Rule mining in knowledge bases

Luis Galarraga del Prado

2016-ENST-0050

EDITE - ED 130

Doctorat ParisTech

THÈSE

pour obtenir le grade de docteur délivré par

TELECOM ParisTech

Spécialité « Informatique »

présentée et soutenue publiquement par

Luis Galárraga

29 septembre 2016

Rule Mining in Knowledge Bases

Directeur de thèse : **Fabian Suchanek**

Jury

M. Stefano CERI, Professor, Politecnico de Milano	Rapporteur
M. Stéphan CLEMENÇON, Professeur, Télécom ParisTech	Examinateur
M. Fabien GANDON, Directeur de Recherches, INRIA Sophia-Antipolis	Examinateur
M. Tom MITCHELL, Professor, Carnegie Mellon University	Rapporteur
M. Joachim NIEHREN, Directeur de Recherches, INRIA Lille	Examinateur
Mme. Marie-Christine ROUSSET, Professeure, Université de Grenoble	Examinatrice
M. Steffen STAAB, Professor, University of Koblenz-Landau	Rapporteur
M. Fabian SUCHANEK, Maître de Conférences, Télécom ParisTech	Directeur de thèse

TELECOM ParisTech
école de l'Institut Mines-Télécom - membre de ParisTech

T H È S E

Abstract

The continuous progress of information extraction (IE) techniques has led to the construction of large general-purpose *knowledge bases* (KBs). These KBs contain millions of computer-readable facts about real-world entities such as people, organizations and places. KBs are important nowadays because they allow computers to "understand" the real world. They are used in multiple applications in Information Retrieval, Query Answering and Automatic Reasoning, among other fields. Furthermore, the plethora of information available in today's KBs allows for the discovery of frequent patterns in the data, a task known as *rule mining*. Such patterns or rules convey useful insights about the data. These rules can be used in several applications ranging from data analytics and prediction to data maintenance tasks.

The contribution of this thesis is twofold : First, it proposes a method to mine rules on KBs. The method relies on a mining model tailored for potentially incomplete web-extracted KBs. Second, the thesis shows the applicability of rule mining in several data-oriented tasks in KBs, namely facts prediction, schema alignment, canonicalization of (open) KBs and prediction of completeness.

Le développement rapide des techniques d'extraction d'information a permis de construire de vastes bases de connaissances généralistes. Ces bases de connaissances contiennent des millions de faits portant sur des entités du monde réel, comme des personnes, des lieux, ou des organisations. Ces faits sont accessibles aux ordinateurs, et leur permettent ainsi de "comprendre" le monde réel. Ces bases trouvent donc de nombreuses applications, notamment pour la recherche d'information, le traitement de requêtes, et le raisonnement automatique.

Les nombreuses informations contenues dans les bases de connaissances peuvent également être utilisées pour découvrir des motifs intéressants et fréquents dans les données. Cette tâche, l'*extraction de règles d'association*, permet de comprendre la structure des données ; les règles ainsi obtenues peuvent être employées pour l'analyse de données, la prédiction, et la maintenance de données, entre autres applications.

Cette thèse présente deux contributions principales. En premier lieu, nous proposons une nouvelle méthode pour l'extraction de règles d'association dans les bases de connaissances. Cette méthode s'appuie sur un modèle d'extraction qui convient particulièrement aux bases de connaissances potentiellement incomplètes, comme celles qui sont extraites à partir des données du Web. En second lieu, nous montrons que l'extraction de règles peut être utilisée sur les bases de connaissances pour effectuer de nombreuses tâches orientées vers les données. Nous étudions notamment la prédiction de faits, l'alignement de schémas, la mise en forme canonique de bases de connaissances ouvertes, et la prédiction d'annotations de complétude.

Acknowledgements

This thesis was accomplished thanks to the financing of the Chair Machine Learning for Big Data of Télécom ParisTech and the Max Planck Institute for Informatics. Moreover, with the following names cloud, I want to thank all the people, who in one way or another, made this adventure of my doctoral studies a wonderful and enriching experience. As the knowledge bases that are studied in this thesis, this name cloud may be *partially complete*.

Esta tesis debe su término al financiamiento otorgado por la Cátedra “Machine Learning for Big Data” de Télécom ParisTech y el “Max Planck Institute for Informatics”. Agradezco también, a través de la siguiente nube de nombres, a todas las personas que hicieron de la aventura de mi tesis doctoral, una experiencia inigualable y enriquecedora. Al igual que las bases de conocimiento que son objeto de estudio en esta tesis, esta nube podría estar *parcialmente completa*.

Table des matières

Chapitre 1

Introduction

1.1 Motivation

Since the inception of the Semantic Web [12], knowledge bases (KBs) have become increasingly prevalent. Initiatives such as DBpedia [7], YAGO [117], NELL [18], Cyc [78], Wikidata [124], and the Knowledge Vault [32] aim at constructing and maintaining large collections of machine readable information about the real world. This information takes the form of facts such as “London is the capital of the United Kingdom", “Barack Obama was born in Honolulu”, or “Every politician is a person". KBs find applications in multiple scenarios such as Information Retrieval, Question Answering and Automatic Reasoning. For instance, when confronted with the query “Barack Obama’s place of birth”, all major search engines nowadays are able to understand that the string “Barack Obama” likely refers to a person and that the user is asking about an attribute (place of birth) of that person. Today’s KBs have been constructed by means of automatic and semi-automatic methods, spanning from Information Extraction and Machine Learning techniques to crowd-sourcing initiatives.

In recent years, KBs have become so large that they can be mined for information. It is possible to find patterns or *rules* in the KBs that describe common correlations in the data. For example, we can mine the rule

$$\textit{livesIn}(x, z) \wedge \textit{marriedTo}(x, y) \Rightarrow \textit{livesIn}(y, z)$$

This rule captures the fact that, very often, the spouse of a person lives in the same place as the person. Such rules do not hold in all cases. Therefore, they have a confidence score associated to them, i.e., the ratio of cases where the rule draws a correct conclusion. In this example, the rule errs for all cases where the spouses reside in different cities.

Finding such rules can serve multiple purposes : First, by applying such rules on the data, new facts can be derived that make the KB more complete. For example, if we know where Barack Obama lives, and if we know that Michelle Obama is his wife, then we can deduce (with high probability) where Michelle Obama lives. Second, such rules can identify potential errors in the knowledge base. If, for instance, the KB contains the statement that Michelle Obama lives in a completely different place, then maybe this statement is wrong. Third, the rules can be used for reasoning. Many reasoning approaches rely on other parties (e.g., human experts) to provide rules [88, 104]. Last, rules describing general regularities can help us understand the data better. We can, for instance, find out that countries often trade with countries speaking the same language, that marriage is a symmetric relationship, that musicians who influence each other often play the same instrument, and so on.

All of this shows that, mining rules on KBs carries a great value for data-oriented tasks such as analytics, prediction and maintenance.

1.2 Contribution

The goal of this thesis is twofold. First, we want to mine "interesting" rules from KBs, i.e., rules that are statistically significant and that draw correct conclusions. Such rules convey useful insights about the data and the domain of knowledge. Our second goal is to apply the rules in multiple KB data maintenance problems. In this thesis, we address tasks in the areas of facts inference, integration of data from independent sources and prediction of completeness.

The thesis is structured as follows :

Preliminaries. The remainder of this chapter is devoted to introduce the basic concepts of KBs and logical rules.

Rule Mining. In Chapter 2 we present the design and implementation of AMIE, a system that can learn logical rules on KBs operating under the Open World Assumption. AMIE can learn rules on large, potentially incomplete KBs in a few minutes without the need for parameter tuning or expert input. AMIE outperforms state-of-the-art rule mining systems both in terms of runtime and quality of rules. The latter dimension is assessed by using the rules for fact inference and measuring the precision of the inferred facts. The work presented in this chapter is based on the following publication :

— Luis Galárraga, Christina Teflioudi, Katja Hose, Fabian Suchanek. *Association Rule Mining Under Incomplete Evidence in Ontological Knowledge Bases*. Proceedings of the 22nd International Conference on World Wide Web. pp 413–422. Rio de Janeiro, Brazil, 2013. *Awarded best student paper.*

Speeding up Rule Mining. Chapter 3 presents AMIE+, an extension of AMIE that implements a set of runtime enhancements aimed at improving the scalability of AMIE. In addition, this chapter introduces an interactive demo that drives users through the process of rule mining with AMIE+ . The contents of this chapter are based on the following publications :

— Luis Galárraga, Christina Teflioudi, Katja Hose, Fabian Suchanek. *Fast Rule Mining in Ontological Knowledge Bases with AMIE+*. International Journal on Very

Large Databases. Volume 24, Issue 6, pp 707–730. December 2015.

— Luis Galárraga. *Interactive Rule Mining in Knowledge Bases*. 31ème Conférence sur la Gestion de Données. Île de Porquerolles, France, 2015.

Wikilinks Semantification. Chapter 4 shows how to apply rule mining to wikilinks. A wikilink is a hyperlink between two Wikipedia pages. In the vast majority of cases, those wikilinks are unsemantified, i.e., there does not exist a relation in the KB that explains the connection between the endpoints of the wikilink. We apply rule mining to predict semantic relations between the entities of a wikilink by mining logical rules such as "politicians that link to a prize have usually won the prize". We then use the rules to predict candidate relations for the entities in unsemantified wikilinks. Such predictions can be used, e.g., to propose new facts for KBs. This work in this chapter was presented in the following publication :

— Luis Galárraga, Danai Symeonidou, Jean-Claude Moissinac. *Rule Mining for Semantifiying Wikilinks*. Proceeding of the 8th Workshop on Linked Open Data. Florence, Italy, 2015.

Schema Alignment. In Chapter 5, we propose to apply rule mining to the task of KB schema alignment. Given several KBs with complementary information about a certain domain, schema alignment is the process of finding mappings between the schemas (relations and classes) of the KBs. The purpose of such mappings is to integrate the KBs, e.g., merge them. This allows us to query the KBs as if they were a single source of information. Our proposal is to express schema alignments as rules, and to mine them automatically with AMIE. This idea was introduced in the following article :

— Luis Galárraga, Nicoleta Preda, Fabian Suchanek. *Mining Rules to Align Knowledge Bases*. Proceedings of the 3rd Workshop on Automated Knowledge Base Construction. pp 43–48. San Francisco, USA, 2013.

Canonicalization of Open KBs. Chapter 6 introduces the problem of canonicalizing open KBs. An open KB is a KB that has been constructed by extracting triples from arbitrary text without enforcing the relations to follow a schema. This schema-free KB may contain facts harvested from independent web sources, and therefore it can express the same fact in different ways. For instance, one web page could contain the statement "President Obama earned a degree from Harvard University", while a second web page could express the same idea as "Barack Obama studied at Harvard University". Canonicalization is the problem of identifying the multiple synonyms of a given entity or relation, so that they are always mentioned in the same way. We propose to use machine learning techniques to cluster synonymous entity names. We then use rule mining to canonicalize the relations. This chapter is based on the following publication :

— Luis Galárraga, Geremy Heitz, Kevin Murphy, Fabian Suchanek. *Canonicalizing Open Knowledge Bases*. Proceedings of the 23rd ACM International Conference on Information and Knowledge Management. pp 1679–1688. Shanghai, China, 2014.

Predicting Completeness. In Chapter 7, we study the problem of predicting whether the KB knows all the answers to a certain query. We focus on queries of the form $\langle$*e, r, y*$\rangle$ such as $\langle$*Obama, hasParent, y*$\rangle$. Here we wish to predict whether the KB knows all the parents (y values) of Barack Obama. We define and study a set of signals that make predictions about the completeness of query answers. We then use rule mining to combine those signals into logical rules that can capture completeness patterns, such as "people with two parents in the KB do not have more parents". Such rules can be used to generate completeness annotations for KBs or generate explicit counter-evidence for rule mining and fact inference. The work presented in this chapter is under revision :

— Luis Galárraga, Simon Razniewski, Antoine Amarilli, Fabian Suchanek. *Predicting Completeness in Knowledge Bases*. Proceeding of the 10th International Conference on Web Search and Data Mining. Cambridge, UK, 2017.

Numerical Rule Mining. Chapter 8 takes a visionary approach and proposes a language for numerical rule mining. This work is an attempt to reconciliate some of the existing work in this area so that it fits within the framework provided by AMIE. The chapter builds upon the following vision paper :

— Luis Galárraga, Fabian Suchanek. *Towards a Numeric Rule Mining Language*. Proceedings of the 4th Workshop on Automated Knowledge Base Construction. Montreal, Canada, 2014.

Conclusion. Chapter 9 concludes this thesis, by summarizing the scope of these contributions and suggesting potential avenues of research.

1.3 Preliminaries

We focus on KBs such as YAGO [117], Wikidata [124] or DBpedia [7]. These KBs store general purpose information about real world entities such as people, places and organizations. They know, for instance, that "London is the capital of UK", "Barack Obama is a politician", and that "all politicians are people". Current KBs have been constructed in multiple ways. Some projects rely on fully automatic Information Extraction (IE) techniques [32, 117]. For instance, the YAGO KB [117] integrates information from the Wikipedia infoboxes, the Wikipedia categories, Wordnet [1], and Geonames [2]. Other KBs extract knowledge from arbitrary web pages in the web, e.g., NELL [18]. In contrast, some KBs rely on knowledge provided by human annotators, e.g., Wikidata [124]. Finally, some of them may use a combination of these techniques [7,78,124].

1. https://wordnet.princeton.edu/
2. http://www.geonames.org/

1.3.1 RDF Knowledge Bases

Conceptually, a knowledge base (KB) is a set of computer-readable facts about a domain of knowledge. Those facts are represented using the RDF[3] data model [128], the standard W3C recommendation for knowledge representation. In RDF, facts take the form of triples ⟨*x, r, y*⟩. Examples of facts are ⟨*UK, hasCapital, London*⟩, ⟨*Barack, type, Politician*⟩ and ⟨*Barack, fullName, 'Barack Hussein Obama'*⟩. The subject is always a resource, i.e., a real-world concept or entity[4], while the object can be either a resource or a literal value, e.g., a string, an integer or a date. To differentiate entity identifiers from literals, we write the first as unquoted strings starting with a capital letter, e.g., *London*, *Politician*. Literal values, in contrast, are represented by single-quoted strings, e.g., 'Barack Hussein Obama'. There are several equivalent alternative representations of facts ; in this thesis we borrow the notation from Datalog and represent a fact ⟨*x, r, y*⟩ as $r(x, y)$. For example, we write *hasCapital(UK, London)*. Since we always assume a fixed KB, we will write $r(x, y)$ to mean $r(x, y) \in \mathcal{K}$.

KBs normally have a *schema* associated to them. The schema consists of a set of statements that define, among other things, the domains and ranges for relations and the class hierarchy. The RDF Schema (RDFS) recommendation [103] provides the vocabulary to define the schema of an RDF KB. For example, the schema of a KB about countries and cities could contain the fact *rdfs :subClassOf*$(EUCountry, Country)$. This fact states that every instance of the class *EUCountry* is also an instance of the class *Country*. The facts *rdfs :domain*$(hasCapital, Country)$ and *rdfs :range*$(hasCapital, City)$ are also typical schema facts. They state that the relation $hasCapital$ is defined from countries to cities. For simplicity, we write $domain(r)$ and $range(r)$ to refer to the domain and range of relation r respectively. In this thesis, we are not concerned with finding rules in the schema of KBs, because the schema is normally defined by the KB designers and it is usually much smaller than the actual KB. However, we may make use of the schema information to improve the quality of the mined rules. Moreover, in the rest of the thesis, we will use the terms *knowledge bases* or simply KBs to refer to RDF KBs.

We now define a KB in a formal way. If $\mathcal{E}$ is a set of entities, $\mathcal{R}$ is a set of relations, and $\mathcal{L}$ is a set of literals, a KB $\mathcal{K}$ is a subset of $\mathcal{E} \times \mathcal{R} \times (\mathcal{E} \cup \mathcal{L})$. In general, we give less attention to the set of literal values $\mathcal{L}$, because literal attributes tend to be unique for instances. This fact makes them less interesting when mining rules that hold frequently in the data[5].

1.3.2 Closed vs. Open World Assumption

The *Closed World Assumption* (CWA) is the assumption that any fact that is not known to be true, is *false*. Under the CWA, e.g., if a KB does not know that Barack

3. Resource Description Framework
4. RDF also defines anonymous entities, called blank nodes. We do not consider them in this thesis.
5. Nevertheless, literal numerical values become crucial in our vision for numerical rule mining presented in Chapter 8

Obama won a Grammy Award, then we can conclude that he did not win a Grammy Award. This is equivalent to say that the KB is *complete* with respect to the real world. Conversely, the *Open World Assumption* (OWA) is the assumption that the truth value of a fact is independent of whether the fact is known to be true. This means that under the OWA, statements not present in the KB are not necessarily false, but just *unknown*. Due to the limitations of human knowledge and information extraction techniques, KBs are inherently incomplete. Hence, it is infeasible to assume a KB knows all the true facts about a given domain of knowledge. For this reason, KBs usually operate under the OWA.

1.3.3 Functions

A *function* is a relation r that has at most one object for every subject, i.e.,

$$\forall x : \mathit{type}(x, \mathit{domain}(r)) : |\{y : r(x,y)\}| \leq 1$$

Similarly, a relation is an *inverse function* if each of its objects has at most one subject.

$$\forall y : \mathit{type}(y, \mathit{range}(r)) : |\{x : r(x,y)\}| \leq 1$$

Since KBs are usually noisy, even relations that are conceptually functions (such as *hasBirthdate*) may exhibit two objects for the same subject. Vice versa, there are relations that are not functions in the strict sense, but that exhibit a similar behavior. For example, *isCitizenOf* can give several citizenships to a person, but the vast majority of people have only one citizenship. Therefore, we use a non-binary notion of *functionality* [116]. The functionality of a relation r is a value between 0 and 1 defined by the formula :

$$\mathit{fun}(r) := \frac{\#x : \exists y : r(x,y)}{\#(x,y) : r(x,y)}$$

where $\#\mathbf{x} : \mathbf{X}$ is an abbreviation for $|\{x : X \in \mathcal{K}\}|$. The formula divides the number of different subjects that occur in relation r by the number of facts of r. Relations will have different functionality scores. For functional relations such as $\mathit{hasBirthDate}$ or $\mathit{hasISBN}$, $\mathit{fun}(r)$ is 1 since there are as many facts as subjects in the relation. Similarly, quasi-functions such as $\mathit{isCitizenOf}$ exhibit values close to 1. Non-functional relations such as $\mathit{hasFriend}$, in contrast, will exhibit values close to 0 as each subject is associated to multiple object values.

Given a relation r, we define its inverse r^{-1} as the relation constructed by swapping the arguments of r, i.e., $\forall r(x,y) : r^{-1}(x,y) := r(y,x)$. For instance, $\mathit{isCitizenOf}^{-1} = \mathit{hasCitizen}$. We define the inverse functionality ifun of a relation r as the functionality of its inverse, that is, $\mathit{ifun}(r) := \mathit{fun}(r^{-1})$.

Some relations have roughly the same degree of functionality and of inverse functionality. Bijections are an example. Usually, however, fun and ifun are different. Manual inspection shows that in web-extracted common sense KBs (e.g., YAGO, DBpedia) the functionality is usually higher than the inverse functionality. For example, a KB is more likely to specify $isCitizenOf : Person \rightarrow Country$ than $hasCitizen : Country \rightarrow Person$. Intuitively, this allows us to consider a fact $r(x, y)$ as a fact about x. In the following, we will assume that for all relations r, $fun(r) \geq ifun(r)$. Whenever this is not the case, r can be replaced by its inverse relation r^{-1}. Then, $fun(r^{-1}) \geq ifun(r^{-1})$. In the following, we assume that all relations have been substituted with their inverses if their inverse functionality is larger than their functionality. This trick may lead to relations with literal domains, thus slightly deviating from the RDF standard. However, it simplifies our approaches considerably without affecting their generality.

1.3.4 Rules

An *atom* is a fact that allows variables at the subject and/or object position. Examples are $livesIn(x, USA)$ or $isCitizenOf(x, y)$. In the rest of this thesis, we denote variables by lowercase letters.

A *Horn rule* consists of a head and a body, where the head is a single atom and the body is a logical conjunction of atoms of zero, one, or more atoms. We denote a rule with head H and body $\{B_1, ..., B_n\}$ by an implication

$$B_1 \wedge B_2 \wedge ... \wedge B_n \Rightarrow H$$

which we abbreviate as $\boldsymbol{B} \Rightarrow H$. An example is the rule

$$livesIn(x, y) \wedge isMarriedTo(x, z) \Rightarrow livesIn(z, y)$$

An *instantiation* σ is an assignment of variables to constants. The application of an instantiation σ to an atom B produces a new atom $\sigma(B)$ where all variables in the instantiation have been replaced by the corresponding constants. For example given the instantiation $\sigma := \{x \rightarrow Oana, y \rightarrow Paris\}$, its application to the atom $livesIn(x, y)$, yields the instantiated fact $livesIn(Oana, Paris)$. This notion is naturally extended to conjunction and rules :

$$\sigma(\boldsymbol{B}) := \sigma(B_1) \wedge \sigma(B_2) \wedge \cdots \wedge \sigma(B_n)$$

$$\sigma(\boldsymbol{B} \Rightarrow H) := \sigma(\boldsymbol{B}) \Rightarrow \sigma(H)$$

We say a fact $r(x, y)$ is a *prediction* of a rule $\boldsymbol{B} \Rightarrow H$ in a KB $\mathcal{K}$ if

$$\exists\, \sigma : \forall B_i \in \boldsymbol{B} : \sigma(H) = r(x, y) \wedge \sigma(B_i) \in \mathcal{K}$$

In other words, a prediction of $\boldsymbol{B} \Rightarrow H$ in $\mathcal{K}$ is an instantiation of the head atom such that the corresponding instantiated body atoms are in $\mathcal{K}$. For example, let us consider a KB containing the facts *livesIn(Barack, Washington)* and $isMarriedTo(Barack, Michelle)$. Let us also consider the rule that predicts that spouses live in the same city :

$$livesIn(x, y) \wedge isMarriedTo(x, z) \Rightarrow livesIn(z, y)$$

We say that the fact $f := \mathit{livesIn}(\mathit{Michelle}, \mathit{Washington})$ is a prediction of this rule under the instantiation $\sigma := \{x \to \mathit{Barack}, y \to \mathit{Washington}, z \to \mathit{Michelle}\}$. If a fact f is a prediction of a rule $\boldsymbol{B} \Rightarrow H$ on a KB $\mathcal{K}$, we say the rule *entails* f, which we denote by $\mathcal{K} \wedge (\boldsymbol{B} \Rightarrow H) \models f$. It is important to remark, that our definition of a prediction does not require f to be in the KB or to hold in reality. This implies that rules can make predictions outside the KB and that some of those predictions may be wrong. In Section 2.3.3 we discuss different strategies to gauge the quality of a rule on a KB based on the number of correct and wrong predictions it makes.

Two atoms in a rule are *connected* if they share a variable or constant. A rule is *connected* if every atom is connected transitively to every other atom of the rule. We restrict our search to connected rules. This restriction avoids mining rules with completely unrelated atoms, such as $\mathit{diedIn}(x, y) \Rightarrow \mathit{wasBornIn}(w, z)$.

A variable in a rule is *closed* if it appears in at least two atoms of the rule. We say, a rule is *closed* if all its variables are closed. If only the head variables are closed, we say the rule is *head-closed*. Consider for instance the following rule :

$$\mathit{hasCapital}(z, y) \wedge \mathit{diedIn}(x, y) \Rightarrow \mathit{wasBornIn}(x, y)$$

This rule is head-closed because the head variables x and y are closed (in contrast to z). Head-closed rules make concrete predictions. Thus, restricting to head-closed rules avoids predicting the mere existence of a fact as in the rule

$$\mathit{diedIn}(x, y) \Rightarrow \mathit{wasBornIn}(x, z)$$

In this example, the variable z is implicitly existentially quantified, that is, the rule is actually equivalent to $\mathit{diedIn}(x, y) \Rightarrow \exists z : \mathit{wasBornIn}(x, z)$. We say a rule is *recursive* if the head relation occurs also in the body as in

$$\mathit{isCitizenOf}(z, y) \wedge \mathit{hasChild}(z, x) \Rightarrow \mathit{isCitizenOf}(x, y)$$

We say an atom is *reflexive* if it contains the same variable in both arguments, e.g., $r(x, x)$.

1.3.5 Language bias

Inspired by the definitions presented in [3], we define a *syntactic language bias* $\mathcal{L}$ as a function that maps rules of the form $\boldsymbol{B} \Rightarrow H$ to truth values. If the function returns *true*, we say the function *accepts* the rule, otherwise the function *rejects* the rule. Language biases are used in rule mining to restrict the size of the search space, by focusing on particular types of rules. An example of a syntactic language bias is the language of rules with at most n atoms, denoted by $\mathcal{L}_n$. For instance, the rule $\mathit{livesIn}(x, y) \wedge \mathit{isMarriedTo}(x, z) \Rightarrow \mathit{livesIn}(z, y)$ is accepted by the language $\mathcal{L}_3$, but rejected by $\mathcal{L}_2$. Other examples are the language of head-closed rules $\mathcal{L}_{hclosed}$, the language of closed rules $\mathcal{L}_{closed}$, the language of rules without reflexive atoms $\mathcal{L}_{not\text{-}reflex}$ or the language of rules with distinct atoms $\mathcal{L}_{distinct}$. Language biases can be combined in a conjunctive fashion, e.g., $\mathcal{L}_{n\text{-}closed} := \mathcal{L}_{closed} \wedge \mathcal{L}_n$ defines the language bias of closed rules up to n atoms. We use the notation $(\boldsymbol{B} \Rightarrow H) \in \mathcal{L}$ to denote that $\mathcal{L}$ accepts the rule $\boldsymbol{B} \Rightarrow H$. For instance

$$(\mathit{isCitizenOf}(z,y) \wedge \mathit{hasChild}(z,x) \Rightarrow \mathit{isCitizenOf}(x,y)) \in \mathcal{L}_{closed}$$

On the other hand

$$(\mathit{livesIn}(x,z) \Rightarrow \mathit{isCitizenOf}(x,y)) \notin \mathcal{L}_{closed}$$

This happens because the variables y and z are not closed.

In this thesis, like in other rule mining approaches, we are not interested in general Horn clauses, but only in the rules of a certain language bias. Language biases offer a trade-off between the expressiveness of the mined rules and the speed of the mining process. As an example, the rules in $\mathcal{L}_3$ can capture more complicated correlations than rules in $\mathcal{L}_2$, but come with a larger search space and thus with a much slower performance. The less restrictive the language bias is, the more expressive the rules can potentially be, the larger the search space grows, and the less tractable the search becomes.

We remark that syntactic language biases are oblivious to the content of the KB on which the rules are mined. For example, one may want to accept rules that make a minimal number of predictions occurring in a given KB. Since no syntactic language bias could enforce such a requirement, we define a *semantic language bias* $\mathcal{L}^{\mathcal{K}}$ as a function that accepts or rejects rules of the form $\boldsymbol{B} \Rightarrow H$, given a KB $\mathcal{K}$. As we will show in Chapter 2, semantic language biases can also have a great impact in the performance of rule mining. All existing rule mining systems resort to a combination of syntactic and semantic language biases to drive the search space exploration.

Chapitre 2

Rule Mining

In this chapter, we present AMIE (Association Rule Mining Under Incomplete Evidence), a system designed to mine Horn rules on KBs. Unlike state-of-the-art approaches, AMIE is tailored for potentially incomplete KBs operating under the Open World Assumption (OWA). This is possible thanks to a novel mining model that estimates the confidence of a rule in the presence of incompleteness. AMIE also relies on a highly concurrent implemention that allows the system to scale to KBs up to 6M facts.

This chapter is structured as follows. Section 2.1 introduces the problem of rule mining in KBs in a formal fashion. Section 2.2 discusses the related work in the area of rule mining on structured data. Section 2.3 describes the AMIE mining model. This is followed by the introduction and study of the Partial Completeness Assumption (PCA) in Section 2.4. The PCA plays a central role in the generation of counter-evidence under the OWA and the definition of confidence for rules. In Section 2.5 we describe the AMIE core algorithm and its implementation. The performance of AMIE and its applicability in the task of data inference are covered in Sections 2.6 and 2.7. Section 2.8 concludes the chapter.

This chapter is based on the following publication :

— Luis Galárraga, Christina Teflioudi, Katja Hose, Fabian Suchanek. *Association Rule Mining Under Incomplete Evidence in Ontological Knowledge Bases*. Proceedings of the 22nd International Conference on World Wide Web. pp 413–422. Rio de Janeiro, Brazil, 2013. *Awarded best student paper*.

2.1 Introduction

Goal. The goal of our work with AMIE is to mine high quality Horn rules from large, potentially incomplete KBs operating under the OWA. In addition, we aim at rules that can predict correct facts beyond the KB. As explained later in this section, these goals pose multiple challenges.

Learning Horn rules. The problem of inducing Horn rules from a set of facts is the central topic of Inductive Logic Programming (ILP). In a standard ILP setting, the facts of a KB $\mathcal{K}$ are divided into two (potentially overlapping) sets : the evidence set $\mathcal{K}_e$ and background knowledge $\mathcal{K}_b$. The set $\mathcal{K}_e$ contains facts about a fixed target relation r and is divided into two disjoint subsets : $\mathcal{K}_e^+$, the set of positive examples, and $\mathcal{K}_e^-$, the set of negative examples. Given a syntatic language bias $\mathcal{L}$, and an input KB $\mathcal{K}$, the problem of inductive logic programming consists of finding a hypothesis (rule) $\boldsymbol{B} \Rightarrow H$, such that :

1. $(\boldsymbol{B} \Rightarrow H) \in \mathcal{L}$
2. $\forall e \in \mathcal{K}_e^+ : (\mathcal{K}_b \wedge \boldsymbol{B} \Rightarrow H) \vDash e$
3. $\forall e \in \mathcal{K}_e^- : (\mathcal{K}_b \wedge \boldsymbol{B} \Rightarrow H) \nvDash e$

In other words, ILP searches for a rule, that based on the background knowledge, (1) meets the given syntactic language bias, (2) predicts all the positive examples in the evidence set and (3) predicts none of the negative examples. Conditions 2 and 3 define the standard ILP semantic language bias, which we denote by $\mathcal{L}_{ILP}^{\mathcal{K}}$. As an example, consider a KB $\mathcal{K}$ and the syntactic language bias $\mathcal{L}_{3-closed}$ with

$$\mathcal{K}_e^+ := \{isCitizenOf(Malia, USA)\}$$
$$\mathcal{K}_e^- := \{isCitizenOf(Malia, UK)\}$$
$$\mathcal{K}_b := \{isCitizenOf(Barack, USA), hasChild(Barack, Malia), visited(Barack, UK)\}$$

Based on the background knowledge in $\mathcal{K}_b$, a standard ILP approach should learn the rule

$$isCitizenOf(z, y) \wedge hasChild(z, x) \Rightarrow isCitizenOf(x, y)$$

because it meets $\mathcal{L}_{3-closed}$ and $\mathcal{L}_{ILP}^{\mathcal{K}}$, that is, it entails the positive example $isCitizenOf(Malia, USA)$ (Condition 2) and not the negative example $isCitizenOf(Malia, UK)$ (Condition 3). Conversely, it would be inacceptable to learn the rule $visited(z, y) \wedge hasChild(z, x) \Rightarrow isCitizenOf(x, y)$ because it does not meet $\mathcal{L}_{ILP}^{\mathcal{K}}$, in particular it predicts the given negative example.

Since real-world data is usually noisy, finding rules that match the definition perfectly may be infeasible. For this reason, ILP approaches normally relax $\mathcal{L}_{ILP}$, allowing rules to cover a subset of the positive examples and hopefully few of the negative examples. We discuss approaches to this problem further down.

Complexity of Rule Mining. Inducing Horn rules from a KB is an inherently hard problem. In [47] it is shown that given a KB $\mathcal{K} := \{\mathcal{K}_b, \mathcal{K}_e\}$ and a syntactic language bias $\mathcal{L}_n$ for some n polynomially bounded on $|\mathcal{K}|$, the problem of deciding whether there exists a Horn rule $\boldsymbol{B} \Rightarrow H$ that satisfies $\mathcal{L}_n$ and $\mathcal{L}_{ILP}^{\mathcal{K}}$ is Σ_2^P-complete, with $\Sigma_2^P = \mathrm{NP}^{NP}$. To understand the extent of such level of complexity, let us define an *oracle machine* as a Turing machine that contains an oracle capable of solving a specific problem in unit time. The oracle resembles a blackbox or function that can be called an arbitrary number of times and provides an answer instantly. We say a decision problem p belongs to a complexity class NP^C, if p can be solved in polynomial time by a non-deterministic oracle machine whose oracle solves a problem in complexity class C. This means NP^{NP} defines the set of problems solvable in polynomial time by a non-deterministic Turing machine equipped with an oracle with complexity in NP. In addition, since rule mining is NP^{NP}-complete, any problem in NP^{NP} can be transformed to rule mining via a polynomial time transformation.

While the problem of rule mining remains intractable in theory, recent approaches [20, 42, 43, 133] have proved it is tractable in practice. Those approaches resort in one way or the other to syntactic language biases and relaxations of the original $\mathcal{L}_{ILP}$ semantic bias, in order to prune uninteresting regions of the search space. In AMIE [43], for instance, we use a support threshold to reduce the size of the search space by avoiding rules with low statistical significance.

Challenges. Mining rules on KBs poses several challenges. First, KBs contain only positive statements and no negations. For instance, they cannot express the fact that Barack Obama does not hold German citizenship. This makes the standard ILP setting inapplicable to KBs because counter-examples for rules are not available, i.e., $\mathcal{K}_e^- = \emptyset$. Furthermore, the basic ILP formulation does not deal with the predictions of the rule that are outside the KB and the evidence set. We can therefore claim that basic ILP is not concerned with prediction of facts, but with the description of the KB and the explanation of evidence set. This is incompatible with our goal of predicting facts beyond the KB.

A second challenge comes from the fact that KBs operate under the Open World Assumption (OWA, see Section 1.3.2). Under the OWA, a statement that is not contained in the KB is not necessarily false ; it is just *unknown*. This means that absent facts cannot take the role of counter-examples. This is a crucial difference to many standard database settings that operate under the Closed World Assumption (CWA). Consider an example KB that does not contain the information that Barack Obama is a citizen of the USA. Under the CWA we would conclude that he is not American. Under the OWA, however, he could be American. It follows that if a rule predicts that Barack Obama is American, rule mining systems do not have a way to know whether this prediction is true or false. This also implies that we cannot accurately measure the correcteness of rules.

Another challenge originates from the size of current KBs. As of 2016, the latest version of the YAGO KB [117] contains 120M facts, about 10M entities and 100 relations. Furthermore, the Knowledge Vault (Google's KB) has been reported to contain 1.6B facts and more than 4000 relations [32]. In particular, learning rules on KBs triggers an instance of the ILP problem formulation for every target relation r in the KB. In such a setting, the evidence set consists of all the facts of the target relation, i.e., $\mathcal{K}_e^+ = \{r(x, y) : r(x, y) \in \mathcal{K}\}$, while the background knowledge comprises all the facts of the KB, i.e., $\mathcal{K}_b = \mathcal{K}$. Since the classical ILP problem is NP^{NP}-complete, it is impossible to solve the problem on KBs of this size.

With our work on AMIE, we address all the aforementioned challenges.

Transaction Label	**Transaction Items**
⟨Elvis, Lisa, Priscilla⟩	{mother(x_3,x_2), father(x_1,x_2), marriedTo(x_1,x_3)}
⟨Barack, Malia, Michelle⟩	{mother(x_3,x_2),father(x_1,x_2),marriedTo(x_1,x_3)}
⟨François, Flora, Ségolène⟩	{mother(x_3,x_2),father(x_1,x_2)}

TABLE 2.1 – Mining Rules with 3 Variables

2.2 Related Work

Rule mining has been an area of active research during the last 25 years. Some approaches mine association rules, some mine logical rules, others mine a schema for the KB, and again others use rule mining for application purposes. In the following, we survey the most pertinent related work along these lines.

2.2.1 Association Rule Mining

Association Rule Mining [4] discovers correlations in shopping transactions. A transaction is a set of items. For example, in the context of sales analysis, a transaction is the set of products bought together by a customer in a specific event. The mined rules are of the form {*Bread, Wine*} ⇒ {*Cheese*}, meaning that people who bought bread and wine usually also bought cheese. However, these are not the kind of rules that we aim to mine in this thesis ; we aim at mining Horn rules as defined in Section 1.3. Thus, association rules are different in nature from the Horn rules we aim at. Still, we can show some similarities between the two approaches. Let us define one transaction for every set of k entities that are connected in the KB. For example, in Table 2.1, we will define a transaction for the entities *Elvis*, *Lisa* and *Priscilla*, because they are connected through the facts *mother(Priscilla,Lisa)*, *father(Elvis,Lisa)*, *marriedTo(Elvis, Priscilla)*. We label the transaction with the set of these entities. Each atom $r(x_i, x_j)$ on variables indexed by $1 \leq i, j \leq n$ corresponds to an item. A transaction with label $\langle C_1, \ldots, C_n \rangle$ contains an item $r(x_i, x_j)$ if $r(C_i, C_j)$ is in the KB. For example, the transaction ⟨*Elvis, Lisa, Priscilla*⟩ contains the items {*mother(x_3,x_2), father(x_1,x_2), marriedTo(x_1,x_3)*}, since the ground atoms *mother(Priscilla,Lisa)*, *father(Elvis,Lisa)* and *marriedTo(Elvis, Priscilla)* are in the KB. In this representation, association rules are Horn rules. In the example, we can mine the association rule

$$\{mother(x_3, x_2), marriedTo(x_1, x_3)\} \Rightarrow \{father(x_1, x_2)\}$$

which corresponds to the Horn rule

$$mother(x_3, x_2) \wedge marriedTo(x_1, x_3) \Rightarrow father(x_1, x_2)$$

Constructing such a table with all possible combinations of entities is practically not viable. If we see a KB as graph with entities as nodes and relations as directed edges, building the table is equivalent to enumerating all connected subgraphs of k nodes, where k is also the number of variables in the rule. Such an enumeration is exponential in k. To see this, consider a fully connected graph (the worst case) with $n = |\mathcal{E}|$. In that case, there are $\binom{n}{k} > (\frac{n}{k})^k$ subgraphs [122]. Apart from that, enumerating all the transactions faces a number of design issues, e.g., how to deal with transactions that contain the same entities in different orderings. Therefore, association rule mining cannot be used directly to mine Horn rules. However, we take inspiration from the parallels between the two types of mining for our system, AMIE.

2.2.2 Inductive Logic Programming

Sherlock [109] is an unsupervised ILP method to learn first-order Horn clauses from open domain facts. Sherlock uses probabilistic graphical models (PGMs) to infer new facts. It tackles the noise of the extracted facts by extensive filtering in a preprocessing step and by penalizing longer rules in the inference part. For mining the rules, Sherlock uses two heuristics : statistical significance and statistical relevance. The statistical significance of a clause $A \Rightarrow B$ measures the number of positive examples of the clause in the extracted corpus, i.e., the cases where both A and B hold. The statistical relevance, on the other hand, measures the divergence between $P(B|A)$, i.e., the probability of the succedent given the antecedent, and $P(B)$, the probability of the succedent per se. Unlike AMIE, it works on facts extracted from free text that are not mapped to crisp relations. Besides, Sherlock works under the Closed World Assumption.

QuickFOIL [133] is a standard ILP system based on a generic top-down greedy algorithm and implemented on top of the QuickStep in-memory storage engine [19]. It learns a set of hypotheses (Horn rules) from positive and negative examples of a target relation and a collection of background facts. When refining a rule, the QuickFOIL algorithm greedily picks the clause that maximizes a scoring function depending on the support and the confidence gain of the new rule. Once a rule is mined, the algorithm removes the positive examples covered by the rule and starts the induction process on the remaining facts. QuickFOIL can scale to problem instances with millions of background facts thanks to a set of aggressive pruning heuristics and multiple database optimizations. However, it is not suitable for mining rules under the Open World Assumption, since it requires explicit negative examples.

A very recent approach called Ontological Pathfinding (OP) [20], exploits the information of the T-Box (specifically relation domains and ranges) to mine all potential candidate rules in a first phase. In a second phase, the support and confidence of the candidates is evaluated in the A-box. The OP algorithm relies on smart data-partitioning and a highly parallel implementation on top of Apache Spark [1], to find rules on a set of 388M facts from Freebase [120]. Like AMIE, OP aims at Horn rules, it does not require explicit counter-examples, and it uses the same definition of support for rules. However, OP resorts to a Closed World Assumption (CWA) to generate counter-examples. The authors compare their approach with AMIE in terms of runtime performance and inference capabilities. In the first category, their implementation provides an important gain

1. `http://spark.apache.org/`

in runtime for very large datasets such as Freebase and YAGO2s, remaining comparable for smaller datasets such as YAGO2. In contrast, OP achieves lower precision at data inference in YAGO2 due to its notion of confidence based on the CWA.

The WARMR system [28, 29] mines patterns in databases that correspond to conjunctive queries. It uses a declarative language bias to reduce the search space. An extension of the system, WARMER [46], modified the approach to support a broader range of conjunctive queries and increase the efficiency of the search space exploration.

ALEPH[2] is a general purpose ILP system that implements Muggleton's Inverse Entailment algorithm [86] in Prolog. It employs a variety of evaluation functions for the rules as well as a variety of search strategies. WARMER and ALEPH are not tailored to deal with large KBs under the Open World Assumption. We compare AMIE to these two systems, which are the only ones available for download. Our experiments do not only show that these systems mine less sensible rules than our approach, but also that they take more time to do so.

2.2.3 Expert Rule Mining

Another rule mining approach over RDF data [90] was proposed to discover causal relations in RDF-based medical data. It requires a domain expert who defines targets and contexts of the mining process, so that the correct transactions are generated. Our approach, in contrast, does not rely on the user to define any context or target. It works out-of-the-box.

2.2.4 Generating Schemas

In this chapter, we aim to generate Horn rules on a KB. Other approaches use rule mining to generate the schema or taxonomy of a KB. [22] applies clustering techniques based on context vectors and formal concept analysis to construct taxonomies. Other approaches use clustering [73] and ILP-based approaches [25]. For the friend-of-a-friend network on the Semantic Web, [49] applies clustering to identify classes of people and ILP to learn descriptions of these groups. Another example of an ILP-based approach is the DL-Learner [67], which has successfully been applied [53] to generate OWL class expressions from YAGO [117]. As an alternative to ILP techniques, [123] proposes a statistical method that does not require negative examples. In contrast to our approach, these techniques aim at generating a schema for a given RDF repository, not logical rules in general.

2.2.5 Relational Learning

Some approaches learn new associations from semantic data without mining explicit logical rules. This task is often referred as *relational machine learning* or *link prediction*.

2. `http://www.cs.ox.ac.uk/activities/machlearn/Aleph/aleph_toc.html`

Methods such as RESCAL [92], among others [93, 112] represent KBs as matrices or tensors. Under this paradigm, for instance, a KB can be represented as a three-dimensional tensor where the fact $r(x, y)$ is encoded as 1 in the cell with coordinates (r, x, y). These methods resort to tensor factorization and latent factor analysis on the matrix representation of the KB, in order to estimate the confidence of the missing cells, i.e., how likely the missing facts are true based on the latent features in the data. Even though the scores are often given a probabilistic interpretation, they are not probabilities in a strict sense.

Another family of approaches [14,45,125] resorts to embedding models to formulate the link prediction problem. In [125], entities are represented as vectors in an embedding space, while relations are defined as transformations on those vectors, e.g., the transformation *nationality* maps the vector of Barack Obama to the vector of USA. Methods based on embedding methods are very effective at predicting values for functional relations, e.g., place of birth and still perform fairly well for one-to-many relations, e.g., children. In a similar fashion, [56] uses multivariate prediction techniques to learn new links on a social graph.

The approach proposed in [65] relies on a graph representation for KBs and applies random walks and path ranking methods to discover new facts in large KBs. In a similar fashion [81] mines frequent meta-paths on data graphs, i.e., sequences of data types connected by labeled edges, and uses them to predict links between entities.

In all these approaches, however, the predictions are *opaque*. It is possible to generate predictions, but not to derive general structural knowledge about the data that can explain the reasons why the predictions were made. For example, these approaches will tell us that Michelle Obama most likely lives in Washington, but they will not tell us that this is because her husband lives in Washington and people tend to live in same place as their spouses. Our approach, in contrast, aims at mining explicit logical rules that capture the correlations in the data. These can then be used to derive new facts and also to explain why these facts were derived.

