monticolo. New strategies for learning knowledge graph embeddings: The recommendation case. In O. Corcho, L. Hollink, O. Kutz, N. Troquard, and F. J. Ekaputra, editors, *Knowledge Engineering and Knowledge Management*, pages 66–80, Cham, 2022. Springer International Publishing. ISBN 978-3-031-17105-5.

J. Hühn and E. Hüllermeier. FURIA: an algorithm for unordered fuzzy rule induction. *Data Mining and Knowledge Discovery*, 19(3):293–319, Dec. 2009. ISSN 1573-756X. doi: 10.1007/s10618-009-0131-8. URL https://doi.org/10.1007/s10618-009-0131-8.

I. F. Jaramillo, J. Garzás, and A. Redchuk. Numerical association rule mining from a defined schema using the vmo algorithm. *Applied Sciences*, 11(13), 2021. doi: 10.3390/app11136154.

G. Ji, S. He, L. Xu, K. Liu, and J. Zhao. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 687–696, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1067. URL https://aclanthology.org/P15-1067.

S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. Yu. A survey on knowledge graphs: representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, Apr. 2021. ISSN 2162-237X. doi: 10.1109/TNNLS.2021.3070843.

E. Jiménez-Ruiz. Logmap family participation in the OAEI 2019. In P. Shvaiko, J. Euzenat, E. Jiménez-Ruiz, O. Hassanzadeh, and C. Trojahn, editors, *Proceedings of the 14th International Workshop on Ontology Matching co-located with the 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 26, 2019*, volume 2536 of *CEUR Workshop Proceedings*, pages 160–163. CEUR-WS.org, 2019. URL http://ceur-ws.org/Vol-2536/oaei19_paper11.pdf.

J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.

R. Kerber. Chimerge: Discretization of numeric attributes. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, AAAI'92, page 123–128. AAAI Press, 1992. ISBN 0262510634.

A. Khajeh Nassiri, N. Pernelle, F. Saïs, and G. Quercini. Generating referring expressions from rdf knowledge graphs for data linking. In J. Z. Pan, V. Tamma, C. d'Amato, K. Janowicz, B. Fu, A. Polleres, O. Seneviratne, and L. Kagal, editors, *The Semantic Web – ISWC 2020*, pages 311–329, Cham, 2020. Springer International Publishing. ISBN 978-3-030-62419-4.

A. Khajeh Nassiri, N. Pernelle, and F. Saïs. Regnum: Generating logical rules with numerical predicates in knowledge graphs. In C. Pesquita,

E. Jimenez-Ruiz, J. McCusker, D. Faria, M. Dragoni, A. Dimou, R. Troncy, and S. Hertling, editors, *The Semantic Web*, pages 139–155, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-33455-9.

R. Kohavi and M. Sahami. Error-based and entropy-based discretization of continuous features. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 114–119. AAAI Press, 1996.

B. Kotnis and V. Nastase. Analysis of the impact of negative sampling on link prediction in knowledge graphs, 2018.

E. Krahmer and K. Van Deemter. Computational generation of referring expressions: A survey. *Computational Linguistics*, 38(1):173–218, 2012.

E. Krahmer, S. v. Erk, and A. Verleg. Graph-based generation of referring expressions. *Computational Linguistics*, 29(1):53–72, 2003.

J. Lajus, L. Galárraga, and F. Suchanek. Fast and exact rule mining with amie 3. In A. Harth, S. Kirrane, A.-C. Ngonga Ngomo, H. Paulheim, A. Rula, A. L. Gentile, P. Haase, and M. Cochez, editors, *The Semantic Web*, pages 36–52, Cham, 2020. Springer International Publishing.

Y. Lin, Z. Liu, H. Luan, M. Sun, S. Rao, and S. Liu. Modeling relation paths for representation learning of knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 705–714, Lisbon, Portugal, Sept. 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1082. URL https://www.aclweb.org/anthology/D15-1082.

H. Liu, Y. Wu, and Y. Yang. Analogical inference for multi-relational embeddings. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 2168–2178. JMLR.org, 2017.

M. Loster, D. Mottin, P. Papotti, J. Ehmüller, B. Feldmann, and F. Naumann. Few-shot knowledge validation using rules. In *Proceedings of the Web Conference 2021*, WWW '21, page 3314–3324, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383127. doi: 10.1145/3442381.3450040. URL https://doi.org/10.1145/3442381.3450040.