2.2.6 Learning Rules From Hybrid Sources

[24] proposes to learn association rules from hybrid sources (RDBMS and Ontologies) under the OWA. For this purpose, the definition of frequency (and thus of support and confidence) is changed so that unknown statements contribute with half of the weight of the true statements. Another approach [70] makes use of an ontology and a constraint Datalog program. The goal is to learn association rules at different levels of granularity w.r.t. the type hierarchy of the ontology. While these approaches focus more on the benefits of combining hybrid sources, our approach focuses on pure RDFS KBs.

2.2.7 Further Applications of Rule Mining

[58] proposes an algorithm for frequent pattern mining in KBs that uses DL-safe rules. Such KBs can be transformed into a disjunctive Datalog program, which allows seeing patterns as queries. This approach does not mine the Horn rules that we aim at.

Some approaches use rule mining for ontology merging and alignment [27, 80, 94]. The AROMA system [27], for instance, uses association rules on extracted terms to find subsumption relations between classes and properties of different ontologies. Again, these systems do not mine the kind of rules we are interested in.

In [1] association rules and frequency analysis are used to identify and classify common misusage patterns for relations in DBpedia. In the same fashion, [2] applies association rules to find synonym predicates in DBpedia. The matched synonyms are then used for predicate expansion in the spirit of data integration. This is a vital task in manually populated KBs where the users may not use canonical names for relations, or for cases when the data is produced by independent providers. In contrast to our work, these approaches do not mine logical rules, but association rules on the co-occurrence of values.

Since RDF data can be seen as a graph, mining frequent subtrees [21,62] is another related field of research. However, as the URIs of resources in knowledge bases are unique, these techniques are limited to mining frequent combinations of classes.

Several approaches, such as Markov Logic [104] or URDF [88] use Horn rules to perform reasoning. These rules are normally provided by human experts, therefore such approaches could be consumers of the rules we mine with AMIE.

2.3 Mining Model

In this section we introduce AMIE's mining model. This encompasses its language bias and the definitions of the metrics used to evaluate the quality of rules. Moreover, we elaborate on the problem of predicting facts in a KB under the OWA and induce a mining model for this setup.

2.3.1 Language bias

Syntactic bias. We are interested in rules that can make concrete predictions (Section 1.3.4), therefore we force the head variables of the rules to be closed, i.e., rules must be in $\mathcal{L}_{hclosed}$. Still $\mathcal{L}_{hclosed}$ allows for existentially quantified variables in the body of rules. For instance $\mathcal{L}_{hclosed}$ would accept the rule

$$\mathit{isLocatedIn}(y, z) \wedge \mathit{livesIn}(x, y) \Rightarrow \mathit{isCitizenOf}(x, y)$$

which is an extension of the simpler rule

$$\mathit{livesIn}(x, y) \Rightarrow \mathit{isCitizenOf}(x, y)$$

Actually $\mathcal{L}_{hclosed}$ would allow any extension of the second rule that contains atoms with existentially quantified variables (z in the first rule). While some of those rules may still be interesting, the vast majority of them are redundant extensions of closed rules. Therefore we restrict the language bias of AMIE to $\mathcal{L}_{closed}$.

We do not explore reflexive atoms, e.g., $r(x, x)$, since they are of little use in our setting. Moreover, we enforce the atoms in a rule to be distinct. Thus, AMIE actually applies the language $\mathcal{L}_{AMIE} \coloneqq \mathcal{L}_{closed} \wedge \mathcal{L}_{not\text{-}reflex} \wedge \mathcal{L}_{distinct}$. Conversely, we allow recursive rules.

Semantic bias. AMIE's semantic bias evaluates the quality of the rule in the input KB. We rely on statistical evidence and correcteness as criteria to accept or reject a rule. We define metrics to evaluate those dimensions and enforce minimum thresholds. That is, if a rule is below the given thresholds, it is rejected. The next sections are devoted to explain the metrics applied by AMIE.

2.3.2 Measures of Significance

Normally, data mining systems define a notion of significance or *support* for rules, which quantifies the amount of evidence for the rule in the data. If a rule applies only to a few instances, it is too risky to use it to draw conclusions. For this reason, data mining systems frequently report only rules above a given support threshold. In the following, we define this metric for AMIE's setting and introduce another notion of significance, the *head coverage*.

2.3.2.1 Support

In our context, the support of a rule quantifies the number of correct predictions in the existing data. One desired property for support is *monotonicity*, that is, the addition of more atoms and constraints to the rule should always decrease its support. As we will show in Section 2.5, such property is crucial for pruning. There are several ways to define the support : it can be the number of instantiations of a rule that appear in the KB. This measure, however, is not monotonic if we add atoms to the body. Consider, for example, the rule

$$\textit{livesIn}(x, y) \Rightarrow \textit{wasBornIn}(x, y)$$

If we add the atom *hasGender*$(x,\ male)$ to the body, the number of instantiations x, y in the KB decreases. In contrast, if we add an atom with a fresh variable, e.g., *hasFriend*(x, z), to the body, the number of instantiations x, y, z increases for every friend of x. This is true even if we add another atom with z to obtain a closed rule. Alternatively, we can count the number of facts in one particular body atom. Under this definition, however, the same rule can have different support values depending on the selected body atom. We can also count the number of facts of the head atom. This measure decreases monotonically if more body atoms are added and avoids equivalent rules with different support values. With this in mind, we define the support of a rule as the number of distinct pairs of subjects and objects in the head of all instantiations that appear in the KB :

$$supp(\boldsymbol{B} \Rightarrow H) \coloneqq \#(vars(H)) : \exists z_1, ..., z_m : \boldsymbol{B} \wedge H$$

$livesIn$	$wasBornIn$
(Jean, Paris)	(Jean, Paris)
(Thomas, Munich)	(Thomas, Munich)
(Antoine, Paris)	(Antoine, Colmar)
(Danai, Marseille)	

TABLE 2.2 – An example KB containing two relations between people and cities.

where $vars(H)$ is the set of variables occurring in the head of the rule and $\{z_1, ..., z_m\} = vars(\boldsymbol{B}) - vars(H)$ are the variables of the rule apart from the head variables. For our example rule the formula becomes :

$$supp(livesIn(x, y) \Rightarrow wasBornIn(x, y)) := \#(x, y) : livesIn(x, y) \wedge wasBornIn(x, y)$$

Table 2.2 shows an example KB with 2 relations and 7 facts. For this KB, our example rule has support 2, because of the cases :

1. $livesIn(Jean, Paris)$, $wasBornIn(Jean, Paris)$
2. $livesIn(Thomas, Munich)$, $wasBornIn(Thomas, Munich)$

Note that the support is defined even for rules that are not yet closed. This allows for early pruning of unpromising candidate rules. For example, consider the rule

$$\textit{livesIn}(x, y) \Rightarrow \textit{wasBornIn}(y, z)$$

This rule is obviously unpromising, because it postulates a birth place for y, which is not a person. The rule is not yet closed (x and z appear only once). Yet, it has support 0. Thus, it can be pruned away and does not need further refinement.

2.3.2.2 Head Coverage

Support is an absolute number. This means that a user has to know the absolute size of the KB in order to define a meaningful threshold. Moreover, if the support threshold is higher than the size of some relation, this relation will be disregarded as head relation for rule mining. To avoid this, we propose a proportional version of support. A naive way would be to use the absolute number of support (as defined in the previous paragraph) over the size of the KB. This definition, however, does not solve the problem for small relations. Therefore, we propose to use the notion of *head coverage* :

$$hc(\boldsymbol{B} \Rightarrow H) := \frac{supp(\boldsymbol{B} \Rightarrow H)}{size(r)}$$

with $size(r) := \#(x', y') : r(x', y')$ denoting the number of facts in relation r. Head coverage quantifies the ratio of the known true facts that are implied by the rule. For the example KB presented in Table 2.2, $hc(livesIn(x, y) \Rightarrow wasBornIn(x, y)) = {}^2/_3$, because the rule predicts 2 out of the 3 facts in the head relation $wasBornIn$.

2.3.3 Measures of Correctness

The support of a rule quantifies the number of known correct predictions of the rule. However, it does not take into account the false predictions of the rule. We will now describe measures that judge the *quality* of a rule. Since KBs do not encode negative information, all existing measures of quality assume counter-evidence for rules in one way or another. We first discuss two existing approaches : the standard confidence and positives-only learning function [113]. Then, we introduce our own measure : the confidence under the assumption of partial completeness.

2.3.3.1 The CWA and Standard Confidence

The standard confidence measure takes all facts that are not in the KB as negative evidence. Thus, the standard confidence of a rule is the ratio of its predictions that are in the KB :

$$conf(\boldsymbol{B} \Rightarrow H) := \frac{supp(\boldsymbol{B} \Rightarrow H)}{\#(vars(H)) : \exists z_1, ..., z_m : \boldsymbol{B}} \tag{2.1}$$

For example, consider again the KB given in Table 2.2 together with the rule

$$livesIn(x, y) \Rightarrow wasBornIn(x, y)$$

This rule has 50% confidence, i.e., $conf(livesIn(x, y) \Rightarrow wasBornIn(x, y)) = {}^2/_4$, because (1) there are two positive examples for the rule, namely $wasBornIn(Jean, Paris)$, $wasBornIn(Thomas, Munich)$ and (2) the predictions $wasBornIn(Antoine, Paris)$ and $wasBornIn(Danai, Marseille)$ are counted as negative examples since they do not appear in the KB.

Standard confidence is the measure traditionally used in association rule mining and market basket analysis, where the Closed World Assumption (CWA) is used : if there is no evidence in any of the transactions of the database that a user bought a specific product, then this user did not buy the product. Albeit natural for the market basket analysis scenario, standard confidence fails to distinguish between "false" and "unknown" facts, which makes it inappropriate for a scenario with Open World semantics like ours. Moreover, we also pursue a different goal than market basket analysis : we aim to maximize the number of true predictions that go beyond the current knowledge, whereas market basket analysis usually tries to mine rules that can describe data that is already known. In that scenario is natural to assume that predictions outside the database are false. In contrast, in our example the standard confidence punishes the rule for inferring the fact $wasBornIn(Danai, Marseille)$, even though our goal is to predict it.

2.3.3.2 Positives-Only Learning

For cases where the KB does not contain negative examples, Muggleton has developed a *positives-only evaluation score* for ILP [75,87]. It takes random facts as negative evidence :

$$Score := \log(P) + \log\frac{R_{size} + 2}{R + 1} - \frac{L}{P} \tag{2.2}$$

Here, P is the number of known true facts covered, R is the number of random examples covered, R_{size} is the total number of random examples, and L is the number of atoms in the hypothesis. The intuition is that a good rule should cover many positive examples, and few or no randomly generated examples. This ensures that the rule is not overly general. Furthermore, in the spirit of Occam's razor[3] the rule should use as few atoms as possible, and thus achieve the best possible compression. This measure is implemented (among others) in the ALEPH system.

The disadvantage of this measure is that it "guesses" negative examples at random, whereas rules usually create false predictions in a non-random way. Even if a rule produces many false predictions, the intersection of these false predictions and the random counter-examples may be very small. Consider again our example rule $\mathit{livesIn}(x, y) \Rightarrow \mathit{wasBornIn}(x, y)$, which produces false predictions for persons who have moved to a different place during their life. By considering random person-location pairs as counter-examples, we might not produce any case for which the rule will give a false prediction. In the example KB from Table 2.2, this method may fail at generating the counter-example $\mathit{wasBornIn}(\mathit{Antoine}, \mathit{Paris})$ simply because such a negative example will have a relatively small probability to be generated. This phenomenon becomes more prevalent as the size of the KB increases. We remark however, that the second term in Muggleton's formula tries to alleviate this phenomenon by giving more weight to scenarios with bigger random samples (R_{size}). In particular, if the set of random negative examples contains all possible missing facts for the head relation, the formula behaves similarly to the standard confidence (plus a penalizing factor for longer rules). In Section 2.6, we compare this measure to our proposed metric, which we introduce next.

2.4 The Partial Completeness Assumption

In AMIE, we generate negative examples for a rule by means of the *Partial Completeness Assumption* (PCA). The PCA is the assumption that

$$r(x, y) \in \mathcal{K} \Rightarrow \forall\, y' : r(x, y') \notin \mathcal{K} : r(x, y') \text{ is false}$$

In other words, we assume that if we know at least one y for a given x and r, then we know all y that hold in reality for that x and r. This assumption allow us to generate counter-examples in a way that is less restrictive than the standard confidence. In our example from Table 2.2, the PCA will assume that any other place of birth for Jean, Thomas and Antoine is false. Conversely, the PCA will not assume anything about the place of birth of Danai, because the KB does not know any. With this notion in mind, we define a new notion of confidence for rules. Under the PCA, the denominator of the confidence formula is not the size of the entire set of conclusions derived from the body of the rule, but the number of facts that we know to be true together with the facts that

3. Among competing hypotheses, the one with the fewest assumptions should be selected

we assume to be false. The PCA is defined according to the following formula :

$$conf_{pca}(\boldsymbol{B} \Rightarrow r(x,y)) := \frac{supp(\boldsymbol{B} \Rightarrow H)}{\#(vars(H)) : \exists z_1, ..., z_m, y' : \boldsymbol{B} \wedge r(x,y')} \tag{2.3}$$

This formula is defined only for head-closed rules and is also applicable when the head atom contains a constant, i.e., for rules of the form $\boldsymbol{B} \Rightarrow r(x,C)$. The PCA confidence normalizes the support of the rule by the number of bindings of the head variables for which there exists a y' with $r(x,y')$. Consider again the KB given in Table 2.2. In this example KB, $conf_{pca}(livesIn(x,y) \Rightarrow wasBornIn(x,y)) = {}^2/_3$. This is because (a) there are 2 positive examples for the rule, $wasBornIn(Jean, Paris)$, $wasBornIn(Thomas, Munich)$, and (b) the prediction $wasBornIn(Antoine, Paris)$ is counted as negative example, because we already know a different place of birth for Antoine. The prediction $wasBornIn(Danai, Marseille)$ is completely disregarded as evidence, because we neither know where Danai was born nor where she was not born.

We notice that Equation 2.3 fixes x and r and implies that rules will try to always predict values for y. In Section 1.3.3 we re-write all relations so that their functionality is larger than their inverse functionality, hence, the most functional direction will be always to predict y given x. To see why this makes sense, recall that it is more intuitive to predict the birthplace of a specific person than predict all the people that were born in a specific city. For this reason, AMIE always predicts in the most functional direction.

In spite of being an assumption, the PCA is certainly true for functions, such as *birthdate* and *gender*. The PCA also holds for relations that are not functions but that have a high functionality, as we shall see in our qualitative analysis of the PCA in next section. The PCA has been picked up by the Google Knowledge Vault [32] and used under the name “local completeness assumption”.

2.4.1 The PCA in real data

The PCA is one of the basic ingredients of AMIE's mining model, therefore studying its accuracy is vital for the purpose of this chapter. In this section we investigate to what extent the PCA holds in a real-world KB. We chose the YAGO2 [55] KB.

Setup. We looked into each of the 31 relations between entities in YAGO2. For each relation r, we randomly sampled 30 subjects. For each subject x, we checked whether the KB knows all y with $r(x,y)$. We measured precision as the ratio of entities in the sample for which the PCA holds, i.e., the ratio of entities for which there are no additional y values in reality. If the relation is more inverse functional than functional ($ifun(r) > fun(r)$, see Section 1.3.3), we considered r^{-1} instead.

As ground truth, we took the Wikipedia page of x and what we could find on the Web by a search engine. It is obvious that such an evaluation cannot be done strictly quantitatively. For example, a person might have worked for a company, but this fact might appear nowhere on Wikipedia – or even on the Web. Or a musician might play 10 instruments at different levels of proficiency, but Wikipedia mentions only the 4 main instruments. Even a search on the Web might not tell us that there are more than 4 instruments. Therefore, we resorted to a qualitative analysis. We analyzed each of the relations manually, and grouped the relations into categories. Some relations fall into multiple categories.

Functions and Quasi-Functions. Table 2.3 shows our results for functions and quasi-functions. By definition, the PCA holds for functions. Our manual analysis, however, did not result in 100% precision for functional relations in Table 2.3. This is because our analysis also counts the cases where the KB contains bugs. If, for instance, YAGO knows the wrong place of death of a person, then there exists another value outside YAGO that is the right value. However the PCA would reject it. Hence, we count this case as a miss. Nevertheless, bugs are rare in YAGO since it guarantees a precision of 95% precision [117].

The PCA extends well to relations that are strictly speaking not functions, but that have a high functionality. These are relations that usually have one object per subject, even though there could be several objects. For example, a person can graduate from several universities, but most people graduate from a single university. We call these relations *quasi-functions*. As shown in Table 2.4, the PCA worked very well also on these, and predicted completeness correctly for $73\% - 100\%$ of the subjects under investigation. Since the PCA takes into account the direction of the functionality, the PCA also holds for quasi inverse-functional relations such as *directed*.

Relation	% of hits
wasBornIn	96.67
diedIn	96.42
hasCapital	93.33

TABLE 2.3 – Validity of the PCA for functions in YAGO2

Relation	% of hits
hasCurrency	75.00
hasOfficialLanguage	73.33
graduatedFrom	64.29
isCitizenOf	96.42
directed^{-1}	90.00
hasAcademicAdvisor	88.89
created^{-1}	86.67
isLeaderOf	89.47
isPoliticianOf	100
isAffiliatedTo	89.47

TABLE 2.4 – Validity of the PCA for quasi-functions in YAGO2

Granularity Differences. Some relations, such as *locatedIn* and *livesIn* (Table 2.5), hold between an entity and a geographical region. In that case, the region can be given at the granularity of a city, a region, a country, or a continent. Naturally, if YAGO contains one of these, the others are possible options. Hence, the PCA fails and we find rather low precision values. However, these cases could be addressed if one restricts the range of the relation (say, to cities). With such a restriction, the relations become functions or quasi-functions, which lifts them into the category where the PCA works well. As we will see in Section 2.7.1, the use of types can significantly improve the performance of AMIE for facts inference.

Implicit Assumptions. Some statements can be inferred from the Wikipedia page even if the page does not mention them. People usually do not state information that can easily be inferred by what they have stated before (following Grice's Maxim of quantity and manner [48]). For example, if someone graduated from a university, people usually do not feel obliged to mention that this person used to live in the country in which the university is located, because this can easily be inferred by the reader. Only less obvious residences will be explicitly mentioned. Therefore, the PCA does not always hold. Note that rules such as $graduatedFrom(x, z) \wedge isLocatedIn(z, y) \Rightarrow livesIn(x, y)$ can only be mined if Grice's maxims are occasionally violated by the authors of the articles. If the authors always follow the maxims, then such rules cannot be mined, because there are not even positive examples for which the rule holds (for lack of support). In the case of YAGO, the only relation that we found in this category is *livesIn*, for which the PCA held in 20.83% of the studied cases.

Relation	% of hits
isLocatedIn	50.00
livesIn	20.83

TABLE 2.5 – Validity of the PCA for relations suffering from granularity differences in YAGO2

Source Incompleteness. For many relations, the source itself (Wikipedia) is incomplete (see Table 2.6). Usually, these relations have, for each subject, some objects that are undisputed. For example, it is undisputed that Albert Einstein is interested in physics. However, these relations also have objects that are less important, disputed, or unknown. For example, Albert Einstein might also be interested in music (he played the violin), but maybe also in pancakes. These less prominent objects are a lot less likely to appear in Wikipedia, or indeed on any Web page. Even if they do, we can never be sure whether there is still something else that Einstein was interested in. For these relations, the knowledge sources are often incomplete by nature. For example, not every single product that a country imports and exports is explicitly mentioned. This phenomenon results in 0% precision for such relations in Table 2.6. The same phenomenon can be observed for the relation $actedIn^{-1}$, since Wikipedia does not include the entire list of actors that played in a movie. Whether or not this poses a problem depends on the application. If ground truth is defined as what is universally true, then source incompleteness is a problem. If ground truth is the source of the KB (i.e., Wikipedia in this case), then source incompleteness is not an issue.

Relation	% of hits
influences^{-1}	34.78
imports	0
exports	0
actedIn^{-1}	0
worksAt	89.66
hasMusicalRole	22.22
dealsWith	10.00

TABLE 2.6 – Validity of the PCA for relations suffering from source incompleteness in YAGO2

Extraction Incompleteness. For a large number of relations, the Wikipedia page contains more objects for a given subject than the KB. These are cases where the extraction process was incomplete. In the case of YAGO, this is due to a strong focus on accuracy, which causes the extraction to discard any extracted fact that cannot be type checked or linked to an entity. This class of relations is the most sensitive category for the PCA. The success of the PCA will depend on how many relations and to what extent they are affected by incomplete extractions. The performance of the PCA for those relations is shown in Table 2.7.

Relation	% of hits
participatedIn^{-1}	48.14
isMarriedTo	79.31
produced^{-1}	56.67
actedIn^{-1}	0
playsFor	20.00
holdsPoliticalPosition	26.67
hasChild^{-1}	26.67
hasWonPrize	31.03
dealsWith	10.00
influences^{-1}	34.78
hasMusicalRole	22.22

TABLE 2.7 – Validity of the PCA for relations suffering from extraction incompleteness in YAGO2

Discussion. In summary, our analysis shows that it depends on the nature of the relation and on its type signature whether the PCA holds or not. There is a large number of relations for which the PCA is reasonable. These are not just functions and inverse functions, but also relations that exhibit a similar behavior.

For many other cases, the PCA does not hold. In these cases, the PCA will falsely assume that a rule is making incorrect predictions – although, in reality, the predictions might be correct. Thus, when the PCA does not hold, we will err on the side of caution. This means that, even if the PCA is not always correct, it is a safe way to underestimate confidence.

At the same time, the PCA is not as restrictive as the Closed World Assumption (CWA) : the PCA admits that there can be facts that are true, but not known to the KB. For example, if a person has a birth date, then both the CWA and PCA would not admit another birth date. However, if a person does not have a birth date, then the PCA will admit that there can be a birth date, while the CWA will assume that there cannot be a birth date. Thus, the PCA is more permissive than the CWA. This encourages us to use the PCA confidence to estimate the correcteness of rules. In Section 2.7.1, we show that this definition of confidence produces more predictive and more accurate rules than the standard confidence, which is based on the CWA.

2.5 AMIE

We now outline the core algorithm of AMIE and its implementation. We follow the description in [43] and extend it with further explanations and details.

2.5.1 Algorithm

Algorithm. Algorithm 1 shows our approach to mine rules. It takes as input a KB $\mathcal{K}$, a maximum rule length l, a threshold $minHC$ on the head coverage of the mined rules, and a threshold $minConf$ on the confidence. We discuss the choice of parameter values later in this section. The algorithm maintains a queue of rules (line 1), which initially contains all possible head atoms, that is, all rules of size 1. It then iteratively dequeues a rule from this queue. If the rule meets certain criteria (line 5), it is pushed to the output. If the rule does not exceed the maximum number of atoms l (line 7), it goes through a refinement process (described below) which expands the rule (the parent) to produce a set of new rules (the children). These new rules, if neither duplicates nor pruned by the head coverage threshold (line 10), are also pushed into the queue. This process is repeated until the queue is empty. In the following, we will see in more detail the different phases of the algorithm.

Refinement. One of the major challenges of rule mining is to find an efficient way to explore the search space. The naive algorithm of enumerating all possible combinations of conjunctions of atoms is infeasible for large KBs. Hence, we explore the search space by iteratively extending rules using a set of *mining operators* (line 8 of Algorithm 1). We

```
Algorithm 1: AMIE
  Input: a KB : $\mathcal{K}$, maximum rule length : $l$, head coverage threshold : $minHC$,
         confidence threshold : $minConf$
  Output: set of Horn rules : $rules$
1 $q = [r_1(x,y), r_2(x,y) \dots r_m(x,y)]$
2 $rules = \langle\rangle$
3 while $\neg q$.isEmpty() do
4     $r = q$.dequeue()
5     if $AcceptedForOutput(r, out, minConf)$ then
6         $rules$.add$(r)$
7     if $length(r) < l$ then
8         $R(r) = Refine(r)$
9         for each rule $r_c \in R(r)$ do
10            if $hc(r_c) \geq minHC \wedge r_c \notin q$ then
11                $q$.enqueue$(r_c)$
12 return $rules$
```

see a rule as a sequence of atoms. The first atom is the head atom and the others are the body atoms. In the process of traversing the search space, we can extend a rule by using one of the following operators :

1. **Add Dangling Atom (**$\mathcal{O}_D$)
   This operator adds a new atom to the body of a rule. The new atom uses a fresh variable for one of its two arguments. The other argument is a variable that is shared with the rule, i.e., it occurs in some other atom of the rule.
2. **Add Instantiated Atom (**$\mathcal{O}_I$)
   This operator adds a new atom to the body of the rule. This atoms uses an entity for one argument and shares the other argument (variable) with the rule.
3. **Add Closing Atom (**$\mathcal{O}_C$)
   This operator adds a new atom with two variables to the body of a rule so that both of its arguments are shared with the rule.

Note that all above operators create connected rules without reflexive atoms. In addition, they do not add atoms that are already in the rule. By repeated application of these operators, we can generate the entire space of rules in the syntactic language $\mathcal{L}_{AMIE}$ (Section 2.3.1) with head coverage greater or equal than $minHC$. In Section 2.5.2 we describe how these operators are implemented and executed on the KB. It is worth mentioning that the operator $\mathcal{O}_D$ always generates non-closed rules (rules outside $\mathcal{L}_{closed}$), therefore AMIE must check, among other requirements, whether a rule is closed before outputing it. We describe this step next.

When to Output. Not every rule that the mining algorithm dequeues is output. This

Algorithm 2: Decide whether to output a rule

```
  Input: a rule : $r$, maximum rule length : $l$, rules output so far : $rules$, confidence
         threshold : $minConf$
  Output: true or false
1 if $r \notin \mathcal{L}_{closed} \vee conf_{pca}(r) < minConf$ then
2     return $false$
3 $parents = parentsOfRule(r, rules)$
4 for each $r_p \in parents$ do
5     if $conf_{pca}(r) \leq conf_{pca}(r_p)$ then
6         return $false$
7 return $true$
```

is because some rules may not be closed, or may not be better than rules that have already been output. Algorithm 2 explains how we decide if a rule should be output or not once it has been dequeued. The refinement operators used by AMIE always produce connected rules. So, at this point, the algorithm only checks if the rule is closed. Then, the algorithm calculates the confidence of the rule and performs a quality check. The rule should have a confidence value that (i) passes the confidence threshold (line 1) and (ii) improves over the confidence of all its parents (line 5). The latter condition implies that the refinements of a rule ($B_1 \wedge ... \wedge B_n \wedge B_{n+1} \Rightarrow H$) must bring some confidence gain with respect to the parent rule ($B_1 \wedge ... \wedge B_n \Rightarrow H$). Since support and head coverage are monotonic metrics, we know that the child rule will never have a higher score than its parent rule. If the child rule has also lower confidence, then its quality is worse in all aspects than the parent rule. Hence, there is no reason to output it.

A rule can have several parents. For example, the rule

$$actedIn(x, y) \wedge directed(x, y) \Rightarrow created(x, y)$$

can be derived by either adding $directed(x, y)$ to $actedIn(x, y) \Rightarrow created(x, y)$ or by adding $actedIn(x, y)$ to $directed(x, y) \Rightarrow created(x, y)$. AMIE requires a confidence gain over all parents of a rule.

Note that the decisions made at this point affect only the output. They do not influence the refinement process, i.e., a rule with low confidence can still be refined to obtain new rules. This is because confidence is a non-monotonic measure, i.e., we might get good rules with further refinement of bad rules.

Duplicate Elimination. As just mentioned, a rule can be derived in multiple ways. For example, the rule $actedIn(x, y) \wedge directed(x, y) \Rightarrow created(x, y)$ can result from the application of the operator $\mathcal{O}_C$ to both $actedIn(x, y) \Rightarrow created(x, y)$ and $directed(x, y) \Rightarrow created(x, y)$. For this reason, AMIE checks for the existence of duplicate rules (line 12) in order to avoid queuing the same rule multiple times. While checking two rules for equality is expensive (it is a graph isomorphism verification task), we observe that two rules can only be equal if they have the same head relation, the same number of atoms and the same head coverage (or support). This reduces drastically the set of rules that have to be checked and therefore the time invested in this task.

Multithreading. To speed up the process, our implementation parallelizes Algorithm 1, that is, the main loop (lines 4 to 17) runs in multiple threads. This is achieved by synchronizing the access to the centralized queue from which the threads dequeue and enqueue and the access to the output.

Parameters and Pruning. If executed naively, Algorithm 1 will have prohibitively high runtimes. The instantiation operator $\mathcal{O}_I$, in particular, generates atoms in the order of $|\mathcal{R}| \times |\mathcal{E} \cup \mathcal{L}|$. For this reason the algorithm defines some parameters that determine when to stop with the exploration of the space. These are the minimal head coverage $minHC$, the maximal length l and the minimal confidence $minConf$. Choosing larger thresholds on head coverage, and choosing a shorter maximum rule length will make the algorithm stop earlier and output fewer rules. Relaxing the values will make the algorithm output the very same rules as before, and find also rules with a smaller head coverage or a larger number of atoms. Thus, these parameters define a trade-off between the runtime and the number of rules.

Interestingly, a larger number of rules is not necessarily a good thing. For instance, a rule that covers only 1% or less of the instances of a relation is probably not interesting. It simply lacks statistical significance. Assuming that a user is not interested in such spurious rules, we set $minHC = 0.01$ by default.

The operator $\mathcal{O}_I$ can drastically increase the size of the search space and the output for certain types of atoms. For instance, an atom of the form $livesIn(x, C)$ may produce a number of candidates proportional to the number of cities in the KB. By default $\mathcal{O}_I$ is disabled in AMIE, even though it can be enabled by the user. In Section 2.6 we show the performance of AMIE with and without this feature.

Additionally, we show in our experiments that rules with more than 3 atoms tend to be very convoluted and not insightful. Hence, we set $l = 3$ by default.

Likewise, rules with low confidence will not be of much use to the application. For example, a rule with confidence 10% will make correct predictions in only one out of ten cases. Assuming that a user is not interested in such kind of rules, we set $minConf = 0.1$ by default.

While the user can change these thresholds, we believe that there is no good reason to deviate from the default values. In particular, relaxing these values will not output better rules. This makes AMIE a system that can be run off the shelf, without the need for parameter tuning.

2.5.2 Mining Operators

AMIE tries to expand a given rule by applying all mining operators defined in the last section. We now explain how the operators are implemented and executed on a KB.

Count Projection Queries. Assume that AMIE needs to add the atom $r(x, y)$ to a rule $B_1 \wedge ... \wedge B_{n-1} \Rightarrow H$. For efficiency reasons, we do not blindly try all possible relations in the place of r. Instead, we first find all relations that lead to a new rule that passes the head-coverage threshold. In other words, we first fire a *count projection query* of the form

SELECT $\boldsymbol{r}$, COUNT(H)
WHERE $H \wedge B_1 \wedge ... \wedge B_{n-1} \ \wedge \ \boldsymbol{r}(\boldsymbol{X}, \boldsymbol{Y})$
SUCH THAT COUNT(H)$\geq k$

where $k := minHC \times size(H)$ (see Section 2.3.2.2) is the translation of the head coverage threshold into an absolute support threshold and the expression COUNT($\cdot$) has COUNT(DISTINCT $\cdot$) semantics (also for the rest of this section). The placeholders $\boldsymbol{X}$ and $\boldsymbol{Y}$ can represent constants or variables that are either fresh or already present in the rule. The results for $\boldsymbol{r}$ are the relations that, once bound in the query, ensure that the head coverage of the rule $B_1 \ \wedge ... \wedge B_{n-1} \ \wedge \ \boldsymbol{r}(\boldsymbol{X}, \boldsymbol{Y}) \Rightarrow H$ is greater or equal than $minHC$. Notice also that for each value of $\boldsymbol{r}$, the expression COUNT(H) gives us the support of the new rule. We now discuss the instantiation of this query for all three operators.

Dangling Atom Operator. As an example, assume that Algorithm 1 dequeues the following intermediate non-closed rule for further specialization :

$$\textit{marriedTo}(x, z) \Rightarrow \textit{livesIn}(x, y)$$

The application of the operator $\mathcal{O}_D$ will fire queries of the form :

SELECT $\boldsymbol{r}$, COUNT($livesIn(x, y)$)
WHERE $livesIn(x, y) \ \wedge \ marriedTo(x, z) \wedge \boldsymbol{r}(\boldsymbol{X}, \boldsymbol{Y})$
SUCH THAT COUNT($livesIn(x, y)$)$\geq k$

with $\boldsymbol{r}(\boldsymbol{X}, \boldsymbol{Y}) \in \{\boldsymbol{r}(y, w), \boldsymbol{r}(z, w), \boldsymbol{r}(w, y), \boldsymbol{r}(w, z)\}$. That is, $\boldsymbol{r}(\boldsymbol{X}, \boldsymbol{Y})$ binds to each possible join combination of a new dangling atom, where w is an arbitrary fresh variable. For intermediate rules, dangling atoms are joined on the non-closed variables ; z and y in this example. If the rule is already closed, dangling atoms are joined on all the variables appearing in the rule.

Closing Atom Operator. The $\mathcal{O}_C$ operator works in the same fashion. In our example, the atom $\boldsymbol{r}(\boldsymbol{X}, \boldsymbol{Y})$ can take values in $\{\boldsymbol{r}(z, y), \boldsymbol{r}(y, z)\}$. The method will produce new atoms so that all non-closed variables are closed. In this example, the method produces the minimum number of specializations required to close the variables y and z. If there is only one closed variable, the method will produce atoms between the open variable and all the other variables. If the rule is already closed, the operator tries with all possible pairs of variables in the rule.

Instantiated Atom Operator. The operator $\mathcal{O}_{\mathcal{I}}$ is implemented in two steps. We first apply the operator $\mathcal{O}_{\mathcal{D}}$ to produce a set of intermediate rules with a new dangling atom and a new fresh variable. Then for each rule, we fire a count-projection query on the fresh variable. This step provides bindings for one of the arguments of the relation. For instance, the application of the $\mathcal{O}_{\mathcal{I}}$ operator to our example rule

$$\textit{marriedTo}(x, z) \Rightarrow \textit{livesIn}(x, y)$$

will first add all possible dangling atoms to the rule. Let us consider one group of such atoms, e.g., those of the form $\boldsymbol{r}(x, w)$. Then for each value of $\boldsymbol{r}$ that keeps the rule above the head coverage threshold $minHC$, the algorithm finds the best bindings for w. For example, imagine we bind $\boldsymbol{r}$ to the relation $citizenOf$. The second step will fire a query of the form :

SELECT $\boldsymbol{w}$, COUNT($livesIn(x, y)$)
WHERE $livesIn(x, y) \wedge marriedTo(x, z) \wedge citizenOf(x, \boldsymbol{w})$
SUCH THAT COUNT($livesIn(x, y)$)$\geq k$

Each binding of $\boldsymbol{w}$ forms a new rule that will be enqueued and later evaluated for output. In some cases, however, AMIE may exclude some of the newly produced rules. This happens when the join variable of the instantiated atom binds only to one entity in the KB. Such cases induce a *quasi-binding* of the join variable, meaning that there is a shorter equivalent rule where the variable has been replaced by the constant. For example, imagine AMIE applies the $\mathcal{O}_I$ operator to the following rule :

$$neighbour(x, y) \Rightarrow dealsWith(x, y)$$

One way to extend the rule is to add the atom $officialLanguage(y, Romansh)$, which leads to :

$$officialLanguage(y, Romansh) \wedge neighbour(x, y) \Rightarrow dealsWith(x, y)$$

This rule is not interesting because y can bind only to *Switzerland*, even if the support of the rule surpasses the given threshold. AMIE does not output this rule because it is equivalent to the simpler rule :

$$neighbour(x, Switzerland) \Rightarrow dealsWith(x, Switzerland)$$

As stated in Section 7, AMIE disables the $\mathcal{O}_I$ operator by default. Its activation changes the initialization step of Algorithm 1. In particular for every atom $r(x, y)$ in the queue, AMIE will also add atoms of the form $r(x, C)$, for every constant C such that the head coverage threshold is met, i.e.,

$$\frac{\#x : r(x, C)}{size(r)} \geq minHC$$

This allows AMIE to mine rules with instantiated atoms in the head, like in our last example. Note that in such cases, both support and confidence are counted on a single variable.

Count-projection queries allow us to choose the relationships and entities for the operators in such a way that the head coverage for the new rules is above $minHC$.

2.5.3 Confidence calculation

Recall from Formula 2.3 that given the support of a rule $\boldsymbol{B} \Rightarrow H$, the calculation of the PCA confidence requires to compute the number of bindings for

$$\#vars(H) : \exists z_1, \dots z_n, y' : \boldsymbol{B} \wedge H'$$

In this expression, H' is a variant of the head H where the least functional argument (see Section 1.3.3) has been replaced by a fresh variable y', i.e., $H' := r_h(x, y')$. For our refined rules, such expression translates into a count query of the form

SELECT COUNT(H) WHERE $H' \wedge B_1 \wedge \cdots \wedge B_n$

Likewise, the calculation of the standard confidence (Formula 2.1) relies on the calculation of the expression $\#vars(H) : \exists z_1, \dots z_n, y' : \boldsymbol{B}$; what we call the *body size*. This translates into a query of the form

SELECT COUNT(H) WHERE $B_1 \wedge \cdots \wedge B_n$

We observe that count queries are actually a less-constrained case of the count-projection queries already introduced. Thus, we focus on the implementation of count-projection queries. This is addressed in the next section.

2.5.4 Query Implementation Details

AMIE relies on *count-projection queries* to determine the relations and instances for new atoms in rules and their support. For a rule $B_1 \wedge ... \wedge B_{n-1} \Rightarrow H$, count-projection queries have the form :

SELECT $\boldsymbol{r}$, COUNT(H)
WHERE $H \wedge B_1 \wedge ... \wedge B_{n-1} \wedge \boldsymbol{r}(\boldsymbol{X}, \boldsymbol{Y})$
SUCH THAT COUNT(H)$\geq k$

where each answer corresponds to a relation r and the support of the refined rule $B_1 \wedge ... \wedge B_{n-1} \wedge r(\boldsymbol{X}, \boldsymbol{Y}) \Rightarrow H$.

SQL and SPARQL. Count-projection queries are essential for the efficiency of our system. Yet, standard database implementations do not provide special support for these types of queries. Assuming that the KB $\mathcal{K}$ is stored as a three-columns table, i.e., each fact is a row with three columns $\langle s, r, o \rangle$, the count-projection query template in SQL would be :

```
SELECT $\mathcal{C}.\boldsymbol{c}$, COUNT($*$) FROM $\mathcal{K}$ AS H,
(SELECT DISTINCT $\boldsymbol{c}$ FROM $\mathcal{K}$) AS $\mathcal{C}$
WHERE $\boldsymbol{P}(H)$ AND EXISTS
  (SELECT * FROM $\mathcal{K}$ AS $B_1, \ldots,$ $\mathcal{K}$ AS $B_n$
   WHERE $\boldsymbol{P'}(H, B_1, \ldots B_n)$)
GROUP BY $\mathcal{C}.\boldsymbol{c}$ HAVING COUNT($*$) >= $k$
```

Here, B_n is the new atom added to the query and $\boldsymbol{c} \in \{s, r, o\}$. The temporary table $\mathcal{C}$ contains the target values for the new atom, e.g., relations if $\boldsymbol{c} = r$. The function $\boldsymbol{P}(H)$ represents the conditions imposed by the head atom, e.g., for a rule of the form $\boldsymbol{B} \Rightarrow \mathit{livesIn}(x, \mathit{USA})$, $\boldsymbol{P}(H)$ is translated into $H.r$ = "$\mathit{livesIn}$" AND $H.o$ = "USA". Likewise, the function $\boldsymbol{P'}(H, B_1, \ldots B_n)$ is replaced by the join conditions between the head and body atoms. If we apply this scheme to the addition of a closing atom $\boldsymbol{r}(z, y)$ in our example rule *marriedTo*$(x, z) \Rightarrow$ *livesIn*(x, y), we would generate the query

```
SELECT $\mathcal{R}.r$, COUNT($*$) FROM $\mathcal{K}$ AS H,
(SELECT DISTINCT $r$ FROM $\mathcal{K}$) AS $\mathcal{R}$
WHERE $H.r$ = "$\mathit{livesIn}$" AND EXISTS
  (SELECT * FROM $\mathcal{K}$ AS $B_1$, $\mathcal{K}$ AS $B_2$,
  WHERE $H.s = B_1.s$ AND $H.o = B_2.o$
    AND $B_2.s = B_1.o$ AND $B1.r$ = "$\mathit{marriedTo}$"
    AND $B_2.r = \mathcal{R}.r$)
GROUP BY $\mathcal{R}.r$ HAVING COUNT($*$) >= $k$
```

The query stores all possible relations for the new atom in the temporary table $\mathcal{R}$ and applies a semi-join with the head atom H. The semi-join is constrained to those bindings of $\mathcal{R}$ for which there exists at least one binding in the rule. The results are then grouped (GROUP BY), aggregated (COUNT(*)) and thresholded (HAVING) per binding of the new atom. Our experience shows that for a KB of a few million facts, such kind of queries can easily take several minutes on an off-the-shelf relational database management system (RDBMS). Hence, efficient SPARQL engines such as RDF-3X [91] are an alternative option. In SPARQL 1.1, the count-projection query template is :

```
SELECT $\boldsymbol{c}$, SUM($?supp$) WHERE {
   SELECT $\boldsymbol{c}$ (COUNT(DISTINCT *) AS $?supp$)
   WHERE {
     SELECT $\boldsymbol{c}$ $?s_H$ $?o_H$ WHERE {
       $?s_H$ $r_H$ $?o_H$ .
       $?s_1$ $r_1$ $?o_1$ .
       ...
       $?s_n$ $?r_n$ $?o_n$ .
     }
  } GROUP BY $\boldsymbol{c}$ $?s_H$ $?o_H$
} GROUP BY $\boldsymbol{c}$ HAVING SUM($?supp$) >= $k$
```

where $\boldsymbol{c} \in \{?s_n, ?r_n, ?o_n\}$ is a variable associated to new atom added to the rule and the triple pattern $\langle ?s_H \; r_H \; ?o_H \rangle$ corresponds to the head atom. This particular template models the case where the head atom does not contain constants. If this is not the case, as in $\boldsymbol{B} \Rightarrow \mathit{livesIn}(x, \mathit{USA})$, we simply replace the argument variables $?s_H$ or $?o_H$ by the corresponding constant and remove it from the GROUP BY condition. As grouping and aggregate functions have only recently become part of the SPARQL 1.1 standard, many existing SPARQL engines do not yet efficiently support the extension. In particular, RDF-3X does not support aggregate functions in this way. Thus, we would need extensive postprocessing of query results to compute a projection query.

Since neither relational databases nor RDF triple stores offer a satisfactory solution for count-projection queries, we resort to a custom implementation.

In-Memory Database. We have implemented an in-memory database that is specifically geared towards count-projection queries. Our implementation indexes the facts aggressively with one index for each permutation of the columns subject (`S`), relation (`R`), and object (`O`). This means that there are six indexes, namely `SRO`, `SOR`, `RSO`, `ROS`, `OSR` and `ORS`. We call them *fact indexes*. Each fact index is a hash table, which maps elements of the first column to a nested hash table. This nested hash table maps elements of the second column to a set of elements of the third column. For example, the index `ORS` has as keys the objects of all triples in the KB. It maps each object o to a hash table. This hash table has as keys all possible relations of the KB. It maps each relation r to a set of subjects $\{s_1, ..., s_n\}$, such that $r(s_i, o)$ for $i = 1...n$. Fact indexes allow us to check the existence of a triple in constant time. They also allow us to efficiently fetch the instantiations of an atom.

In addition to the fact indexes, our database relies on three *aggregated indexes* `S`, `P`, `O`. These store the aggregated number of facts for each key of the fact indexes. For example, the aggregated index `P` stores the number of triples for each relation in the KB, whereas the aggregated index `S` stores the number of triples where each entity appears as subject.

Size Queries. Fact indexes in combination with aggregated indexes can be used to determine the *size of an atom* ($\mathit{size}(a, \mathcal{K})$), i.e., its number of bindings in the KB $\mathcal{K}$. We call this operation a *size query*. Size queries are the building blocks for the implementation of more complex types of queries and can be used to calculate the size of relations and the support of single atom queries. For example, the size (or support) of the atom $\mathit{livesIn}(x, y)$ can be retrieved by a simple look-up in the aggregated index `P`. The size of the atom $\mathit{livesIn}(x, \mathit{USA})$ requires two lookups in the fact index `ROS` : the first lookup to get the object values of $\mathit{livesIn}$ and the second to retrieve the list of subjects for the object value *USA*.

Existence Queries. One of the central tasks of the in-memory database is to determine whether there exists a binding for a conjunctive query. Algorithm 3 shows how this can be implemented. The algorithm requires as input a conjunctive query and a KB $\mathcal{K}$. If the query is a single atom (line 2), we can directly verify if its size is greater than zero using the indexes (line 3). Otherwise, we select the atom B_s with fewest instantiations using the indexes (line 5), and run through all of its instantiations (lines 7 to 10). We apply such instantiations to the remaining atoms (line 8) and repeat this process recursively (line 9) until we end up with a single atom. Since rules are connected query patterns, the atom B_s must share at least one variable with the remaining atoms. This means that by instantiating B_s, some variables in the remaining atoms become instantiated, making the atoms more selective with every recursive step.

Algorithm 3: Existence Queries

```
    Input: a query : $B_1 \wedge ... \wedge B_n$, a KB : $\mathcal{K}$
    Output: true or false
 1  $q := B_1 \wedge ... \wedge B_n$
 2  if $n = 1$ then
 3      return size($B_1$, $\mathcal{K}$) $> 0$
 4  else
 5      $s := argmin_i \{size(B_i, \mathcal{K})\}$
 6      $q := q \setminus \{B_s\}$
 7      for each instantiation $\sigma_s \in B_s$ do
 8          $q' := \sigma_s(q)$
 9          if Exists($q'$, $\mathcal{K}$) then
10              return $true$
11  return $false$
```

Select Queries. Algorithm 4 describes the implementation of SELECT DISTINCT queries on one projection variable for a conjunction of atoms. The algorithm starts finding the atom with the fewest number of instantiations B_s. If the projection variable x is in B_s (lines 4 to 8), the algorithm goes through all the instantiations $\hat{x}$ of variable x, instantiates the query accordingly and checks whether there exists a solution for the instantiated query pattern in the KB (line 7). If there is, the solution $\hat{x}$ is added to the result set. In contrast, if the projection variable is not in the most restrictive atom B_s (lines 10 to 13), the algorithm iterates through the instantiations of B_s and recursively selects the distinct bindings of x in the remaining atoms (line 13).

Count Queries. To compute the PCA confidence of a rule $\boldsymbol{B} \Rightarrow r_h(x, y)$ or $\boldsymbol{B} \Rightarrow r_h(x, C)$ for some constant C, AMIE must fire a *count query* to estimate the denominator of the confidence formula. For the PCA confidence, such queries have the form :

SELECT COUNT(x, y) WHERE $r_h(x, y') \wedge \boldsymbol{B}$

SELECT COUNT(x) WHERE $r_h(x, y') \wedge \boldsymbol{B}$

depending on the number of variables of the head atom. $\boldsymbol{B} = B_1 \wedge B_2 \ldots \wedge B_n$ are the body atoms, and $r_h(x, y')$ is a variant of the head atom where the least functional

Algorithm 4: Select Distinct Queries

```
   Input:  a variable : $\boldsymbol{x}$, a query : $B_1 \wedge ... \wedge B_n$, a KB : $\mathcal{K}$
   Output: Set of bindings for variable $\boldsymbol{x}$ that match the query
 1 $q := B_1 \wedge ... \wedge B_n$
 2 $s := argmin_i \; \{size(B_i, \mathcal{K})\}$
 3 $result := \{\}$
 4 if $\boldsymbol{x} \in B_s$ then
 5     for each instantiation $\hat{x} \in \boldsymbol{x}$ do
 6         $q' := q$ instantiated with $\boldsymbol{x} \leftarrow \hat{x}$
 7         if Exists($q'$, $\mathcal{K}$) then
 8             $result$.add($\hat{x}$)
 9 else
10     $q := q \setminus \{B_s\}$
11     for each instantiation $\sigma_s \in B_s$ do
12         $q' := \sigma_s(q)$
13         $result.add(Select(\boldsymbol{x}, q', \mathcal{K}))$
14 return $result$
```

argument has been replaced by a fresh variable y' (see Section 2.4). These queries return the number of distinct bindings of the head variables that fulfill the pattern $\boldsymbol{B} \wedge r(x, y')$. They are used to calculate the confidence of rules. For the two variables case, the in-memory database first fires a SELECT query on variable x :

SELECT DISTINCT x WHERE $\boldsymbol{B} \wedge r(x, y')$

Then, for each binding of x, it instantiates the query and fires another select query on variable y, adding up the number of instantiations.

Count Projection Queries. Count projection queries take the form

SELECT $\boldsymbol{x}$, COUNT(H)
WHERE $H \wedge B_1 \wedge ... \wedge B_n$
SUCH THAT COUNT(H)$\geq k$

These are the types of queries used to determine the relations and instances for new atoms in the refinement phase of AMIE. Algorithm 5 shows how we answer these queries. The algorithm takes as input a selection variable $\boldsymbol{x}$, a projection atom $H := R(\boldsymbol{X}, \boldsymbol{Y})$, remaining atoms $B_1, ..., B_n$, the support threshold k, and a KB $\mathcal{K}$. The algorithm returns a hash table with each instantiation of the selection variable $\boldsymbol{x}$ as key and the number of distinct bindings of the projection atom H as value.

We first check whether $\boldsymbol{x}$ appears in the projection atom (line 3). If that is the case (lines 4 to 7), we run through all instantiations of the projection atom, instantiate the query accordingly (line 5), and check for existence (line 6). Each existing instantiation increases the counter for the respective value of the selection variable $\boldsymbol{x}$ (line 7). If the selection variable does not appear in the projection atom (lines 9 to 13), we iterate through all instantiations of the projection atom. We instantiate the query accordingly, and fire a SELECT DISTINCT query for $\boldsymbol{x}$ (line 11). We then increase the counter for each value of $\boldsymbol{x}$ (line 13).

Algorithm 5: Count Projection Queries

```
Input: the projection variable : $\boldsymbol{x}$, a query : $R(\boldsymbol{X},\boldsymbol{Y}) \wedge B_1 \wedge ... \wedge B_n$, minimum
       support threshold : $k$, a KB : $\mathcal{K}$,
Output: A map $\langle \hat{x} \rightarrow k \rangle$ for each binding of the projection variable $\boldsymbol{x}$
1  $map := \{\}$
2  $q := B_1 \wedge ... \wedge B_n$
3  if $\boldsymbol{x} \in \{R, \boldsymbol{X}, \boldsymbol{Y}\}$ then
4      for each instantiation $(\sigma := \{R \leftarrow r, X \leftarrow x, Y \leftarrow y\}) \in R(\boldsymbol{X},\boldsymbol{Y})$ do
5          $q' := \sigma(q)$
6          if Exists($q'$, $\mathcal{K}$) then
7              $map[x]++$
8  else
9      for each instantiation $(\sigma := \{R \leftarrow r, X \leftarrow x, Y \leftarrow y\}) \in R(\boldsymbol{X},\boldsymbol{Y})$ do
10         $q' := \sigma(q)$
11         $\mathcal{X}$ := Select($\boldsymbol{x}$, $q'$, $\mathcal{K}$)
12         forall the $x \in \mathcal{X}$ do
13             $map[x]++$
14 $map := \{\langle x \rightarrow n \rangle \in map : n \geq k\}$
15 return $map$
```

2.6 Performance Experiments

In this section we evaluate the runtime performance of AMIE by comparing it with two competitor rule mining systems, namely ALEPH [113] and WARMR [28, 29] (Sections 2.6.2 and 2.6.3 respectively). We also measure the runtime of AMIE with different settings on YAGO and DBpedia.

2.6.1 Experimental Setup

Hardware. All experiments were run on a server with 48GB of RAM, 8 physical CPUs (Intel Xeon at 2.4GHz, 32 threads) and using Fedora 21.

Datasets. For the purpose of comparing AMIE with its competitors, we used a subset of YAGO2 consisting of 948K facts about 470K entities and 32 relations. We also tested AMIE on a subset of DBpedia containing 13.7M facts about 1.4M entities and 1595 relations. These subsets were obtained by removing all facts with literals (numbers and strings). Literal values, such as geographical coordinates, are shared by only very few entities, therefore they are less interesting for rule mining.

Settings. In its default settings, AMIE uses a 1% head coverage threshold (i.e., $minHC = 0.01$), and a maximum of 3 atoms for rules (i.e., $l = 3$, see Section 2.5). By default, AMIE does not impose a confidence threshold, i.e., $minConf = 0$. Unless explicitly mentioned, the instantiation operator $\mathcal{O}_I$ is disabled. ("without constants"). The system uses as many threads as available logical cores in the system (32 in our hardware platform). Any deviation from these settings will be explicitly stated.

Metrics. We compared AMIE in terms of output quality and runtime to two popular state-of-the-art systems : WARMR [28, 29] and ALEPH[2]. To have an equal basis for the comparison with these systems, we made AMIE simulate their metrics. AMIE can threshold on support, head coverage, standard confidence, and PCA confidence, and can rank by any of these. She can also deviate from the default setting and count support on one of the head variables (like WARMR). In that case, AMIE counts on the most functional variable of the relation (see Section 1.3.3 about functions). Again, any such deviation from the default behavior will be mentioned explicitly.

All rules and all experimental results are available at `http://www.mpi-inf.mpg.de/departments/ontologies/projects/amie/`.

2.6.2 AMIE vs. WARMR

WARMR is a system that unifies ILP and association rule mining. Similar to APRIORI algorithms [5], it performs a breadth-first search in order to find frequent patterns. WARMR generates Datalog queries of the form "$? - A_1, A_2, ..., A_n$", where A_i are logical atoms. WARMR applies a closed world assumption for assessing the quality of the produced rules.

Usability. To discover frequent patterns (as in association rule mining), the user must provide WARMR a notion of frequency (support). Given that WARMR considers queries as patterns and that queries can have variables, it is not immediately obvious what the frequency of a given query is. Therefore, the user needs to specify the predicate that is being counted by the system (the *key predicate*). In the usual scenario of market basket analysis, e.g., the system counts customer transactions. In a scenario in which the database is a KB, one solution is to count entities. Since the key predicate determines what is counted, it must be present in all queries. Therefore, we add a predicate *entity*(x), which we fill with all entities of the KB. AMIE does not require such a choice.

For WARMR, the user needs to provide specific information about which predicates can be added to a query, which of their variables can be fresh, and which arguments of predicates are allowed to be unified (type declarations). In contrast, AMIE requires none of these. AMIE simply takes as input the KB in triple format.

WARMR is also able to mine rules with constants. The user can define which predicates and arguments should be instantiated with constants (we call this mode MODE1). WARMR then checks all the constants appearing in the facts of that specific predicate and argument and afterwards uses them in the queries. MODE1 naturally entails an increase of the branching factor in the search space and an explosion in the number of candidates that need to be evaluated. Alternatively, WARMR allows the user to set a maximum number of constants to be used for each predicate and argument (MODE2). Unfortunately, though, it does not provide a way for the user to influence the selection of these constants. In other words, there is no guarantee that the constants that WARMR will use are the most promising ones.

WARMR produces rules as output. These rules are not necessarily connected. For example, WARMR mines

$$\mathit{isMarriedTo}(b, c) \wedge \mathit{isLeaderOf}(a, d) \Rightarrow \mathit{hasAcademicAdvisor}(c, e)$$

This rule is not only nonsensical from a semantic perspective, but also redundant, because the second atom does not influence the implication. Therefore, the user has to filter out these rules from the output. Thus, we conclude that the broader mission and the broader applicability of WARMR entails that much more configuration, acquaintance, and expert knowledge is needed to make it mine Horn rules on semantic KBs.

Runtime. We first compare WARMR with AMIE in terms of runtime only. For a fair comparison, we have to make sure that both systems run in the same settings. Hence, we tweaked AMIE to simulate WARMR's notion of support. We run all systems with an absolute support threshold of 5 entities. We also use the standard confidence as quality metric for rules, instead of the PCA confidence.

In our initial experiment, WARMR was not able to terminate on YAGO2 in a time period of 1 day. Therefore, we created a random sample of YAGO2. We first recall that a KB can be seen as a graph with nodes as entities and facts as directed edges from the subject to the object. We also observe that sampling facts randomly from a KB is equivalent to sample edges (and their endpoints) in a graph. Since this strategy could break the interesting links between entities in the resulting sample, we randomly selected 10K nodes (we call them seed entities) from the graph and then collected their directed 3-hop neighborhood. This yielded 47K facts about 14K entities. This sample contains all available information in a radius of 3 hops around the seed entities, but much less information about the entities at the periphery of the subgraph. For this reason, we restricted the values for the key predicate to the seed entities only.

Table 2.8 summarizes the runtime results for WARMR and AMIE on this dataset. We see that AMIE mines her rules in 6.02 seconds. WARMR, in contrast, took 18 hours.

We also ran both systems in a mode that allows them to mine rules with constants. For AMIE, this means enabling the instantiation operator $\mathcal{O}_I$ (see Section 12). For WARMR, this corresponds to MODE1. AMIE completed the task in less than 2 minutes. WARMR, in contrast, did not terminate in 3 days. Therefore, we ran it only for

Constants	WARMR	AMIE
no	18h	6.02s
yes	(48h)	1.43min

TABLE 2.8 – Runtimes on YAGO2 sample

the relations *diedIn*, *livesIn*, *wasBornIn*, for which it took 48h. To understand this drastic difference, one has to take into account that WARMR is an ILP algorithm written in a logic programming environment, which makes the evaluation of all candidate queries inefficient.

Results. After filtering out non-connected rules, WARMR mined 41 closed rules. AMIE, in contrast, mined 75 closed rules, which included the ones mined by WARMR. We checked back with the WARMR team and learned that for a given set of atoms $B_1, \ldots, B_n$, WARMR will mine only one rule, picking one of the atoms as head atom (e.g., $B_1 \wedge ... \wedge B_{n-1} \Rightarrow B_n$). AMIE, in contrast, will mine one rule for each possible choice of head atom (as long as the thresholds are met). In other words, AMIE with the standard support and confidence measures simulates WARMR, but mines more rules. Furthermore, it runs orders of magnitude faster. Especially for large datasets for which the user would have needed to use complicated sampling schemes in order to use WARMR, AMIE can be a very attractive alternative. Even for smaller datasets with rules with constants, AMIE can provide results while WARMR cannot. Moreover, AMIE does not make a closed world assumption as WARMR does. In Section 2.7.1 we show that the PCA confidence defined by AMIE is more suitable than the standard confidence to identify predictive rules in a web-extracted KB designed under the Open World Assumption.

2.6.3 AMIE vs. ALEPH

ALEPH is an ILP system that implements a variety of scoring functions for measuring a rule's quality. For our experiments we used the *Positives-only* evaluation function [75,87] described by Equation 2.2 which is the most interesting for our setting, since it does not require the existence of explicit negative examples. ALEPH's positives-only learning function rewards rules that cover many positive examples, and few or no randomly generated examples. It also benefits rules evaluated on larger sets of negative examples and that use as few atoms as possible.

Usability. ALEPH can be run with different commands that influence the search strategy. We chose the *induce* command, which runs fastest. To run ALEPH, the user has to specify the target predicate for learning (the head predicate of the rules). In the following, we ran ALEPH successively with all predicates of the KB as targets. In addition, the user has to specify a series of type and mode declarations (similar to WARMR),

KB	ALEPH	AMIE
YAGO2 full	4.96s to $>$ 1 day	4.41min
YAGO2 Sample	0.05s to $>$ 1 day	5.65s

TABLE 2.9 – Runtimes ALEPH vs. AMIE

Relations	Runtime
isPoliticianOf, hasCapital, hasCurrency	$<$ 5min
dealsWith, hasOfficialLanguage, imports	$<$ 5min
isInterested, hasMusicalRole	$<$19min
hasAcademicAdvisor, hasChild	$>$ 1 day
isMarriedTo, livesIn, worksAt, isLocatedIn	$>$ 1 day

TABLE 2.10 – Runtimes of ALEPH on YAGO2

which will be used as a language bias in order to restrict the search space. Also, the user needs to provide ALEPH with files containing the background knowledge and positive examples for the target predicate (in the spirit of the original ILP formulation, see Section 2.1). In contrast, AMIE requires no such input. It will run on a KB without any prespecified choices of predicates.

Runtime. We ran AMIE and ALEPH on YAGO2. For ALEPH, we used the positives-only evaluation function with $R_{size} = 50$ and we considered only clauses that were able to explain at least 2 positive examples, so that we will not get grounded facts as rules in the output. For a fair comparison, we also instructed AMIE to run with a support threshold of 2 facts.

Table 2.9 shows the results on YAGO2 and the sample of 47K facts we constructed for WARMR. AMIE terminated in 4.41 minutes on YAGO2 and 5.65s on the sample and found rules for all relations. ALEPH runs for one head relation at a time. For some relations (e.g., *isPoliticianOf*), it terminated in a few seconds. For others, however, we had to abort the system after 1 day without results as Tables 2.10 and 2.11 show. For each relation, ALEPH treats one positive example at a time. Some examples need little processing time, others block the system for hours.

Results. We compared the output of ALEPH with the positives-only evaluation function to the output of AMIE using the PCA confidence on the sample of YAGO2 used for the runtime experiments. Since ALEPH required more than one day for some relations, we used only rules for which the head relation runs in less than one day. ALEPH mined 56 rules, while AMIE mined 302 rules including the ones mined by ALEPH.

Summary. Our experimental results show that AMIE can be up to 3 orders of magnitude

Relations	Runtime
diedIn, directed, hasAcademicAdvisor	< 2min
graduatedFrom, isPoliticianOf, playsFor	< 2min
wasBornIn, worksAt, isLeaderOf	< 2min
exports, livesIn, isCitizenOf	< 1.4h
actedIn, produced, hasChild, isMarriedTo	> 1 day

TABLE 2.11 – Runtimes of ALEPH on YAGO2 sample

Dataset	Runtime	Rules
YAGO2	3.62min	138
YAGO2 ($l = 4$)	27.14min	645
YAGO2 const	17.76min	18886
DBpedia 2.0 ($l = 2$)	2.89min	6963

TABLE 2.12 – AMIE on Different Datasets

faster than other state-of-the-art systems, namely WARMR [46] and ALEPH [87].

2.6.4 AMIE with Different Settings

As a proof of concept, we ran AMIE in its default settings, with constants, and with maximum rule length $l = 4$ on YAGO2. We also ran the system on DBpedia 2.0. Due to the large number of relations in this dataset, there is an enormous number of rules to be found. We therefore show the time taken to mine rules with 2 atoms. As Table 2.12 shows, AMIE can produce rules with or without constants in a reasonable amount of time. We also show some of those rules in Table 2.13. In particular, rules with 4 atoms motivate us to keep the default rule length at 3 atoms.

y :isMarriedTo$(x, y) \wedge$ *y :livesIn*$(x, z) \Rightarrow$ *y :livesIn*(y, z)
y :hasAdvisor$(x, y) \wedge$ *y :graduatedFrom*$(x, z) \Rightarrow$ *y :worksAt*(y, z)
y :wasBornIn$(x, y) \wedge$ *y :isLocatedIn*$(y, z) \Rightarrow$ *y :isCitizenOf*(x, z)
y :hasWonPrize$(x, G.\ W.\ Leibniz) \Rightarrow$ *y :livesIn*$(x, Germany)$
y :hasWonPrize$(x, Grammy) \Rightarrow$ *y :hasMusicalRole*$(x, Guitar)$
y :advisor$(z, w) \wedge$ *y :citizenOf*$(w, y) \wedge$ *y :livesIn*$(z, x) \Rightarrow$ *y :deals*(x, y)
y :diedIn$(x, z) \wedge$ *y :locatedIn*$(z, y) \wedge$ *y :livesIn*$(x, z) \Rightarrow$ *y :politician*(x, y)
d :capital$(x, y) \Rightarrow$ *d :largestCity*(x, y)
d :nationality$(x, y) \Rightarrow$ *d :residence*(x, y)

TABLE 2.13 – Some Rules by AMIE on YAGO (y :) and DBpedia (d :)

2.7 Fact Inference with AMIE

In Section 2.6 we showed the performance of AMIE in terms of runtime. However, we did not analyze the quality of the output rules. This section evaluates AMIE in terms of the produced rules, more specifically, in how correct are the conclusions drawn by those rules. In Section 2.1 we stated that, one of the goals of the AMIE mining model is to learn rules that make correct predictions, in and beyond the KB. The task of predicting facts outside the KB is often referred in the literature as *link prediction* [14, 45, 65, 81, 93, 112, 125] because it aims at finding semantic links (relations) between entities in the KB. This is a difficult endeavor : It amounts to guessing the places of residence for people, their birth place, or even their death place. Naturally, we may not assume a high precision in the prediction of the future. We may only expect educated guesses.

To evaluate the precision of these guesses for the studied rule mining systems, we proceeded as follows : We ran the system on the YAGO2 dataset and ranked the obtained rules by confidence. For each rule, we evaluated whether the predictions that go beyond YAGO2 were true. We did this by either checking if the prediction appears in a newer version of the KB (YAGO2s), or by manually checking them in Wikipedia. If we could find the predicted fact in neither, we evaluated it as false. Since the number of predictions can be large, we conducted the evaluation of the precision on a random sample of the predictions. We applied this experimental setup to assess the precision of the predictions for AMIE using the standard confidence, AMIE using the PCA confidence (Section 2.7.1) and ALEPH using the positives-only learning function (Section 2.7.2).

2.7.1 Standard vs. PCA Confidence

Our first goal is to see whether the PCA confidence or the standard confidence perform better for the task of fact prediction. For this purpose, we ran AMIE on YAGO2, and sorted the resulting rules first by descending PCA confidence, and then by descending standard confidence. We looked at the top ranked rules in each case, and evaluated the precision of the predictions. The two bottom curves of Figure 2.1 plot the aggregated predictions versus the aggregated precision for the standard and the PCA confidence. The n-th dot from the left represents the total number of unique predictions and the total precision of these predictions (ratio of correct predictions), aggregated over the first n rules. As we see, ranking the rules by standard confidence is a very conservative approach : It identifies rules with reasonable precision, but these do not produce many predictions. Going down in the list of ranked rules, the rules produce more predictions – but at lower precision. The top 30 rules produce 113K predictions at an aggregated precision of 34%. In contrast, if we rank the rules by PCA confidence, we quickly get large numbers of predictions. The top 10 rules already produce 135K predictions – at a precision of 45%. The top 30 rules produce 3 times more predictions than the top 30 rules by standard confidence – at comparable precision. This is because the PCA confidence is less conservative than the standard confidence. We thus conclude that the PCA confidence is better suited for making predictions than the standard confidence. In addition, we investigate how the confidence metrics correlate with the actual

	Top 20 rules	Top 30 rules	All rules
Confidence	0.76	0.63	0.33
PCA Confidence	0.32	0.29	0.29

TABLE 2.14 – Average Absolute Error to Precision

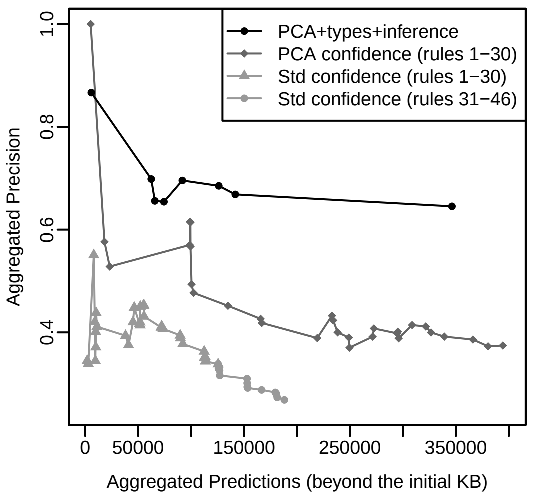


FIGURE 2.1 – Std. confidence vs. PCA confidence

precision of the rule, e.g., in case we want to use them as estimators of the real precision. Table 2.14 summarizes our findings. We rank the mined rules by their precision and report the average absolute error of the standard and PCA confidence weighted by the number of predictions produced by the rules. We can observe that, on average, the PCA confidence estimates the precision of the rules better than the standard confidence. Thus, reasoning approaches could use the PCA confidence as a weight for the rule.

Using Type Information. The previous experiment showed us the precision of individual rules for prediction. To make more accurate predictions, we have to combine these rules with more signals. We proceed as follows. In Section 2.4.1 we discussed the granularity differences in relations. For instance, the relation $livesIn$ is used to express a person's city or country of residence. This implies that, for example, the rule $livesIn(x, y) \Rightarrow isCitizenOf(x, y)$ can predict that some people are citizens of cities. Such spurious predictions decrease the precision of the inference process. Therefore, we configured AMIE to mine *typed rules*. These have the form :

$$\boldsymbol{B} \wedge \textit{rdf :type}(x, D) \wedge \textit{rdf :type}(y, R) \Rightarrow r(x, y)$$

where $D := domain(r)$ and $R := range(r)$ correspond to the domain and range of the head relation r in the schema of YAGO3. We resorted to the YAGO3 [74] types and schema because the type signatures in older versions of YAGO are too general. For instance, the relation *livesIn* is defined from person to location in YAGO2, whereas in YAGO3 it is defined from person to city. To allow AMIE to find such rules, we augmented the YAGO2 dataset by adding the *rdf :type* statements about the subjects and objects of the triples in YAGO3. In addition we configured AMIE to enforce the type atoms *rdf :type*$(x, D) \wedge$ *rdf :type*(y, R) right after the creation of the head atom.

Joint Prediction. We observe that a prediction can be fired from multiple rules. If we consider rules as signals of evidence, then facts predicted by more rules should get a higher confidence score. In YAGO2, 9% of the predictions are fired by more than one rule (with a PCA confidence threshold of 0.1). To take this into account, we changed the way predictions are ranked. In the original experimental setup, if multiple rules $R_1, \dots R_k$ made a prediction p, the prediction was only counted the first time it was fired. Since the rules were ranked by decreasing PCA confidence, this was equivalent to ranking the predictions according to their highest PCA confidence :

$$score(p) := max\{conf_{pca}(R_1), \dots, conf_{pca}(R_k)\}$$

where $\mathcal{K} \wedge R_i \vDash p$ and $i = 1...k$. We propose an alternative score instead :

$$score^*(p) := 1 - \prod_{i=1}^{k}(1 - conf_{pca}(R_i)) \quad (2.4)$$

Equation 2.4 aggregates the PCA confidence of the rules so that the predictions concluded by multiple rules are ranked higher. It also confers a probabilistic interpretation to the PCA confidence. The score of a prediction is the probability that at least one of the rules in $R_1, \dots R_k$ concludes p. This is computed as 1 minus the probability that none of the rules concludes p. The probability of a rule not concluding p is defined as 1 minus the PCA confidence of the rule. The probability that none of the rules concludes p is the product of the individual probabilities. Although this scoring-scheme is very simplistic (it assumes independence of the rules, and confers a probabilistic interpretation to the confidence), it can still serve as a proof of concept. In real applications, more involved methods [88, 104] can be used for joint prediction.

Results. The upper curve in Figure 2.1 shows the precision of the predictions made with both heuristics. We proceeded as in the previous experiment, that is, we first used the rules to fire predictions, and then we ranked these predictions by descending score and computed their cumulative precision. Unlike in the original experimental setup, the n-th point from the left in the new curve corresponds to the cumulative precision of the predictions up to the n-th bucket. We bucketized the predictions by score using a bucket size of 0.1, i.e., the first point corresponds to the predictions with score between 1 and 0.9, the next one accounts for the predictions with score between 0.9 and 0.8 and so on.

As we can observe, our heuristics have a significant effect on the precision of the predictions. The precision is much higher at each level of recall, compared to the original experiment. We can make 100,000 predictions at a precision of 70%. At 400K

	Top n	Predictions	Precision
Positives-only	7	2997	27%
PCA Confidence	12	2629	62%
Positives-only	9	5031	26%
PCA Confidence	22	4512	46%
Positives-only	17	8457	30%
PCA Confidence	23	13927	43%

TABLE 2.15 – PCA confidence vs. positives-only score : aggregated precision of rules mined on YAGO2 sample.

predictions, we still achieve a precision of 60%. While these predictions should not be added directly to a KB, they could be sent to human evaluators to check their correctness. It is much easier for a person to check fact candidates for their correctness than to invent them from scratch. In addition, this experimental setup can serve as a baseline for more sophisticated inference approaches.

2.7.2 AMIE vs. ALEPH

In Section 2.7.1, we showed empirically that the PCA confidence outdoes the standard confidence as a ranking metric for rules and predictions in a web-extracted KB, namely YAGO2. In this section we compare AMIE and the PCA confidence with ALEPH and its positives-only learning function (see Section 2.3.3.2). We build upon the experimental setup introduced in Section 2.6.3 where we compared the runtime of AMIE and ALEPH on YAGO2. We ran both systems on our sample of YAGO2 with a minimum support of 2 facts and without a confidence threshold, that is, $minConf = 0$ for AMIE. ALEPH generates a random set of facts as counter-examples for rules. The size of the counter-examples set is provided as an argument by the user (R_{size} in Equation 2.2). We chose $R_{size} = 50$ since this argument has a direct impact in ALEPH's runtime. The higher the value of R_{size}, the more accurate is the estimation of the confidence of the rule and the slower the system performs.

As we did in Section 2.7.1, we ordered the rules by decreasing score (ALEPH) and decreasing PCA confidence (AMIE). We computed the precision of the rules by evaluating whether a prediction made by the rule is correct or not. Table 2.15 shows the number of predictions and their total precision. We show the aggregated values at the points where both approaches have produced around 3K, 5K, and 8K predictions. AMIE's PCA confidence succeeds in sorting the rules roughly by descending precision, so that the initial rules have an extraordinary precision compared to ALEPH's. AMIE needs more rules to produce the same number of predictions as ALEPH (but she also mines more).

We suspect that ALEPH's positives-only evaluation function manages to filter out overly general rules only to some extent. The reason is that this measure "guesses" negative examples at random (Section 2.3.3.2), whereas rules usually create false predictions in a non-random way.

2.8 Conclusion

In this chapter, we have presented AMIE, an approach to mine Horn rules on large RDFS knowledge bases. AMIE is based on a formal model for rule mining under the Open World Assumption, a method to simulate counter-examples, and a scalable mining algorithm and implementation. In contrast to traditional ILP systems, AMIE requires no input other than the KB and does not need configurations or parameter tuning. As our experiments have shown, AMIE outperforms state-of-the-art approaches not only in terms of runtime, but also in terms of the number and quality of the output rules. If we combine these rules with simple heuristics for type checking and joint prediction, we can use them to predict facts with a precision of about 70%.

Chapitre 3

Speeding Up Rule Mining

The performance experiments presented in Section 2.6 show the suitability of AMIE for rule mining on a KB with 1M facts. In this chapter, we present AMIE+, an extension of AMIE that implements a set of heuristics that improve the scability of the system. Our extensions to the basic AMIE algorithm aim at speeding up two phases of the main rule mining algorithm introduced in Section 2.5 : (i) the refinement phase and (ii) the confidence evaluation.

The contents of this chapter are structured as follows : Section 3.1 describes three strategies that optimize the refinement phase of the AMIE algorithm. Section 3.2 describes two heuristics to efficiently identify low quality rules before computing their actual confidence. By identifying such rules in advance, we can discard them without investing time in calculating their confidence. In Section 3.3 we evaluate the runtime gain carried by our heuristics by comparing AMIE+ with AMIE. In Section 3.4 we introduce an interactive demo that illustrates the inner workings of AMIE+ to end users. Section 3.5 concludes the chapter.

The work presented in this chapter is based on the following publications :

— Luis Galárraga, Christina Teflioudi, Katja Hose, Fabian Suchanek. *Fast Rule Mining in Ontological Knowledge Bases with AMIE+*. International Journal on Very Large Databases. Volume 24, Issue 6, pp 707–730. December 2015.

— Luis Galárraga. *Interactive Rule Mining in Knowledge Bases*. 31ème Conférence sur la Gestion de Données. Île de Porquerolles, France, 2015.

3.1 Speeding Up Rule Refinement

In this section, we discuss how AMIE+ speeds up the rule refinement phase for specific kinds of rules. These techniques do not alter AMIE's output in any way.

Maximum Rule Length. Recall from Section 2.5 that the maximum rule length l is an input parameter for our system. AMIE stops exploring the search space as soon as all rules with a length of at most l have been produced. During the mining process, AMIE creates connected rules by applying all possible mining operators (line 10 in Algorithm 1) on previously created rules. Given a maximum rule length l and a non-closed Horn rule of length $l-1$, AMIE+ will refine it only if it is possible to close it before exceeding the length constraint. This means that for a not-yet-closed rule of length $l-1$, AMIE+ will not apply the add-dangling-atom operator $\mathcal{O}_D$, because this results in a non-closed rule, which will be neither output nor refined. In the same spirit, if the same rule contains more than two non-closed variables (see Section 2.5.2), AMIE+ will skip the application of the add-closing atom operator $\mathcal{O}_C$. This happens because an application of the operator $\mathcal{O}_C$ can close at most two variables with one atom. This reasoning also applies to $\mathcal{O}_I$, the instantiation operator : rules with more than one non-closed variable are not refined with instantiated atoms, because the addition of an instantiated atom can close at most one variable.

Perfect Rules. By definition, a rule cannot achieve a PCA confidence that is higher than 100%. Thus, once a rule has achieved 100% PCA confidence, we can stop adding new atoms. This is because the confidence cannot increase and the support can only decrease. Hence, any refinement is futile and will be discarded by the output routine described in Algorithm 2. We call rules with 100% PCA confidence *perfect rules*.

Simplifying Projection Queries. Support is by definition monotonically decreasing with the length of the rule (Section. 2.3.2.1). Hence, whenever we apply an add-dangling-atom operator to a rule R_p (the parent rule) to produce a new rule R_c (the child rule), the support of R_c will likely be smaller than the support of R_p. However, there is one case in which the addition of a dangling atom cannot reduce the support. This happens when R_c (i) already contains atoms with the same relation as the dangling atom and (ii) these atoms have a variable in common with the dangling atom. An example is the parent rule

$$R_p: \Rightarrow isCitizenOf(x, y)$$

and the child rule

$$R_c: \quad isCitizenOf(z, y) \Rightarrow isCitizenOf(x, y)$$

We observe that the addition of the dangling atom *isCitizenOf*(z, y) cannot further restrict the support of R_p because the new atom is a less restrictive version of the atom $isCitizenOf(x, y)$. This means that z will always bind to the same values as x. From this observation, it follows that the support of R_c can be rewritten as

$$supp(R_c) = \#(x, y) : isCitizenOf(x, y) \wedge isCitizenOf(x, y)$$

$$supp(R_c) = \#(x, y) : isCitizenOf(x, y)$$

which is the same as $supp(R_p)$. Thus both R_p and R_c have the same support. This observation can be leveraged to speed up projection queries. In particular, the application of the dangling atom operator $\mathcal{O}_D$ to R_c requires to fire count projection queries of the form $\boldsymbol{r}(\boldsymbol{X},\boldsymbol{Y}) \wedge R_c$ with $\boldsymbol{r}(\boldsymbol{X},\boldsymbol{Y}) \in \{\boldsymbol{r}(z,w), \boldsymbol{r}(w,z), \boldsymbol{r}(x,w), \boldsymbol{r}(w,x)\}$ for an arbitrary fresh variable w. Since R_c has the same support as R_p, we can replace R_c by R_p resulting in an equivalent query with fewer atoms. We remark, though, that this rewriting is not applicable to all cases, e.g., $\boldsymbol{r}(z,w) \wedge \mathit{isCitizenOf}(z,y) \Rightarrow \mathit{isCitizenOf}(x,y)$. In general, if the dangling atom contains the originally replaceable variable (z in our example), AMIE cannot replace R_c with R_p. In our example this happens because the atom *isCitizenOf*(z,y) becomes mandatory due to the additional constraints imposed on variable z in $\boldsymbol{r}(z,w)$. Conversely, in this example the rewriting is valid for the dangling atoms $\boldsymbol{r}(x,w)$ and $\boldsymbol{r}(w,x)$.

3.2 Speeding up Confidence Evaluation

3.2.1 Motivation

A significant part of the runtime of Algorithm 1 is spent on computing confidence scores (up to 35% in our experiments). The reason is that the calculation of confidence (both PCA and standard) requires the calculation of the number of instantiations of the rule body. If the body contains atoms with many instantiations, the joins can be very expensive to compute.

At the same time, we will not output rules with a confidence below the threshold $\mathit{minConf}$ (Section 2.5). This means that the system might spend a significant amount of time evaluating expensive confidence queries only to find out that the rule was of low confidence and will not be output. An example of such a rule is :

$$\mathit{directed}(x,z) \wedge \mathit{hasActor}(z,y) \Rightarrow \mathit{marriedTo}(x,y)$$

This rule concludes that a director is married to all the actors that acted in his/her movies, producing a total of 74249 married couples in YAGO2. AMIE needs more than 500ms (more than twice the average cost : 200ms) to calculate the confidence of this intuitively bad rule.

We have developed a method to approximate the confidence value of such a rule very quickly. Our approximation is based on statistics, such as the functionalities of the atoms, or the size of the joins between two relations. We pre-compute these quantities, so that they can be accessed in constant time. As a result, AMIE+ prunes the example-rule above in less than 1ms. In addition, our approximation is designed such that it is more likely to overestimate confidence than to underestimate it. This is important, because we use it to prune rules, and we want to avoid pruning rules that have a higher confidence in reality.

In Section 3.2.2, we provide an overview of the confidence approximation and we explain for which form of rules we use it. Section 3.2.3 discusses the underlying assumptions, how to compute the approximation and how AMIE+ uses it. Finally, Section 3.2.4 derives upper bounds for the confidence of a particular class of rules.

3.2.2 Confidence Approximation

Recall that the standard confidence and the PCA confidence (see Sections 2.3.3.1 and 2.4) for a rule of the form $\boldsymbol{B} \Rightarrow r_h(x,y)$ are defined as :

$$conf(\boldsymbol{B} \Rightarrow r_h(x,y)) := \frac{supp(\boldsymbol{B} \Rightarrow r_h(x,y))}{\#(x,y) : \exists z_1, \dots, z_m : \boldsymbol{B}}$$

and

$$conf_{pca}(\boldsymbol{B} \Rightarrow r_h(x,y)) := \frac{supp(\boldsymbol{B} \Rightarrow r_h(x,y))}{\#(x,y) : \exists z_1, \dots, z_m, y' : \boldsymbol{B} \wedge r_h(x,y')}$$

We restrict our analysis to rules with two variables in the head because count queries on a single variable are cheap to compute. By the time AMIE has to calculate the confidence of a rule, the system already knows the support of the rule. Hence, the remaining step is to fire the queries for the denominators of the confidence expressions. We denote them by d_{std} and d_{pca} :

$$d_{std}(\boldsymbol{B} \Rightarrow r_h(x,y)) := \#(x,y) : \exists z_1, \dots, z_m : \boldsymbol{B} \tag{3.1}$$

$$d_{pca}(\boldsymbol{B} \Rightarrow r_h(x,y)) := \#(x,y) : \exists z_1, \dots, z_m, y' : \boldsymbol{B} \wedge r_h(x,y') \tag{3.2}$$

Our aim is to derive a conservative approximation for d_{pca} and d_{std} denoted by $\widehat{d_{pca}}$ and $\widehat{d_{std}}$ respectively. By plugging this expression into the PCA confidence formula, we get

$$\widehat{conf}_{pca}(R) := \frac{supp(R)}{\widehat{d_{pca}}(R)} \tag{3.3}$$

Let us reconsider Equation 3.2 and rewrite it as follows :

$$d_{pca}(\boldsymbol{B}(x,y) \Rightarrow r_h(x,y)) := \#(x,y) : \exists z_1, \dots, z_m, y' : \boldsymbol{B}(x,y) \wedge r_h(x,y')$$

Here, we resort to an abstraction that treats the body of the rule $\boldsymbol{B}$ as a relation $\boldsymbol{B}(x,y)$ on the head variables, i.e., $\boldsymbol{B}(x,y) := \{x, y : \exists z_1, \dots z_n : \boldsymbol{B}\}$. If $\boldsymbol{B}$ has functionality $fun(\boldsymbol{B})$, on average each entity in variable x relates to $\#y_{per\ x} = {}^1/_{fun(\boldsymbol{B})}$ bindings in y. Let us define the effective domain and range of a relation r as :

$$dom(r) := \{x : \exists y : r(x,y) \in \mathcal{K}\}$$

$$rng(r) := \{y : \exists x : r(x,y) \in \mathcal{K}\}$$

These are the sets of entities that occur as subjects and objects in the relation respectively. Given the definitions of $dom(r)$ and $rng(r)$ respectively, the following equation provides an estimate for the body size of the rule, i.e., $d_{std}(\boldsymbol{B} \Rightarrow r_h(x,y))$:

$$\widehat{d_{std}}(\boldsymbol{B} \Rightarrow r_h(x,y)) := |dom(\boldsymbol{B})| \cdot \#y_{per\ x} \tag{3.4}$$

However, for the PCA confidence, the denominator is restricted also by the entities in the effective domain of the head relation. This consideration leads us to the expression :

$$\widehat{d}_{pca}(\boldsymbol{B} \Rightarrow r_h(x,y)) := |dom(\boldsymbol{B}) \cap dom(r_h)| \cdot \#y_{per\ x} \tag{3.5}$$

In the following, we describe when it makes sense to use this approximation and then, in Section. 3.2.3, we discuss how to calculate the terms of Equation 3.5 in an efficient way. The analysis for Equation 3.4 is analogous.

When to Use the Approximation. Using any form of confidence approximation always involves the risk of pruning a good rule. At the same time, if the exact confidence value is cheap to compute, the potential gain of using an approximation is small. For this reason, we only use the confidence approximation for rules whose exact confidence is relatively "expensive" to compute. These rules typically have a large number of bindings in the body because of the presence of *intermediate variables*. This translates into higher runtimes and memory usage. An example is the rule we saw before :

$$directed(x,z) \wedge hasActor(z,y) \Rightarrow marriedTo(x,y)$$

In this example, a director x is related to many movies z (the intermediate variable) that have different actors y. Hence, we consider a rule *expensive* if its body (i) contains variables other than the variables appearing in the head atom (z in our example) and (ii) if these additional variables define a single path between the head variables ($x \rightarrow z \rightarrow y$ in our example). Note that rules without intermediate variables are usually associated with more selective queries. An example is the rule

$$livesIn(x,y) \wedge bornIn(x,y) \Rightarrow diedIn(x,y)$$

Rules that contain multiple paths between the head variables are also selective. Consider the rule :

$$livesIn(x,z_1) \wedge locatedIn(z_1,y) \wedge bornIn(x,z_2) \wedge locatedIn(z_2,y) \Rightarrow isCitizenOf(x,y)$$

In these two examples, both $livesIn$ and $bornIn$ join on x in the body and restrict the size of the result.

We therefore use the confidence approximation only for rules where the head variables x, y are connected through a single chain of existentially quantified variables $z_1, \ldots, z_{n-1}$. These rules have the form :

$$r_1(x,z_1) \wedge r_2(z_1,z_2) \wedge ... \wedge r_n(z_{n-1},y) \Rightarrow r_h(x,y)$$

In order to write a rule in this canonical form, we may have to replace some relations by their inverses (i.e., substitute $r_i(z_{i-1},z_i)$ with $r_i^{-1}(z_i,z_{i-1})$) and change the order of the atoms. We will now see how to compute the approximation for this type of rules.

3.2.3 Computing the Approximation

In the following, we use the shortcut notations $ov_{dr}(r_1, r_2)$, $ov_{rd}(r_1, r_2)$, $ov_{dd}(r_1, r_2)$, $ov_{rr}(r_1, r_2)$ for the size of the overlap sets between the effective domains and ranges of pairs of relations. For example number of common entities between the domain of relation r_1 and the range of relation r_2 is denoted by

$$ov_{dr}(r_1, r_2) := |dom(r_1) \cap rng(r_2)|$$

Let us now consider again the rule

$$directed(x, z) \wedge hasActor(z, y) \Rightarrow marriedTo(x, y)$$

which implies that a director is married to all actors that acted in his movies. In this case, $d_{pca}(R)$ is defined as

$$d_{pca}(R) := \#(x, y) : \exists\, z, y' : directed(x, z) \wedge hasActor(z, y) \wedge marriedTo(x, y')$$

Here $\boldsymbol{B}(x, y) = \{\boldsymbol{x}, \boldsymbol{y} : \exists z : directed(\boldsymbol{x}, z) \wedge hasActor(z, \boldsymbol{y})\}$. To calculate the approximation defined in Equation 3.5, we need to calculate (1) the number of directors in $\boldsymbol{B}(x, y)$ that are married, i.e., $|dom(\boldsymbol{B}) \cap dom(marriedTo)|$ and (2) the number of actors y associated to each director x, i.e., $\#y_{per\ x}$. We focus on (2). The estimation of $\#y_{per\ x}$ requires us to walk from the most to the least functional variable, i.e., through the path $x \rightarrow z \rightarrow y$, connecting a director to his potential actors. If $fun(r)$ and $ifun(r)$ denote the functionality and inverse functionality of the relation r, respectively, then walking through this path involves the following steps :

1. For each director x, the relation $directed$ will produce on average $\frac{1}{fun(directed)}$ movies z.
2. Some or all of these movies z will find join partners in the first argument of $hasActor$.
3. For each movie z, $hasActor$ will produce on average $\frac{1}{fun(hasActor)}$ actors y.
4. Each of these actors in y acted on average in $\frac{1}{ifun(hasActor)}$ movies of the $hasActor$ relation.