H. Ma, M. Alipourlangouri, Y. Wu, F. Chiang, and J. Pi. Ontology-based entity matching in attributed graphs. *Proc. VLDB Endow.*, 12(10):1195–1207, 2019.

C. Meilicke, M. W. Chekol, D. Ruffinelli, and H. Stuckenschmidt. Anytime bottom-up rule learning for knowledge graph completion. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 3137–3143, 7 2019.

C. Meilicke, M. W. Chekol, M. Fink, and H. Stuckenschmidt. Reinforced anytime bottom up rule learning for knowledge graph completion. *arXiv preprint arXiv:2004.04412*, 2020.

A. Melo, M. Theobald, and J. Völker. Correlation-based refinement of rules with numerical attributes. In W. Eberle, editor, *Proceedings of the twenty-seventh International Conference of the Florida Artificial Intelligence Research Society (FLAIRS) : May 21 - 23, 2014 Pensacola Beach, Florida, USA*, pages 345–350. AAAI Press, 2014. URL https://madoc.bib.uni-mannheim.de/35956/.

B. Minaei-Bidgoli, R. Barmaki, and M. Nasiri. Mining numerical association rules via multi-objective genetic algorithms. *Information Sciences*, 233:15–24, 2013. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2013.01.028. URL https://www.sciencedirect.com/science/article/pii/S0020025513001072.

A. K. Nassiri, N. Pernelle, F. Saïs, and G. Quercini. Re-miner for data linking results for OAEI 2020. In P. Shvaiko, J. Euzenat, E. Jiménez-Ruiz, O. Hassanzadeh, and C. Trojahn, editors, *Proceedings of the 15th International Workshop on Ontology Matching co-located with the 19th International Semantic Web Conference (ISWC 2020), Virtual conference (originally planned to be in Athens, Greece), November 2, 2020*, volume 2788 of *CEUR Workshop Proceedings*, pages 211–215. CEUR-WS.org, 2020. URL https://ceur-ws.org/Vol-2788/oaei20_paper12.pdf.

A. K. Nassiri, N. Pernelle, F. Sais, and G. Quercini. Knowledge graph refinement based on triplet bert-networks, 2022.

A. K. Nassiri, N. Pernelle, and F. Saïs. Enrichissement de règles de horn par des predicats numériques. In C. Faron and S. Loudcher, editors, *Extraction*

et Gestion des Connaissances, EGC 2023, Lyon, France, 16 - 20 janvier 2023, volume E-39 of *RNTI*, pages 425–432. Editions RNTI, 2023a. URL http://editions-rnti.fr/?inprocid=1002846.

A. K. Nassiri, N. Pernelle, and F. Saïs. Enrichissement de règles de horn par des predicats numériques. In C. Faron and S. Loudcher, editors, *Extraction et Gestion des Connaissances, EGC 2023, Lyon, France, 16 - 20 janvier 2023*, volume E-39 of *RNTI*, pages 425–432. Editions RNTI, 2023b. URL http://editions-rnti.fr/?inprocid=1002846.

G. Navas-Palencia. Optimal binning: mathematical programming formulation. abs/2001.08025, 2020. URL http://arxiv.org/abs/2001.08025.

M. Nayyeri, C. Xu, Y. Yaghoobzadeh, H. S. Yazdi, and J. Lehmann. Toward understanding the effect of loss function on then performance of knowledge graph embedding, 2019.

A.-C. Ngonga Ngomo and S. Auer. Limes – a time-efficient approach for large-scale link discovery on the web of data. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain*, pages 2312–2317, 01 2011.

A.-C. Ngonga Ngomo and K. Lyko. Eagle: Efficient active learning of link specifications using genetic programming. In E. Simperl, P. Cimiano, A. Polleres, O. Corcho, and V. Presutti, editors, *The Semantic Web: Research and Applications*, pages 149–163, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-30284-8.

A.-C. Ngonga Ngomo, M. A. Sherif, K. Georgala, M. Hassan, K. Dreßler, K. Lyko, D. Obraczka, and T. Soru. LIMES - A Framework for Link Discovery on the Semantic Web. *KI-Künstliche Intelligenz, German Journal of Artificial Intelligence - Organ des Fachbereichs "Künstliche Intelligenz" der Gesellschaft für Informatik e.V.*, 2021. URL https://papers.dice-research.org/2021/KI_LIMES/public.pdf.