Up to step 2, we can approximate the number of distinct movies that bind to the variable z for each director in the variable x as :

$$\#z_{per\ x} := \frac{ov_{rd}(directed, hasActor)}{|rng(directed)| \times fun(directed)}$$

Here, $|rng(directed)|$ is the number of distinct movies in the effective range of $directed$ and $ov_{rd}(directed, hasActor)$ denotes the distinct movies in the overlap between the objects of $directed$ and the subjects of $hasActor$. The term $^1/_{fun(directed)}$ corresponds to step 1. Our join estimation assumes that the movies in the overlap of $directed$ and $hasActor$ are uniformly distributed among the different directors in the relation $directed$.

For steps 3 and 4, we can approximate the number of actors in the variable y for each movie in the variable z as follows :

$$\#y_{per\ z} := \frac{ifun(hasActor)}{fun(hasActor)}$$

The term $^1/_{fun(hasActor)}$ corresponds to step 3. At the end of this step, we already have, for a single director x, a bag of actors y associated to him. However, these are not necessarily distinct actors, since x and y are connected through the variable z (movies). Therefore, a duplicate elimination step is needed. To see why, assume that each director has directed on average 3 movies and that each movie has 5 actors. Then, the rule will produce on average 15 actors y for each director x. However, there is no guarantee that these actors are distinct. If the director trusts specific actors and collaborates repeatedly with them in some or all of his movies, there will be less than 15 distinct actors.

The term $ifun(hasActor)$ achieves this duplicate elimination : since each actor participated in $^1/_{ifun(hasActor)}$ different movies, the actor contributes to the final count with a weight that is inversely proportional to this number.

In this way of performing duplicate elimination, a single actor y belongs to $^1/_{ifun(hasActor)}$ different movies z, which are chosen from *all* the movies in the relation $hasActor$. In reality, we want the number of different movies to be chosen from those that remain after step 2, i.e., the average number of movies by the same director that an actor acts in. This number is obviously smaller, which implies that the factor $ifun(hasActor)$ is a pessimistic estimator. This makes our approximation an underestimation of the real confidence denominator, and the overall confidence approximation an overestimation of the actual confidence.

We can now estimate the number of actors y that are supposed to be married with each director x as :

$$\#y_{per\ x} := \#z_{per\ x} \times \#y_{per\ z}$$

To calculate $\widehat{d_{pca}}$ of Equation 3.5, we are now only missing the expression $|dom(\boldsymbol{B}) \cap dom(marriedTo)|$. Here we make the simplifying assumption that $dom(\boldsymbol{B}) = dom(directed)$, so that the expression becomes the size of the join between the relations $directed$ and $marriedTo$, on the subject argument, i.e., $ov_{dd}(directed, marriedTo)$.

To summarize, the factor $\widehat{d_{pca}}(R)$ for a rule $r_1(x,z) \wedge r_2(z,y) \Rightarrow r_h(x,y)$ can be approximated by :

$$\widehat{d_{pca}}(R) := \frac{ov_{dd}(r_1, r_h) \cdot ov_{rd}(r_1, r_2) \cdot ifun(r_2)}{fun(r_1) \cdot |rng(r_1)| \cdot fun(r_2)}$$

For the more general case of a rule that contains $n-1$ existential variables forming a single path from x to y

$$r_1(x, z_1) \wedge r_2(z_1, z_2) \wedge ... \wedge r_n(z_{n-1}, y) \Rightarrow r_h(x, y)$$

the formula becomes :

$$\widehat{d_{pca}}(R) := \frac{ov_{dd}(r_1, r_h)}{fun(r_1)} \times \prod_{i=2}^{n} \frac{ov_{rd}(r_{i-1}, r_i)}{|rng(r_{i-1})|} \frac{ifun(r_i)}{fun(r_i)} \tag{3.6}$$

Estimating standard confidence. Equation 3.4 provides an approximation $\widehat{d_{std}}$ for the body size of a rule, i.e., the denominator of the standard confidence formula d_{std}. As we did for the general case of $\widehat{d_{pca}}$ (Equation 3.6), we assume that $\boldsymbol{B}$ has been rewritten as a path between the head variables x, y, i.e., $\boldsymbol{B} := r_1(x, z_1) \wedge r_2(z_1, z_2) \wedge \cdots \wedge r_n(z_{n-1}, y)$. Under the assumption that $dom(\boldsymbol{B}) := dom(r_1)$, the final formula becomes :

$$\widehat{d_{std}}(R) := |dom(r_1)| \times \prod_{i=2}^{n} \frac{ov_{rd}(r_{i-1}, r_i)}{|rng(r_{i-1})|} \frac{ifun(r_i)}{fun(r_i)} \tag{3.7}$$

Implementation. We precompute the functionalities, the inverse functionalities, and the overlaps between the domains and ranges of each pair of relations when the KB is loaded into the in-memory database. This results in longer loading times, but pays off easily during rule mining. The sizes of the ranges of the relations are given by our indexes in constant time. After this preprocessing, the approximation of the confidence can be computed as a simple product of precomputed values without actually firing a single query. We apply the approximation only if the query is expensive (see Section 3.2.2). If the approximated value is smaller than the threshold, we abandon the rule. Otherwise, we compute the exact PCA confidence and proceed as usual.

Assumptions. The approximations defined by Equations 3.4 and 3.5 make a series of assumptions. First, we make use of functionalities as average values. In other words, we assume that for any relation all objects are uniformly distributed among the subjects (which corresponds to a zero variance). In reality, this is not always the case. Additionally, the estimation of the expression $\#z_{per\ x}$ uses the term $\frac{ov_{rd}(r_{i-1}, r_i)}{|rng(r_{i-1})|}$. This term assumes that the entities in the overlap are uniformly distributed among the entities in the range of r_{i-1}. This also introduces some error that depends on the variance of the real distribution. Nevertheless, the duplicate elimination largely underestimates the count of $\#y_{per\ x}$, and therefore we expect our approximation to usually result in an overestimation of the actual confidence. This is indeed the case, as our experiments in Section 3.3 show.

3.2.4 Confidence Upper Bounds

In some particular cases, we can derive lower bounds for the confidence denominator (d_{pca}, d_{std}) instead of using the approximation described in Section 3.2.2. Consider a rule of the form :

$$r(x, z) \wedge r(y, z) \Rightarrow r_h(x, y)$$

Here, the standard confidence denominator is given by

$$d_{std} := \#(x, y) : \exists z : r(x, z) \wedge r(y, z)$$

Since both atoms contain the same relation, we know that all the entities of z in the first atom will join with the second atom. Furthermore, we know that the join will result in *at least* one y-value for each binding of x, i.e., the case where $y = x$. This allows us to deduce

$$d_{std} \geq \#(x, x) : \exists z : r(x, z) \wedge r(x, z)$$

$$d_{std} \geq \#x : \exists z : r(x, z) \tag{3.8}$$

This expression can be calculated in constant time with the indexes of our in-memory database (Section 2.5.4). A similar analysis can be applied for rules of the form $r(z, x) \wedge r(z, y) \Rightarrow r_h(x, y)$.

The same reasoning applies to the denominator of the PCA confidence, yielding

$$d_{pca} \geq \#x : \exists\ z, y' : r(x, z) \wedge r_h(x, y') \tag{3.9}$$

Although this expression requires to fire a query, it contains fewer atoms than the original expression and counts instances of a single variable instead of pairs. It is therefore much cheaper than the original query.

Both Inequalities 3.8 and 3.9 define lower bounds for the number of pairs in the denominator expressions of the standard and the PCA confidence, respectively. Thus, AMIE+ uses them to upper-bound the respective confidence scores. If the upper bound is below the threshold, the rules can be pruned even before computing the approximate confidence denominator.

3.3 Experiments

In this section, we evaluate the runtime improvements of AMIE+ over the previous version AMIE. Recall from Section 2.5 that the AMIE algorithm consists of three main phases :

— Refinement (i.e., rule expansion).

— Output, which includes confidence calculation.

— Duplicate elimination.

Table 3.1 shows the proportion of time spent by AMIE in each phase when running on YAGO2 – first without constants and then with constants. We observe that the refinement and output phases dominate the system's runtime. When constants are not enabled, most of the time is spent in the refinement phase. In contrast, the addition of the instantiation operator increases the number of rules and therefore the time spent in the output and duplicate elimination phases. In both cases, the duplicate elimination is the least time-consuming phase. The enhancements introduced by AMIE+ aim at reducing the time spent in the refinement and output phases.

Dataset	Rules	Refinement	Output	Dup. elim.
YAGO2	135	87.48%	8.79%	3.74%
YAGO2 (c)	19132	53.54%	35.64%	10.82%

TABLE 3.1 – Time spent in the different phases of the AMIE algorithm on YAGO2, first without the instantiation operator and then with this operator.

KB	Facts	Subjects	Relations
YAGO2 core	948K	470K	32
YAGO2s	4.12M	1.65M	37
DBpedia 2.0	6.70M	1.38M	1595 [1]
DBpedia 3.8	11.02M	2.20M	650
Wikidata	8.4M	4.00M	431

TABLE 3.2 – Knowledge bases used to test AMIE and AMIE+.

3.3.1 Experimental setup

Hardware. We compared AMIE and AMIE+ on the same runtime environment used for the experiments in Section 2.6.1, i.e., a server with 48GB of RAM, 8 physical CPUs (Intel Xeon at 2.4GHz, 32 threads) with Fedora 21 as operating system.

Datasets. We ran our experiments on different KBs. Table 3.2 shows a summary of the KBs used for our experiments. As in Section 2.6.1, we removed all facts with literals (numbers and strings). For both DBpedia 2.0 and 3.8, we used the person data and mapping-based properties datasets. For Wikidata, we used a dump from December 2014, available for download at `http://tools.wmflabs.org/wikidata-exports/rdf/exports/20140420/`.

Settings. AMIE+ inherits the default settings of AMIE described in Section 2.6.1. In addition, AMIE+ enables the improvements introduced in Section 3.1, namely pruning by maximum rule length (MRL), the query rewriting (QRW) and the identification of perfect rules (PR). These are lossless runtime enhancements that optimize the refinement phase and do not alter AMIE's output. If an additional confidence threshold is given, AMIE+ can use the heuristics presented in Section 3.2 to prune rules below the confidence threshold. This reduces the time spent in the confidence calculation and hence the output phase. For this reason, we override the default settings and set a confidence threshold of 0.1 for AMIE+. Any deviation from these settings will be explicitly stated.

Runtime Comparison. Table 3.3 shows the runtimes of AMIE and AMIE+. We set a threshold of 0.1 PCA confidence for AMIE to make it comparable with AMIE+. For the

1. Relations with more than 100 facts only.

KB	AMIE	AMIE+					
		Only Refinement	Only Output	Output +			Full
				MRL	QRW	PR	
YAGO2	3.17min	29.37s	2.82min	29.03s	38.16s	2.80min	28.19s
YAGO2 (const)	37.57min	11.72min	37.05min	8.90min	12.04min	36.48min	9.93min
YAGO2 (4)	27.14min	9.49min	26.48min	8.65min	15.69min	24.20min	8.35min
YAGO2s	$>$ 1 day	$>$ 1 day	$>$ 1 day	1h 7min	1h 12min	$>$ 1 day	59.38min
DBpedia 2.0	$>$ 1 day	$>$ 1 day	$>$ 1 day	45.11min	46.37min	$>$ 1 day	46.88min
DBpedia 3.8	$>$ 1 day	$>$ 1 day	11h 46min	8h 35min	7h 33min	10h 11min	7h 6min
Wikidata	$>$ 1 day	$>$ 1 day	$>$ 1 day	1h 14min	7h 56min	$>$ 1 day	25.50min

TABLE 3.3 – Runtime and output comparison between AMIE and AMIE+ on different KBs. On YAGO2 (4), $l = 4$. On YAGO2 (const), the instantiation operator was switched on.

latter, we show the results in several categories :

1. Only refinement : only the improvements affecting the refinement process (Section 3.1) are active, namely the maximum rule length (MRL), the query rewriting (QRW) and the perfect rules (PR).
2. Only output : only the improvements affecting the output process are active, i.e., the confidence approximation and the confidence upper bounds, both with confidence threshold 0.1 (Section 3.2).
3. Output + MRL/QRW/PR : the output improvements and one of the refinement improvements are active.
4. Full : All improvements are active.

We first note that AMIE is not able to finish within a day for YAGO2s, DBPedia 2.0, DBpedia 3.8, and Wikidata. In contrast, AMIE+ can mine rules on all these datasets in a matter of hours, and even minutes. For YAGO2 (const), we can see that the full version of AMIE+ is 3.8x faster than AMIE. For YAGO2, this speed-up nearly doubles to 6.7x. This boost is mainly due to the improvements in the refinement process : AMIE+ with only these improvements is already 3.2x faster on YAGO2 (const) and 6.5x faster on YAGO2 than AMIE. This is not surprising since for YAGO2 most of the time is spent on refining rules (Table 3.1). Therefore, the improvements in this phase result in a significant gain.

Notice also that AMIE+ (only output) is only marginally faster than AMIE for the YAGO2 family of datasets. This is because the confidence approximation heuristic requires computing the join cardinalities for every pair of relations in the KB. This means that there is a trade-off between an initial additional cost for pre-computing these values and the potential savings. For the case of YAGO2, the output phase takes only around 9% of the overall mining time, i.e., the confidence evaluation is not really a problem.

For YAGO2s, DBpedia 2.0, DBpedia 3.8, and Wikidata, we see that using only the refinement improvements or only the output refinements is not enough. If we activate all improvements, however, AMIE+ is able to terminate in the majority of cases within an hour or in the worst case over-night.

Table 3.3 also shows the benefit of individual refinement improvements over the baseline of AMIE+ (only output). The improvement that offers the highest speedup (up to 6.7x) is the maximum rule length (MRL), closely followed by query rewriting (QRW, up to 5.1x speedup), whereas perfect rules (PR) rank last. This occurs because MRL is much more often applicable than QRW and PR. Besides, perfect rules are relatively rare in KBs. AMIE found, for instance, 1 perfect rule on YAGO2s and 248 (out of 5K) in DBpedia 3.8.

All in all, we find that AMIE+ can run on several datasets on which AMIE was not able to run. Furthermore, on datasets on which both can run, AMIE+ achieves a speed-up of up to 6.7x.

Output Comparison. Table 3.5 shows a comparison of AMIE and AMIE+ in terms of output (number of rules). For AMIE+ (full), we report the number of rules that were pruned by the confidence approximation. To assess the quality of the confidence approximation, we report in addition the *pruning precision*. The pruning precision is the ratio of rules for which the confidence approximation introduced in Section 3.2.2 overestimates the actual confidence. We calculate this ratio by counting the number of times that the heuristics produce a higher value than the real confidence (among the rules on which the approximation is applicable). For example, a pruning precision of 96% means that in 4% of the cases the system erroneously pruned rules with a confidence higher than 0.1. As for the runtime comparison, we set a threshold of 0.1 PCA confidence for AMIE. We also interrupted the system if it ran more than one day. In those cases, we report the output and the pruning precision until the point of interruption (denoted by a "*" in Table 3.5). We remark that in those cases, the pruning precision in Table 3.5 is not exact since it was computed by comparing the output of AMIE+ to the output of AMIE in a subset of the rules, i.e., those that were found until AMIE was interrupted.

As we can see, the pruning by approximation does not entail a serious decrease in the quality of the output : AMIE+ does not miss more than 5% of the rules with confidence above 10%. At the same time, the pruning yields a speed-up by a factor of up to 3, as Table 3.3 shows. Table 3.4 shows some examples of rules with high confidence that we mined.

3.4 AMIE+ Demo

This section is based on our work published in [41] and presents an interactive demo of AMIE+. Our demo lets the user understand the heuristics that govern the system's search strategy (Section 2.5.1) by allowing the user to navigate through the search space together with the system. The demo is available to users under `http://luisgalarraga.de/amie-demo`.

y :isCitizenOf(x, y) ⇒ *y :livesIn*(x, y)
y :wasBornIn$(x, y)\wedge$ *y :isLocatedIn*(y, z) ⇒ *y :citizenOf*(x, z)
y :hasWonPrize$(x, G.\ W.\ Leibniz)$ ⇒ *y :livesIn*$(x, Germany)$
y :hasWonPrize$(x, Grammy)$ ⇒ *y :musicalRole*$(x, Guitar)$
d :countySeat(x, y) ⇒ *d :largestCity*(x, y)
d :jurisdiction$(z, y)\wedge$ *d :successor*(x, z) ⇒ *d :jurisdiction*(x, y)
w :ownedBy(x, y) ⇒ *w :subsidiary*(y, x)
w :relative$(y, z)\wedge$*w :sister*(z, x) ⇒ *w :relative*(x, y)

TABLE 3.4 – Some rules mined by AMIE on different datasets (y : YAGO, w : Wikidata, d : DBpedia).

	AMIE	AMIE+(full)		
KB	Rules	Rules	Pruned	Prun. prec.
YAGO2	68	68	24	100.00%
YAGO2 (c)	15634	15634	24	100.00%
YAGO2 (4)	645	645	203	100.00%
YAGO2s	94*	94	78	100.00%
DBpedia 2.0	24308*	112865	5380	98.26%
DBpedia 3.8	2470*	5087	2621	98.41%
Wikidata	889*	1512	773	95.35%

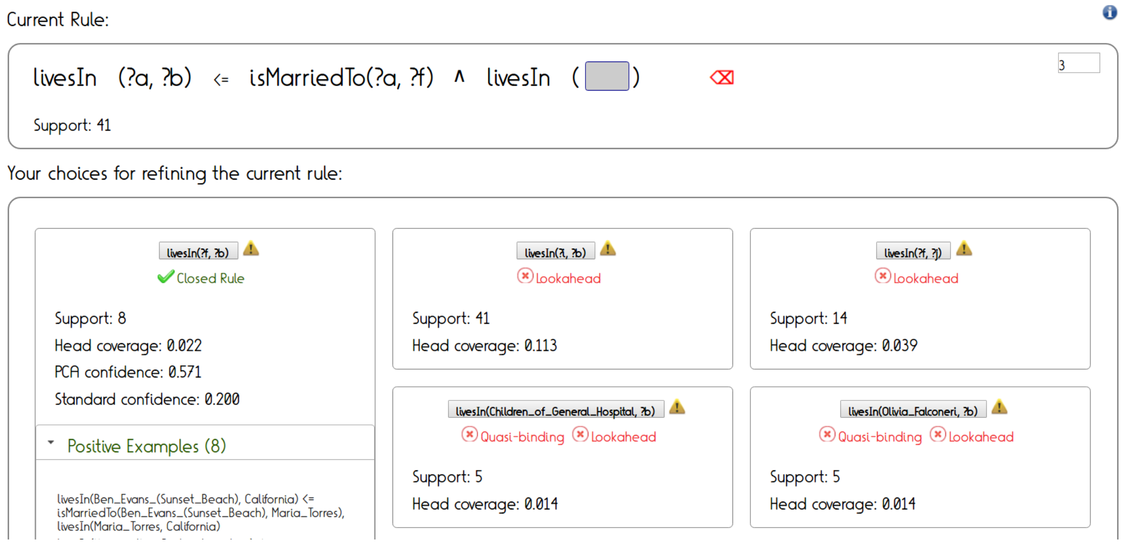
TABLE 3.5 – Output comparison of AMIE (PCA conf ≥ 0.1) and AMIE+ full. Starred : output after processing for 1 day. On YAGO2 (4), $l = 4$. On YAGO2 (const), the instantiation operator was switched on.

3.4.1 Interface

To demonstrate the AMIE+ algorithm to the user, our demo shows the rule construction process, starting from the empty rule. At each step, the user can add a new atom $r(x, y)$. This involves two steps : (1) choosing the relation r and (2) choosing the arguments x, y. The second step offers various permutations of arguments : Each argument can be a variable (either known or fresh) or an entity (chosen from the set of entities that lead to the highest support). In this section we depart from the traditional notation $\boldsymbol{B} \Rightarrow H$ and write rules as $H \Leftarrow \boldsymbol{B}$ for usability reasons, i.e., rules are constructed in a left to right fashion as the user were writing. For instance, imagine the user wants to follow the path to mine the rule

$$livesIn(a, b) \Leftarrow isMarriedTo(a, f) \wedge livesIn(f, b)$$

The demo starts with the empty rule "... ⇐ ..." and asks the user to select a relation for the head atom H. The system shows as choices a list of relations ranked by size (support). Once the user picks a relation (i.e., *livesIn* in our example), the system displays all possible head atoms using that relation, ranked by support. This includes, in our


Current Rule:

livesIn (?a, ?b) <= isMarriedTo(?a, ?f) ∧ livesIn ( )

3

Support: 41

Your choices for refining the current rule:

livesIn(?f, ?b)
✓ Closed Rule

Support: 8
Head coverage: 0.022
PCA confidence: 0.571
Standard confidence: 0.200

Positive Examples (8)

livesIn(Ben_Evans_(Sunset_Beach), California) <=
isMarriedTo(Ben_Evans_(Sunset_Beach), Maria_Torres),
livesIn(Maria_Torres, California)

livesIn(?l, ?b)
Lookahead

Support: 41
Head coverage: 0.113

livesIn(?f, ?j)
Lookahead

Support: 14
Head coverage: 0.039

livesIn(Children_of_General_Hospital, ?b)
Quasi-binding Lookahead

Support: 5
Head coverage: 0.014

livesIn(Olivia_Falconeri, ?b)
Quasi-binding Lookahead

Support: 5
Head coverage: 0.014

FIGURE 3.1 – Mining a closed Horn rule

example, atoms such as $livesIn(a, b)$ and *livesIn*(a, *California*). Constants are always bound to the least functional argument of relations (see Section 1.3.3). In our example, it is more reasonable to predict the place of residence of a person, that all the people that live in a place (Section 2.4). Once the arguments are fixed, the atom is appended to the rule, and the user can choose the relation for the next atom. In the example, the user would select the relation $isMarriedTo$. In the next step, the system shows the user all possible atoms produced by the mining operators presented in Section 2.5. For the relation $isMarriedTo$, the $\mathcal{O}_D$ operator yields the atoms $isMarriedTo(a, f)$ and $isMarriedTo(f, a)$. Our example rule can be produced by selecting the first atom followed by a new atom using the relation $livesIn$ (Figure 3.1). The user can also, at any point, decide to backtrack and to remove the last atom from the rule in order to explore a different path in the search space.

At each step in the process, the user has complete freedom to choose his options. However, the system also shows the metrics and pruning strategies that AMIE+ would apply. These techniques include :

1. **Perfect rules.** If the rule is closed and has a PCA confidence of 1.0, then no refinement can make the rule better, because confidence cannot increase and support can only decrease. Hence, the rule is output and removed from the queue (Section 3.1).
2. **Skyline technique.** If the new rule has a lower confidence than the rule that it is derived from, then the rule is just silently enqueued for further refinement, but not output. This is because it will have lower confidence and lower support than the previous rule. This heuristic was designed to avoid over-specifications of the same logical rule (Section 2.5).
3. **Maximum rule length.** If the maximum permitted length of a rule is n, then AMIE

will not apply the operator $\mathcal{O}_\mathcal{D}$ at length $n-1$. This is because the operator adds a new variable, and thus the rule cannot be closed within the limit of n atoms. This technique is called the *lookahead* (Section 3.1).

4. **Quasi-bindings.** When AMIE adds a new instantiated atom, it will exclude atoms where the variable of this atom can bind to only one entity in the KB. This is because these cases induce a *quasi-binding* of the free variable, meaning that there is a shorter equivalent rule (Section 2.5.2).

Our demo shows for every possible choice of atoms the support, the head coverage, the standard confidence, and the PCA confidence that this atom would achieve. Thus, the user can see which choice achieves the best metrics, and why – much like AMIE+ does during the mining process. For illustration, we also show positive examples and negative examples under both the CWA and the PCA for the new rule.

3.4.2 Implementation

Our demo is implemented as a client-server application that lets the user drive the AMIE+ system step by step. We use YAGO [117] as KB. In order to guarantee reasonable response times in the interactive client-server setting, we created a sample of YAGO following the same procedure as in Section 2.6.2 : We took 10K random seed entities, and collected all the facts within a range of 3 hops from the seed entities, producing a sample of 35K facts. The server consists of a Java servlet that serves as interface to the AMIE+ codebase, namely the mining operators described in Section 2.5 and the in-memory triple store. The KB is loaded into memory only once when the servlet is initialized in the servlets container. The client side is a lightweight user-interface written in Javascript and HTML.

For the AMIE algorithm, it does not matter whether an unknown person or President Obama is an example or counter-example for a rule. For the user, in contrast, it is more illustrative to show prominent entities rather than unknown ones. Hence, we computed a relevance score for each entity e as :

$$relevance(e) := log(wikilength(e)) \times (incoming(e) + 1)$$

Here, $wikilength(e)$ is the length of the Wikipedia article of the entity (in bytes), and $incoming(e)$ is the number of Wikipedia articles linking to the article of e. Both numbers can be easily obtained from YAGO. We add 1 in the second term to guarantee that the score is a positive number. The relevance of a fact $r(x, y)$ is defined as the sum of the relevance scores of its arguments. This score is used to rank facts when displaying examples for rules, so that facts about prominent entities are preferred.

3.5 Conclusion

In this chapter, we have extended AMIE to AMIE+ by a series of pruning and query rewriting techniques, both lossless and approximate. As our extensive experiments have shown, AMIE+ runs on millions of facts in only a few minutes, being able to handle KBs up to 11M facts in a few hours.

The scalability boost provided by AMIE+ clears the way for multiple applications in data analytics and maintenance. In the upcoming chapters, we explore some of those applications. The framework provided by AMIE will help us in the tasks of wikilinks semantification, ontology schema alignment, canonicalization of open knowledge bases and prediction of completeness.

Chapitre 4

Wikilinks Semantification

In Chapter 2 we presented AMIE, a system capable of learning Horn rules on large, potentially incomplete KBs. In this chapter, we show one of the applications of rule mining with AMIE, namely the task of wikilinks semantification. This is the task of predicting the semantic relation between two entities that are connected by a hyperlink in Wikipedia. We refer to those hyperlinks as wikilinks.

The content of this chapter is organized in five sections. Section 4.1 provides a motivation for this work. Section 4.2 discusses a few existing approaches that make use of wikilinks for inference tasks in KBs. Section 4.3 describes how to mine semantification rules on a KB with wikilinks. It also details the process of using semantification rules to predict candidate relations for wikilinks in DBpedia. Section 4.4, conversely, shows how wikilinks can improve rule mining, i.e., increase the quality of the rules. Section 4.5 concludes the chapter.

This chapter builds upon the work published in the following workshop article :

— Luis Galárraga, Danai Symeonidou, Jean-Claude Moissinac. *Rule Mining for Semantifiying Wikilinks*. Proceeding of the 8th Workshop on Linked Open Data. Florence, Italy, 2015.

4.1 Motivation

Wikipedia-centric KBs such as DBpedia [7] or YAGO [117] store the hyperlink structure between articles in Wikipedia. That is, if the page of Barack Obama links to the page of the Nobel Peace Price, these KBs store a triple of the form $linksTo$(*BarackObama*, *NobelPeacePrize*). We call such facts, *wikilinks*. Even though wikilinks account for more than 25% of the non-literal facts in DBpedia, they are rarely exploited, with a few approaches using them for the type prediction [95, 96]. Nevertheless, the fact that two entities are connected via a wikilink often suggests a semantic connection between them. We aim at discovering the exact meanings of such connections.

Some wikilinks are already semantified in KBs. YAGO and DBpedia, for example, know that Barack Obama links to USA and is also a citizen and the President of that country. KBs can extract such information because it is usually available in the infoboxes. However, if the information lies somewhere outside the infoboxes, KBs will not see it, leading to unsemantified wikilinks (see [64, 129] for automatic population of infoboxes from text). This is the case for 89% of wikilinks in DBpedia. For instance, the Wikipedia article of Barack Obama links to the article of the 2009 Nobel Prize, but DBpedia does not know that he won the Nobel Prize. In some other cases, the semantic connection encoded in a wikilink can be meaningless or too vague to be modeled in the KB. For example, Obama's article also links to the articles for cocaine and ovarian cancer.

In this work, we show how to leverage the already semantified wikilinks to semantify the others. This is achieved by learning frequent semantic patterns from the relations in the KB and the wikilinks. If we observe that politicians often link to the prizes they have won, we can suggest that unsemantified wikilinks from politicians to prizes convey a $hasWon$ relationship. This pattern can be expressed as a logical rule :

$$linksTo(x,y) \wedge type(x, Politician) \wedge type(y, Prize) \Rightarrow hasWon(x,y)$$

In our example with Barack Obama, this rule would predict the concrete fact $hasWon(BarackObama, NobelPeacePrize)$. Such predictions could be proposed as candidate facts to populate KBs.

We use AMIE [43] to mine logical rules like in our example. We then use the rules to draw conclusions and compute a list of the most likely candidate relations between the entities of unsemantified wikilinks. Using a straightforward inference method, we can discover (with high precision) meanings for 180K unsemantified wikilinks in DBpedia.

Rule Mining can semantify wikilinks, but vice versa, wikilinks can also improve rule mining. We observe that sometimes, wikilinks can increase the confidence of the obtained rules. For instance, assuming that a rule mining approach learns the rule :

$$type(y, SportsTeam) \wedge currentMemberOfTeam(x,y) \Rightarrow hasTeam(x,y)$$

We observe that by requiring a wikilink between the entities, i.e., adding an atom of the form $linksTo(x,y)$ to the body, we achieve higher confidence. This observation could be leveraged by data inference and link prediction approaches. It also provides additional insights about the KB.

4.2 Related Work

Wikilinks semantification is a form of link prediction in KBs. Also referred as relational learning, this task aims at predicting relations between entities in a KB. For a detailed discussion of the state of the art in link prediction, we refer to Section 2.2.5.

All previous link prediction approaches tackle the problem in a general way. In this chapter we target the special case of predicting semantic links for entities for which there exists a signal of semantic connection in the form of a wikilink. Some approaches have leveraged the semantic value conveyed by wikilinks for type inference in KBs. The

Domain	Range	Relation - % occurrences					
Person	Person	successor	18%	assocBand	11%	assocMusicArtist	11%
Person	Place	birthPlace	56%	deathPlace	18%	nationality	8%
Person	Org.	team	53%	almaMater	8%	party	5%
Place	Place	isPartOf	29%	country	28%	location	13%
Place	Person	leaderName	42%	architect	32%	saint	12%
Place	Org.	owner	24%	tenant	16%	operatedBy	12%
Org.	Org.	sisterStation	18%	assocBand	15%	assocMusicArtist	15%
Org.	Person	currentMember	22%	bandMember	20%	formerBandMember	20%
Org.	Place	location	19%	city	17%	hometown	13%

TABLE 4.1 – Top-3 relations encoded in wikilinks between instances of Person, Place and Organization in DBpedia.

work presented in [96] represents the set of wikilinks as a directed graph where each entity is replaced by its more specific type in the DBpedia type hierarchy. The method discovers frequent subgraph patterns on the resulting graph. These are called Encyclopedic Knowledge Patterns (EKP). EKPs can be used to describe classes of entities and therefore predict the types for untyped entities, e.g., instances of soccer players will often link to instances of coaches and soccer leagues. While this method also makes use of the instance information to mine patterns, it does not aim at discovering relations between entities. Thus, it does not make use of any other relations holding between the endpoints of the wikilinks. In the same spirit, [95] builds upon EKPs and uses the instance information to map both entities and classes to a vector space. A similarity function on this space is used to compute the distance of an entity to the prototypical vectors of classes and predict the types for untyped entities.

4.3 Predicting Semantics for Wikilinks

4.3.1 Mining Semantification Rules

Our approach to semantify wikilinks relies on the intuition that (a) wikilinks often convey a semantic connection between entities, (b) some of them are already semantified in KBs, (c) the types of the entities in the wikilink define the signature of its implicit relation, and (d) the already semantified wikilinks can help us semantify the others. The already semantified wikilinks constitute our training set. From this training set, we mine a set of semantic patterns in the form of logical rules.

To justify our intuition, we look at the types of the endpoints of semantified wikilinks in DBpedia. We restrict our analysis to the classes *Person*, *Place* and *Organization*. Table 4.1 shows the most common relations holding between pairs of those entities for which there exists at least one wikilink.

For example, we observe that when a person links to a place, in 56% of the cases, the person was born in that place. Similarly, when an organization links to a place, in 19% of the cases, this corresponds to its location. We also observe that in our dataset,

$linksTo(x,y) \wedge is(x, Town) \wedge is(y, Country) \Rightarrow country(x,y)$
$linksTo(x,y) \wedge lieutenant(y,x) \wedge is(x, OfficeHolder) \wedge is(y, Governor) \Rightarrow governor(x,y)$
$linksTo(x,y) \wedge largestCity(y,x) \wedge is(x, City) \wedge is(y, AdminRegion) \Rightarrow partOf(x,y)$

TABLE 4.2 – Some semantification rules mined by AMIE on DBpedia.

81% of the links for these classes are not semantified. Rule mining techniques can help us learn the patterns suggested by Table 4.1 and semantify more links.

We start by constructing a training set $\mathcal{K}$ from DBpedia 3.8[1] consisting of 4.2M facts and 1.7M entities, including people, places and organizations. We enhance this dataset with the type information about the entities, i.e., 8M *rdf :type* statements, and the wikilinks between those entities. Since we can only learn from already semantified wikilinks, we restrict the set of wikilinks to those where both endpoints participate in a relation in the data, i.e.,

$$linksTo(a,b) \in \mathcal{K} \Leftrightarrow \exists\, r, r', x, y : (r(x,a) \vee r(a,x)) \wedge (r'(y,b) \vee r'(b,y))$$

This procedure led us to a training set $\mathcal{K}$ with a total of 18M facts. We ran AMIE on this dataset and configured it to mine *semantification rules*. These are closed Horn rules of the form :

$$linksTo^*(x,y) \wedge \boldsymbol{B} \wedge type(x,C) \wedge type(y,C') \Rightarrow r(x,y)$$

where *linksTo* is an alias for *wikiPageWikiLink* (DBpedia relation for wikilinks), *linksTo** denotes either *linksTo* or *linksTo*$^{-1}$, "*type*" is a synonym for *rdf :type* and $\boldsymbol{B}$ is a conjunction of up to 2 atoms. To mine semantification rules, we tweaked AMIE to enforce atoms of the form $linksTo^*(x,y)$ and $type(x,C)$, $type(y,C')$ right after the creation of the head atom. We explore all C, C' that produce rules above the given support and confidence thresholds. With support and PCA confidence thresholds 100 and 0.2 respectively, AMIE found 3546 semantification rules on the training set $\mathcal{K}$. Table 4.2 shows examples of those rules.

4.3.2 Predicting Relations for Wikilinks

Once we have mined semantification rules on a KB $\mathcal{K}$, we use them to draw a set of predictions of the form $p := r(a,b) \notin \mathcal{K}$ as in Section 2.7. We restrict even further the set of predictions by requiring the arguments to be the endpoints of unsemantified wikilinks, i.e., $r(a,b) : \nexists\, r' : r' \neq linksTo \;\wedge\; r'(a,b) \in \mathcal{K}$.

Recall that those predictions may be deduced by multiple rules since AMIE explores the search space of rules in an exhaustive fashion. Moreover, those rules have different degrees of confidence. We resort to Formula 2.4 for joint-prediction to account for these observations. Given an unsemantified wikilink $w := linksTo(a,b)$, Formula 2.4 allows us to propose a list of candidate meanings for w. If among the set of predictions there are

1. We learn rules on DBpedia 3.8 to corroborate some of their predictions automatically in DBpedia 3.9

Precision@1	**Precision@3**
0.77 ± 0.10	0.67 ± 0.07

TABLE 4.3 – Average MAP@1 and MAP@3 scores for semantification of wikilinks on DBpedia.

WikiLink	**Semantification candidates**
Interstate 76 (west) → Colorado State Highway	*routeJunction* (1.0)
J. Bracken Lee → Herbert B. Maw	*predecessor* (1.0), parent(0.998), governor(0.882)
WHQX → WTZE	*sisterStation* (1.0)
Set the Tone (band) → Zones (band)	*associatedMusicalArtist*(1.0), *associatedBand*(1.0)

TABLE 4.4 – Some examples of semantification candidates for wikilinks. The correct candidates are in italics.

several facts of the form $r_i(a, b)$, then each relation r_i is a semantification candidate for w with confidence $score^*(r_i(a, b))$ (Equation 2.4). For each unsemantified link, we propose a list of semantification candidates sorted by PCA confidence. Our procedure proposes relation candidates for 180K unsemantified wikilinks in the training set. Since we can only corroborate 1% of our predictions in DBpedia 3.9, we evaluate the precision of our approach on a random sample of 60 unsemantified wikilinks as follows : For each wikilink we count the number of correct candidates at top 1 and top 3 of the ranking ; we then add up these counts and divide them by the total number of candidates at top 1 and top 3 respectively. This gives us an estimation of the precision of our approach. Table 4.3 shows the estimated precision values drawn from the sample as well as the size of the Wilson Score Interval [16] at confidence 95%. The results imply that, for example, the precision at top 1 for the whole set of wikilinks lies in the interval 77% ± 10% with 95% probability.

Table 4.4 shows some examples of wikilinks and the ranking of semantification candidates proposed by our approach. The number in parentheses corresponds to the confidence of the semantification candidate. The candidates evaluated as correct according to the our evaluation are in italics.

4.4 Wikilinks for Rule Mining

If AMIE finds two rules $\boldsymbol{B} \Rightarrow H$ and $\boldsymbol{B} \wedge B_{n+1} \Rightarrow H$ and the latter has lower confidence, the system will not output it because it is worse in all dimensions (Section 2.5.1). We therefore investigate the confidence gain carried by the addition of wikilink atoms in rules.

We first run standard AMIE on the DBpedia mapping-based triples. In a second run, we add the wikilinks to the mapping-based triples and instruct AMIE to mine rules of the form $\boldsymbol{B} \wedge linksTo^*(x, y) \Rightarrow r(x, y)$. We did not change the standard pruning strategies, therefore AMIE is allowed to prune the longer rule with the $linksTo^*(x, y)$ atom, if it

Rules without wikilink	857
Rules with wikilink	1509
Rules with confidence gain	1389
Weighted average gain (wag)	0.03
Rules with gain ≥ 0.1	139

TABLE 4.5 – Statistics about rule mining with and without wikilinks.

Rule	**Δ-conf**
$operator(x, y) \land is(x, Stadium) \Rightarrow location(x, y)$	0.53
$debutTeam(x, y) \Rightarrow team(x, y)$	0.28
$officialLanguage(x, y) \Rightarrow spokenIn(x, y)$	0.19

TABLE 4.6 – Confidence gain for some rules when specialized with a *linksTo* atom on the head variables.

does not lead to a confidence gain. In both cases, we set a threshold of 100 positive examples for support and no confidence threshold. We report our findings in Table 4.5. We observe that requiring the head variables to be connected via a wikilink increases the number of rules from 857 to 1509. This occurs because in the second run, AMIE sometimes mines extensions of the rules with the $linksTo^*$ atom. In other words, for some rules, the addition of a wikilink atom provides a confidence gain. This is the case for 1389 rules as Table 4.5 shows. We are interested in finding how much confidence gain is carried by those rules. Thus, we define the *gain* of a wikilink rule as a variant of the gain metric used in association rule mining [11] :

$$gain(R) := supp(R) \times (pcaconf(R) - pcaconf(R_{\neg linksTo}))$$

That is, the gain of a wikilink rule is the product of its support and the difference in confidence with respect to the rule without the $linksTo^*$ atom. Table 4.5 reports an average gain of 0.03. This indicates that, in the majority of cases, the wikilinks do not provide a significant confidence gain to rule mining in DBpedia. The reason lies on the fact that for 99% of the triples in the DBpedia mapping-based dataset, there is a wikilink between the arguments of the triples. This implies that the addition of a wikilink atom does not provide additional information to most of the rules. On the other hand, for 10% of the rules the gain can be higher than 0.1. We show some of those rules with their corresponding confidence gain in Table 4.6. This occurs because more than 100K triples in our dataset do not have a wikilink between the subject and the object. Thus, for those entities, the atoms of the form $linksTo^*$ make a difference, which leads to a confidence gain that can be used to improve the quality of the rules.

4.5 Conclusion

While none of the major Wikipedia-centric KBs make further use of the wikilinks, in this work we have shown that they often encode latent relations between entities. Such relations may not be captured in KBs. We have shown that rule mining techniques and naive inference methods are a feasible method to accurately discover those implicit semantics. This wikilink semantification task can be seen as a particular case of the link prediction problem in KBs. With this work, we aim at turning the attention to the wikilinks, as they convey valuable information that can help improve the completeness of KBs. All the datasets and experimental results produced by this work are available under `http://luisgalarraga.de/semantifying-wikilinks`.

Chapitre 5

Schema Alignment

This chapter illustrates the applicability of rule mining to the problem of KB schema alignment, an important data integration task. Data integration is the process of reconciliating data produced by independent sources about a common domain. Data integration is crucial for the Semantic Web, given its primary goal of providing a unified web of entities in the presence of numerous independent contributors.

This chapter is structured as follows. Section 5.1 discusses in detail the problem of interoperability in the Semantic Web and motivates our work on KB schema alignment. Section 5.2 presents the related work for the tasks of instance and schema alignment in KBs. In Section 5.3 we present the requirements of our method for schema alignment. In Section 5.4 we show the family of alignment mappings that can be found with our method on real-world KBs. We conclude the chapter in Section 5.5 with a brief outlook of future work in this area.

The work presented in this chapter is based on the following publication :

— Luis Galárraga, Nicoleta Preda, Fabian Suchanek. *Mining Rules to Align Knowledge Bases*. Proceedings of the 3rd Workshop on Automated Knowledge Base Construction. pp 43–48. San Francisco, USA, 2013.

5.1 Motivation

Many of publicly available KBs share a lot of information. For instance, KBs that feed from Wikipedia, such as YAGO, DBpedia, Wikidata, Freebase, and Google's knowledge graph, share a large part of their entities. Consequently, many of the (public) KBs have been connected in the Linked Open Data Cloud [71] (LOD). The LOD cloud provides instance alignments in the form of *sameAs* links between equivalent entity identifiers across KBs. Altogether, the LOD cloud provides hundreds of millions of such links, thus interconnecting billions of statements.

These links, however, concern mostly instances. The schema of the KBs, i.e., the class hierarchy and the relations of the KBs, have not yet been mapped at large scale. This entails that, although the instances are aligned, the data is not interoperable. A query formulated in the schema of one KB will not have answers in another KB – even if the desired information is there, and even if the instances have been linked between the two resources. For example, assume that a user of YAGO asks for the parents of Elvis Presley, and assume that YAGO knows the father of Elvis, and DBpedia knows his mother. The user will not get the desired results, if she does not know that the *hasChild* relationship of YAGO is formulated as its inverse, i.e., *hasParent* in DBpedia. Hence, despite the great advances of linked open data, the KBs are still to a large degree disconnected databases.

A standard data integration solution based on manually defined mappings would not scale-up to the hundreds of KBs on the Semantic Web. Recent work [116] has allowed finding relation and class alignments across KBs at large scale. However, this approach requires the KBs to have the same structure : One relation in one KB has to correspond to one relation in the other. Real-data examples show us that this is not always the case. For example, if the user asks for the country of birth of Elvis, then one KB may express this by the relationship *wasBornInCountry*. Another KB, in contrast, may require a join of the relationship *wasBornInCity* with the relationship *locatedInCountry*. This problem is particularly prevalent in cases where one KB distinguishes between the entity and its label, while the other one does not. For example, one KB could say *sang(Elvis, 'All Shook Up')*, while the other one could say *sang(Elvis, AllShookUp)*, *label(AllShook-Up, 'All Shook Up')*. Such a structural mismatch would derail any alignment system that assumes isomorphic KBs. Consequently, any query evaluation algorithm that uses them would miss relevant query plans.

Another important issue faced by data integration systems is the translation of literal values from one KB to another. For example, different KBs may use different labels for the same entities. MusicBrainz uses abbreviated names for countries, stating for example *livedIn(Elvis, 'DE')*. YAGO, on the other hand, uses complete names for countries. If the data of MusicBrainz were integrated directly into YAGO, the result would be inconsistent.

This leaves us to conclude that, even though KBs talk about the same things in the same representation format, and even though they may complement each other, they speak in different "languages". The reason is a structural mismatch between the schemas of the KBs. Our vision is that this structural mismatch be overcome. We would like to see a Semantic Web where the KBs are not just linked by their instances, but also by their relations and classes – irrespective of structural differences, and across different schemas. This would allow users to make full use of the data and links of the Semantic Web.

Contribution. The contribution of this chapter is to illustrate how to mine schema alignments between two KBs. For this purpose, we resort to a simple, yet very effective technique. We assume that some of the instances of two KBs have already been aligned. Then we coalesce the two KBs to a single KB, and apply rule mining [43]. This yields what we call ROSA rules (Rule for Ontology Schema Alignment) : Logical rules that reveal complex relationships between properties and concepts of the KBs. This idea subsumes state-of-the-art ontology matching [116], in that it can find equivalences of relationships, equivalence to the inverse of a relationship, and the subsumption of relationships. However, it goes further by finding that one “hop” in one KB can correspond to a several hops in the other. It can find that one literal in one KB corresponds to a different constant in the other KB, or that one class in one KB corresponds to a relationship in the other. We hope that, by this illustration, we can open up the door for a new, and exciting, area of research.

5.2 Related work

Scope. Several aspects of KB alignment have been addressed in recent work : the alignment of classes [51, 57], the alignment of classes together with relations (T-Box) [8, 23, 110], and the alignment of instances (A-Box) [13, 63, 71]. Holistic approaches have been investigated in [36, 116, 121].

We agree that the alignment of instances has been solved to a large degree. However, we are concerned that state-of-the-art solutions for the schema alignment of KBs are not yet mature enough.

Instance-Based Approaches. Some approaches [36,116,121] can align the instances and the schema of two KBs at the same time. ILIADS [121] combines a clustering algorithm based on lexical, graph-structure and data instance statistics with a logical inference to check the consistency of the mappings. Most large KBs, however, do not come with the OWL axioms on which ILIADS relies. PARIS [116] develops a probabilistic model for graph alignment. The approach presented in [36] models a KB, expressed in OWL Lite, as a graph. Given two KBs, a set of similarity metrics are used to find equivalences between the nodes of the input KBs. These metrics are applicable to both the A-Box and T-Box of the KB and exploit multiple features such as neighbourhood similarity. However, all these approaches assume that one relation or class in one KB corresponds to one relation or class in the other KB. In practice, this is often not the case.

Schema Alignment. We refer to [111] for a comprehensive survey on ontology schema alignment. We mention the most relevant work surveyed in [111] and complement the discussion with more recent approaches. The CLIO system [82] implements schema alignment of relational databases. The system is based on a query discovery method that depends on database constraints provided by the user. Such constraints are normally not available in KBs, thus they must be defined by the user. Moreover we focus on the RDF data model. COMA++ [8] is a customizable utility designed to align the schema of two RDF KBs. It takes OWL descriptions and instance alignments as input and offers a set of strategies to align the different components of the schema, i.e., classes and properties. AgreementMaker [23] provides a layered architecture consisting of a set of matchers that implement specific tasks, e.g., alignmnent of classes. The output of a matcher can serve as input to another matcher, allowing the user to design a customized strategy to align the schemas of the input KBs. Both COMA++ and

AgreementMaker provide a user interface to drive the user in the alignment process. They also offer methods to evaluate the quality of the resulting alignments. Unlike our approach, these systems require heavy parameter tuning and some knowledge about the structure of the datasets.

The iMAP system [31] can learn schema alignments involving arbitrary functions like in $address \equiv concat(city, state)$, given a set of integrity constraints provided by the users. In our vision, the alignment of KBs would happen fully automatically and without the need for database constraints or expert knowledge. The work in [106] goes beyond the alignment of binary predicates and allows for schema integration of classes and relations via a mediated schema and a set of reification rules (to deal with n-ary relations). Such approach can learn, e.g., an alignment between the class $BirthEvent$ and the binary relations $birthPlace$ and $birthDate$. Our approach, in constrast, does not deal with reified facts as we focus on binary predicates.

Data Translation. The approach of [17] argued for the introduction of more complex mapping rules, which are able to express data translations. However, such formalisms have not yet been widely adopted due to scalability issues. In this work, we propose to focus on mining a small, yet expressive set of mappings patterns, which capture many real-data cases. The actual translation of data from one KB to another KB shares resemblance to the query discovery problem [82]. Query discovery and schema matching are seen as complementary and independent problems [72]. Query discovery solutions are not directly applicable to KBs as they are data model dependent or rely on database constraints. Furthermore, the data transformations that we envision go beyond data restructuring.

Association Rule Mining for Ontology Alignment. To our knowledge, there are only few works that mine rules for KB alignment [27, 119]. These works focus exclusively on the alignment of hierarchies of entities. Our vision goes beyond this goal. We would like to express more complex mappings and actual data transformations.

5.3 Method

We are interested in discovering complex schema relationships between two given KBs $\mathcal{K}_1$ and $\mathcal{K}_2$ in RDF format. We denote the set of instances and relations of $\mathcal{K}_1$ and $\mathcal{K}_2$ as $\mathcal{E}_1$, $\mathcal{E}_2$, $\mathcal{R}_1$ and $\mathcal{R}_2$ respectively (Section 1.3.1). We assume that some instances of the KBs have already been aligned, e.g., by means of $sameAs$ links. This can be done by a (partial) substitution ϕ, which maps the instances of $\mathcal{E}_1$ to the *sameAs* counterparts from $\mathcal{E}_2$ if any, or to themselves otherwise. The substitution ϕ leaves literals unchanged. As pointed out in [116], different KBs may use the same relation (as given by a URI) with different meanings. Therefore, we use a substitution t that substitutes all relation names in $\mathcal{E}_1$ so as to make sure they are different from the relation names in $\mathcal{E}_2$. With

$$r(x,y) \Rightarrow r'(x,y) \qquad \text{(R-subsumption)}$$
$$r(x,y) \Leftrightarrow r'(x,y) \qquad \text{(R-equivalence)}$$
$$type(x,C) \Rightarrow type'(x,C') \qquad \text{(C-subsumption)}$$
$$r_1(x,y) \wedge r_2(y,z) \Rightarrow r'(x,z) \qquad \text{(2-Hops alignment)}$$
$$r(z,x) \wedge r(z,y) \Rightarrow r'(x,y) \qquad \text{(Triangle alignment)}$$
$$r_1(x,y) \wedge r_2(x,V) \Rightarrow r'(x,y) \qquad \text{(Specific R-subsumption)}$$
$$r(y,V) \Rightarrow r'(x,V') \qquad \text{(Attr-Value translation)}$$
$$r_1(x,V_1) \wedge r_2(x,V_2) \Rightarrow r'(x,V') \qquad \text{(2-Value translation)}$$

FIGURE 5.1 – ROSA Rules, $r, r_1, r_2 \in \mathcal{R}_1, r' \in \mathcal{R}_2$.

this in mind, we coalesce the two KBs as follows :

$$\mathcal{K} := \{\hat{r}(\phi(x), \phi(y)) \mid r(x,y) \in \mathcal{K}_1 \wedge \hat{r} = t(r)\} \cup \mathcal{K}_2$$

Our coalesced KB subsumes $\mathcal{K}_1$ and $\mathcal{K}_2$. We could restrict ourselves to the part that is common to both KBs, but then many alignments are lost because of missing data (KBs are highly incomplete). We leave the detailed study of different coalescing techniques for future work.

On the coalesced KB $\mathcal{K}$, we will mine rules. We are particularly interested in rules that express KB alignments. We call them ROSA rules :

Definition 1. A ROSA rule from a KB $\mathcal{K}_1$ to a KB $\mathcal{K}_2$ is a rule mined on the coalesced KB $\mathcal{K}$, such that the relations of the body belong to $\mathcal{R}_1$, and the relation of the head belongs to $\mathcal{R}_2$.

This definition is asymmetric in the sense that we can mine ROSA rules from $\mathcal{K}_1$ to $\mathcal{K}_2$ and from $\mathcal{K}_2$ to $\mathcal{K}_1$. ROSA rules express one type of cross-schema alignments.

Rule Patterns. Figure 5.1 groups some useful ROSA rules of up to 3 atoms into patterns. Arguments in lower case denote variables, whereas uppercase letters refer to constant values. Since we assume that every relation r is also present in its inverse form r^{-1}, the patterns also comprise rules that involve a swap of arguments. In the next section, we discuss each of these patterns in detail with examples on real-worlds KBs.

5.4 Experiments

Our goal is to qualitatively analyze the complex schema alignments that are necessary to make two KBs interoperable. Some of those alignments are given by ROSA rules.

5.4.1 Setup

Given two KBs, we can define schema mappings from the first KB to the second, and from the second to the first. In the following, we assume that we mine ROSA rules from a KB $\mathcal{K}_1$ to a KB $\mathcal{K}_2$ on their coalesced KB $\mathcal{K}$. We used the following KBs for our experiments :

— **YAGO 2s** : We used the facts about instances contained in the datasets *yagoFacts* and *yagoLiteralFacts*, with 2.9M entities and 22.8M facts in total. We also used, though separately, the instance data contained in *yagoSimpleTypes*, which comprises 5.4M *rdf :type* statements.

— **DBpedia 3.8** : We used the *person data* and *raw infobox properties* datasets, which together contain 11M facts about 2.1M entities. We also used the *ontology infoboxes* dataset, which has 13.2M *rdf :type* statements about 2.3M entities.

— **Freebase** : We used information about people, which comprises 19M facts about 2.7M subjects. In addition, we used the instance facts, which comprise 140M *rdf :type* statements about 74M entities.

— **IMDb** : We used a custom crawl of IMDb, similar to the one in [116]. It comprises 722K entities and 12M facts.

We use the namespace prefixes Y, D, F, and I for these KBs, respectively. ROSA rules require pre-existing instance alignments. For the first three KBs, we used the instance alignments from the Linked Data cloud. For IMDb, we used the gold standard instance alignments provided by [116]. For the actual mining of rules, we use AMIE with its default settings (1% head coverage).

Quality. The confidence scores provided by AMIE serve as quality metrics for our ROSA rules. Due to the inherent incompleteness of KBs and the open world assumption, we prefer the PCA confidence over the standard confidence as correctness metric.

5.4.2 Simple Mappings

In this section we provide examples of simple schema mappings mined with our approach on our test KBs. These mappings correspond to ROSA rules of 2 atoms, without arbitrary constants except for type statements of the form $type(x, C)$. For each rule pattern, we show the top 3 examples in terms of confidence.

R-subsumption. The R-subsumption rule holds when two ontologies contain semantically close relations with different levels of specificity. In this respect, the rule can capture *subPropertyOf* relationships, as defined in RDF Schema [103]. For instance, in YAGO the relationship between an author and his œuvre is labelled generically with *Y :created*, while DBpedia defines more specific relations such as *D :writer* for authors or *D :musicalArtist* for singers. To show this, we ran AMIE on a coalesced KB built from

DBpedia and YAGO. AMIE found 360 R-subsumption alignments. Due to the incompleteness and noise of the data, it is unlikely to find perfect subsumption rules, i.e., rules with confidence 100% are rare. We show the top 3 with their PCA confidences :

$$\textit{D :musicalArtist}(x,y) \Rightarrow \textit{Y :created}(y,x) \quad (90\%)$$
$$\textit{D :musicalBand}(x,y) \Rightarrow \textit{Y :created}(y,x) \quad (90\%)$$
$$\textit{D :mainInterest}(x,y) \Rightarrow \textit{Y :isInterestedIn}(x,y) \quad (88\%)$$

We look only at R-subsumptions $A \Rightarrow B$ whose inverse $B \Rightarrow A$ has a low PCA confidence (i.e., $< 50\%$) because otherwise the subsumption is an R-equivalence.

R-equivalence. If two relations r and r' subsume each other, then they are *semantically equivalent*. In our framework, this translates to two ROSA implications. We define the confidence of an equivalence rule as the minimum of the PCA confidences of the two implications. On YAGO and DBpedia, we mined a total of 88 R-equivalence rules. The top 3 by confidence are :

$$\textit{Y :directed} \Leftrightarrow \textit{D :director}^{-1} \quad (98\%)$$
$$\textit{Y :wroteMusicFor} \Leftrightarrow \textit{D :musicBy} \quad (97\%)$$
$$\textit{Y :isCitizenOf} \Leftrightarrow \textit{D :nationality} \quad (96\%)$$

C-subsumption. A C-subsumption is a rule of the form

$$type(x, C) \Rightarrow type'(x, C')$$

where *type* is the *rdf :type* relationship in one KB, and $type'$ is the *rdf :type* relationship in the other KB. If all instances of class C are also instances of class C', then C' subsumes C, i.e. *rdfs :subClassOf*(C, C'). Class alignment is a crucial task in ontology integration, because instance information frequently constitutes a significant part of the contents of any ontology. As we did for R-subsumption rules, we used AMIE to align the instance data of YAGO and DBpedia. We show the top 3 alignments (from a total of 59) where the inverse implication holds with low confidence.

$$\textit{Y :type}(x, \textit{Y :Site}) \Rightarrow \textit{D :type}(x, \textit{D :PopulatedPlace}) \quad (97\%)$$
$$\textit{Y :type}(x, \textit{Y :Site}) \Rightarrow \textit{D :type}(x, \textit{D :Settlement}) \quad (95\%)$$
$$\textit{D :type}(x, \textit{D :Athlete}) \Rightarrow \textit{Y :type}(x, \textit{Y :Person}) \quad (91\%)$$

C-equivalence patterns follow immediately by combining C-subsumption rules with their corresponding inverse implications.

5.4.3 More complex patterns

Rules with arbitrary constants or with more than 2 atoms increase the size of the search space drastically. As a consequence, rule mining finds many more rules. Some of the rules are *soft rules*. These do not express an a priori alignment, but rather a correlation that happens to be in the data. The rules can still be of use for reasoning [17], but are of more contingent nature. The following patterns all produce some soft rules with high PCA confidence. Therefore, we show as examples some handpicked interesting rules from the results. Our goal is to show that some quite complex alignments do exist between the KBs, even if we cannot identify them fully automatically yet.

2-Hops subsumption. A KB can express a relationship between two entities by passing through an intermediate entity (a blank node for example), while another KB expresses the same relationship by a single fact. This structural difference between two KBs can be captured by 2-hops subsumption rules of the form

$$r_1(x,y) \wedge r_2(y,z) \Rightarrow r'(x,z)$$

For instance, in IMBb, the attribute *country-of-birth* points to the name of the country, while in YAGO it points to the country entity, which has a *label* attribute. By running AMIE on YAGO and IMDb, we mined the ROSA rule

$$\textit{Y :wasBornIn}(x,y) \wedge \textit{Y :label}(y,z) \Rightarrow \textit{I :bornIn}(x,z) \qquad (37\%)$$

Some of the top 2-hops subsumption alignments found by AMIE between YAGO and Freebase are soft rules :

$$\textit{Y :married}(x,y) \wedge \textit{Y :child}(y,z) \Rightarrow \textit{F :children}(x,z) \qquad (73\%)$$

Soft rules may not hold for all instances. However, they should not be considered as wrong. Recent work [17] proposed a reasoning framework that can make use of such rules when dealing with incomplete data.

Triangle Alignments. The case where the body atoms of a 2-hops subsumption have the same relation is particularly interesting, because it can capture sibling relationships, co-author links and other relationships that denote co-participation. We call such rules *triangle alignments*. As an example, consider the following rule that we mined on YAGO and Freebase :

$$\textit{Y :child}(x,y) \wedge \textit{Y :child}(x,z) \Rightarrow \textit{F :sibling}(y,z) \qquad (37\%)$$

Specific R-subsumption. An R-subsumption may become more specific by adding an attribute-value constraint to one of the arguments in the body of the rule. The following examples were extracted from YAGO and Freebase :

$$\textit{Y :graduated}(x,y) \wedge \textit{Y :type}(y, \textit{University}) \Rightarrow \textit{F :institution}(x, \textit{y}) \qquad (98\%)$$
$$\textit{Y :bornIn}(x,y) \wedge \textit{Y :type}(y, \textit{City}) \Rightarrow \textit{F :birthPlace}(x, \textit{y}) \qquad (97\%)$$

For the first example, the corresponding R-subsumption *Y :graduated*$(x,y) \Rightarrow$ *F :institution*(x,y), has a PCA confidence of only 88%. In this example, the type constraint strengthened the precision of the mapping by also aligning the ranges of the relations. In general, if a relation in a $\mathcal{K}_2$ subsumes multiple relations in $\mathcal{K}_1$, specific R-subsumptions provide additional information, e.g., type constraints, to distinguish among those relations.

Attribute-Value Translation. Different conventions for entity labels, as well as missing links between the literal values across KBs, require ROSA rules of the form

$$r(x, V) \Rightarrow r'(x, V')$$

We call these rules attribute-value translations, because they provide a bridge between the semantics of predicate-object combinations across KBs. By allowing constants in the arguments of the rules, AMIE mined translations between genders in YAGO and IMDb (99% confidence), as they are represented as URIs in YAGO and as literals in the IMDb crawl.

We also found translations between places in YAGO and their corresponding timezones in DBpedia, such as :

$$\textit{Y :locatedIn}(x, \textit{Italy}) \Rightarrow \textit{D :timeZone}(x, \textit{CET}) \qquad (100\%)$$
$$\textit{Y :locatedIn}(x, \textit{California}) \Rightarrow \textit{D :timeZone}(x, \textit{PST}) \qquad (100\%)$$

2-Value Translation. Differences in the verbosity of two ontologies can be captured by ROSA rules of the form

$$r_1(x, V_1) \wedge r_2(x, V_2) \Rightarrow r'(x, V')$$

This rule says that two facts about an entity in $\mathcal{K}_1$ are mapped to a single fact about the same entity in $\mathcal{K}_2$. An example mined from YAGO and Freebase is :

$$\textit{F :type}(x,\textit{Royal}) \wedge \textit{F :gender}(x,\textit{female}) \Rightarrow \textit{Y :type}(y,\textit{Princess}) \qquad (55\%)$$

5.5 Conclusion

In this chapter, we have argued that complex schema alignments are needed if we want to knit the Semantic Web together. To illustrate this, we have defined one class of such alignments, the ROSA rules. We have shown that these are easy to mine, and that they already comprise some interesting types of alignments. Preliminary experiments on real large-scale KBs have shown that there exist quite some alignments that go beyond simple one-to-one mappings.

However, many challenges remain : Incompleteness in the KBs means that it is challenging to make the distinction between subsumption semantics and equivalence semantics, or to distinguish between soft and hard ROSA rules. This is particularly true for complex patterns like 2-hop subsumption or value translations, where the large number of soft correlations means that we cannot yet identify meaningful mappings fully automatically. Moreover, ROSA rules are just one particular possibility of schema alignment, that uses Horn rules. We envision that allowing functions [31] in the translation rules could address even more schema incompatibilities. For instance, the following rule could translate between a KB that concatenates first name and last name, and a KB that does not :

$$K_1 \textit{ :firstName}(x, y) \wedge K_1 \textit{ :lastName}(x, z) \Rightarrow K_2 \textit{ :name}(x, \textit{concatenate}(y, z))$$

Such kind of rules would suppose an important step towards a full interoperability between between the KBs of the Semantic Web. We leave this idea as future work.

Chapitre 6

Canonicalization of open KBs

This chapter studies the problem of canonicalization of open KBs. An open KB is a schema-free KB that has been constructed by automatically harvesting triples from text in the web.

We structure the contents of this chapter as follows. We start with an introduction to the problem in Section 6.1. This is followed by the related work in the areas of open information extraction and synonym detection for both noun and verbal phrases (Section 6.2). Section 6.3 studies the problem of canonicalization of noun phrases (entities). In Section 6.4, we address the canonicalization of verbal phrases (relations). Section 6.5 evaluates the performance of our methods on an open KB extracted from the web. We discuss our conclusions in Section 6.6.

The work presented in this chapter builds upon the following publication :

— Luis Galárraga, Geremy Heitz, Kevin Murphy, Fabian Suchanek. *Canonicalizing Open Knowledge Bases*. Conference on Information and Knowledge Management, 2014.

6.1 Introduction

Techniques based on Open Information Extraction (open IE [10, 35]), such as ReVerb [37], allow the subjects and objects in the KB to be arbitrary noun phrases, and the predicates to be arbitrary verb phrases. For instance, they may extract the facts ⟨*DC, is capital of, United States*⟩ and ⟨*Washington, capital city of, USA*⟩. At the other extreme, techniques based on "closed IE" require that the subjects, predicates and objects are canonical, i.e., that they have unique ids. This is the approach taken by many KBs, such as YAGO, Wikidata, DBpedia, and Knowledge Vault.

Open IE techniques normally achieve higher coverage than closed IE approaches. However, the triples in an open KB are "dirty", that is, it is not clear if they are talking about the same entities, or even about the same predicate. This means that when we query the KB for facts about an entity by one name, we cannot be sure that we get all facts about the entity. Conversely, we cannot be sure that all the facts we get are about the same entity. There are also systems such as NELL [18], which applies a mixed approach, in which the predicates are defined in a schema, but the entity names are open.

Contribution. The main contribution of this chapter is an approach that takes a large "open" KB, such as produced by ReVerb or NELL, and convert it into a canonicalized form. In this new form, entities and relation names are mapped to canonical clusters. More precisely, our contributions are as follows :

— *Canonicalization of noun phrases :* In Section 6.3, we show how standard clustering techniques, with simple blocking strategies and similarity functions, give surprisingly good results for entity clustering in both NELL and Reverb data. The resulting clusters contain noun phrases talking about the same real world entity.

— *Canonicalization of verbal phrases :* We show how to use AMIE, to learn high quality clusters of semantically equivalent verbal phrases (relations) from Open IE triples (Section 6.4).

6.2 Related Work

6.2.1 Open Information Extraction

Open IE systems extract triples of the form ⟨*subject, predicate, object*⟩ from natural language text. For example, given the sentence "McCain fought hard against Obama, but finally lost the election", an Open IE system will extract two triples, ⟨*McCain, fought against, Obama*⟩ and ⟨*McCain, lost, the election*⟩. Early systems [10, 130] typically restricted the subject and object to noun phrases. These can be named entities, such as *Obama*, but also common noun phrases, such as *the election*. The predicate can be any sequence of words that appear between the two arguments. This basic approach can harvest a huge number of triples from Web corpora. However, it will also extract uninformative triples, such as ⟨*Hamas, claimed, responsibility*⟩ (where it is not clear for what Hamas claimed responsibility).

The ReVerb approach [37] restricted the predicates to lightweight verbal phrases, which greatly improved the precision of the triples. The Ollie approach [108] relaxed this restriction by expanding the syntactic scope of relation phrases to cover a much larger number of relation expressions, and by adding context information such as attribution and clausal modifiers. Some approaches use dependency parsing [130], or employ hand-crafted patterns on the parse trees. ClausIE [30] can reason on the dependency trees, and thus extract more robust triples.

All of these approaches have in common that their relationships are not canonicalized. The approaches cannot see that *was born in* and *'s birth place is* denote the same semantic relationship. The approach used in the NELL project [18], in contrast, works with a predefined set of relationships. It will extract triples of the form ⟨*NP, relation, NP*⟩, where the *NP* are noun phrases and *relation* is one of the predefined relation names. Thus, NELL is not strictly speaking an Open IE system. Still, it shares many properties of Open IE systems. Most notably, all of the Open IE approaches, and NELL, cannot extract canonical entities. They cannot see that *Barack Obama* and *President Obama* are two names for the same person. We discuss some solutions to this below.

6.2.2 Linking and clustering entities

One approach to resolving entity names is to try to map them to an existing list of known entities, such as Wikipedia or Freebase. This is known as entity linkage (or "Wikification" if the target KB is Wikipedia). Typically each mention generates a candidate list of entities (based on string matching), and then this list is re-ranked using a machine learned model. There are several kinds of models : some link each entity mention in isolation using local features [50], some jointly link sets of mentions within a page using local features and global context [100], and some jointly link mentions across sets of pages [69].

One problem with the above approaches is that many pages may refer to new entities that are not already in a KB. This is particularly true for tail entities (i.e., ones that are not popular enough to have a Wikipedia entry), and/or for new or emerging entities. The standard approach to this is to allow each mention to either map to an entity on the shortlist, or to a special NIL or OOKB (out-of-KB) entity (see e.g., [54]). However, that still leaves the issue of how to cluster these NIL values to create new entities.

The problem of clustering equivalent entities has been widely studied, and is closely related to the problems of cross-document entity resolution in the NLP community [9, 126] and record linkage in the DB community [33]. Most methods use some variant of the following basic technique : define (or learn) a pairwise similarity function for comparing two candidate entities, and then apply hierarchical agglomerative clustering (HAC) to find the mention clusters. For example, this approach was used in the RESOLVER system of [131] to cluster the entities derived from TextRunner's Open IE triples. They defined the similarity of two entities in terms of their string similarity, as well as the similarity of their attribute values. We use a similar technique in this chapter ; see Section 6.3 for the details.

We compare our technique to Concept Resolver [61], a state-of-the-art system that clusters entity mentions on NELL data. Concept Resolver operates in two phases. The first phase performs disambiguation under the *one-sense-per-category* assumption. This assumption states that within a given NELL category, noun phrases are unambiguous. For example, *Apple* can be a company and a fruit, but there cannot be two companies called Apple (nor two fruits). We can infer the type of a particular noun phrase by the type signature of the accompanying verb. For instance, for the triple $\langle$*Apple, hasCeo, Tim Cook*$\rangle$, the domain of *hasCeo* tells us that *Apple* refers to the company. This triple is then rewritten as $\langle$*Apple :company, hasCeo, Tim Cook :person*$\rangle$. The second phase clusters all these type-augmented mentions. For this, they use HAC with a machine learned similarity metric, similar to our approach.

6.2.3 Clustering relations

There has been less work on clustering synonymous relations than on clustering synonymous entities. The database community has studied schema alignment, but this usually relies on knowing the type signature of the relations, which are unavailable for Open IE triples.

The RESOLVER system [131] used HAC to cluster Open IE relations in TextRunner data. They used the set of subjects and objects associated with each relation to define a feature vector ; they then constructed a generative model (known as the "Extracted Shared Property" model) to compute the probability that two relations are equivalent, based on counting the number of entity pairs that they had in common. Finally they used this similarity metric inside HAC. The disadvantage of this approach is that it defines the feature vector for a relation in terms of the raw string names for the entities, and these can be very ambiguous. For example, suppose the dataset contains the triples ⟨*Indian Airlines, is headquartered in, Mumbai*⟩ and ⟨*Indian Airlines, has headquarters in, Bombay*⟩. We cannot determine that *is headquartered in* is equivalent to *has headquarters in* unless we know that *Mumbai* is equivalent to *Bombay*.

One solution to this is to jointly cluster the entities and relations at the same time ; this has been called "knowledge graph identification" [98]. We adopt a simpler two-stage approach. We first perform entity linkage on the triples, mapping the subject and object to a unique id (e.g., both *Bombay* and *Mumbai* map to the Freebase id */m/04vmp*). We then pass these partially-disambiguated triples to the AMIE rule mining system, which can discover equivalences between synonymous relations. We then use these learned equivalences to create clusters of semantically equivalent relations. See Section 6.4 for the details.

The PATTY system [89] uses pattern mining techniques to find subsumption rules between syntactic patterns (e.g., *daughter of* $\sqsubset$ *child of*), extracted from a corpus. Like our approach, PATTY links the arguments of phrases to a KB (YAGO) to find subsumption rules. However, their goal is to construct a taxonomy of verbal phrases, whereas we are interested in finding equivalences between verbal phrases.

The WEBRE approach [84] can cluster verb phrases with close meaning in the presence of ambiguity. For instance, the verb phrase *be part of* holds different semantics in the sentences "New York is part of the United States" (*location* is part of *location*) and "Sun Microsystems is part of Oracle" (*company* is part of *company*). WEBRE first disambiguates the relational concepts, producing a set of typed relations called type A relations (e.g. *company* is part of *company*). Then, WEBRE performs synonym resolution on such concepts. For this purpose, WEBRE uses both the Open IE triples and the source text corpus to construct a hypernym graph, an entity similarity graph and verb phrase similarity graph. Such data structures are used to construct features for a clustering implementation based on HAC. Our approach also deals with ambiguous verbal phrases by enforcing type constraints on the arguments of the equivalence mappings mined from the Open IE KB. However, unlike WEBRE, our methods rely solely on the Open IE triples. We compare our approach for relation clustering with WEBRE in Section 6.5.2.

6.3 Canonicalizing noun phrases

6.3.1 Mentions

Given an Open IE KB, our first goal is to canonicalize its noun phrases. For simplicity, we concentrate here on canonicalizing the subjects ; the same approach can be used to canonicalize the objects. We note that the same string can have different meanings if it appears on two different pages. For example, ⟨*Obama, won, an award*⟩ can refer to Barack Obama or Michelle Obama, depending on where the triple was found. We assume, however, that the same subject phrase on the same Web page will always refer to the same entity. For example, a news article that uses *Obama* to refer to the president may not use that word (without other qualification) to refer to his wife. This is a common assumption in linguistics [44].

With this in mind, we define a mention as a triple $m = (n, u, \mathcal{A})$ where n is a subject noun phrase such as *Barack Obama*, u is the url of a Web document such as *bbc.com* where the mention was found, and $\mathcal{A}$ is the set of attributes about n that were extracted from u. Each attribute is a predicate-object pair such as ⟨*was born in, Honolulu*⟩, or ⟨*won, an award*⟩. In the rest of this chapter, we model $\mathcal{A}$ as a table with columns *pred* and *obj* and resort to relational algebra (with set semantics) to express queries on this table. For instance $\pi_{pred}(\mathcal{A})$ denotes the set of predicates occurring in $\mathcal{A}$. We conclude that a mention defines the profile of a noun phrase n in a particular Web source u.

6.3.2 Clustering

Our goal is to partition the set of mentions, so that all mentions in one partition refer to the same real-world entity. This task can be seen as a clustering problem, where the real number of clusters, i.e., the number of different entities in the data, is unknown. To solve this problem, we use Hierarchical Agglomerative Clustering (HAC) on the set of mentions built from the Open IE triples.

In its general formulation, HAC has $\mathrm{O}(N^3)$ time complexity, where N is the number of mentions [76]. This makes it inadequate for large datasets. To alleviate this fact, we used token blocking [97], a method that resembles the canopies method introduced in [79]. This method first assigns each mention to one or several groups, called *canopies*. One standard approach is to assign the noun phrases to canopies based on the words that the noun phrases contain. For example, a noun phrase *President Obama* will be assigned to the canopy for *President* and to the canopy for *Obama*. Then, standard HAC is applied within each canopy. This partitions each canopy into a set of clusters. Finally, the clusters that live in different canopies but share a mention are merged. In the example, the cluster {*Barack Obama*, *President Obama*} from the *Obama* canopy will be merged with the cluster {*President Obama*, *US President*} from the *President* canopy. This technique reduces the number of pairwise comparisons required by the standard HAC implementation. Furthermore, it facilitates parallelism, because each canopy can run HAC on a smaller subset of data.

Canopies can drastically affect the recall of the clustering algorithm : If two mentions refer to the same real-world entity, but are assigned to different canopies, then they might never end up in the same cluster. For example, *Mumbai* and *Bombay* will go to different canopies, and unless there is a mention that connects them, they will remain in different clusters. Thus, we face a trade-off, where assigning a mention to a small number of canopies will improve efficiency (decrease the runtime), but hurt recall and precision.

We propose the following solution to this problem, which we call "object canopy expansion" : We assign two mentions $m_1 = \langle n_1, w_1, \mathcal{A}_1 \rangle$ and $m_2 = \langle n_2, w_2, \mathcal{A}_2 \rangle$ to the same canopy, if (1) n_1 and n_2 share a word that is not a stopword, or if (2) there exist two objects $o_1 \in \pi_{obj}(\mathcal{A}_1), o_2 \in \pi_{obj}(\mathcal{A}_2)$ that share a word. In this way, we can merge *Mumbai* and *Bombay*, if, e.g., they both appear in triples ⟨*Mumbai, is located in, the Republic of India*⟩ and ⟨*Bombay, is a city in, India*⟩.

6.3.3 Similarity Functions

HAC requires a similarity function on the mentions. In this paper, we study different similarity functions, with the goal of determining which ones work best under which conditions. Many of the functions use the Jaccard coefficient :

$$jaccard(\mathcal{S}, \mathcal{S}') = \frac{|\mathcal{S} \cap \mathcal{S}'|}{|\mathcal{S} \cup \mathcal{S}'|}$$

Given two mentions $m = (n, u, \mathcal{A})$ and $m' = (n', u', \mathcal{A}')$, we define the following similarity functions (called *features*) :

Attribute Overlap. The attribute overlap is the Jaccard coefficient of the set of attributes :

$$f_{attr}(m, m') := jaccard(\mathcal{A}, \mathcal{A}')$$

Two attributes $(p, o) \in \mathcal{A}$, $(p', o') \in \mathcal{A}'$ are equal if $p = p'$ and $o = o'$.

String Similarity. We use the Jaro-Winkler similarity [127] between n and n' as a feature

$$f_{strsim}(m, m') := jarowinkler(n, n')$$

String Identity. As a special case of the string similarity (and as a baseline), we consider the string identity :

$$f_{strid}(m, m') := \begin{cases} 1, & \text{if } n = n' \\ 0, & \text{else} \end{cases}$$

IDF Token Overlap. If two noun phrases share a word, they are more likely to be similar, but not if many other mentions share that word. For example, the fact that *Rhine River* and *Ruhr River* share the word *River* is not very significant, because this word also appears in a plethora of other river names. Therefore, we introduce a weighted word overlap, in which a word is given more importance if it appears in fewer mentions. We follow the standard Inverse Document Frequency (IDF) approach :

$$f_{itol}(m, m') = \frac{\sum_{w \in w(n) \cap w(n')} log(1 + df(w))^{-1}}{\sum_{w \in w(n) \cup w(n')} log(1 + df(w))^{-1}}$$

Here, $w(\cdot)$ is the set of words of a string, excluding stop words. $df(w)$ is the frequency of the word in the collection of all words that appear in the subjects and the objects of the OpenIE triples.

Word Overlap. If two Web pages share many words, then this can be an indication that two mentions on these pages refer to the same entity. We define

$$f_{wol}(m, m') = jaccard(t(u), t(u'))$$

where $t(\cdot)$ is the set of the top 100 words on a page, ranked in terms of TF-IDF [77].

Entity Overlap. Since words can be ambiguous, we also experimented with an overlap of entities. We applied a standard entity linkage algorithm [40] on the pages, and identified the set $e(u)$ of linked Freebase entities on the page u. Then we define

$$f_{eol}(m, m') = jaccard(e(u), e(u'))$$

Type Overlap. Shared types can be a signal for merging two mentions. We used the results of the type induction algorithm [105] on the verbal phrases of the attributes (as provided by the authors of [105]). This approach proposes types for the subject arguments of verbal phrases, e.g., *type(is the capital of)* = *country*, which is equivalent to provide candidate types for the subject noun. We define

$$f_{tol}(m, m') = jaccard(types(\pi_{pred}(\mathcal{A})), types(\pi_{pred}(\mathcal{A}'))$$

So far, we have described how to compute the similarity between two mentions. The similarity between two clusters is calculated using the single linkage criterion [52], that is, the maximum of the intercluster pairwise similarities. Our experience suggests that the complete linkage criterion (the policy that uses the minimum intercluster similarity) and even the average linkage criteria are too conservative and therefore lead to low clustering recall.

6.3.4 Combined Feature

In addition to the individual features, we also study a combined feature, which is a logistic function :

$$f_{ml}(m, m') = \frac{1}{1 + e^{-f_{sim}(m, m')}}$$

Here, $f_{sim}(m, m')$ is a linear combination of features :

$$f_{sim}(m, m') = c_0 + \sum_{i=1}^{N} c_i f_i(m, m')$$

The $f_1, \ldots, f_N$ are the similarity functions discussed in Section 6.3.3. We also study a simple combined feature f_{sml} that excludes the metrics that depend on information outside the open KB, namely the words overlap, the entity overlap, and the type overlap. In both cases, the weights c_i are determined by training a logistic regression classifier.

To train the combined similarity function, we need labeled training data, i.e., a set of mention pairs that are labeled as being *equal* or *unequal*. Such training data can be obtained, for instance, from approaches that perform entity disambiguation (Section 6.2.2), i.e., that map Open IE triple subjects to entities in a KB. In our case (i.e., in the case of ReVerb triples), half of the triples already have their subjects mapped to Freebase entities [40]. We use these mappings for training, so that we can partition the remaining, yet-unmapped, mentions into synonymous clusters.

To learn robust weights for our features, we have to make sure that our training set contains hard cases, where the same entity appears with different names, because otherwise we will learn to put too much weight on just the string similarity. To do this, we randomly sample 200 Freebase entities that appear with different names in our triples. For example, our set of entities contains the Greek poet Homer, because he appears as *Homer* and as *Homerus* in our triples. Every one of these names, in turn, can refer to several entities (homonyms). For example, *Homer* does not just refer to the Greek poet, but also to Homer Simpson, the cartoon character from “The Simpsons”. Hence, we add to our set also Homer Simpson, and all names that refer to Homer Simpson. This results in 436 entities with 714 names in total. Next we collect all triples that contain one of these 714 names (in mapped form), and construct their mentions (using the provenance information in the Reverb triples). This results in 43K mentions. From these mentions, we can construct a training set of pairs of mentions that are labeled as *equal* or *unequal*. We construct a set of 1137 pairs, balanced between equal and unequal pairs. We also make sure that each entity contributes with at least two examples, because standard random sampling would penalize entities with few mentions. This set of pairs is then used to train our weighted similarity function. In Section 6.5.1, we compare this learned similarity function to the baseline approaches.

6.3.5 Canonicalization

Given a cluster of synonym mentions $m = (n, w, \mathcal{A})$, the canonicalization consists of selecting a representative noun phrase *n̂* that will replace the other noun phrases in the canonicalized KB. We propose a simple approach that selects the noun phrase *n̂* with the highest number of different Web sources u. In case of a tie, an alternative is to select the longest noun phrase.

6.4 Canonicalizing verbal phrases

In this section, we describe our approach for clustering verbal phrases that have equivalent meaning. The basic idea is to learn rules that predict when one phrase is semantically equivalent to another, and then to perform clustering by applying the transitivity of the equivalence relation.

6.4.1 A semi-canonicalized KB

Our approach to verbal phrase clustering requires that the subjects and objects of the Open IE triples are already canonicalized. There are two ways to achieve this. Either we can use our noun phrase clustering algorithm of Section 6.3, or we can make use of the existing mappings to Freebase. We consider both approaches. In particular, for the latter case, we take a subset of all the Reverb triples where the subject was already mapped to Freebase (as provided in the data in [40]), and where we can unambiguously map the object to Freebase using a simple string matching technique.

With either of these techniques, we obtain a "semi-canonicalized" KB, where the subjects and objects are fully canonicalized, and the predicates are still uncanonicalized verbal phrases (the "dual" of NELL). This KB may contain, e.g., ⟨*Barack Obama, was born in, Honolulu*⟩ and ⟨*Barack Obama, 's birthplace is, Honolulu*⟩.

6.4.2 Rule Mining

Suppose we have the two Open IE relations r=*was born in* and r' =*'s birthplace is*. We would like to discover that these are equivalent, i.e., that $r \sqsubset r'$ and $r' \sqsubset r$, where $r \sqsubset r'$ means that r' is more general than r (r' subsumes r), i.e.,

$$\forall r(x,y) \in \mathcal{K} : r(x,y) \Rightarrow r'(x,y)$$

Unfortunately not all triples with r will necessarily appear with r'. Conversely, relations that are very sparse may occur with the same subject and object by chance, even though they are not equivalent. For example, if we find ⟨*Woody Allen, married, Soon-Yi Previn*⟩ and ⟨*Woody Allen, 's stepdaughter, Soon-Yi Previn*⟩, we should not deduce *'s stepdaughter* $\sqsubset$ *married*, even if all triples with the former verbal phrase appear with the latter. Therefore, we resort to our rule mining approach AMIE from Chapter 2.

We apply AMIE to our semi-canonicalized KB to mine subsumption rules of the form $r(x,y) \Rightarrow r'(x,y)$. As suggested in Section 5.4, we infer equivalence rules of the form $r(x,y) \Leftrightarrow r'(x,y)$ by merging subsumptions in both directions. We score each subsumption with the PCA confidence, and score an equivalence with the minimum of the two subsumption scores. The output of the AMIE system is a set of equivalent verb phrases like *be-named-after(x, y)* ⇔ *take-its-name-from(x, y)*

6.4.3 Phrase Clustering

The rule mining has given us a set of weighted equivalence relations between verbal phrases. Since the equivalence relation is transitive, we iteratively merge equivalence mappings with at least one verbal phrase in common. For instance, given the equivalences

— *stand-for(x, y)* ⇔ *be-an-acronym-for(x, y)*

— *be-short-for(x, y)* ⇔ *be-an-acronym-for(x, y)*

— *refer-to(x, y)* ⇔ *be-short-for(x, y)*

we merge the relations *stand-for*, *be-an-acronym-for*, *short-for* and *refer-to* into a single cluster.

6.4.4 Canonicalization

We propose to canonicalize clusters of verbal phrases by mapping them to Freebase relations. To achieve this, we resort to the ROSA approach presented in Chapter 5 to coalesce our open KB with Freebase. This is possible thanks to the instance alignment provided in the form of links from noun phrases to Freebase entities. To coalesce our open KB with Freebase, we restrict our set of triples to those that have a subject and an object linked to Freebase. Then we join this set with Freebase as proposed in Section 5.3, so that we obtain a set with facts of the form $vp(x, y)$ and $fr(x, y)$. Here, x and y are Freebase entities, vp is a verbal phrase and fr is a Freebase relation. Then, we run AMIE on our coalesced KB in order to mine cross-ontology equivalence rules of the form $vp(x, y) \Leftrightarrow fr(x, y)$. The rule

$$bought(y, x) \Leftrightarrow acquiring\_company(x, y)$$

is an example. For each cluster of Reverb verb phrases, we collected all the Freebase relations implied by the verbal phrases in these equivalences. In Section 6.5.2, we show that in some cases it is possible to map clusters unambiguously to Freebase. If a cluster cannot be mapped to Freebase (e.g., because it represents a novel relation that is not part of the Freebase schema), a representative for the cluster can be chosen by selecting either the verbal phrase that appears in most equivalences, or the phrase with the largest number of triples.

6.5 Experiments

We conducted two groups of experiments, one to evaluate different entity clustering features, and one to evaluate the relation clustering.

6.5.1 Entity clustering

6.5.1.1 Evaluation metrics

To evaluate the quality of a clustering of mentions, we assume each mention m can be mapped to a corresponding entity $e(m)$ from Freebase according to a gold standard clustering E. Each cluster $e \in E$ contains all mentions that map to the same entity. Given

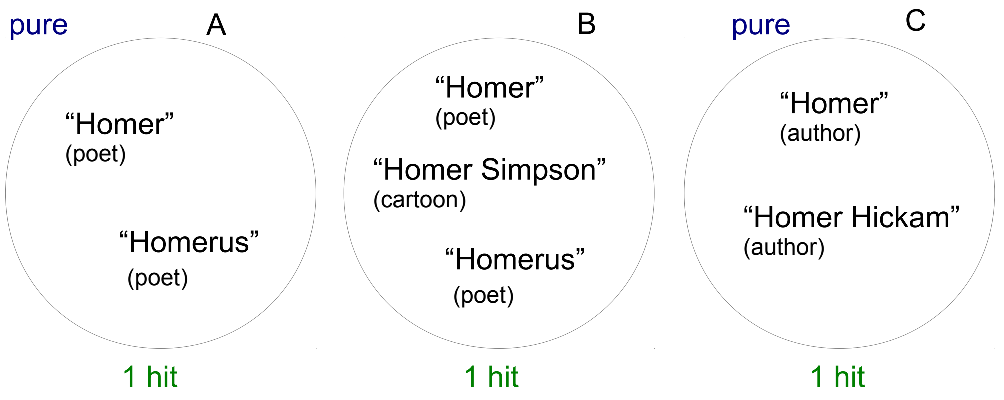


FIGURE 6.1 – An example of a clustering of N=7 mentions and a gold standard with 3 Freebase entities, labeled in parentheses below the mentions.

this, we can measure precision and recall of the clustering in 3 different ways, which we call macro analysis, micro analysis and pairwise analysis.