D. Nguyen, D. Q. Nguyen, T. Nguyen, and D. Phung. A convolutional neural network-based model for knowledge base completion and its application to search personalization. *Semantic Web*, 10:1–14, 08 2018. doi: 10.3233/SW-180318.

D. Q. Nguyen, T. D. Nguyen, and D. Phung. A relational memory-based embedding model for triple classification and search personalization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 3429—-3435, 2020.

M. Nickel, L. Rosasco, and T. Poggio. Holographic embeddings of knowledge graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 1955–1961. AAAI Press, 2016.

S. Ortona, V. V. Meduri, and P. Papotti. Rudik: Rule discovery in knowledge bases. *Proc. VLDB Endow.*, 11(12):1946–1949, Aug. 2018. ISSN 2150-8097.

H. Paulheim. How much is a triple? estimating the cost of knowledge graph creation. In *International Workshop on the Semantic Web*, 2018.

S. Perera, P. Mendes, A. Sheth, K. Thirunarayan, A. Alex, C. Heid, and G. Mott. Implicit entity recognition in clinical documents. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 228–238, Denver, Colorado, June 2015. Association for Computational Linguistics. doi: 10.18653/v1/S15-1028. URL https://aclanthology.org/S15-1028.

S. Perera, P. N. Mendes, A. Alex, A. P. Sheth, and K. Thirunarayan. Implicit entity linking in tweets. In H. Sack, E. Blomqvist, M. d'Aquin, C. Ghidini, S. P. Ponzetto, and C. Lange, editors, *The Semantic Web. Latest Advances and New Domains*, pages 118–132, Cham, 2016. Springer International Publishing. ISBN 978-3-319-34129-3.

N. Pernelle, F. Saïs, and D. Symeonidou. An automatic key discovery approach for data linking. *Journal of Web Semantics*, 23:16–30, 2013. ISSN 1570-8268. doi: https://doi.org/10.1016/j.websem.2013.07.001. URL https://www.sciencedirect.com/science/article/pii/S1570826813000528. Data Linking.

P. Pezeshkpour, Y. Tian, and S. Singh. Revisiting evaluation of knowledge base completion models. In *Conference on Automated Knowledge Base Construction*, 2020.

U. Qudus, M. Röder, M. Saleem, and A.-C. Ngonga Ngomo. Hybridfc: A hybrid fact-checking approach fornbsp;knowledge graphs. In *The Semantic*

Web – ISWC 2022: 21st International Semantic Web Conference, Virtual Event, October 23–27, 2022, Proceedings, page 462–480, Berlin, Heidelberg, 2022. Springer-Verlag. ISBN 978-3-031-19432-0. doi: 10.1007/978-3-031-19433-7_27. URL https://doi.org/10.1007/978-3-031-19433-7_27.

E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, dec 2001. ISSN 1066-8888. doi: 10.1007/s007780100057. URL https://doi.org/10.1007/s007780100057.

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 https://arxiv.org/abs/1908.10084.

Y. Ren, K. Van Deemter, and J. Z. Pan. Charting the potential of description logic for the generation of referring expressions. In *Proceedings of the 6th International Natural Language Generation Conference*, pages 115–123. ACL, 2010a.

Y. Ren, K. Van Deemter, and J. Z. Pan. Generating referring expressions with owl2. In *23rd International Workshop on Description Logics DL2010*, page 420, 2010b.

A. Sadeghian, M. Armandpour, P. Ding, and D. Z. Wang. *DRUM: End-to-End Differentiable Rule Mining on Knowledge Graphs*. Curran Associates Inc., Red Hook, NY, USA, 2019.

T. Safavi and D. Koutra. CoDEx: A Comprehensive Knowledge Graph Completion Benchmark. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8328–8350, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.669. URL https://aclanthology.org/2020.emnlp-main.669.

A. Salleb-Aouissi, C. Vrain, and C. Nortet. Quantminer: A genetic algorithm for mining quantitative association rules. In *IJCAI*, pages 1035–1040, 2007.

E. Salvat and M.-L. Mugnier. Sound and complete forward and backward chainings of graph rules. In P. W. Eklund, G. Ellis, and G. Mann, editors, *Conceptual Structures: Knowledge Representation as Interlingua*, pages 248–262, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg. ISBN 978-3-540-68730-6.

M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In A. Gangemi, R. Navigli, M.-E. Vidal, P. Hitzler, R. Troncy, L. Hollink, A. Tordai, and M. Alam, editors, *The Semantic Web*, pages 593–607, Cham, 2018. Springer International Publishing. ISBN 978-3-319-93417-4.