We will explain these metrics using the example in Figure 6.1, which illustrates a set of $|M| = 7$ mentions distributed across $|C| = 3$ clusters. There are $|E| = 3$ Freebase entities, all called "Homer" : the Greek poet, the cartoon character Homer Simpson, and the author Homer Hickam. (Each entity also has other aliases.)

Macro-analysis. We define the macro precision of the clustering as the fraction of clusters that are pure, i.e., where all the mentions in the cluster are linked to the same entity :

$$precision_{macro}(C,E) = \frac{|c \in C : \exists_{=1}\, e \in E : e \supseteq c|}{|C|}$$

Macro recall is calculated by swapping the roles of the ground truth and the resulting clustering, i.e., $recall_{macro}(C,E) = precision_{macro}(E,C)$. This corresponds to the fraction of Freebase entities that get assigned to a unique cluster. In Figure 6.1, we can see that clusters *A* and *C* are pure as they do not mix mentions of different entities. Hence $precision_{macro} = {}^{2}/_{3}$. Conversely, the cartoon character and the author are pure entities because they occur only in one cluster, therefore $recall_{macro} = {}^{2}/_{3}$.

Micro analysis. Micro precision is defined as the purity [76] of the resulting clustering. This is computed by assuming that the most frequent Freebase entity of the mentions in a cluster is the correct entity. That is, we compute

$$precision_{micro}(C,E) = \frac{1}{N}\sum_{c \in C} max_{e \in E}|c \cap e|$$

where N is the number of mentions in the input. Macro recall is symmetrically defined as $recall_{micro}(C,E) = precision_{micro}(E,C)$. For example, in Figure 6.1, the most frequent

	Macro			Micro			Pairwise		
	Prec.	Recall	F1	Prec.	Recall	F1	Prec.	Recall	F1
String identity	**1.000**	0.436	0.607	**1.000**	0.798	0.888	**1.000**	0.740	0.851
String similarity	0.995	0.658	0.792	0.998	0.844	0.914	0.999	0.768	**0.986**
IDF token overlap	0.994	0.879	0.933	0.996	0.969	0.982	0.999	**0.973**	**0.986**
Attribute overlap	**1.000**	0.05	0.102	**1.000**	0.232	0.377	**1.000**	0.094	0.173
Entity overlap	0.996	0.436	0.607	0.995	0.934	0.964	0.999	0.932	0.964
Type overlap	0.987	**0.926**	**0.956**	0.995	**0.973**	**0.984**	0.999	0.972	0.985
Word overlap	0.988	0.913	0.949	0.995	**0.973**	**0.984**	0.999	**0.973**	**0.986**
Simple ML	0.994	0.899	0.944	0.996	0.972	**0.984**	0.999	**0.973**	**0.986**
Full ML	0.994	0.906	0.948	**1.000**	0.937	0.967	**1.000**	**0.973**	0.869

TABLE 6.1 – Precision and recall on ReVerb's *Base* dataset. The highest values in each column are in bold.

entity for clusters A and B is the poet (2 mentions in each cluster) and for C is the author (2 mentions), so the micro precision is $^6/_7$. Analogously $recall_{micro} = {}^5/_7$ because the highest frequencies in a cluster for the entities are : 2 times for the poet, 2 times for the author, and 1 time for the cartoon character.

Pairwise analysis. In the pairwise evaluation, we measure the precision and recall of individual pairwise merging decisions. To be more precise, let us say that two mentions from the same cluster produce a *hit* if they refer to the same Freebase entity. We define the pairwise precision as

$$precision_{pairwise}(C,E) = \frac{\sum_{c \in C} \#hits_c}{\sum_{c \in C} \#pairs_c}$$

$\#pairs_c = |c| \times (|c|-1)/2$ is the total number of mention pairs in a cluster. Likewise, we define recall as

$$recall_{pairwise}(C,E) = \frac{\sum_{c \in C} \#hits_c}{\sum_{e \in E} \#pairs_e}$$

In Figure 6.1, clusters A and C produce 1 hit out of 1 pairwise decision, whereas cluster B produces 1 hit out of 3 pairwise decisions. Hence the pairwise precision is $\frac{3}{5}$. To compute $recall_{pairwise}$, we calculate the total number of pairwise decisions in the gold standard E : $\#pairs_{poet} + \#pairs_{author} + \#pairs_{cartoon} = 6 + 1 + 0$, so $recall_{pairwise} = {}^3/_7$.

In all cases, the F1 measure is defined as the harmonic mean of precision and recall, i.e.,

$$F1_x(E,C) = \frac{2 \times precision_x(C,E) \times recall_x(C,E)}{precision_x(C,E) + recall_x(C,E)}$$

	Macro			Micro			Pairwise		
	Prec.	Recall	F1	Prec.	Recall	F1	Prec.	Recall	F1
String identity	**1.000**	0.436	0.607	**1.000**	0.798	0.888	**1.000**	0.740	0.851
String similarity	0.948	0.477	0.634	0.971	0.811	0.884	0.973	0.743	0.842
IDF token overlap	0.994	0.879	**0.933**	0.996	0.969	**0.982**	0.999	0.973	**0.986**
Attribute overlap	0.994	0.054	0.102	0.990	0.232	0.376	0.990	0.094	0.172
Entity overlap	0.000	0.805	0.000	0.169	0.987	0.289	0.051	0.981	0.097
Type overlap	0.750	0.980	0.850	0.157	**1.000**	0.272	0.051	0.999	0.097
Word overlap	0.000	**1.000**	0.000	0.157	**1.000**	0.271	0.051	**1.000**	0.097
Simple ML	0.979	0.490	0.653	0.824	0.916	0.868	0.405	0.937	0.565
Full ML	0.990	0.154	0.267	0.776	0.889	0.829	0.396	0.931	0.555

TABLE 6.2 – Precision and recall on ReVerb's *Base* dataset, without canopies. Highest values in bold.

Datasets	freebase entities	mentions	triples
Base	150	8.5K	9.1K
Ambiguous	446	34K	37K

TABLE 6.3 – Testing datasets for the mentions clustering on ReVerb

6.5.1.2 Clustering ReVerb

ReVerb[1] [37] is a state-of-the-art Open IE system that was run on the Clueweb09 corpus[2]. It produced 3M triples. Half of these triples have their subject linked to Freebase, as in [40]. To evaluate the different clustering features, we built a gold standard as follows : We sampled 150 Freebase entities that appear with at least 2 names in our dataset, and collected all their mentions in our triples. This resulted in 8.5K mentions. We call this dataset *Base*. We then enrich this dataset with all the mentions of homonym entities, as in Section 6.3.4. We name this dataset *Ambiguous*. This results in 446 Freebase entities and 34K mentions. For both datasets, we constructed a gold standard clustering by grouping those mentions that are linked to the same Freebase entity. Table 6.3 summarizes the information about the testing datasets.

Results on *Base*. We ran our implementation of HAC on the *Base* dataset. On a Intel Core i7 (8 logical cores, 2.40 GHz) with 16 GB of RAM, our implementation (using the full ML similarity function, plus expanded canopies) created 157 clusters in 54.3 seconds (averaged across 3 runs). Our memory footprint reaches a peek of 5.7GB. Next, we assess the quality of the different similarity features introduced in Section 6.3.2. We show the results in Table 6.1, using a confidence threshold[3] that was chosen to maximize the F1 score. Our first observation is that all the features deliver very good

1. `http://OpenIE.cs.washington.edu/`
2. `http://www.lemurproject.org/clueweb09.php/`
3. In HAC the confidence threshold defines the minimum similarity between two clusters that is required to merge them.

precision. This means that they rarely produce a cluster than contains two different entities. Thus, the features behave rather conservatively.

Let us now look at recall. The macro-recall is very low in general, because as soon as one entity appears in more than one cluster, the metrics decreases (even if all other mentions of the entity are clustered correctly). Let us therefore look at the micro-recall and the pairwise-recall. All features perform decently. This is mainly because all features can build on the pre-clustering that the canopies already established, i.e., every feature is applied only to pairs of mentions with overlapping words. When comparing the recall of the features, the attribute overlap stands out with lowest recall. This is because our triples are very sparse : It rarely happens that two different mentions of the same entity share many attributes. The type overlap, in contrast, works very well : The fact that two mentions with similar names share a type is a strong signal that they refer to the same entity. The word overlap performs well for a similar reason. Among the features that do not require preprocessing, the IDF token overlap is a clear winner.

Rather surprisingly, the combined features (Full ML and Simple ML) are not the best performing methods. We thought that this would be because of the canopies, which make the resulting data distribution at test time quite different from what was used to train the model. To assess this conjecture, we also ran HAC without canopies on this dataset (Table 6.2). We can observe that the string identity, the IDF tokens overlap, and the attributes overlap are insensitive to the canopies policy. The other features, in contrast, perform much worse without canopies. This suggests that they provide little or no extra evidence of synonymy by themselves. They are noisy signals that mainly mislead the ML methods. This, in turn, explains the performance of the ML methods with and without canopies.

Table 6.4 lists some examples of pure entity clusters that the Simple ML feature could find. We can for instance cluster the mentions "Phoenix Arizona" and "Phoenix" using as signals the tokens overlap and common attributes such as ⟨*located in, Arizona*⟩. Moreover we can avoid mixing these mentions with the mythological creature of the same name. On the other hand, the sparseness of the attribute overlap still leads to losses in recall. For example, we could not cluster the mentions "Beijing National Stadium" and "National Stadium" together, even though they are the same entity.

Results on *Ambiguous*. On the *Ambiguous* dataset and with the same setup as for *Base*, our implementation produces 823 clusters in 15.045 minutes on average with a peek memory footprint of 6.5GB. The results are shown in Table 6.5. Not surprisingly, precision is lower on this dataset. This is mainly caused by the single linkage criterion for clusters. Under this policy, the similarity of a pair of clusters is determined by the highest intercluster similarity score. While this aggressive strategy was shown to improve recall significantly, it also propagates errors more easily. Furthermore, this phenomenon is amplified by the reclustering phase and the ambiguity of the test set. To see this, consider a set of mentions with labels *Barack Obama*, *Michelle Obama*, and *Obama*, the latter referring ambiguously to both Barack and Michelle Obama. A single clustering error on the ambiguous mentions (*Obama*) will move the three entities in the same cluster, and thus decrease precision. Conversely, the baseline approach is less sensitive to ambiguity in this particular case because the precision will only penalize the merge of the ambiguous mention *Obama*, as *Barack Obama* and *Michelle Obama* will be never clustered together. Hence, all features produce lower precision. The attribute overlap has highest precision – but this is mainly because it is a very sparse feature that hesitates to merge mentions. Let us therefore look at the micro-F1 and the pairwise-F1. We see that the IDF token overlap is the strongest feature, even in presence of ambiguity. It is followed by the combined features. For comparison, the table also shows the Simple ML without the "object canopy expansion" (Section 6.3.2). We see that the expansion increases recall slightly, and has a negligible effect on precision. Therefore, we conclude that the technique is indeed useful. Table 6.4 shows some examples of impure clusters.

Lessons Learned. Our study of features shows that, among the more advanced features, the type overlap and the word overlap produce significant leverage if combined with canopies – both on the random dataset and on the ambiguous one. They are closely followed by the IDF token overlap, which stands out as the strongest simple feature, with and without canopies. The combined features also perform decently. All in all, the best features can cluster entities in the general case with nearly 100% precision, and a pairwise-recall of more than 98%.

6.5.1.3 Clustering NELL

The NELL project [18] also extracts triples from the ClueWeb09 corpus. The Concept Resolver approach [61] aims to cluster the entities of these triples. Concept Resolver operates under the one-sense-per-category assumption, which states that within one category, names are unambiguous. We note that we can now give a justification for this assumption based on the good performance of the Type Overlap feature on the ReVerb data. Once the type for a noun phrase has been extracted, Concept Resolver collects all the type compatible triples about the noun phrase and builds a "sense" for that noun phrase, the equivalent of what we call a "mention". The authors of [61] provi-

Pure clusters
{*Phoenix Arizona*, *Phoenix*}
{*Hannibal Hamlin*, *Hamlin, Hannibal*}
{*The Colorado Rockies*, *The Rockies*}
Impure clusters
{*John's Gospel*, *John Peel*, *Peel*}
{*Suns*, *Phoenix Coyotes*, *Phoenix Suns*}
{*Ethanol*, *Cellulosic Ethanol* }

TABLE 6.4 – Some examples of successfull and unsuccessful synonym identification

	Macro			Micro			Pairwise		
	Prec.	Recall	F1	Prec.	Recall	F1	Prec.	Recall	F1
String identity	0.734	0.390	0.510	0.932	0.771	0.844	0.942	0.565	0.706
String similarity	0.607	0.442	0.511	0.792	0.873	0.831	0.809	0.574	0.671
IDF token overlap	0.643	0.509	0.568	0.913	0.847	**0.879**	0.900	0.703	**0.789**
Attribute overlap	**0.997**	0.083	0.153	**0.998**	0.162	0.279	**0.997**	0.024	0.047
Entity overlap	0.905	0.480	**0.627**	0.663	0.939	0.777	0.458	0.892	0.606
Type overlap	0.467	0.917	0.619	0.626	**0.970**	0.760	0.401	0.914	0.558
Word overlap	0.390	**0.926**	0.549	0.625	**0.970**	0.760	0.401	0.915	0.557
Simple ML, no obj.can.	0.711	0.444	0.546	0.808	0.909	0.855	0.630	0.889	0.738
Simple ML	0.709	0.444	0.546	0.808	0.923	0.862	0.649	0.968	0.777
Full ML	0.685	0.552	0.611	0.671	0.955	0.788	0.302	**0.989**	0.463

TABLE 6.5 – Precision and recall on ReVerb's *Ambiguous* dataset. The highest values in each column are in bold.

	Micro-evaluation			Pairwise		
	Precision	Recall	F1	Precision	Recall	F1
Simple ML	0.660	0.578	0.616	0.376	0.188	0.250
Concept Resolver	0.778	0.633	0.699	0.542	0.335	0.415
IDF Token Overlap	0.700	0.475	0.566	0.356	0.067	0.113

TABLE 6.6 – Comparison of entity clustering methods on the NELL data.

ded us with the data and the training examples used to evaluate Concept Resolver. As described in Section 6.3.4, we used the triples to construct mentions.

The challenge is to cluster together several names that refer to the same entity. Since not all the triples contained the source from which they were extracted, we defined a default source for those without provenance. We also restricted our dataset to the categories considered in the evaluation of Concept Resolver. The resulting dataset consists of 18K mentions and 20K triples.

Gold Standard. Concept Resolver comes with a manually compiled set of noun phrases that should be clustered together as one entity. We estimated our precision by sampling a random subset of 100 clusters from the output and checking them manually. For Concept Resolver we report the precision value from [61], averaged across all categories and weighted by the number of senses (mentions in our approach) per category.

We note that our notion of precision is not fully comparable to theirs : the one-sense-per-category assumption merges all mentions of the same type into one mention – no matter if the triples come from different sources. This can cause problems. For example, *Obama :Person* is assumed to map to only one entity, whereas we allow it to map to multiple entities, depending on which page the mention was seen. Moreover, Concept Resolver removes singleton clusters from the evaluation of precision. For us, in contrast, such a singleton cluster can correspond to several successfully merged mentions from

	Conf.	Phrases	Clusters	Precision			In Freebase	Triples covered
				Macro	Micro	Pairwise		
Linked KB	0.8	522	118	0.900	0.936	0.946	18%	15%
	0.5	967	143	0.896	0.690	0.337	25%	29%
Linked KB (types)	0.8	752	303	0.946	0.980	0.997	9%	21%
	0.5	1185	319	0.861	0.892	0.779	14%	27%
Clustered KB	0.8	826	234	0.940	0.716	0.273	6%	16%
	0.5	1185	264	0.813	0.665	0.292	8%	33%

TABLE 6.7 – Quality of relation clusters for two different confidence thresholds.

Verb phrases	Freebase relation
be an abbreviation-for, be known as, stand for, be an acronym for	-
be spoken in, be the official language of, be the national language of	`location.country.official_language`
be bought, acquire	`organization.organization.acquired_by`
be the birth place of, be-the-hometown-of	`ns:location.location.people_born_here`

TABLE 6.8 – Examples of clusters of verbal phrases. The last two were mapped to Freebase.

different pages. This also makes trivial any comparison to our string identity baseline. Therefore, we restricted our manual evaluation to non-singleton clusters according to Concept Resolver and used the IDF token overlap as baseline. Concept Resolver does not report values for the macro dimension, we show results for the micro and pairwise evaluation.

Results. We ran our synonym resolution machinery on the NELL triples and computed precision and recall. The results are shown in Table 6.6. The Simple ML feature can achieve decent precision and recall, albeit inferior to the values of Concept Resolver. We believe this is because the logistic regression model implemented by Concept Resolver leverages information from the ontology, which is not possible in an Open IE scenario. Whereas our Simple ML uses a Jaccard score on the attributes of mentions as signal for synonymy, Concept Resolver also builds features from the properties of the relations. For instance, their approach will penalize the similarity for the mentions *Auburn* and *Auburn Hills* as they have different values for the functional predicate *cityLocatedInState* (the first located in Maine and the second in Michigan) even if they have other attributes in common (e.g. both located in USA). Additionally, Concept Resolver makes use of inverse and quasi-inverse functions as they can identify mentions uniquely. The fact that the mentions *Mumbai* and *Bombay* share the inverse functional predicate ⟨*cityCapitalofState, Maharastra*⟩ is used as strong evidence for synonymy. Still, our Simple ML and the IDF token overlap deliver reasonable results even without

Datasets	freebase entities	triples	verbal phrases
Clustered KB	25K	600K	17K
Linked KB	206K	1.3M	33K

TABLE 6.9 – Testing datasets for verbal phrase clustering on ReVerb

this additional information.

Lessons Learned. We can see here that the relation schema that Concept Solver uses improves the entity clustering results. If such data is not available, a synonym resolution approach based on Open IE attributes and string similarity features can deliver reasonable results. Even a simple baseline such as the IDF token overlap should not be outright discarded for the task of synonym resolution.

6.5.2 Relation clustering

6.5.2.1 Dataset

Since the relations of NELL are already canonicalized, we run the relation clustering only on the ReVerb set of triples. As discussed in Section 6.4.1, the relation clustering requires a semi-canonicalized KB, in which the subjects and objects have been canonicalized. There are two ways to achieve this : either by our entity clustering ("Clustered KB") or by a mapping to Freebase ("Linked KB"). The Clustered KB was constructed by sampling 25K freebase ids from the Linked KB and gathering all their triples. This results in 600K triples. Both the Clustered and the Linked KB are given as input to the AMIE system in order to mine relation equivalences. We ran AMIE using a support threshold of 5, meaning that two verbal phrases must have at least 5 pairs in common. This also implies that relations with less than 5 cases are discarded. Table 6.9 summarizes the information about the input datasets.

6.5.2.2 Ambiguous phrases

In [84] it is suggested that approximately 22% of the Reverb phrases are polysemous. The phrase *belongs-to*, e.g., conveys different meanings in the sentences "The Wii belongs to Nintendo" (*invention created by organization*) and "Mallorca belongs to Spain" (*island belongs to country*). Polysemy hurts precision, since phrases with unrelated meanings will be transitively clustered as synonyms due to the polysemic relations. To alleviate this effect, we also run AMIE on triples from the Linked KB where the entities are augmented with types. This option makes AMIE enforce type constraints on the arguments of the equivalence mappings when possible. Thus, a single verbal phrase can generate multiple specific relations that differ only in their signatures. Since the Freebase ontology has approximately 23K different data types, we allowed type enhacement only with the most common types, i.e., person, organization, location, and string.

6.5.2.3 Evaluation metrics

Since evaluating recall would require identifying all the actual relations occurring in the set of 1.3M triples, we focus on measures of precision. As in Section 6.5.1.1, we can measure the precision of a relation cluster at the macro, micro or pairwise level. The pairwise precision measures the quality of a set of clusters as the ratio of correct pairwise merges. A pairwise merge is counted as correct if the corresponding verbal phrases mean the same, or if one of them is slightly more general than the other. For instance, the phrases *be-spoken-in* and *be-the-official-language-of* count as a correct merge. The micro-precision assumes the most frequent meaning in a cluster as ground truth. Hence, it is calculated by adding up the frequency of the most common meaning in each of the clusters and dividing this number by the total number of phrases. Conversely, the macro-precision is the ratio of pure clusters, i.e., the proportion of clusters where all the phrases belong to the same meaning. For micro and macro precision, phrases were labeled with the most general meaning.

6.5.2.4 Results

On the Clustered KB, AMIE mined 3.5K equivalence mappings, whereas mining the Linked KB produced 4.3K equivalence rules. When the type enhancement is enabled, the number rises to 22K mappings. For example, we find *use-truck-for* $\Leftrightarrow$ *use-van-for* with confidence 1.0 and support 19, or *stand-for* $\Leftrightarrow$ *be-an-acronym-for* with confidence 0.88 and support 44. With the type enhancement, AMIE can discriminate between a country changing location, e.g., “Israel moved to Mont Hor”, a person visiting a place, e.g., “Barack Obama moved to Los Angeles”, and an organization changing location, e.g., “Fostoria Glass moved to Moundsville”. This results in different equivalence rules for the phrase *move-to* such as *moved-to*$(location \rightarrow location)$ $\Leftrightarrow$ *located-in*$(location \rightarrow location)$ (support 8, confidence 0.5) and *moved-to*$(person \rightarrow location)$ $\Leftrightarrow$ *move-permanently-to*$(person \rightarrow location)$ (support 5, confidence 0.5). In these examples our coarse-grained type constraints are enough to avoid mixing phrases that denote permanent location (e.g. *situated-in*) with phrases that denote place of residence (e.g. *now-lives-in*).

The mappings have different confidence scores, and therefore we tested the phrase clustering at two confidence thresholds : 0.8 and 0.5. Table 6.7 shows the results, together with the number of clusters and phrases. As we see, the precision of our clusters is very good, meaning that we do not merge phrases that do not belong together. Naturally, a higher confidence threshold always leads to fewer phrases being clustered, and to fewer clusters. Our results are better on the cleaner Linked KB than on the Clustered KB. We can also observe the benefit of the type enhancement. In general, we can cluster only a very small portion of the verbal phrases. However, our clusters are non-trivial and contain an average of 4-5 phrases.

Comparison to WEBRE. We compare our results to the WEBRE system [84] for verbal phrase clustering. WEBRE operates in two phases. In the first phase, the system identifies typed relations, e.g., "*location* is part of *location*", called type A relations. In the second phase synonym resolution is applied to the set of type A relations. This stage produces a final set of latent relations, called type B relations. We use the precision on the Linked KB with types because the type-enhanced phrases resemble the type A relations introduced by WEBRE. To be comparable, we report a weighted micro-precision, where the correct assignments of phrases to clusters are weighted by the number of triples with the verbal phrase. We get a score of 0.981, which is slightly better than WEBRE's score of 0.897. Nevertheless, this comparison must be taken with a grain of salt because the evaluation performed in WEBRE is somewhat different from ours (see Section 5 in [84]). First, their method to generate typed verbal phrases is different. In particular, WEBRE extracts the instance information (types) from the source web corpus of the triples, while we we use fixed set of types from Freebase. Second, we could not have access to their gold standard for precision and recall. Third, the micro-precision formula of WEBRE uses are more granular definition of synonymy : a phrase can be a synonym of the true relation of a cluster (score 1), somehow related (score 0.5) or unrelated (score 0). Nevertheless, it is safe to say that our approach is not far off from the state-of-the-art in terms of precision.

Mapping to Freebase. As described in Section 6.4.4, we used AMIE and ROSA rules to find equivalences between verbal phrases and Freebase relations. We ran AMIE on a combination of Freebase and Reverb with support threshold 5, producing 5.1K cross-ontology mappings. We then applied the same confidence thresholds as for relation clustering (0.5 and 0.8) and used the rules to map the clusters of verb phrases to Freebase. We counted clusters that were mapped to one Freebase relation or to two mutually inverse Freebase relations as "correctly mapped". For example, Freebase expresses the fact that Steven Spielberg directed the Titanic movie with a pair of mutually inverse relations, ⟨*Steven Spielberg, directed, Titanic*⟩ and ⟨*Titanic, directed by, Steven Spielberg*⟩. The last column in Table 6.7 shows the proportion of triples whose relation could be mapped. We find a large number of very interesting mappings, some of which are shown in Table 6.8. Going manually through the mappings, we find an average precision of 88% for the Clustered KB with threshold 0.8.

Lessons Learned. We conclude that rule mining can be used to cluster verbal phrases, and to map them to canonical Freebase relations. A cleaner KB and the type enhancement both help. This method can produce such clusterings and mappings only for a small portion of the verbal phrases, but it can do so at a high precision and for a significant percentage of the Reverb triples.

6.6 Conclusions

We have shown that it is possible, using fairly simple and standard machine learning techniques, to identify synonym mentions in a reasonable fraction of the triples coming from standard Open IE systems, such as Reverb and NELL. In addition, we have illustrated how rule mining and the notions of schema alignment can help us solve this problem when it comes to synonym verbal phrases.

Regarding the canonicalization of noun phrases, our results suggest that, even with a certain level of ambiguity, the IDF token overlap is the strongest signal of synonymy for noun phrases on the Web, whereas more sophisticated features extracted from the sources are insufficient for this task on their own. We also provided useful and novel insights about the impact of canopies in the performance of Hierarchical Agglomerative Clustering, a standard technique for record linkage and identification of synonyms.

In relation to the canonicalization of verbal phrases, we have shown how our rule mining techniques can identify verbal phrases of close meaning with high precision in an open KB. The resulting clusters of relations are semantically meaningful ; some of them could be mapped to existing relations in Freebase using the schema alignment techniques proposed in Chapter 5.

We believe this hybrid approach — whereby we use high recall open extractors, followed by clustering methods to improve the precision — shows great promise for bridging the gap between Open and Closed IE methods for knowledge base construction.

Finally, we remark that we have studied the problem of canonicalization of open KBs as two separate subproblems. A potential direction of research in this matter would be to intertwine the solutions for those subproblems. We have shown that canonicalizing the noun phrases serves the canonicalization of verbal phrases. In our vision, this process could be iterative and incremental, i.e., the canonicalized verbal phrases could help us improve the canonicalization of mentions, which in a later stage could help us canonicalize even more verbal phrases.

Chapitre 7

Predicting Completeness in Knowledge Bases

This chapter addresses the problem of predicting completeness in KBs. This problem is of great importance to KB maintainers due to the inherent incompleteness and the open semantics of KBs.

This chapter is structured as follows. In Section 7.1, we motivate the problem of predicting completeness in KBs. Section 7.2 summarizes the related work in this field. In Section 7.3, we define completeness for KBs. Section 7.4 proposes a set of signals to estimate completeness for given pairs of entities and relations. Section 7.5 describes how to combine completeness signals and learn completeness rules using AMIE. In Section 7.6, we evaluate the performance of the proposed signals and our method to predict completeness. Section 7.7 shows an application scenario for completeness assertions in KBs. We draw our conclusions and discuss directions of future work in Section 7.8.

At the time of writing, the contents of this chapter are under revision :

— Luis Galárraga, Simon Razniewski, Antoine Amarilli, Fabian Suchanek. *Predicting Completeness in Knowledge Bases*. Proceeding of the 10th International Conference on Web Search and Data Mining. Cambridge, UK, 2017.

7.1 Introduction

Motivation. Current KBs suffer from data quality problems. These problems include false data, missing information, or schema inconsistencies. Traditionally, many approaches have tried to clean up erroneous information [132]. YAGO, e.g., guarantees that 95% of its facts are correct w.r.t the extraction source. In contrast, completeness (recall) has remained relatively unexplored. While we often know what proportion of the facts in the KB are correct, we usually do not know what proportion of the facts in the real world are known to KB.

For example, as of 2016, Wikidata knows the father for only 2% of all people in the KB – even though in the real world everyone has a father. DBpedia contains only 6 Dijkstra Prize winners – but in the real world there are 35. Likewise, according to YAGO, the average number of children per person is 0.02. In general, between 69% and 99% of instances in popular KBs lack at least one property that other entities in the same class have [83, 114]. Thus, we know that today's KBs are highly incomplete, but we do not know where the information is missing.

This unknown degree of completeness poses several problems [102]. First, users do not have any guarantee that a query run against the KB yields all the results that match the query in the real world. Second, the data providers themselves may not know where the data is incomplete, and thus do not know where to focus their efforts. If they knew, e.g., which people are missing their alma mater, they could focus on tracing these pieces of information and adding them. Third, completeness information can help identify wrong facts. If we know, e.g., that Barack Obama has only 2 children, then a KB that contains 3 children has to be erroneous. Finally, completeness information can be insightful on its own, because it induces counter-evidence for KBs. If it is stated the KB knows all the children of Barack Obama, then any other child is a certified counter-example. This can be exploited by rule mining and machine learning algorithms that require negative evidence.

Thus, it would be of tremendous use for both data providers and data consumers if we could know where the information in the KB is complete. In the ideal case, we would want to make what we call *completeness assertions*, which say, e.g., *This KB contains all children of Barack Obama*.

Challenges. The main obstacle to establish such completeness assertions is the Open World Assumption (OWA), which nearly all KBs make (Section 1.3.2). Furthermore, today's KBs provide no way to store completeness information. For example, YAGO says that Barack Obama is married to Michelle Obama, but it does not say that Barack Obama is not (and was never) married to any other woman. In fact, there is not even a way that YAGO and similar KBs could express this idea. The KBs are not just incomplete, but also, by design, unable to provide any indications of completeness.

Contribution. In this chapter, we make a first step towards generating completeness information *automatically*. Our goal is to determine automatically whether certain properties of certain objects are complete : whether a person has more children in reality than in the KB, whether a person graduated from a university in real life even though the KB does not know about it, or whether a person has had more spouses in reality than are known to the KB. More precisely :

— We conduct a systematic study of signals that can indicate completeness of properties of objects in a KB.

— We show how completeness assertions can be learned through a rule mining system, AMIE – based on training data obtained through crowd-sourcing.

— We find that completeness can be predicted for certain relations with up to 100% precision on real KBs (YAGO and Wikidata).

— As a use case, we show that our completeness assertions can increase the precision of rule mining.

7.2 Related Work

Knowledge Bases. Many of today's KBs provide estimations of their precision. YAGO [118] was manually evaluated and found to be 95% correct. NELL [18] is regularly checked by humans for precision. Facts in the Knowledge Vault [32] are annotated with an estimated precision. However, little is known about the recall/completeness of these KBs. Of course, larger sizes may indicate higher completeness, but size is only a very coarse proxy for completeness.

Incompleteness Studies. Some studies have found that KBs are indeed quite incomplete. For instance, a watermarking study [114] reports that 69%–99% of instances in popular KBs lack at least one property that other entities in the same class have. Google found that 71% of people in Freebase have no known place of birth, and 75% have no known nationality [32]. This tells us that KBs are incomplete in general, but it does not tell which parts of the KB are complete.

Manual Indicators. The Wikidata community maintains lists that explain where information is still missing – e.g., a list of people without birth dates [1]. Also, Wikidata contains *no-value statements*, which say that an empty relation is complete for an entity [34]. An extension for Wikidata allows contributors to manually add recall information [26]. However, these annotations are mostly provided manually : our work aims at deducing such annotations *automatically*.

Partial Completeness Assumption. Some approaches simply *assume* that KBs are complete in certain areas. In Section 2.4 we introduced the *partial completeness assumption* (PCA), which is used by AMIE [42, 43]. This same assumption is applied in [32] under the name of *local closed world assumption*. We discuss the PCA as a signal of completeness in Section 7.4.

Rule Mining. Inductive Logic Programming and Rule Mining approaches [66] find rules such as *If a person lives in a city, then their spouse lives most likely in the same city*. These rules can then be used to predict new information (here : where the spouse lives). As a side effect, this procedure determines where the KB is incomplete. However, such approaches can only ever mine new facts between instances that are already known to the KB. They cannot tell us that a spouse is missing if that spouse is not in the KB. We will show in our experiments how rule mining can benefit from the techniques we develop in this paper.

1. `https://www.wikidata.org/wiki/Wikidata:Database_reports/top_missing_properties_by_number_of_sitelinks/P569`

Completeness Reasoning. On the database side, some work has investigated how to combine completeness information about parts of databases to deduce completeness annotations on query results [68, 85, 101]. However, this work assumes that the KB has already been annotated with completeness assertions. Our goal, in contrast, is to *generate* such assertions.

7.3 Preliminaries

Completeness. In line with work in databases [85, 102], we define completeness via a hypothetical ideal KB $\mathcal{K}^*$, which contains all facts of the real world. A KB $\mathcal{K}$ is *complete* for a query q, if q delivers the same results on $\mathcal{K}$ as on $\mathcal{K}^*$. In this chapter, we focus on a particular type of queries, namely those that ask for the objects of a given subject and relation. Thus, a pair of an entity s and a relation r is *complete* in a KB $\mathcal{K}$, if

$$\{o : r(s, o) \in \mathcal{K}\} \supseteq \{o : r(s, o) \in \mathcal{K}^*\}$$

For example, a KB is complete for the subject *Barack Obama* and the relation *hasChild*, if it contains both of Obama's children. If the KB is complete for a subject s and a relation r, we make a *completeness assertion* of the form $complete(s, r)$. Our goal is to establish such completeness assertions.

In general, completeness assertions make less sense for relations with low functionality (see Section 1.3.3 for the definition of functionality). For example, it does not make sense to ask a KB if it knows all citizens of France, but it is more sensible to ask whether the KB knows all nationalities of one person. Therefore, we consider completeness primarily for relations with high functionality, and for the more functional direction of relations. When a relation is incomplete for a subject, we could also try to estimate *how many* objects are missing. This would amount to a cardinality estimation. In this chapter, however, we focus on the simpler task of establishing completeness assertions, and leave cardinality estimations for future work.

Completeness Considerations. The notion of completeness is not well-defined for all relations [102]. Take, e.g., the relation *hasHobby*. It is not always clear whether an activity counts as a hobby or not. Thus, it is difficult to establish whether a KB is complete on the hobbies of a certain person. Even for seemingly well-defined relations such as *hasOfficialLanguage*, completeness is not easy to establish : a country may have de facto official languages that are not legally recognized (e.g., the US) ; languages that are official in some regions but not in the country (e.g., India) ; or an official language that is not a spoken language (e.g., New Zealand). In this paper, we manually selected relations for which completeness is well-defined, and concentrate on these.

Completeness Oracles. A *completeness oracle* tries to guess whether a given relation is complete for a given subject in the fixed KB $\mathcal{K}$. Technically, a completeness oracle is a binary relation on entities and relations that holds whenever the oracle predicts that a given subject is complete for a given relation. The Partial Completeness Assumption (PCA) is an example of a simple completeness oracle. It predicts completeness for a subject s and a relation r if there exists an object x with $r(s, x)$. For instance, if Barack Obama has one child in the KB, then the PCA oracle will (wrongly) assume that Barack Obama is complete for relation *hasChild*, i.e., *pca(BarackObama, hasChild)* will be true.

The precision and recall of an oracle o are defined as follows, where *complete* denotes the completeness assertions that are true relative to the ideal KB $\mathcal{K}^*$:

$$precision(o) := \frac{\#(e,r) : o(e,r) \wedge complete(e,r)}{\#(e,r) : o(e,r)}$$

$$recall(o) := \frac{\#(e,r) : o(e,r) \wedge complete(e,r)}{\#(e,r) : complete(e,r)}$$

The *F1 measure* is defined as usual from precision and recall.

7.4 Completeness Oracles

We now present various completeness oracles, of which we study two kinds : *simple* oracles and *parametrized* oracles.

7.4.1 Simple Oracles

Closed World Assumption. The Closed World Assumption (CWA) assumes that any fact that is not in the KB does not hold in the real world. That is, the CWA assumes that the entire KB is complete, i.e., $\mathcal{K} \supseteq \mathcal{K}^*$. In general, the CWA is incompatible with the philosophy of the Semantic Web. Still, the CWA may be suitable under certain conditions. For instance, if a person is not known to be the president of any country, then most likely the person is indeed not the president of any country. Formally, the CWA completeness oracle is simply defined as :

$$cwa(s,r) := true$$

Partial Closed World Assumption (PCA). The PCA is an oracle that has proven useful for rule mining in Chapter 2. Under the PCA, a subject-relation pair s, r is considered complete if there is an object o with $r(s, o)$. In other words, we assume that, if the KB knows some r-values for s, then it knows all its values. The PCA is more cautious at predicting completeness than the CWA : it predicts the completeness only if objects are already known. This implies that the PCA makes predictions only for those entities that have an object for the relationship, and remains silent otherwise. For instance, according to the CWA, a person that has no nationality in the KB has no nationality in reality, but the PCA will not make such claims. Formally, the PCA oracle is :

$$pca(s,r) := \exists o : r(s,o)$$

The PCA is especially well suited for *functional relations*, where an entity can have at most one object. Indeed, if an entity has some object for a functional relation, then it is complete. As shown in the evaluation carried in Section 2.4, the PCA also performs well for quasi-functions such as nationality.

Cardinality. A more cautious oracle than the PCA is the *cardinality* oracle. For an integer value k, the oracle says that a subject s is complete for a relation r if s has at least k different objects for r. Formally :

$$card_k(s,r) := \#(o : r(s,o)) \geq k$$

This oracle subsumes the CWA and PCA : $card_0$ is cwa, and $card_1$ is pca. Other values for k can be useful ; for instance, $card_2$ can be effectively used as a predictor for the $hasParent$ relation. In our experience, however, larger values are rarely useful, and hence we categorize this oracle as a simple oracle.

Popularity. The previous oracles have looked at properties of entities in isolation. We can also look at the entities in the context of the entire KB. For example, we can hypothesize that entities which are popular (by some measure) are more likely to be complete. For example, we expect that Wikipedia-based KBs are more complete for famous entities such as Albert Einstein than for entities that have only stub-articles. For a Boolean measure pop that indicates whether an entity is popular or not, we can define this completeness oracle as

$$popularity_{pop}(s,r) := pop(s)$$

No Change. So far, we have only looked at a single snapshot of the KB. An alternative is to study how the KB changes over time. If the objects of a particular subject do not change, then this may suggest that the subject is complete. Given a Boolean measure of change chg, where $chg(s,r)$ indicates whether the set of objects for entity s and relation r has changed over time, we define :

$$nochange_{chg}(s,r) := \neg chg(s,r)$$

7.4.2 Parametrized Oracles

We now move on to the study of oracles that depend on parameters that are difficult to determine upfront, such as classes and relations.

Star Patterns. Instead of trying to estimate the completeness for a relation by looking only at that relation, we can look at facts involving *other* relations. For example, if someone has won a Nobel Prize, then we probably know their alma mater. Formally, we consider “star-shaped patterns” of certain relations around the subject, and predict completeness if they are all present :

$$star_{r_1 \dots r_n}(s,r) := \forall i \in \{1,\dots,n\} : \exists o : r_i(s,o)$$

Class Information. In some circumstances, the class to which an entity belongs can indicate completeness with respect to some relations. For example, the instances of the class *LivingPeople* should not have a death date. If we assume that the KB is correct, this implies that any instance of that class is complete with respect to that relation. Formally, the class-based oracle for a class expression c is

$$class_c(s, r) := type(s, c) \in \mathcal{K}$$

We restrict our study to two types of class expressions : plain class names such as *Living_people* or negated class expressions of the form $\hat{t} \wedge \neg t$ where t is a subclass of $\hat{t}$, like in $Person \wedge \neg Adult$.

Others. Many other completeness oracles can be envisaged. For example, we could extract information from the Web to find out whether we can find more objects ; we could ask a crowd of users for more objects ; we could compare two KBs to see if one contains more information than the other ; or we could pull in yet other sources. In this work, however, we limit ourselves to a single source, and leave other such approaches to future work.

7.5 Learning Completeness

7.5.1 Combining Oracles

Some completeness oracles cannot be used out-of-the-box. For example, there exists a star oracle for every possible sequence of relations $r_1 \ldots r_n$ that form a star-shaped query in the KB. Another example is the class oracle. Since YAGO has more than 200K classes, there is a large number of potential class oracles that could be conceived in this KB. Furthermore, oracles may work best in combination : for instances of some classes, the PCA may be the best oracle, while the cardinality oracle may be better in other cases. Our goal is thus to *generalize and learn* completeness oracles from the simple and parametrized ones that we presented.

Towards this goal, we assume that we already have a certain number of gold standard completeness assertions as *training data*. We show in Section 7.6 how to obtain such assertions from a crowd of users with good precision. Based on these gold standard annotations, we can then learn combinations and parametrizations of the oracles. To this end, we adapt the AMIE rule mining approach presented in Section 2.5.

7.5.2 Enhancing AMIE

Recall from Chapter 2 that AMIE learns rules of the form $\boldsymbol{B} \Rightarrow H$, such as $wasBornIn(x, y) \Rightarrow livesIn(x, y)$, on KBs. AMIE starts with rules with an empty body, i.e., with rules of the form $\Rightarrow r(\boldsymbol{X}, \boldsymbol{Y})$ [2], and refines them using a number of operators.

2. In $r(\boldsymbol{X}, \boldsymbol{Y})$ at least one of the arguments is a variable

Each of the operators takes a rule as input, and produces a set of refined rules as output, by adding one particular type of atom to the body of the rule :

— **Add Dangling Atom :** A *dangling atom* joins the rule on an existing variable and introduces a new variable in the other position.

— **Add Closing Atom :** A *closing atom* is an atom that joins on two existing variables in the rule.

— **Add Instantiated Atom :** An *instantiated atom* has one instantiated argument (a constant/entity) and joins with the rule on the other argument.

The operators always produce rules with less support than the original rule. AMIE applies them iteratively to find all rules above a given support threshold.

In order to mine rules about completeness, all of the completeness oracles (Section 7.4) have to be translated into the AMIE framework. We represent unary atoms $p(x)$ as $p(x, \text{`}True\text{'})$ so that the AMIE operators can apply. Then, we define the following new types of atoms :

— ***complete*(*x*, *R*), *incomplete*(*x*, *R*)** : Our training data (Section 7.5.1) takes the form of completeness assertions $complete(x, R)$ and $incomplete(x, R)$, where x binds to entities in the domain of relation R, i.e., $type(x, domain(R))$. We add these assertions to the KB.

— ***isPopular*(*x*) :** The popularity oracle relies on an external measure pop of entity popularity. We considered three such measures : (i) number of facts for that entity, (ii) length of the article in the English Wikipedia, and (iii) number of ingoing links to the Wikipedia page. Manual inspection revealed that (i) correlated best with completeness. Thus, we add $isPopular(x)$ to the KB if x is among the 5% entities with the most facts in the KB.

— ***hasNotChanged*(*x*, *R*) :** Given an older version of the KB, we add facts of the form $hasNotChanged(x, R)$ to the new KB if x has exactly the same R-objects in the new KB as in the old KB. These type of facts are required by the $nochange$ oracle. In our experiments, we used YAGO1 and YAGO3.

— ***notype*(*x*, *T*) :** The $notype(x, T)$ atom states that an entity is not an instance of class T. Such atoms are always used in conjunction with instantiated atoms of the form $type(x, \hat{T})$ where $\hat{T}$ is a super-class of T. These types of atoms allow us to integrate class expressions of the form $c = \hat{T} \wedge \neg T$ as defined for the $class_c$ oracle. Moreover, they assume the instance information in the KB is complete.

— ***lessThan*$_\mathbf{n}$(x, R)**, ***moreThan*$_\mathbf{n}$(x, R)** : An atom of the form $lessThan_n(x, R)$ with $k > 0$ is satisfied if x has less than k objects for relation R in the KB. The *moreThan*$_k$ atom is defined analogously. These kind of atoms allow AMIE to learn the pca and $card_k$ oracles.

We let AMIE mine only rules with heads of the form $c(x, R)$, where c is either $complete$ or $incomplete$, R is a relation, and x is a variable. For performance reasons, we enable the “Add Instantiated Atom” operator only for $isPopular(x)$, $hasNotChanged(x, R)$,

$type(x,T)$ and $notype(x,T)$. Unlike the standard AMIE syntactic language bias, we only enforce the head variables of the rule to be closed. This relaxation of the language bias allows to mine rules with star patterns like for example

$$hasWonPrize(x,z) \land isPoliticianOf(x,w) \Rightarrow complete(x, isCitizenOf)$$

Still, we do not allow open variables in the new types of atoms, e.g., $isPopular$ and $hasNotChanged$. We also forbid atoms with the relation R in the body of the rules. Furthermore, we define five additional mining operators to capture the oracles that we defined :

— **Add Type :** Given a rule $\boldsymbol{B} \Rightarrow c(x,R)$, this operator adds an atom of the form $type(x,T)$, where $T = domain(R)$. The operator is applied only if the rule does not yet contain a type atom.