B. Shi and T. Weninger. Proje: Embedding projection for knowledge graph completion. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 1236–1242. AAAI Press, 2017.

L. Simonne, N. Pernelle, F. Saïs, and R. Thomopoulos. Differential causal rules mining in knowledge graphs. In *Proceedings of the 11th on Knowledge Capture Conference*, K-CAP '21, page 105–112, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384575. doi: 10.1145/3460210.3493584. URL https://doi.org/10.1145/3460210.3493584.

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

T. Soru, E. Marx, and A.-C. Ngonga Ngomo. Rocker: A refinement operator for key discovery. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, page 1025–1033, Republic and Canton of Geneva, CHE, 2015a. International World Wide Web Conferences Steering Committee. ISBN 9781450334693. doi: 10.1145/2736277.2741642. URL https://doi.org/10.1145/2736277.2741642.

T. Soru, E. Marx, and A.-C. Ngonga Ngomo. Rocker: A refinement operator for key discovery. In *Proceedings of the 24th International Conference on WWW*, pages 1025–1033, 2015b.

R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *ACM SIGMOD Conference*, 1996.

F. M. Suchanek, S. Abiteboul, and P. Senellart. Paris: Probabilistic alignment of relations, instances, and schema. *Proc. VLDB Endow.*, 5(3):157–168, Nov. 2011. ISSN 2150-8097.

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. URL https://openreview.net/forum?id=HkgEQnRqYQ.

D. Symeonidou, N. Pernelle, and F. Sais. Kd2r: A key discovery method for semantic reference reconciliation. In *On the Move to Meaningful Internet Systems, Crete, Greece. Proceedings*, volume 7046, pages 392–401, 09 2011.

D. Symeonidou, V. Armant, N. Pernelle, and F. Saïs. Sakey: Scalable almost key discovery in rdf data. In *International Semantic Web Conference*, pages 33–49. Springer, 2014.

D. Symeonidou, L. Galárraga, N. Pernelle, F. Saïs, and F. Suchanek. Vickey: Mining conditional keys on knowledge bases. In *International Semantic Web Conference*, pages 661–677. Springer, 2017.

T. P. Tanon, D. Stepanova, S. Razniewski, P. Mirza, and G. Weikum. Completeness-aware rule learning from knowledge graphs. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 5339–5343. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi: 10.24963/ijcai.2018/749. URL https://doi.org/10.24963/ijcai.2018/749.

Y. C. Teach, D. Ruffinelli, S. Broscheit, and R. Gemulla. You can teach an old dog new tricks! on training knowledge graph embeddings. In *International Conference on Learning Representations*, 2020.

B. D. Trisedya, J. Qi, and R. Zhang. Entity alignment between knowledge graphs using attribute embeddings. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press, 2019. ISBN 978-1-57735-809-1. doi: 10.1609/aaai.v33i01.3301297. URL https://doi.org/10.1609/aaai.v33i01.3301297.

T. Trouillon, C. R. Dance, É. Gaussier, J. Welbl, S. Riedel, and G. Bouchard. Knowledge graph completion via complex tensor factorization. *Journal of Machine Learning Research (JMLR)*, pages 1–38, 2017.

O. Udrea, L. Getoor, and R. J. Miller. Leveraging data and structure in ontology integration. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD '07, page 449–460. Association for Computing Machinery, 2007. ISBN 9781595936868.

B. Veseli, S. Singhania, S. Razniewski, and G. Weikum. Evaluating language models fornbsp;knowledge base completion. In *The Semantic Web: 20th International Conference, ESWC 2023, Hersonissos, Crete, Greece, May 28–June 1, 2023, Proceedings*, page 227–243, Berlin, Heidelberg, 2023. Springer-Verlag. ISBN 978-3-031-33454-2. doi: 10.1007/978-3-031-33455-9_14. URL https://doi.org/10.1007/978-3-031-33455-9_14.

J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Silk - a link discovery framework for the web of data. In *LDOW*, 2009a.

J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Discovering and maintaining links on the web of data. In *International Workshop on the Semantic Web*, 2009b.

H. Wang, V. Kulkarni, and W. Y. Wang. {DOLORES}: Deep contextualized knowledge graph embeddings. In *Automated Knowledge Base Construction*, 2020a. doi: 10.24432/C5SG64. URL https://openreview.net/forum?id=ajrveGQBl0.

P.-W. Wang, D. Stepanova, C. Domokos, and J. Z. Kolter. Differentiable learning of numerical rules in knowledge graphs. In *International Conference on Learning Representations*, 2020b. URL https://openreview.net/forum?id=rJleKgrKwS.

Q. Wang, Z. Mao, B. Wang, and L. Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017a. doi: 10.1109/TKDE.2017.2754499.

Q. Wang, Z. Mao, B. Wang, and L. Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017b. doi: 10.1109/TKDE.2017.2754499.

Q. Wang, P. Huang, H. Wang, S. Dai, W. Jiang, J. Liu, Y. Lyu, Y. Zhu, and H. Wu. Coke: Contextualized knowledge graph embedding, 2020c.

X. Wang, Y. Jiang, Y. Luo, H. Fan, H. Jiang, H. Zhu, and Q. Liu. FTRLIM results for OAEI 2019. In P. Shvaiko, J. Euzenat, E. Jiménez-Ruiz, O. Hassanzadeh, and C. Trojahn, editors, *Proceedings of the 14th International Workshop on Ontology Matching co-located with the 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 26, 2019*, volume 2536 of *CEUR Workshop Proceedings*, pages 146–152. CEUR-WS.org, 2019. URL http://ceur-ws.org/Vol-2536/oaei19_paper9.pdf.

Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge graph embedding by translating on hyperplanes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 28(1), Jun. 2014a. doi: 10.1609/aaai.v28i1.8870. URL https://ojs.aaai.org/index.php/AAAI/article/view/8870.

Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge graph embedding by translating on hyperplanes. AAAI'14, page 1112–1119. AAAI Press, 2014b.

G. Weikum, X. L. Dong, S. Razniewski, F. Suchanek, et al. Machine knowledge: Creation and curation of comprehensive knowledge bases. *Foundations and Trends® in Databases*, 10(2-4):108–490, 2021.

R. West, E. Gabrilovich, K. Murphy, S. Sun, R. Gupta, and D. Lin. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, page 515–526, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450327442. doi: 10.1145/2566486.2568032. URL https://doi.org/10.1145/2566486.2568032.

J. Wu, Z. Pan, C. Zhang, and P. Wang. Lily results for OAEI 2019. In P. Shvaiko, J. Euzenat, E. Jiménez-Ruiz, O. Hassanzadeh, and C. Trojahn, editors, *Proceedings of the 14th International Workshop on Ontology Matching co-located with the 18th International Semantic Web Conference (ISWC 2019), Auckland, New Zealand, October 26, 2019*, volume 2536 of *CEUR Workshop Proceedings*, pages 153–159. CEUR-WS.org, 2019. URL http://ceur-ws.org/Vol-2536/oaei19_paper10.pdf.

H. Xiao, M. Huang, and X. Zhu. From one point to a manifold: Knowledge graph embedding for precise link prediction. In *International Joint Conference on Artificial Intelligence*, 2015.

R. Xie, Z. Liu, J. Jia, H. Luan, and M. Sun. Representation learning of knowledge graphs with entity descriptions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), Mar. 2016a. URL https://ojs.aaai.org/index.php/AAAI/article/view/10329.

R. Xie, Z. Liu, and M. Sun. Representation learning of knowledge graphs with hierarchical types. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 2965–2971. AAAI Press, 2016b. ISBN 9781577357704.

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*, pages 1980–1990, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1223. URL https://aclanthology.org/D18-1223.

B. Yang, S. W.-t. Yih, X. He, J. Gao, and L. Deng. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the International Conference on Learning Representations (ICLR) 2015*, May 2015.

F. Yang, Z. Yang, and W. W. Cohen. Differentiable learning of logical rules for knowledge base reasoning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

L. Yao, C. Mao, and Y. Luo. KG-BERT: BERT for knowledge graph completion. *CoRR*, abs/1909.03193, 2019. URL http://arxiv.org/abs/1909.03193.

C. Zhang, H. Yao, C. Huang, M. Jiang, Z. Li, and N. V. Chawla. Few-shot knowledge graph completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3041–3048, 2020a.

S. Zhang, Y. Tay, L. Yao, and Q. Liu. *Quaternion Knowledge Graph Embeddings*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Z. Zhang, J. Cai, Y. Zhang, and J. Wang. Learning hierarchy-aware knowledge graph embeddings for link prediction. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 3065–3072. AAAI Press, 2020b.