— **Specialize Type :** Given a rule $type(x,T) \land \boldsymbol{B} \Rightarrow c(x,R)$, this operator produces a new rule $type(x,T') \land \boldsymbol{B} \Rightarrow c(x,R)$ where T' is a subclass of T.

— **Add Negated Type :** Given a rule $type(x,T) \land \boldsymbol{B} \Rightarrow c(x,R)$, the application of this operator produces a new rule $notype(x,T') \land type(x,T) \land \boldsymbol{B} \Rightarrow c(x,r)$, where T' is a subclass of T.

— **Add Cardinality Constraint :** Given a rule $\boldsymbol{B} \Rightarrow c(x,R)$, this operator adds an atom of the form $moreThan_0(x,R)$ or $lessThan_k(x,R)$, where k is the highest number of objects seen for any subject in the relation R.

— **Tighten Cardinality Constraint :** Given a rule $lessThan_k(x,R) \land \boldsymbol{B} \Rightarrow c(x,R)$, this operator replaces k by the largest value k' ($k' < k$) that decreases the support of the original rule. Likewise, given a rule $moreThan_k(x,R) \land \boldsymbol{B} \Rightarrow c(x,R)$, we replace k by the smallest value k' ($> k$) that decreases the support.

Examples. In the following, we show how AMIE uses the new mining operators to mine rules about completeness.

Example 1 : $notype(x, Adult) \land type(x, Person) \Rightarrow complete(x, hasChild)$

This rule says that if a person is not an adult, then the KB is complete for the children of that person (most likely zero). AMIE starts with the simple rule $\Rightarrow complete(x, hasChild)$ and applies all the mining operators described in Section 7.5.2. Among the different new rules generated by this step, the "Add Type" operator produces the rule $type(x, Person) \Rightarrow complete(x, hasChild)$. In the next step, the operator "Add Negated Type" produces new rules of the form $notype(x,T) \land type(x, Person) \Rightarrow complete(x, hasChild)$, where T is a subclass of $Person$. In particular, for $T = Adult$, we obtain our example rule.

Example 2 : $lessThan_1(x, isCitizenOf) \Rightarrow incomplete(x, isCitizenOf)$

This rule states that if a person has less than one citizenship, then the KB is incomplete in the citizenship relation for that person. AMIE starts with the rule $\Rightarrow$ $incomplete(x, isCitizenOf)$, and applies the "Add Cardinality Constraint". Assuming that in the KB nobody has more than 3 nationalities, the operator produces the rule $lessThan_3(x, isCitizenOf) \Rightarrow incomplete(x, isCitizenOf)$ with some support s. In a later step, AMIE tries to tighten the cardinality constraint by means of the "Tighten Cardinality Constraint" operator. The operator searches for the closest $k < 3$ such that the support of the new rule drops. If the number of incomplete people with less than 2 nationalities is smaller than s, the system will chose $k = 2$ and the rule becomes $lessThan_2(x, isCitizenOf) \Rightarrow incomplete(x, isCitizenOf)$. An additional call to the operator "Tighten Cardinality Constraint" on the new rule will produce $lessThan_1(x, isCitizenOf) \Rightarrow incomplete(x, isCitizenOf)$. We remark that depending on the data distribution, AMIE may need a single call to the "Tighten Cardinality Constraint" to produce the target rule, i.e., it may skip the intermediate step where $k = 2$.

7.5.3 AMIE as Completeness Oracle

AMIE will learn rules that predict completeness as well as rules that predict incompleteness. Unlike the standard setting, AMIE counts on explicit counter-examples for such rules. For the first type of rules, she uses the $complete(x, r)$ statements of the training data as examples, and the $incomplete(x, r)$ statements as counter-examples. For the second type of rules, the roles are reversed. This implies that confidence for completeness and incompleteness rules follows the formula :

$$conf(\boldsymbol{B} \Rightarrow c(x, R)) := \frac{supp(\boldsymbol{B} \Rightarrow c(x, R))}{supp(\boldsymbol{B} \Rightarrow c(x, R)) + supp(\boldsymbol{B} \Rightarrow \neg c(x, R))}$$

where $c \in \{complete, incomplete\}$.

Predicting completeness. Once the rules have been learned, we can define a new completeness oracle, the *AMIE oracle*. For a given entity e and a relation r, the AMIE oracle checks whether any of the learnt rules predicts $complete(e, r)$. If so, and if there is no rule with higher or equal confidence that predicts $incomplete(e, r)$, the oracle returns *true*. If there is a rule with equal confidence that predicts $incomplete(e, r)$, the oracle returns *true* if the support of the completeness rule is higher. In all other cases, the oracle returns *false*.

By restricting AMIE to only star atoms or only class atoms, we can produce a *Star oracle* and a *Class oracle*, respectively, analogously to the AMIE oracle.

7.6 Experiments

7.6.1 Setup

Knowledge bases. Our goal is to measure the precision and recall of the completeness oracles on real data. We conducted our study on two KBs : YAGO3, released in September 2015, and a dump of Wikidata of December 2015. For both datasets, we used the facts between entities, the facts with literal object values (except for the relation *rdfs :label*) and the instance information. These choices leave us with a KB of 89M facts (78M type statements) for YAGO, and a KB of 15M facts (3.6M type statements) for Wikidata. We studied completeness on a set of relations covering a large variety of cases. For one type of relations, basically every entity of the domain has to have exactly one object : *hasGender*, *wasBornIn* in YAGO ; *sex_or_gender* (P21), *mother* (P25), *father* (P22), *place_of_birth* (P19) in Wikidata. For other relations, entities do not need to have an object, but can have at most one : *diedIn* in YAGO ; *place_of_death* (P20) in Wikidata. Again others usually have one object, but can have more : *isCitizenOf* and *director*(*Movie*, *Person*) in YAGO ; *country_of_citizenship* (P27) and *director* (P57) in Wikidata. In the most general case, a subject can have zero, one, or several objects : *hasChild*, *graduatedFrom*, *isConnectedTo*(*Airport*, *Airport*), and *isMarriedTo*[3] in YAGO ; *child* (P40), *alma_mater*[4] (P69), *brother*, and *spouse* (P26) in Wikidata. One relation has to have 2 objects : *hasParent*[5] in YAGO. Our relations cover people, movies, and geographical locations.

Ground Truth. In order to evaluate our completeness oracles, we need a set of completeness assertions and incompleteness assertions as a gold standard. For some relations, we could generate this gold standard automatically. Namely, for the relations where every subject has to have exactly one object, we have $complete(s,r)$ iff $\exists o : r(s,o)$. For the relations where every subject must have at least one object, we can directly label as incomplete all subjects without a value. For the relations with at most one object, all subjects with one object are considered complete. For the relation *isConnectedTo*, we used the OpenFlights[6] dataset as ground truth, which we assumed to be complete for all airports in this dataset (identified by their IATA code). However, due to data inaccuracies, in some cases YAGO knew more flights than OpenFlights : we discarded those airports.

Crowdsourcing. For the remaining relations, we used *crowdsourcing* to obtain ground truth data. Given an entity, a relation, and the objects known in the KB, we asked crowd workers whether they could find any additional objects on the Web. If they could, we labelled the entity-relation pair as incomplete, otherwise as complete. To make the task well-defined and manageable, we asked workers to look only at a set of given Web pages. We manually defined queries for each relation (e.g., "x died" for $diedIn(x,y)$ or "x child" for $hasChild(x,y)$), and then gave workers the first 4 URLs retrieved using the Bing search API. We used the Crowdflower platform[7] for crowdsourcing, and paid 1 cent per answer. For each entity, we collected 3 opinions.

3. Despite the name, this relation captures also past spouses.
4. We use the same semantics as in YAGO : places a person graduated from.
5. This is how we call the inverse of *hasChild* in YAGO.
6. `http://openflights.org/data.html`
7. `https://www.crowdflower.com`

Quality Control. To monitor quality, we generated 20–29 test questions for each relation, for which we manually checked the correct answer. Annotators had to pass a qualification test of 10 questions with at least 80% correct answers ; further, the remaining test questions were mixed with the data, and annotators had to maintain 80% correctness while working. About a quarter of annotators failed at the initial tests, and about 5% fell below the correctness threshold while working. Their answers were discarded. Furthermore, we used only the annotations where all 3 answers were unanimous. These make up 55% of the annotations. We carried out a second round of annotations to compensate for the discarded answers.

Sampling. In order to evaluate the popularity oracle, we took a random sample of 100 entities among the popular entities in the domain of the studied relations. We call this gold standard *popular*. Likewise, we produced another gold standard of 100 entities among those whose objects did not change w.r.t YAGO1. This sample is called *no-change*. For the other experiments, we constructed a gold standard consisting of 200 randomly picked entities in the domain of the studied relations. We call this sample *uniform*. The uniform sample is not always useful. For example, only 1% of people have a citizenship in YAGO. Thus, in a sample of 200 people, we may expect a citizenship for only 2 of them. This is too few to learn a rule. Therefore, for relations where less than 10% of the subjects have an object we construct a *biased sample* instead. Instead of choosing 200 entities randomly, we choose 100 entities randomly among those that have an object, and 100 among those that do not. In our experiments, we mark the relations where we used the biased sample. For the calculation of precision and recall, we carried out a de-biasing step. This means that the values we report reflect the true population of entities in the KBs. For the learned oracles, we use 80% of our gold standard for training, and the rest for testing.

7.6.2 Basic Completeness Oracles

Experiment. Our completeness oracles from Section 7.4 try to guess whether a pair of a subject and a relation is complete. We considered the subject–relation pairs where we had a gold standard, and computed precision and recall values as described in Section 7.3. Table 7.2 shows the results for the oracles for YAGO3, and Table 7.4 for Wikidata. Table 7.3 and Table 7.5 show the corresponding F1 measures.

Cardinality Oracles. The first column in the tables shows the CWA. It trivially achieves a recall of 100% : for all pairs that are complete in reality, it makes a correct prediction. However, its precision is lower. This precision value corresponds to the actual completeness of the KB with respect to the real world. We see, e.g., that YAGO is complete for the death place for 44% of the people. This means that these people are either alive, or dead with a known death place in YAGO. We also observe that Wikidata is generally more complete than YAGO.

The next oracle is the PCA. It achieves 100% precision for all functional relations : if a subject has an object, the PCA rightly assumes that the subject is complete. For quasi-functions, such as *isCitizenOf*, the PCA still performs decently, failing only for people with several nationalities. The PCA has a recall of 100% for relations that are mandatory (such as *hasGender*) : whenever this relation is complete in the gold standard, the PCA indeed predicts it. For the other relations, the PCA has much lower values in precision and recall.

The $card_2$ oracle has a much lower recall. We could not compute it for relations where the sample did not contain any entity with sufficiently many objects. This oracle basically makes sense only for the *hasParent* relation, where it performs perfectly. As $card_3$ behaves worse that $card_2$ on both datasets, we omitted it.

Popularity Oracle. The fourth column shows the performance of the popularity oracle computed on the *popular* sample. The oracle was not computed for *isConnectedTo* due to noise in the data. The popularity oracle generally has a low recall, because there are not many popular entities. Its precision is generally good, indicating that popular entities (those that have many facts in general) are indeed more complete than unpopular ones. However, even popular entities are incomplete for parents and citizenship in YAGO, and for parents in Wikidata.

NoChange Oracle. The next column shows the NoChange oracle on YAGO, for those relations that exist in both YAGO1 and YAGO3. It has a very low recall, indicating that most entities did indeed change their objects over time (they most likely gained more objects). The precision is decent, but not extraordinary.

7.6.3 Learned Completeness Oracles

Learning. We took 80% of our gold standard to train our modified AMIE approach (Section 7.5) with 4-fold cross-validation. The training phase measures the performance of AMIE at different configurations, i.e., different values for the support and confidence thresholds. We tested values for support in the range from 10 to 100 entities (in steps of 10), while confidence was tested on values from 0.3 to 1.0 (in steps of 0.1). We report the best configuration in terms of F1 measure for each relation, and use it to measure performance in the testing set (remaining 20% of the gold standard). Training took 44 hours on YAGO, and 4 hours in Wikidata. This difference is mainly due to the much larger type hierarchy in YAGO (78M type assertions as opposed to 3.6M). Table 7.1 shows some of the rules that AMIE learned. The first rule says that a person who has a date of death, but no place of death, is incomplete for the place of death. In the second rule, the IMDb id acts as a substitute for the type *movie*, which is not always consistently used in Wikidata. Thus, the rule basically says that if a movie has a producer, then it is most likely complete on the director.

dateOfDeath(x, y) $\wedge$ *lessThan*$_1$*(x, placeOfDeath)* $\Rightarrow$ *incomplete(x, placeOfDeath)*
IMDbId(x, y) $\wedge$ *producer(x, z)* $\Rightarrow$ *complete(x, director)*
notype(x, Adult) $\wedge$ *type(x, Person)* $\Rightarrow$ *complete(x, hasChild)*
lessThan$_2$*(x, hasParent)* $\Rightarrow$ *incomplete(x, hasParent)*

TABLE 7.1 – Example rules that AMIE learned (2 on Wikidata, 2 on YAGO)

Results. AMIE learns her rules using the best configuration from the training set. After the rules have been learned, making the actual predictions on the testing set takes only seconds. We evaluated these predictions against the remaining 20% of our gold standard, and report the precision, recall, and F1 values in the three last columns of Tables 7.2 and 7.3 for YAGO, and in Tables 7.4 and 7.5 for Wikidata. For the star oracle, we use a star size of $n = 1$ for YAGO, and $n = 3$ for Wikidata. We observe that this oracle can improve the F1 value for the *isMarriedTo* relation. The class oracle, likewise, performs well for certain relations. In particular, the oracle learned that the YAGO class *LivingPeople* means that the *diedIn* relation must be complete, boosting F1 from 60% to 99%. This shows that parametrized oracles can be useful.

In general, the oracles complement each other. Only the complete AMIE approach can nearly always perform best. This is because AMIE learns the strengths of the individual oracles, and combines them as is most useful. As explained in Section 7.5.3, AMIE uses the most confident rules to make completeness predictions. For functional relations, AMIE learned a rule that mimics the PCA, predicting completeness for a subject whenever one object is present : $moreThan_0(X, r) \Rightarrow complete(X, r)$. For *diedIn*, AMIE learned a rule that mimics the class oracle : $type(x, LivingPeople) \Rightarrow complete(x, diedIn)$. When such relation-specific rules are not available, AMIE learns the CWA. This is the case for difficult relations such as *brother*, *graduatedFrom* or *isConnectedTo*. In particular, AMIE learns the CWA in rules of the form

$$type(x, domain(r)) \Rightarrow complete(x, r)$$

All this shows that it is indeed possible to predict completeness with very good precision and recall for a large number of relations. We can predict whether people are alive, or whether they have siblings – all by just looking at the incomplete KB. Only for *spouse* and *alma_mater*, our oracles perform less well. However, guessing whether someone is married, or whether someone graduated from university, is close to impossible even for a human.

7.7 Application

Rule mining is generally used to find arbitrary rules in a KB, not just completeness rules. These rules can be used for *fact prediction*, i.e. to predict which person lives where, or which city is located in which country (Section 2.7). The predictions can be compared to the real world, with the proportion of correct predictions being the precision of the approach. This section shows how to improve the precision of fact prediction

Relation	CWA		PCA		$card_2$		Pop.		No change		Star		Class		AMIE	
	Pr	Rec	Pr	Rec	Pr	Rec	Pr	Rec	Pr	Rec	Pr	Rec	Pr	Rec	Pr	Rec
diedIn	43%	100%	100%	13%	—	—	97%	2%	74%	8%	100%	33%	100%	97%	96%	96%
directed	25%	100%	93%	100%	72%	11%	91%	3%	90%	59%	0%	0%	0%	0%	100%	100%
graduatedFrom	80%	100%	70%	2%	79%	1%	89%	1%	82%	6%	84%	94%	85%	100%	77%	100%
hasChild	55%	100%	36%	1%	41%	0%	78%	1%	70%	7%	83%	26%	63%	100%	65%	100%
hasGender	64%	100%	100%	100%	—	—	98%	1%	—	—	92%	81%	91%	100%	100%	100%
hasParent*	0%	100%	37%	100%	100%	100%	—	—	—	—	0%	0%	0%	0%	100%	100%
isCitizenOf*	2%	100%	97%	100%	93%	6%	2%	1%	2%	7%	6%	33%	2%	53%	100%	100%
isConnectedTo	77%	100%	67%	23%	60%	12%	—	—	—	—	77%	62%	79%	100%	81%	100%
isMarriedTo*	38%	100%	84%	4%	92%	0%	66%	1%	51%	7%	25%	74%	40%	100%	29%	100%
wasBornIn	16%	100%	100%	100%	—	—	73%	3%	33%	5%	0%	0%	0%	0%	100%	100%

TABLE 7.2 – Precision and recall of all completeness oracles on YAGO3. Relations with a biased sample are marked with *.

Relation	CWA	PCA	card$_2$	Pop.	No change	Star	Class	AMIE
diedIn	60%	22%	—	4%	15%	50%	**99%**	96%
directed	40%	96%	19%	7%	71%	0%	0%	**100%**
graduatedFrom	89%	4%	2%	2%	10%	89%	**92%**	87%
hasChild	71%	1%	1%	2%	13%	40%	**78%**	**78%**
hasGender	78%	**100%**	—	2%	—	86%	95%	**100%**
hasParent*	1%	54%	**100%**	—	—	0%	0%	**100%**
isCitizenOf*	4%	98%	11%	1%	4%	10%	5%	**100%**
isConnectedTo	87%	34%	19%	—	—	68%	88%	**89%**
isMarriedTo*	55%	7%	0%	3%	12%	37%	**57%**	46%
wasBornIn	28%	**100%**	—	5%	8%	0%	0%	**100%**

TABLE 7.3 – F1 measure of all completeness oracles on YAGO3. Relations with a biased sample are marked with *.

Relation	CWA		PCA		$card_2$		Pop		Star		Classes		AMIE	
	Pr	Rec	Pr	Rec	Pr	Rec	Pr	Rec	Pr	Rec	Pr	Rec	Pr	Rec
alma_mater	82%	100%	80%	8%	95%	2%	57%	1%	76%	100%	76%	100%	76%	100%
brother	86%	100%	57%	0%	—	—	61%	1%	92%	96%	92%	100%	92%	100%
child	54%	100%	15%	1%	—	—	25%	0%	73%	86%	58%	95%	79%	68%
c._of_citizenship*	27%	100%	95%	100%	100%	5%	38%	1%	0%	0%	0%	0%	96%	100%
director	68%	100%	100%	100%	—	—	95%	1%	89%	100%	85%	94%	100%	100%
father*	3%	100%	100%	100%	100%	3%	16%	6%	100%	80%	4%	82%	100%	100%
mother*	1%	100%	100%	100%	100%	1%	12%	9%	52%	96%	2%	86%	100%	100%
place_of_birth	36%	100%	100%	100%	100%	4%	90%	3%	86%	41%	0%	0%	100%	100%
place_of_death	81%	100%	100%	21%	100%	1%	97%	1%	77%	87%	77%	87%	93%	100%
sex_or_gender	69%	100%	100%	100%	100%	3%	96%	1%	87%	98%	85%	97%	100%	100%
spouse*	40%	100%	88%	4%	—	—	29%	1%	38%	99%	37%	99%	38%	100%

TABLE 7.4 – Precision and recall of all completeness oracles on Wikidata. Relations with a biased sample are marked with *.

Relation	**CWA**	**PCA**	**card$_2$**	**Pop.**	**Star**	**Class**	**AMIE**
alma_mater	**90%**	14%	5%	1%	87%	87%	87%
brother	93%	1%	—	1%	94%	**96%**	**96%**
child	70%	1%	—	1%	**79%**	72%	73%
country_of_citizenship*	42%	97%	10%	3%	0%	0%	**98%**
director	81%	**100%**	—	3%	94%	89%	**100%**
father*	5%	**100%**	6%	9%	89%	8%	**100%**
mother*	3%	**100%**	3%	10%	67%*	5%	**100%**
place_of_birth	53%	**100%**	7%	5%	55%	0%	**100%**
place_of_death	89%	35%	1%	2%	81%	81%	**96%**
sex_or_gender	81%	**100%**	6%	3%	92%	91%	**100%**
spouse*	**57%**	7%	—	1%	54%	54%	55%

TABLE 7.5 – F1 measure of all completeness oracles on Wikidata. Relations with a biased sample are marked with *.

using completeness assertions. For this purpose, we filter out a fact prediction $r(s, o)$, if we know that $complete(s, r)$. For example, if the fact prediction says that a person has a parent, but the KB already knows 2 parents, then we discard the prediction.

Setup. We follow the same experimental setup as in Section 2.7 : We ran the standard AMIE system on YAGO3 and used the obtained rules to infer new facts. Each rule (and thus each prediction) comes with a confidence score. We grouped the predictions in buckets by confidence score. For each bucket, we resorted to crowd-workers to evaluate the precision of the predictions on a sample of 100 facts. The lower line in Figure 7.1 shows the number of predictions versus the cumulative precision estimated on the samples. Each data point corresponds to a predictions bucket, i.e., the first point on the left corresponds to the predictions with confidence score between the 0.9 and 1, the second point to those with confidence between 0.8 and 0.9, and so on. In the second phase of the experiment, we use completeness assertions to filter out predictions. We produce completeness assertions as in Section 7.6.3, by training AMIE with cross-validation on our entire set of gold standard completeness assertions. The upper line in Figure 7.1 shows the cumulative precision and number of predictions for each bucket after filtering.

Results. As we can observe, the filtering could successfully filter out all wrong predictions. The remaining predictions have a precision of 100%. This high precision has to be taken with a grain of salt : the remaining predictions are mainly about the citizenship, which is guessed from the place of residence or place of birth. The completeness assertions filter out any predictions that try to assign a second citizenship to a person, and thus drastically increase the precision. However, there are also a few other relations among the predictions, e.g., death place, or alma mater (guessed from the workplace of the academic advisor).

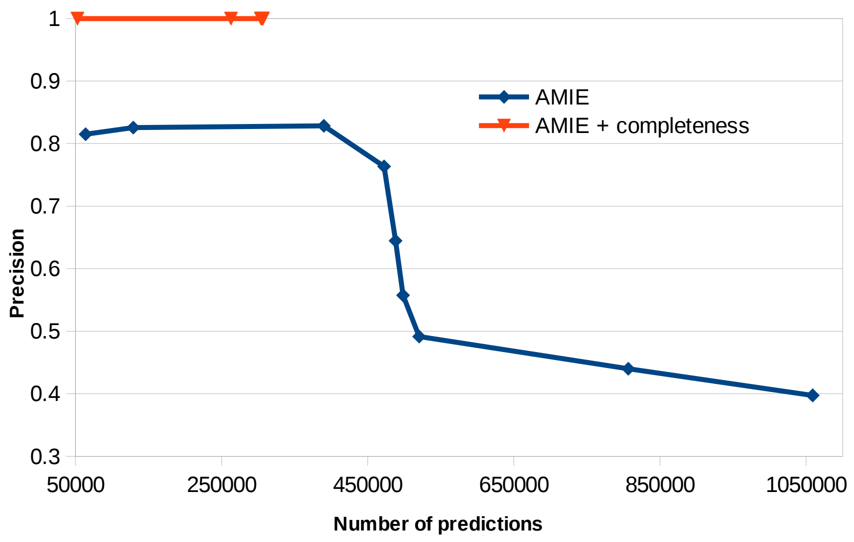


FIGURE 7.1 – Precision of fact prediction

This precision comes at a price. In total, AMIE made 1.05M predictions. Of these, 400K were correct. From these, the filtering incorrectly removed 110K. Thus, the filtering removes roughly 25% of the correct predictions as a side-effect. Still, we believe that our experiments make the case that completeness assertions can significantly improve the performance of fact prediction.

7.8 Conclusion

Completeness is an important dimension of quality, which is orthogonal to the dimension of correctness, and which has so far received less attention. In this chapter, we have defined and analyzed a range of simple and parametrized completeness oracles. We have also shown how to combine these oracles into more complex oracles by rule mining. Our experiments on YAGO and Wikidata prove that completeness can be predicted with high precision for many relations. Completeness estimations can be then used to improve fact prediction to 100% precision in specific cases.

We hope that this work can lead to new research avenues, aiming to design knowledge bases that are not only highly accurate, but also highly complete. All the data and rules produced for this chapter are available at
`http://luisgalarraga.de/completeness-in-kbs`.

Chapitre 8

Vision on Numerical Rule Mining

This chapter takes the first step towards numerical rule mining by defining a language to express numerical rules. We rely on the framework for rule mining presented in Chapter 2. The content of this chapter is based on the following publication :

— Luis Galárraga, Fabian Suchanek. *Towards a Numeric Rule Mining Language*. Automated Knowledge Base Construction Workshop, 2014.

8.1 Introduction

In chapter 2 we have proposed AMIE, a system to learn Horn rules on potentially incomplete KBs. The results presented in Section 2.6 show that AMIE can learn interesting rules on KBs, and that those rules can be used to infer new facts with high precision (Section 2.7). Nevertheless, our experimental setup made use of a subset of the KBs, namely the facts holding between entities. We deliberately discarded literal facts because they tend to hold unique values. For instance, it is rare for distinct entities in a KB to have the exact same string representation (relation *rdfs :label* in RDF Schema). Traditional rule mining on such relations would lead to rules with low support. Moreover, unlike relations defined between sets of entities, e.g., $livesIn : Person \rightarrow City$, relations such as surfaces or geographical locations are defined on a range, which is uncountably infinite. Also, prediction on real numbers is substantially different from link prediction and is normally addressed via regression models. While one could think of literal relations with finite countable or bounded ranges, such as *hasAge*, *hasHeight*, this is not the case for most of the literal relations we find in KBs.

In this chapter, we propose to exploit some of the literal relations in KBs to extend the scope of rule mining. This can be achieved by enlarging our language bias to accept *numerical rules*. A numerical rule contains variables that do not bind to named entities, but to literal numerical values. Consider, e.g., the rule

$$type(x, MarketEcon) \wedge import(x, y) \wedge export(x, z) \wedge cad(x, w) \Rightarrow w \approx 1.2 \times (y - z)$$

This rule says that if x is a market economy, then its current account deficit (*cad*) is roughly 120% of the difference between its import and export (measured in dollars,

Constraint	**Example rule with this constraint**
$x \geq \phi$	$age(x, y) \Rightarrow y \geq 0$
$x > \phi$	$type(x, solvent) \wedge balance(x, y) \Rightarrow x > 0$
$x = \phi$	$type(x, dog) \wedge numLegs(x, y) \Rightarrow y = 4$
$x \in [a, b]$	$lat(x, y) \Rightarrow y \in [-90, 90]$
$x \approx \phi$	$wikiLinks(x, y) \wedge articleLength(x, z) \Rightarrow z \approx 3.4 \times log(y)$

TABLE 8.1 – Numerical constraints and some examples

for example). Such rules would add tremendous value to today's KBs : First, the rules could be used to spot errors in the data. If the cad of a country does not fall in the expected range, then the value could be flagged as erroneous or less confident. Second, such rules could be used to compute missing information. If, e.g., we do not have the cad of France, we could compute it from the data. Finally, the rules carry human-understandable insight. The finding that the cad of a country correlates with its import and export has a value in itself [115].

Mining such rules is challenging for several reasons. First, there are infinitely many numerical constants. In the example, the factor could be refined to 1.21 or 1.3. Second, there are infinitely many ways in which numerical variables can correlate. For example, the cad could be the sum, product or any other function of the ages of its members of parliament. Thus, the search space becomes unbearably large. Finally, numerical rules bridge two traditionally different areas of research : inductive logic programming (ILP) and statistical data mining. While statistical data mining techniques can spot correlations of values, they are less able to simultaneously derive logical constraints such as *type(x, MarketEcon)*. ILP, on the other hand, traditionally does not deal with numerical values.

In this chapter, we study different types of numerical rules. We develop a unified language for writing down numerical rules, and we explain which types of rules can already be mined by existing work. With this thrust, we hope to lay the ground for research that will enable us to mine numerical rules just as efficiently as logical rules.

8.2 Numerical Rules

8.2.1 Numerical Constraints

One of the most intuitive ways to add numerical constraints to Horn rules is to permit *constraints*, i.e., atoms of the form $x \circ \phi$. Here, x is a numerical variable, $\circ$ is a binary operator that returns a truth value, and ϕ is an expression that may or may not involve variables, e.g., $x = 1.2 \times y$. The atom *holds* under an instantiation σ (Section 1.3.4), if $\sigma(x) \circ \sigma(\phi) = true$. Atoms that are not constraints are called *categorical atoms*. Categorical atoms that contain a numerical variable are called *numerical atoms*. Table 8.1 shows some types of numerical constraints and example rules.

We could also aim to mine cardinality constraints such as

$$hasWonPrize(x, EmmyAward) \wedge actedIn(x, y) \Rightarrow \#y > 10$$

Using the notation proposed in Section 7.5.2, this rule is equivalent to

$$hasWonPrize(x, EmmyAward) \Rightarrow moreThan_{10}(x, actedIn)$$

Cardinality constraints in the head of rules are particularly interesting because they allow us to mine and predict negations by setting $\#y = 0$ or $\#y < c$ for $c \in \mathbb{N}$. Negations, however, are inconsistent with the Open World Assumption that KBs make.

We could also mine rules such as :

$$type(x, City) \wedge hasPopulation(x, y) \Rightarrow y \sim N(\mu; \sigma)$$

This rule states that the population of cities follows a certain distribution. We leave such rules as future work and concentrate on the operators from Table 8.1.

Pathological cases. We notice that the addition of numerical constraints may lead to pathological cases such as

$$p(x,y) \Rightarrow z = y \quad (8.1)$$
$$p(x,y) \wedge z = y \Rightarrow r(x,y) \quad (8.2)$$
$$p(x,y) \wedge z > y \Rightarrow r(x,z) \quad (8.3)$$

In cases (1) and (2), we are binding a variable z that serves no purpose. In case (3), we are predicting $r(x, z)$ where z has no defined value. These phenomena are a particularity of constraints. If we replace the constraints by categorical atoms, the problem does not appear :

$$p(x,y) \Rightarrow q(z,y) \quad (8.4)$$
$$p(x,y) \wedge q(z,y) \Rightarrow r(x,y) \quad (8.5)$$
$$p(x,y) \wedge q(z,y) \Rightarrow r(x,z) \quad (8.6)$$

To avoid the pathological rules, we impose that every numerical variable has to appear in at least one categorical atom. We also impose that the head atom can only contain *bound* variables, i.e., variables that appear in the body either in a categorical atom or in a constraint that restricts it to one particular value. This is usually the case for the operator "=" [1], although not necessarily, cf. $x = x$. Unlike the standard AMIE language bias, we do not impose that every variable has to appear at least in two categorical atoms, because we want to allow rules such as $hasAdvisor(x, y) \wedge age(y, z) \Rightarrow z > 30$.

8.2.2 Descriptive and Predictive rules

Consider the following rule :

$$gdp(x,y) \wedge natDebt(x,z) \Rightarrow z = 0.9 \times y$$

1. The constraint $x \in [a, a]$ also binds.

This rule says that if x has a GDP and a national debt, then the debt can be computed as 90% of the GDP. If x has no national debt, then the rule will always hold trivially. We say that the rule is a *descriptive rule* because it will not try to *predict* the debt of x. Any rule with a numerical constraint in the head is descriptive. Now consider the following variant of the rule :

$$gdp(x,y) \ \Rightarrow\ \exists z : natDebt(x,z)\ \wedge z = 0.9 \times y$$

This rule says that every x that has a GDP also has a national debt, and that this debt amounts to 90% of the GDP. This formula allows for prediction of the national debt of a country given only its GDP. However, it does not qualify as a Horn rule, because it contains a conjunction and an existential quantifier in the head. We propose to write this rule as

$$gdp(x,y)\ \wedge\ z = 0.9 \times y\ \Rightarrow\ natDebt(x,z)$$

This rule has the same semantics, but does not require the existential quantifier in the head[2]. We call numerical rules with a categorical atom in the head *predictive rules*.

8.2.3 Functional notation

A *function* is a relation that has, for each subject, at most one object in the KB (Section 1.3.3). It is very common for rule mining systems to assume that numerical atoms bind to relations that are functions. In our survey of the state of the art in Section 8.3, we did not find any approach that would not make this assumption. For example, a country can have only one national debt value. This assumption makes sense only if we do not take into account the dimension of time.

If all numerical atoms are functions, we can use them in constraints. For example, given the rule :

$$gdp(x,y)\ \wedge\ natDebt(x,z)\ \wedge\ z > 5y\ \Rightarrow\ hasProblem(x, \text{‘}true\text{’})$$

We could rewrite it as :

$$natDebt(x) > 5{\times}gdp(x)\ \ \Rightarrow\ \ hasProblem(x, \text{‘}true\text{’})$$

In general, any expression of the form $f(x,y)\ \wedge\ y \circ \phi$ where f is a functional relation, can be rewritten as $f(x) \circ \phi$. This rewriting applies also if the function term is in the head :

$$type(x, Indebted)\ \wedge\ natDebt(x,y)\ \wedge\ gdp(x,z)\ \Rightarrow\ y = 5{\times}z$$

This rule becomes :

$$type(x, Indebted)\ \Rightarrow\ natDebt(x) = 5{\times}gdp(x)$$

This transformation, however, allows only the descriptive semantics of the operator “=”. There is no way to predict that x has a national debt, if x does not have that property

2. The FORS system [59] uses the same notation, albeit with the Prolog predicate *is/2*.

already. To allow for this subtlety, we permit the assignment atom $f(x) := \phi$ in the head. If a rule contains the pattern

$$\ldots \;\; y = \phi \;\; \Rightarrow f(x, y)$$

We can rewrite it as

$$\ldots \; \Rightarrow f(x) := \phi$$

Thus, given the rule

$$type(x, Indebted) \;\wedge\; gdp(x, z) \;\wedge\; y = 5 \times z \;\Rightarrow\; natDebt(x, y)$$

We can rewrite it as

$$type(x, Indebted) \;\Rightarrow\; natDebt(x) := 5 \times gdp(x)$$

In the way we have defined our rewriting, every rule with function terms can be transformed into an equivalent rule without function terms. Vice versa, every rule in which all numerical atoms are functions can be rewritten to an equivalent rule without numerical variables. As we will show in the next section, function terms allow us to extend the notions of support and confidence – designed originally for categorical rules – to rules with constraints. For this reason, we propose to use the language of Horn rules with function terms as a unified language for rules with numerical constraints.

8.2.4 Evaluation Metrics

Support. The statistical significance of a rule is measured by the *support* metric. Recall from Section 2.3.2.1 that the support of a rule $\boldsymbol{B} \Rightarrow H$ is the absolute number of cases in the KB where the rule holds. The support follows the formula :

$$supp(\boldsymbol{B} \Rightarrow H) := \#(vars(H)) : \exists z_1, ..., z_m : \boldsymbol{B} \wedge H$$

This definition of support does not transfer well to rules that have a constraint in the head, such as $natDebt(x, y) \wedge gdp(x, z) \Rightarrow y > z$. This is because we would want to count the different countries x for which this rule holds, not the different pairs of countries and debt values y, z. In our language of function terms, we get this behavior for free : The rule becomes $\Rightarrow natDebt(x) > gdp(x)$ (with an empty body). In such a rule, $vars(H) = \{x\}$ thanks to the function terms.

Confidence. The standard (Section 2.3.3.1) and PCA confidence (Section 2.4) measure the ratio of correct conclusions of a rule according to the formulas :

$$conf_{std}(\boldsymbol{B} \Rightarrow H) := \frac{supp(\boldsymbol{B} \Rightarrow H)}{\#vars(H) : \exists z_1, \ldots, z_k : \boldsymbol{B'}}$$

$$conf_{pca}(\boldsymbol{B} \Rightarrow H) := \frac{supp(\boldsymbol{B} \Rightarrow H)}{\#vars(H) : \exists z_1, \ldots, z_k, y' : \boldsymbol{B'} \wedge r(x, y')}$$

Here, $\boldsymbol{B}'$ is the body of the rule after all function terms have been rewritten. For example, consider the following descriptive rule

$$\Rightarrow \ natDebt(x) = 0.9 \times gdp(x)$$

From Section 8.2.3 it follows that this rule denotes

$$gdp(x,y) \ \wedge \ natDebt(x,z) \ \Rightarrow \ z = 0.9 \times y$$

The standard confidence of this example rule is given by the expression :

$$conf_{std}(\boldsymbol{B} \Rightarrow H) := \frac{\#x : \exists\, y,z : gdp(x,y) \wedge natDebt(x,z) \wedge z = 0.9{\times}y}{\#x : \exists\, y,z : gdp(x,y) \wedge natDebt(x,y)}$$

We notice that for descriptive rules, the PCA confidence boils down to the standard confidence. To see this, recall that if the head of the rule contains an expression of the form $x \circ \phi(y)$ in its function term free form, the head atom becomes $x \circ \phi(y')$ in the denominator, like in

$$conf_{pca}(\boldsymbol{B} \Rightarrow H) := \frac{\#x : \exists\, y,z : gdp(x,y) \wedge natDebt(x,z) \wedge z = 0.9{\times}y}{\#x : \exists\, y,z : gdp(x,y) \wedge natDebt(x,y) \wedge z = 0.9{\times}y'}$$

Since y' is an unbound variable, the expression $z = 0.9 \times y'$ is trivially fulfilled and can be omitted.

Now let us consider the predictive rule

$$type(x, Indebted) \Rightarrow natDebt(x) := 5{\times}gdp(x)$$

denoting

$$type(x, Indebted) \wedge gdp(x,z) \wedge y = 5{\times}z \Rightarrow natDebt(x,y)$$

The standard confidence of such rule is defined according to the formula :

$$conf_{std}(\boldsymbol{B} \Rightarrow H) := \frac{\#x : \exists\, y,z : type(x, Indebted) \wedge gdp(x,z) \wedge y = 5{\times}z \wedge natDebt(x,y)}{\#x : \exists\, y,z : type(x, Indebted) \wedge gdp(x,z) \wedge y = 5{\times}z}$$

where the equality constraint in the denominator is trivially satisfied and could be omitted :

$$conf_{std}(\boldsymbol{B} \Rightarrow H) := \frac{\#x : \exists\, y,z : type(x, Indebted) \wedge gdp(x,z) \wedge y = 5{\times}z \wedge natDebt(x,y)}{\#x : \exists\, z : type(x, Indebted) \wedge gdp(x,z)}$$

We remark that the standard confidence was originally designed for descriptive rules under the Closed World Assumption, that is, it punishes rules concluding facts that are not in the KB. In this example, any indebted country with a GDP will lower the confidence value, no matter if it has the wrong debt value or no value at all. For KBs operating under the Open World Assumption, we proposed the PCA confidence. In our example, this yields :

$$conf_{pca} := \frac{\#x : \exists\, y,z : type(x, Indebted) \wedge gdp(x,z) \wedge y = 5{\times}z \wedge natDebt(x,y)}{\#x : \exists\, y,z,y' : type(x, Indebted) \wedge gdp(x,z) \wedge y = 5{\times}z \wedge natDebt(x,y')}$$

Again, the equality constraint in the denominator is trivially fulfilled because y is unbound, thus the constraint can be omitted. We see that the PCA confidence counts as counter-examples only those countries for which a national debt value is known. While the standard and PCA confidence are equivalent for descriptive rules, we prefer the PCA confidence for predictive rules.

Confidence and support measure two orthogonal dimensions of the quality of a rule, which can be combined [74], but are kept distinct in most rule mining approaches [4, 43, 46, 113].

Error measures. If a rule contains a numerical constraint in the head, other measures may become applicable. For example, if the rule concludes $y \approx 3x$, then the quality of the rule could depend on the difference $|y - 3x|$. If we know that x follows certain probability distribution, $x \sim F(\mu, \sigma)$, then the confidence could be the probability that our predictions were drawn from such a distribution, i.e., $P(y \mid y = 3x \wedge x \sim F(\mu; \sigma))$. While such measures abound, the difficulty is to make numeric measures interoperable with classical measures. For example, if a rule mining algorithm produces both numerical and non-numerical rules, and if numerical rules have an error measure but no confidence, then it is unclear how these rules can be ranked together. We propose to replace a constraint of the form $y \approx \phi$ by $y{>}\phi{-}\epsilon \;\wedge\; y{<}\phi{+}\epsilon$ for a suitable error margin ϵ. If the constraint appears in the head, the rule has to be split into two rules that each have one of the conjuncts in the head (to still meet the language of Horn rules). The value of ϵ depends on the domain of y. If the domain of y operates under a *value scale*, i.e., scales for which ratios are not defined (such as temperatures or geographic coordinates), ϵ can be defined as an *absolute error*. This requires the choice of an appropriate numerical constant, which depends, e.g., on the unit system that y uses. An absolute error of 2 units for predicting temperatures in Celsius may be too loose if the scale is changed to Fahrenheit or if we are trying to predict latitudes. If y operates under a *ratio scale* (as is the case for populations or distances), we can use a *relative error* $\epsilon = \alpha y$, $\alpha \in (0, 1)$. Then, α is independent of the unit system and can be set to a default value.

With this change, all error measures become binary measures, and hence interoperable with the categorical support and confidence measures.

8.3 Existing Work

We survey here existing work in the area of numerical rule mining, and express the rules in our proposed language with function terms. The work of [99] studies the problem of mining optimized rules of the form $A \in [a, b] \wedge C_1 \Rightarrow C_2$ on a database table. Here, A is a column name and C_1, C_2 are equality constraints on columns, as in the rule

$$year \in [1990, 2000] \wedge src\_city = NY \Rightarrow dest\_city = Paris$$

In our language, we would define a unique identifier for each row of the table and express the columns as binary relations and function terms, e.g.,

$$year(id) \in [1990, 2000] \wedge src\_city(id, NY) \Rightarrow dest\_city(id, Paris)$$

The problem consists in finding the values of a and b that maximize either support or confidence given a minimum threshold for the other metric. The general problem with an arbitrary disjunction of inequality constraints is shown to be NP-Hard [99]. One way to reduce the search space is to *discretize* it. [39] uses equi-depth bucketing and computational geometry techniques to calculate a and b almost optimally, providing bounds for the approximation error. The approach described in [15] proposes to optimize for the gain, a function of support and confidence. A method based on genetic algorithms [107] allows mining confidence-optimized rules with categorical atoms and multiple interval constraints. In our language these rules take the form, e.g., $height(x) \in [150, 170] \Rightarrow weight(x) \in [50, 90]$.

The work presented in [6] incorporates interval constraints into the standard ILP framework. In our language, the rules look like $age(x) \in [a, b] \Rightarrow status(x, Single)$. The goal is to find a, b that maximize the rule confidence. Before refining the rule, the system performs an independence test between the age of people and their marital status. If the variables are not independent, then the algorithm reports intervals where it is more likely to find single people, i.e., intervals where confidence is maximized.

Regression trees [60] are a common method to learn functional correlations in numerical structured data. Imagine we want to predict the length of a Wikipedia article as a function of its number of links. In our language for numerical rules, such a rule is written as

$$\Rightarrow wikiLength(x) := g(wikiLinks(x))$$

Most mining systems use the writing

$$wikiLinks(x, y') \Rightarrow \exists\, y : wikiLength(x, y) \wedge y = g(y')$$

The learning process starts with a training set consisting of all the articles x where the dependent variable y is known. If some quality conditions are met, a regression model $y \approx g(y')$ is induced and the algorithm stops. Otherwise, the training set is split into positive and negative examples and their hypothesis refined with new categorical atoms, e.g., $type(x, Scientist)$. The algorithm is then recursively applied to the new hypotheses and training sets. FORS [59] was one of the first systems in integrating functional correlations using regression trees into the ILP framework. In the syntax of FORS, the rule is expressed as follows :

$$wikiLinks(x, y') \wedge is(y, g(y')) \Rightarrow wikiLength(x, y)$$

FORS cannot mine the descriptive variant of such a rule. Furthermore, the system was tried out only on toy examples and is unlikely to scale to the millions of facts in today's KBs. A more recent approach [38] relies on regression trees for the prediction of numerical attributes in OWL KBs. The approach learns rules that predict the value of one fixed numerical attribute of an entity, based on the properties of that entity. In our language, these rules look like $type(x, TalkShow) \wedge dealsWith(x, Sports) \Rightarrow popularity(x) := 12\%$.

8.4 Conclusion

We have seen that there has been quite some work on numeric rule mining. In our proposed rule language, these approaches become comparable. However, so far, these works solve only islands of numeric rule mining. None of the approaches can mine full-fledged rules with function terms at scale. This, however, may be about to change : As we make progress on small scale numerical rule mining, and as categorical rule mining starts to scale, these islands may soon become connected. A unified syntax for numerical rules may be the first step.

Chapitre 9

Conclusion

9.1 Summary

In this thesis we have illustrated the great value that rule mining techniques can bring to knowledge bases (KBs). The contribution of this thesis can be categorized along two major lines : (1) how to efficiently mine rules on KBs (rule mining), and (2) how to apply those rules in multiple tasks related to the maintenance of semantic data (applications).

Rule Mining. Regarding the extraction of rules on KBs, the contributions of this thesis are as follows :

— We propose a novel way to generate counter-evidence for rule mining based on the Partial Completeness Assumption (PCA, Section 2.4). Our empirical results on YAGO, show that the PCA is much more sensible at generating counter-evidence than the Closed World Assumption (CWA), made by several rule mining systems [20, 46]. Moreover, the definition of confidence based on the PCA identifies predictive rules more accurately than the standard CWA confidence. The PCA confidence in combination with a simple joint-prediction approach, and the use of types, can predict hundreds of thousands of facts on YAGO with a precision of up to 70% (Section 2.7.1).

— We have shown that, in spite of being inherently a hard problem in its general form, rule mining can be efficiently solved on large KBs. In this regard, we have proposed AMIE (Section 2.5), a system designed to mine Horn rules on potentially incomplete KBs operating under the Open World Assumption (OWA). AMIE relies on a monotonic definition of support, a parallelized implementation, and a set of heuristics to drive the exploration of the search space in an efficient manner. This allows the system to mine rules on large KBs with millions of facts in a matter of minutes.

Applications. We have discussed multiple applications of rule mining in this thesis.

— *Wikilinks Semantification.* In Chapter 4, we have explained how to use rule mining to semantify wikilinks in DBpedia. Our simple approach finds semantification rules, a particular type of Horn rules. Those rules are then used to make predictions about the semantic relations that hold between the endpoints of a wikilink. With this approach, we are able to semantify 180K wikilinks in DBpedia with high precision.

— *ROSA rules.* In Chapter 5, we have presented ROSA rules. These rules express complex alignments between the schemas of two KBs. Such alignments can be used to integrate the data from different KBs in the spirit of a unified web of objects. We have shown how to mine ROSA rules from two instance-aligned KBs. We have also provided some examples of schema alignments between real-world KBs.

— *Canonicalization of open KBs.* We have introduced a method to canonicalize open KBs, i.e., schema-free KBs with triples extracted from arbitrary text (Chapter 6). This process encompasses two subproblems : clustering of synonym noun phrases and clustering of verbal phrases (relations). We have performed a systematic study of multiple signals of synonymy for noun phrases in a web-extracted open KB. Our results show that the IDF tokens overlap constitutes a simple, yet strong signal of synonym for noun phrases. This signal can still be combined appropiately with other signals via machine learning methods and clustering techniques for detection of synonyms in web extractions. The clustering of verbal phrases, on the other hand, can be accurately solved via rule mining techniques, given a canonicalization of the noun phrases.

— *Prediction of Completeness.* In Chapter 7, we have conducted a systematic study of a set of signals for completeness in KBs. These signals span from simple heuristics, such as the Partial Completeness Assumption, to more complex oracles based on rule mining techniques and inference. We have illustrated how to obtain completeness annotations from crowd-workers. We have also described how to learn rules from those annotations. The obtained rules can predict completeness assertions for entities in the domains of a set of relations in YAGO and Wikidata, with very high precision in some cases. We have made use of predicted completeness assertions to improve the precision of a fact inference task. In this scenario, the completeness statements play the role of counter-evidence that allow us to discard spurious predictions and achieve 100% on a sample of 1000 predictions, drawn from a population of 1M facts.

9.2 Outlook

We now provide an outlook of potential avenues of research in the field of rule mining in knowledge bases. As we did for Section 9.1, we frame our outlook within the lines of rule mining per se and its potential use cases.

Rule Mining. AMIE mines Horn rules in the language of closed rules $\mathcal{L}_{closed}$ (Section 2.3.1). While we have justified the choice $\mathcal{L}_{closed}$ for the purposes of prediction and tractability, this language may be too restrictive for some applications. For example, our work on completeness in KBs (Chapter 7) required us to relax the AMIE language bias to allow for existentially quantified variables in the body of rules. Besides, our vision on numerical rule mining in Chapter 8 illustrates the value that numerical rules could bring to rule mining and KBs.

To address such a limitation, we envision to develop a framework for general rule mining inspired by the AMIE algorithm. We believe this is feasible since AMIE mines rules in a top-down, still fairly generic fashion. In this spirit, users could customize the AMIE framework by defining, among other things, (a) their own mining operators, (b) their own metrics to gauge quality, and (c) their own data sources. Requirement (a) would allow users to mine non-closed Horn rules, rules with numerical constraints, or even patterns outside the language of Horn rules. Requirement (c) would let users exploit other scalable data storage solutions. In [20], the authors use Apache Spark to store the KB and compute the scores for rules. This design decision allows their approach to scale to datasets that are even larger than those discussed in this work.

Numerical Rule Mining. We could extend our vision on numerical rule mining to accept expressions on strings, like in the rules :

$$\Rightarrow label(x) := firstName(x) + lastName(x)$$

$$type(x, Country) \Rightarrow alpha3Code(x) := label(x)[1,3]$$

Applications. In the following we discuss challenges and potential future work in the context of the applications we have studied in this thesis.

— *ROSA rules and schema alignment.* In Chapter 5 we proposed a method to mine schema mappings from a KB $\mathcal{K}_1$ to a KB $\mathcal{K}_2$ on a coalesced KB $\mathcal{K}$. The coalescing step depends on the existing instance alignments. In this line, the study of different coalescing strategies is a potential research direction. In addition, the problem of discriminating between soft constraints and crisp schema mappings for some types of ROSA rules remains a challenge in this context.

— *Joint prediction.* In Section 2.7 we have proposed a fairly simple method for joint prediction of facts. Our approach assigns predictions a score that depends on all the rules that entail the prediction. The strategy makes the strong assumption that rules are independent events. In general, however, this is not case. Thus, an interesting direction for future work is the design of joint prediction mechanisms that take into account the correlations between rules in a KB. Furthermore, in order to improve the recall of fact inference, we envision an iterative approach that deduces new facts and uses them as evidence to mine new rules. Desirable properties of such an approach are *determinism* and *convergence*, which are challenging for two reasons. First, rules can make contradictory predictions, which due to the OWA, cannot be trivially spotted. Second, since fact inference is not perfect, the evidence provided by those predictions should not have the same weight as the evidence contained originally in the KB. This natural requirement poses tractability challenges, if we confer weights a probabilistic meaning. Such interpretation would require us to redefine our queries and metrics (support and confidence) to operate in a setting where facts have also a confidence or probability of being correct.

— *Completeness as counter-evidence for Rule Mining.* In Section 7.7, we applied predicted completeness assertions to generate explicit counter-evidence for an inference approach. These counter-examples allowed us to discard spurious predictions and to improve the precision of fact inference. While this could also help us for the vision of joint prediction we just discussed, it could also provide counter-evidence for the rule mining itself. In this line of thinking, for example, a new definition of confidence could use as counter-examples those predictions that are believed to be wrong by a completeness oracle, and the PCA otherwise.

— *Canonicalization of open KBs.* We have studied the canonicalization of open KBs as a two-dimensional problem in Chapter 6, where we solve the canonicalization of noun phrases first and use it to canonicalize the verbal phrases. We envision an iterative approach, in which the clustering of verbal phrases can also help improve the clustering of noun phrases. Moreover, we could also use rule mining and inference to deduce new facts for open KBs. This, in combination with schema alignment techniques could serve the task of populating existing KBs.

The rise of large machine-readable KBs has given computers the power to handle the contents of the web in a deeper and more insightful way. KBs provide a model of the real-world that allow computer programs to reason about the available knowledge. For example, KBs have changed the way we ask search engines. Thanks to the available semantic data, search engines have departed from the traditional bag-of-words query model to a more semantic concept-centric model. This allows search engines to deliver accurate answers to queries about e.g., artists we like or places we want to visit.

With this thesis, we have a made a step forward towards an even more semantic web, by making sense out of the existing semantic information. Rule Mining finds trends and insights in data. Such insights are powerful tools that can make computers more proactive and “smarter”. Rules allow computer programs to reason and predict factual information, to describe the rules that govern a given domain of knowledge (taking the role of human experts), and in a broader sense, to maintain the quality of the knowledge that is available to us. We have shown the great value that rule mining techniques can bring to the field of knowledge management and, we hope this will motivate further research in this direction. We believe that study of more expressive rule languages and the develop of more scalable methods for rule mining will be an important factor in the realization of the Semantic Web.

Bibliographie

[1] Z. Abedjan, J. Lorey, and F. Naumann. Reconciling ontologies and the web of data. In *CIKM*, 2012.

[2] Ziawasch Abedjan and Felix Naumann. Synonym analysis for predicate expansion. In *ESWC*, 0 2013.

[3] Hilde Adé, Luc Raedt, and Maurice Bruynooghe. Declarative bias for specific-to-general ilp systems. *Machine Learning*, 20, 1995.

[4] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *SIGMOD*, 1993.

[5] Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. Fast discovery of association rules. In *Advances in knowledge discovery and data mining*, 1996.

[6] Martin Theobald André Melo and Johanna Völker. Correlation-based refinement of rules with numerical attributes. In *FLAIRS*, 2014.

[7] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. DBpedia : A nucleus for a Web of open data. In *ISWC*, 2007.

[8] David Aumueller, Hong Hai Do, Sabine Massmann, and Erhard Rahm. Schema and ontology matching with coma++. In *SIGMOD*, 2005.

[9] Amit Bagga and Breck Baldwin. Entity-based cross-document coreferencing using the vector space model. In *COLING*, 1998.

[10] Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. Open information extraction from the web. In *IJCAI*, 2007.

[11] Roberto J. Bayardo. Mining the most interesting rules. pages 145–154, 1999.

[12] Tim Berners-Lee, James Hendler, Ora Lassila, et al. The semantic web. *Scientific american*, 284(5) :28–37, 2001.

[13] Christoph Böhm, Gerard de Melo, Felix Naumann, and Gerhard Weikum. Linda : distributed web-of-data-scale entity matching. In *CIKM*, 2012.

[14] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, 2013.

[15] Sergey Brin, Rajeev Rastogi, and Kyuseok Shim. Mining optimized gain rules for numeric attributes. In *SIGKDD*, 1999.

[16] Lawrence D. Brown, T. Tony Cai, and Anirban DasGupta. Interval estimation for a binomial proportion. *Statistical Science*, 2001.

[17] Andrea Calì, Thomas Lukasiewicz, Livia Predoiu, and Heiner Stuckenschmidt. Rule-based approaches for representing probabilistic ontology mappings. In *URSW (LNCS Vol.)*, 2008.

[18] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.

[19] Craig Chasseur and Jignesh M. Patel. Design and evaluation of storage organizations for read-optimized main memory databases. *Proc. VLDB Endow.*, 6(13), August 2013.

[20] Yang Chen, Sean Goldberg, Daisy Zhe Wang, and Soumitra Siddharth Johri. Ontological pathfinding : Mining first-order knowledge from large knowledge bases. In *Proceedings of the 2016 ACM SIGMOD international conference on Management of data*. ACM, 2016.

[21] Yun Chi, Richard R. Muntz, Siegfried Nijssen, and Joost N. Kok. Frequent Subtree Mining - An Overview. *Fundam. Inf.*, 66(1-2), 2004.

[22] Philipp Cimiano, Andreas Hotho, and Steffen Staab. Comparing Conceptual, Divisive and Agglomerative Clustering for Learning Taxonomies from Text. In *ECAI*, 2004.

[23] Isabel F. Cruz, Flavio Palandri Antonelli, and Cosmin Stroe. Agreementmaker : Efficient matching for large real-world schemas and ontologies. *PVLDB*, 2(2), 2009.

[24] Claudia d'Amato, Volha Bryl, and Luciano Serafini. Data-driven logical reasoning. In *URSW*, 2012.

[25] Claudia d'Amato, Nicola Fanizzi, and Floriana Esposito. Inductive learning for the Semantic Web : What does it buy ? *Semant. web*, 1(1,2), April 2010.

[26] F. Darari, S. Razniewski, R. Prasojo, and W. Nutt. Enabling fine-grained RDF data completeness assessment. In *ICWE*, 2016.

[27] Jérôme David, Fabrice Guillet, and Henri Briand. Association Rule Ontology Matching Approach. *Int. J. Semantic Web Inf. Syst.*, 3(2), 2007.

[28] Luc Dehaspe and Hannu Toironen. Discovery of relational association rules. In *Relational Data Mining*. Springer-Verlag New York, Inc., 2000.

[29] Luc Dehaspe and Hannu Toivonen. Discovery of frequent DATALOG patterns. *Data Min. Knowl. Discov.*, 3(1), March 1999.

[30] Luciano Del Corro and Rainer Gemulla. Clausie : clause-based open information extraction. In *WWW*, 2013.

[31] Robin Dhamankar, Yoonkyong Lee, AnHai Doan, Alon Halevy, and Pedro Domingos. imap : Discovering complex semantic matches between database schemas. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, pages 383–394, New York, NY, USA, 2004. ACM.

[32] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault : A web-scale approach to probabilistic knowledge fusion. In *KDD*, 2014.

[33] Xin Dong, Alon Halevy, and Jayant Madhavan. Reference reconciliation in complex information spaces. In *SIGMOD*, 2005.

[34] F. Erxleben, M. Günther, M. Krötzsch, J. Mendez, and D. Vrandecic. Introducing Wikidata to the linked data web. In *ISWC*, 2014.

[35] O. Etzioni, A. Fader, J. Christensen, S. Soderland, and Mausam. Open Information Extraction : the Second Generation. In *IJCAI*, 2011.

[36] Jérôme Euzenat and Petko Valtchev. Similarity-based ontology alignment in OWL-Lite. In *Proc. 16th european conference on artificial intelligence (ECAI)*, Proc. 16th european conference on artificial intelligence (ECAI), pages 333–337, Valencia, Spain, August 2004. IOS press. euzenat2004c.

[37] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *EMNLP*, 2011.

[38] N. Fanizzi, C. d'Amato, and F. Esposito. Towards numeric prediction on owl knowledge bases through terminological regression trees. In *ICSC*, 2012.

[39] Takeshi Fukuda, Yasuhiko Morimoto, Shinichi Morishita, and Takeshi Tokuyama. Mining optimized association rules for numeric attributes. *Journal of Computer and System Sciences*, 58(1) :1 – 12, 1999.

[40] Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. FACC1 : Freebase annotation of ClueWeb corpora, version 1, June 2013.

[41] Luis Galárraga. Interactive Rule Mining in Knowledge Bases. In *31ème Conférence sur la Gestion de Données (BDA 2015), Île de Porquerolles*, October 2015. Papier Demo.

[42] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. Fast Rule Mining in Ontological Knowledge Bases with AMIE+. *VLDB Journal*, November 2015.

[43] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. Amie : Association rule mining under incomplete evidence in ontological knowledge bases. In *WWW*, 2013.

[44] William A. Gale, Kenneth W. Church, and David Yarowsky. One sense per discourse. In *Workshop on Speech and Natural Language*, 1992.

[45] Alberto García-Durán, Antoine Bordes, and Nicolas Usunier. Effective blending of two and three-way interactions for modeling multi-relational data. In *ECML-PKDD*, 2014.

[46] Bart Goethals and Jan Van den Bussche. Relational Association Rules : Getting WARMER. In *Pattern Detection and Discovery*, volume 2447. Springer Berlin / Heidelberg, 2002.

[47] Georg Gottlob, Nicola Leone, and Francesco Scarcello. On the complexity of some inductive logic programming problems. In *Proceedings of the 7th International Workshop on Inductive Logic Programming*, ILP '97, pages 17–32, London, UK, UK, 1997. Springer-Verlag.

[48] Paul Grice. Logic and conversation. *J. Syntax and semantics*, 3, 1975.

[49] Gunnar Aastrand Grimnes, Peter Edwards, and Alun D. Preece. Learning Meta-descriptions of the FOAF Network. In *ISWC*, 2004.

[50] B. Hachey, W. Radford, J. Nothman, M. Honnibal, and J. Curran. Evaluating entity linking with wikipedia. *Artificial Intelligence*, 194, 2013.

[51] Michael Hartung, Anika Groß, and Erhard Rahm. Conto-diff : generation of complex evolution mappings for life science ontologies. *J. of Biomedical Informatics*, 46(1), 2013.

[52] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[53] Sebastian Hellmann, Jens Lehmann, and Sören Auer. Learning of OWL Class Descriptions on Very Large Knowledge Bases. *Int. J. Semantic Web Inf. Syst.*, 5(2), 2009.

[54] Johannes Hoffart, Yasemin Altun, and Gerhard Weikum. Discovering emerging entities with ambiguous names. In *WWW*, 2014.

[55] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. YAGO2 : a spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence Journal*, 2013.

[56] Yi Huang, Volker Tresp, Markus Bundschus, Achim Rettinger, and Hans-Peter Kriegel. Multivariate prediction for learning on the semantic web. In *ILP*, 2011.

[57] Prateek Jain, Pascal Hitzler, Amit P. Sheth, Kunal Verma, and Peter Z. Yeh. Ontology alignment for linked open data. In *ISWC*, 2010.

[58] Joanna Jozefowska, Agnieszka Lawrynowicz, and Tomasz Lukaszewski. The role of semantics in mining frequent patterns from knowledge bases in description logics with rules. *Theory Pract. Log. Program.*, 10(3), 2010.

[59] Aram Karalič and Ivan Bratko. First order regression. *Machine Learning*, 26(2-3) :147–176, 1997.

[60] Stefan Kramer. Structural regression trees. In *AAAI*, 1996.

[61] Jayant Krishnamurthy and Tom M. Mitchell. Which noun phrases denote which concepts ? In *HLT*, 2011.

[62] Michihiro Kuramochi and George Karypis. Frequent Subgraph Discovery. In *ICDM*. IEEE Computer Society, 2001.

[63] S. Lacoste-Julien, K. Palla, A. Davies, G. Kasneci, T. Graepel, and Z. Ghahramani. Sigma : Simple greedy matching for aligning large knowledge bases. In *KDD*, 2013.

[64] Dustin Lange, Christoph Böhm, and Felix Naumann. Extracting structured information from wikipedia articles to populate infoboxes. In *CIKM*, 2010.

[65] Ni Lao, Tom Mitchell, and William W. Cohen. Random walk inference and learning in a large scale knowledge base. In *EMNLP*, 2011.

[66] N. Lavrac and S. Dzeroski. Inductive logic programming. In *WLP*, 1994.

[67] Jens Lehmann. DL-Learner : Learning Concepts in Description Logics. *Journal of Machine Learning Research (JMLR)*, 10, 2009.

[68] A. Y. Levy. Obtaining complete answers from incomplete databases. In *VLDB*, 1996.

[69] Thomas Lin, Mausam, and Oren Etzioni. Entity linking at web scale. In *AKBC-WEKEX*, 2012.

[70] Francesca A. Lisi. Building rules on top of ontologies for the semantic web with inductive logic programming. *TPLP*, 8(3), 2008.

[71] http ://linkeddata.org/.

[72] Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. Generic schema matching with cupid. In *VLDB*, 2001.

[73] Alexander Maedche and Valentin Zacharias. Clustering Ontology-Based Metadata in the Semantic Web. In *PKDD*, 2002.

[74] Farzaneh Mahdisoltani, Joanna Biega, and Fabian M. Suchanek. Yago3 : A knowledge base from multilingual wikipedias. In *CIDR*, 2015.

[75] T Mamer, CH Bryant, and JM McCall. L-modified ilp evaluation functions for positive-only biological grammar learning. In F Zelezny and N Lavrac, editors, *Inductive logic programming*, number 5194 in LNAI. Springer-Verlag, 2008.

[76] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*, chapter Hierarchical Clustering. Cambridge University Press, 2008.

[77] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*, chapter Scoring, term weighting and the vector space model. Cambridge University Press, 2008.

[78] Cynthia Matuszek, John Cabral, Michael Witbrock, and John Deoliveira. An introduction to the syntax and content of Cyc. In *AAAI Spring Symposium*, 2006.

[79] Andrew McCallum, Kamal Nigam, and Lyle H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *KDD*, 2000.

[80] Deborah L. McGuinness, Richard Fikes, James Rice, and Steve Wilder. An Environment for Merging and Testing Large Ontologies. In *KR*, 2000.

[81] Changping Meng, Reynold Cheng, Silviu Maniu, Pierre Senellart, and Wangda Zhang. Discovering meta-paths in large heterogeneous information networks. In *WWW*, 2015.

[82] Renée J. Miller, Laura M. Haas, and Mauricio A. Hernández. Schema mapping as query discovery. In *VLDB*, 2000.

[83] B. Min, R. Grishman, L. Wan, C. Wang, and D. Gondek. Distant supervision for relation extraction with an incomplete knowledge base. In *NAACL*, 2013.

[84] Bonan Min, Shuming Shi, Ralph Grishman, and Chin Y. Lin. Ensemble semantics for large-scale unsupervised relation extraction. In *EMNLP-CoNLL*, 2012.

[85] A. Motro. Integrity = Validity + Completeness. *TODS*, 1989.

[86] Stephen Muggleton. Inverse entailment and progol. *New Generation Comput.*, 13(3&4), 1995.

[87] Stephen Muggleton. Learning from positive data. In *ILP*, 1997.

[88] Ndapandula Nakashole, Mauro Sozio, Fabian Suchanek, and Martin Theobald. Query-time reasoning in uncertain rdf knowledge bases with soft and hard rules. In *Workshop on Very Large Data Search (VLDS) at VLDB*, 2012.

[89] Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. Patty : A taxonomy of relational patterns with semantic types. In *EMNLP*, 2012.

[90] Victoria Nebot and Rafael Berlanga. Finding association rules in semantic web data. *Knowl.-Based Syst.*, 25(1), 2012.

[91] Thomas Neumann and Gerhard Weikum. RDF-3X : a RISC-style engine for RDF. *Proc. VLDB Endow.*, 1(1), August 2008.

[92] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *ICML*, 2011.

[93] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. Factorizing yago : scalable machine learning for linked data. In *WWW*, 2012.

[94] Natalya Fridman Noy and Mark A. Musen. PROMPT : Algorithm and Tool for Automated Ontology Merging and Alignment. In *AAAI/IAAI*. AAAI Press, 2000.

[95] Andrea Giovanni Nuzzolese, Aldo Gangemi, Valentina Presutti, and Paolo Ciancarini. Type inference through the analysis of wikipedia links. In *LDOW*, 2012.

[96] AndreaGiovanni Nuzzolese, Aldo Gangemi, Valentina Presutti, and Paolo Ciancarini. Encyclopedic knowledge patterns from wikipedia links. In *ISWC*. 2011.

[97] George Papadakis, Ekaterini Ioannou, Claudia Niederée, and Peter Fankhauser. Efficient entity resolution for large heterogeneous information spaces. In *WSDM*, 2011.

[98] Jay Pujara, Hui Miao, Lise Getoor, and William Cohen. Knowledge graph identification. In *ISWC*, 2013.

[99] R. Rastogi and K. Shim. Mining optimized association rules with categorical and numeric attributes. *IEEE Trans. on Knowl. and Data Eng.*, 14(1) :29–50, January 2002.

[100] L. Ratinov, D. Roth, D. Downey, and M. Anderson. Local and global algorithms for disambiguation to wikipedia. In *NAACL*, 2011.

[101] S. Razniewski, F. Korn, W. Nutt, and D. Srivastava. Identifying the extent of completeness of query answers over partially complete databases. In *SIGMOD*, 2015.

[102] S. Razniewski, F. M. Suchanek, and W. Nutt. But what do we actually know ? *AKBC*, 2016.

[103] W3C Recommendation. RDF Schema 1.1, 2014.

[104] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62(1-2), 2006.

[105] Alan Ritter and Oren Etzioni. A latent dirichlet allocation method for selectional preferences. In *ACL*, 2010.

[106] Jacobo Rouces, Gerard de Melo, and Katja Hose. Complex schema mapping and linking data : Beyond binary predicates. In *Proceedings of the WWW 2016 Workshop on Linked Data on the Web (LDOW 2016)*, 2016.

[107] Ansaf Salleb-aouissi, Christel Vrain, and Cyril Nortet. Quantminer : A genetic algorithm for mining quantitative association rules. In *IJCAI*, pages 1035–1040, 2007.

[108] Michael Schmitz, Robert Bart, Stephen Soderland, Oren Etzioni, et al. Open language learning for information extraction. In *EMNLP-CoNLL*, 2012.

[109] Stefan Schoenmackers, Oren Etzioni, Daniel S. Weld, and Jesse Davis. Learning first-order Horn clauses from web text. In *EMNLP*, 2010.

[110] Len Seligman, Peter Mork, Alon Y. Halevy, Kenneth P. Smith, Michael J. Carey, Kuang Chen, Chris Wolf, Jayant Madhavan, Akshay Kannan, and Doug Burdick. Openii : an open source information integration toolkit. In *SIGMOD*, 2010.

[111] Pavel Shvaiko and Jérôme Euzenat. *A Survey of Schema-Based Matching Approaches*, pages 146–171. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[112] Ajit P. Singh and Geoffrey J. Gordon. Relational learning via collective matrix factorization. In *KDD*, 2008.

[113] Ashwin Srinivasan. The aleph manual, 2001.

[114] F. M. Suchanek, D. Gross-Amblard, and S. Abiteboul. Watermarking for Ontologies. In *ISWC*, 2011.

[115] Fabian Suchanek and Nicoleta Preda. Semantic culturomics (vision paper). In *VLDB*, 2014.

[116] Fabian M. Suchanek, Serge Abiteboul, and Pierre Senellart. PARIS : Probabilistic Alignment of Relations, Instances, and Schema. *PVLDB*, 5(3), 2011.

[117] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO : A core of semantic knowledge. In *WWW*, 2007.

[118] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago : A Core of Semantic Knowledge. In *WWW*, 2007.

[119] Christos Tatsiopoulos and Basilis Boutsinas. Ontology mapping based on association rule mining. In *ICEIS (3)*, 2009.

[120] Metaweb Technologies. The freebase project. `http://freebase.com`.

[121] Octavian Udrea, Lise Getoor, and Renée J. Miller. Leveraging data and structure in ontology integration. In *SIGMOD*, 2007.

[122] Ryuhei Uehara. The number of connected components in graphs and its applications, 1999.

[123] Johanna Völker and Mathias Niepert. Statistical schema induction. In *ESWC*, 2011.

[124] D. Vrandečić and M. Krötzsch. Wikidata : a free collaborative knowledgebase. *Communications of the ACM*, 2014.

[125] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, 2014.

[126] Michael Wick, Sameer Singh, and Andrew McCallum. A discriminative hierarchical model for fast coreference at large scale. In *ACL*, 2012.

[127] William E. Winkler. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In *Section on Survey Research*, 1990.

[128] Word Wide Web Consortium. RDF Primer (W3C Recommendation 2004-02-10). `http://www.w3.org/TR/rdf-primer/`, 2004.

[129] Fei Wu and Daniel S. Weld. Autonomously semantifying wikipedia. In *CIKM*, 2007.

[130] Fei Wu and Daniel S Weld. Open information extraction using wikipedia. In *ACL*, 2010.

[131] Alexander Yates and Oren Etzioni. Unsupervised methods for determining object and relation synonyms on the web. *J. Artif. Int. Res.*, 34(1), March 2009.

[132] A. Zaveri, A. Rula, A. Maurino, R. Pietrobon, J. Lehmann, and S. Auer. Quality assessment for linked data : A survey. *Semantic Web Journal*, 2015.

[133] Qiang Zeng, Jignesh Patel, and David Page. QuickFOIL : Scalable Inductive Logic Programming. In *VLDB*, 2014.

Appendices

Annexe A

Résumé en français

A.1 Introduction

Depuis la conception du Web Sémantique en 2001, les bases de connaissances sont devenues de plus en plus incontournables. Des initiatives telles que DBpedia [7], YAGO [117], NELL [18], Cyc [78], Wikidata [124], et Knowledge Vault [32] ont toutes pour objectif de construire et d'entretenir des collections volumineuses de données sur le monde réel. Ces données consistent en des ensembles de faits tels que “Londres est la capitale du Royaume-Uni”, ou encore “Tous les politiciens sont des personnes”. Ces faits sont normalement stockés dans des formats lisibles par des machines. De telles bases de connaissances trouvent donc naturellement de nombreuses applications, notamment dans la recherche d'information, le traitement de requêtes, ainsi que le raisonnement automatique. Par exemple, confrontés avec une requête sur le lieu de naissance de Barack Obama, tous les moteurs de recherche de nos jours sont capables de comprendre que la chaîne de caractères “Barack Obama” fait référence à une personne et que l'utilisateur pose une question sur un attribut de cette personne. Les bases de connaissances actuelles ont été construites á partir de méthodes automatiques et semi-automatiques d'extraction d'information. Ces méthodes varient de techniques d'extraction d'information (IE) aux techniques d'apprentissage automatique (Machine Learning) en passant par les mécanismes de crowd-sourcing.

Ces dernières années, les bases de connaissances sont devenues suffisamment volumineuses pour rendre possible l'extraction d'informations intelligibles. Il est ainsi devenu possible de découvrir des motifs intéressants et fréquents dans les données. Par exemple, nous pouvons trouver la règle ci-dessous :

$$\textit{livesIn}(x, z) \wedge \textit{marriedTo}(x, y) \Rightarrow \textit{livesIn}(y, z)$$

Cette règle dit que si deux personnes sont mariées, elles résident dans la même ville. De telles règles font parfois de fausses prédictions. C'est pourquoi on leur assigne normalement un score de confiance, qui mesure la taux de cas dans lesquels les règles se trompent. Dans notre exemple, la règle fait des erreurs chez les couples qui habitent dans des villes différentes.

L'extraction de règles peut donc servir plusieurs objectifs. D'abord, les règles peuvent être utilisées pour déduire de nouveaux faits, qui peuvent être ajoutés aux bases de connaissances. Deuxièmement, les règles peuvent aider à identifier des erreurs dans les données. Troisièmement, plusieurs méthodes de raisonnement automatique [88, 104] dépendent des règles, normalement fournies par des experts. Finalement, les règles nous permettent de mieux comprendre les données. Par exemple, on peut constater que le mariage est une relation symétrique, ou que les musiciens sont souvent influencés par d'autres musiciens, qui jouent des mêmes instruments.

Tout cela montre que l'extraction de règles dans des bases de connaissances porte une grande valeur pour des tâches telles que l'analyse, la prédiction et la maintenance de données.

A.2 Contribution

Cette thèse présente deux contributions principales. D'une part, elle présente une méthode pour extraire des règles “intéressantes” à partir d'une base de connaissances. Ces règles doivent (1) être supportées par les données (être statistiquement significatives), et (2) faire des conclusions correctes. D'autre part, cette thèse montre comme appliquer telles règles dans plusieurs tâches orientées données. Nous étudions ainsi des applications sur la prédiction de faits, l'intégration de données et la prédiction de complétude. La structure de la thèse est détaillée ci-dessous :

1. Préliminaires
2. Extraction de règles
3. Accélérer l'extraction de règles
4. Semantification de wikilinks
5. Alignement de schémas
6. Mise en forme canonique de bases de connaissances ouvertes
7. Prédiction de complétude
8. Extraction de règles numériques
9. Conclusion

A.3 Préliminaires

A.3.1 Bases de connaissances RDF

Une base de connaissances $\mathcal{K}$ est un ensemble de faits lisibles par une machine. Nous nous intéressons aux bases de connaissances RDF (Resource Description Framework) [128]. Dans ce format, un fait est un triplé $\langle$*sujet, relation, objet*$\rangle$, tel que $\langle$*UK, hasCapital, London*$\rangle$ ou $\langle$*Barack, type, Politician*$\rangle$. Les bases de connaissances RDF possèdent aussi un schéma qui définit les relations et la hiérarchie de classes. Par exemple, le triplé ⟨*Politicien, rdfs :subClassOf, Person*⟩ dit que tous les politiciens sont aussi des personnes. Dans cette thèse nous utilisons le format Datalog ; le triplé ⟨*x, r, y*⟩ devient $r(x, y)$. Nous écrivons $r(x, y)$ pour dire $r(x, y) \in \mathcal{K}$.

A.3.2 Hypothèses du monde clos et du monde ouvert

L'hypothèse du monde clos est une supposition selon laquelle tous les faits qui ne sont pas connus par une base de connaissances sont *faux* ; ainsi elle présume que la base de connaissances est complète par rapport au monde réel. En revanche l'hypothèse du monde ouvert ne fait pas cette supposition et suppose que les faits absents sont seulement *inconnus*. Les base de connaissances qui sont étudiées dans cette thèse utilisent l'hypothèse du monde ouvert.

A.3.3 Clauses de Horn (Règles)

Un *atome* est un fait qui peut contenir des variables dans le sujet ou l'objet. Quelques exemples sont $livesIn(x, USA)$ or $isCitizenOf(x, y)$. Une règle $\boldsymbol{B} \Rightarrow H$ est un clause de Horn. Elle consiste en un antécédent et une conséquence. L'antécédent est une conjonction logique de plusieurs atomes $\boldsymbol{B} := B_1 \wedge ... \wedge B_n$, tandis que la conséquence consiste en un seul atome H. Un exemple de règle est celle qui traduit le fait que des couples mariés habitent dans la même ville :

$$livesIn(x, y) \wedge isMarriedTo(x, z) \Rightarrow livesIn(z, y)$$

Une instanciation σ est une fonction qui associe à chaque variable une constante. L'application d'une instanciation σ à un atome B produit un nouvel atome $\sigma(B)$, dont les variables ont été remplacées par les constantes correspondantes dans l'instanciation σ. Cette définition est facilement extensible aux conjonctions d'atomes et aux règles de la forme $\boldsymbol{B} \Rightarrow H$. Nous disons qu'un fait $r(x, y)$ est une prédiction d'une règle $\boldsymbol{B} \Rightarrow H$ dans une base de connaissances $\mathcal{K}$, si $\exists\ \sigma : \forall B_i \in \boldsymbol{B} : \sigma(H) = r(x, y) \wedge \sigma(B_i) \in \mathcal{K}$. Prenons par exemple une base de connaissances contenant les faits *livesIn(Barack, Washington)* et $isMarriedTo(Barack, Michelle)$ et la règle ci-dessus. Dans ce cas, le fait $livesIn(Michelle, Washington)$ est une prédiction de la règle sous l'instanciation $\sigma := \{x \rightarrow Barack, y \rightarrow Washington, z \rightarrow Michelle\}$.

Deux atomes sont *connectés* dans une règle s'ils partagent un argument (variable ou constante). Une règle est *connectée* si tous ses atomes sont transitivement connectés les uns aux autres. Une variable est *fermée* si elle apparaît au moins dans deux atomes. Une règle est fermée si toutes ses variables sont fermées. Nous nous intéressons aux règles fermées, car elles nous permettent de faire des conclusions concrètes, plutôt que de prédire la simple existence de faits comme dans la règle $diedIn(x, y) \Rightarrow wasBornIn(x, z)$.

A.4 Extraction de règles

A.4.1 Introduction

Objectif. L'objectif de ce chapitre est d'extraire des règles d'association dans des bases de connaissances de grandes tailles et potentiellement incomplètes. Par ailleurs, nous nous intéressons aux règles qui prédisent correctement de nouvelles connaissances.

Défis. L'extraction de règles dans les bases de connaissances reste encore un défi pour deux raisons. En premier lieu, les techniques traditionnelles d'induction d'hypothèses [1] ont besoin de contre-exemples sous la forme de faits négatifs. Or les bases de connaissances RDF ne disposent que d'affirmations. Le modèle de données RDF ne permet pas de représenter, par exemple, le fait que Barack Obama n'est pas citoyen français. À cause de l'hypothèse du monde ouvert, les faits qui ne sont pas connus par la base de connaissances ne peuvent pas servir comme contre-évidence non plus.

En deuxième lieu, la taille des bases de connaissances courantes rend l'extraction de règles plus difficile. En 2016 la dernière version de YAGO [117] consiste en 120 millions de faits sur 10 millions d'entités. En outre le Knowledge Vault (la base de connaissances de Google) contient 1,6 milliards de faits avec plus de 4000 relations. Comme le problème d'induire une hypothèse à partir de données est infaisable (NP^{NP}-complète [47]), il est impossible de résoudre ce problème pour de tels volumes de données.

Nous présentons une solution à ces problèmes dans ce chapitre.

A.4.2 Mesures de Significativité

Tous les systèmes d'extraction de règles définissent des mesures de significativé pour les règles. Ces mesures ont pour objectif de quantifier l'évidence d'une règle dans les données. Une règle avec une faible évidence n'est pas utile pour faire des conclusions. Ci-dessous nous décrivons deux mesures de signifiance : le *support* et la *couverture*.

Support. Le support compte le nombre de prédictions de la règle, qui sont connues – et donc correctes – par la base de connaissances. Nous calculons le support d'une règle $\boldsymbol{B} \Rightarrow H$ avec la formule suivante :

$$supp(\boldsymbol{B} \Rightarrow H) := \#(vars(H)) : \exists z_1, ..., z_m : \boldsymbol{B} \wedge H$$

Dans cette formule $vars(H)$ représente l'ensemble de variables présentes dans la conséquence H et $\{z_1, ..., z_m\} = vars(\boldsymbol{B}) - vars(H)$. Notre définition de support est monotone, donc l'addition d'atomes à une règle ne peut jamais augmenter son support.

1. Étudiées dans le domaine de *Inductive Logic Programming* (ILP)

Couverture. La couverture mesure la proportion de prédictions correctes d'une règle par rapport à la taille de la relation dans la conséquence. La couverture est calculée à partir de la formule suivante :

$$hc(\boldsymbol{B} \Rightarrow H) := \frac{supp(\boldsymbol{B} \Rightarrow H)}{size(r)}$$

où $size(r) := \#(x', y') : r(x', y')$ dénote le nombre de faits de la relation r.

A.4.3 Mesures de confiance

Les mesures de significativé quantifient le nombre de prédictions vérifiables d'une règle dans la base de connaissances. Elles ne prennent pas en compte les prédictions fausses. Comme les bases de connaissances ne stockent pas de faits négatifs, toutes les mesures de confiance doivent présumer de la contre-évidence de certaines règles. Dans cette section, nous discutons de deux mesures de confiance : la confiance standard, et la confiance sous la présomption de complétude partielle.

Confiance Standard. La confiance standard est la proportion de prédictions d'une règle, qui sont dans la base de connaissances. Elle est calculée à partir de la formule suivante :

$$conf(\boldsymbol{B} \Rightarrow H) := \frac{supp(\boldsymbol{B} \Rightarrow H)}{\#(vars(H)) : \exists z_1, ..., z_m : \boldsymbol{B}} \tag{A.1}$$

La confiance standard est la mesure transitionnelle utilisée par les systèmes d'extraction de règles d'association. Elle s'appuie sur l'hypothèse du monde clos, donc elle n'est pas adéquate pour les bases de connaissances que nous étudions dans cette thèse. Nous proposons une mesure plus appropriée pour des scénarios qui opèrent sous l'hypothèse du monde ouvert.

A.4.4 La Présomption de Complétude Partielle

Nous proposons de tenir pour acquis des contre-exemples sous la Présomption de Complétude Partielle ou PCA[2]. La PCA est la supposition que

$$r(x, y) \in \mathcal{K} \Rightarrow \forall\ y' : r(x, y') \notin \mathcal{K} : r(x, y') \text{ is false}$$

C'est-à-dire, si la base de connaissances connaît au moins un objet y pour une entité x et une relation r, elle connaît tous les objets y qui sont vrais dans le monde réel. Cette présomption implique des contre-exemples de façon moins restrictive que ne le permet la confiance standard. Si la base de connaissances ne contient aucun lieu de naissance pour une personne, la confiance standard assume que la personne n'a pas un lieu de naissance. Par contre, la PCA ne fait aucune supposition dans ce cas, donc

2. Abréviation de *Partial Completeness Assumption*

elle accepte que certains faits puissent être inconnus. Cette notion nous permet de définir une nouvelle mesure de confiance, que nous appelons la *confiance PCA* :

$$conf_{pca}(\boldsymbol{B} \Rightarrow r(x,y)) := \frac{supp(\boldsymbol{B} \Rightarrow H)}{\#(vars(H)) : \exists z_1, ..., z_m, y' : \boldsymbol{B} \wedge r(x,y')} \tag{A.2}$$

A.4.5 AMIE

Dans cette section nous décrivons AMIE, notre système d'extraction de règles pour les bases de connaissances.

Algorithme. L'algorithme 6 décrit notre approche d'extraction de règles. L'algorithme prend en entrée une base de connaissances $\mathcal{K}$, un seuil maximal l pour le nombre d'atomes, un seuil minimal de couverture et un seuil minimal de confiance. Nous détaillons la sélection des valeurs pour ces arguments plus tard dans cette section.

L'algorithme commence avec l'initialisation d'une file d'attente (ligne 1) avec toutes les règles de taille 1 – de la forme $\Rightarrow r(x,y)$, avec un antécédent vide. Ensuite l'algorithme opère de façon itérative en retirant une règle à chaque itération. Si la règle remplit certains critères (ligne 5), elle est délivrée. Si la règle n'excède pas la longueur maximale l (ligne 7), elle est spécialisée. La spécialisation produit de nouvelles règles dérivées à partir d'une règle mère. Si une règle dérivée n'est pas le doublon d'une autre règle et qu'elle dépasse le seuil de couverture donné (ligne 10), elle est ajoutée dans la file. L'algorithme termine lorsque la file d'attente est vide.

Algorithm 6: AMIE

```
Input: a KB : $\mathcal{K}$, longueur maximale : $l$, seuil de couverture : $minHC$, seuil de
       confiance : $minConf$
Output: ensemble de clause de Horn : $rules$
1  $q = [r_1(x,y), r_2(x,y) \dots r_m(x,y)]$
2  $rules = \langle\rangle$
3  while $\neg q$.isEmpty() do
4      $r = q$.dequeue()
5      if $AcceptedForOutput(r, out, minConf)$ then
6          $rules$.add($r$)
7      if $length(r) < l$ then
8          $R(r) = Refine(r)$
9          for each rule $r_c \in R(r)$ do
10             if $hc(r_c) \geq minHC \wedge r_c \notin q$ then
11                 $q$.enqueue($r_c$)
12 return $rules$
```

Spécialisation. AMIE explore l'espace de recherche en dérivant de nouvelles règles à partir de règles plus simples. Cela est accompli à travers l'application d'un ensemble d'opérateurs aux règles. Ces opérateurs sont décrits ci-dessous :

1. **Add Dangling Atom (**$\mathcal{O}_D$) produit de nouvelles règles en ajoutant un nouvel atome à l'antécédent de la règle prise en entrée. Le nouvel atome partage une variable avec la règle et introduit une nouvelle variable (la variable "dangling").
2. **Add Instantiated Atom (**$\mathcal{O}_I$) ajoute un nouvel atome instancié à l'antécédent de la règle prise en entrée. L'atome partage une variable avec la règle et a une valeur constante pour l'autre argument.
3. **Add Closing Atom (**$\mathcal{O}_C$) ferme une règle en ajoutant un atome dont les variables existent dans la règle.

À travers l'application itérative de cet ensemble d'opérateurs, AMIE explore l'espace de recherche et rapporte toutes les règles fermées en ayant au plus l atomes qui excèdent le seuil de support et confiance.

Critères de sortie. La ligne 5 de l'algorithme 6 décide si une règle est délivrée dans la sortie. Les critères pour accepter une règle incluent :

1. Être fermée (Section A.3.3)
2. Excéder le seuil de confiance $minConf$.
3. Avoir une confiance supérieure à celle de toutes ses mères. Les mères sont toutes les règles déjà délivrées, telles que l'application d'un des opérateurs produit la règle originelle. La raison de cette condition est d'éviter des règles redondantes.

Paramètres. L'algorithme 6 est infaisable pour des bases de connaissances volumineuses à cause de la taille de l'espace de recherche. Par exemple l'opérateur $\mathcal{O}_I$ produit un nombre d'atomes de l'ordre de $|\mathcal{R}| \times |\mathcal{E} \cup \mathcal{L}|$. Pour cette raison, AMIE définit des paramètres qui réduisent l'espace de recherche, à savoir le seuil de couverture $minHC$ et la longueur maximale l. Par défaut AMIE désactive l'opérateur $\mathcal{O}_I$ et utilise un seuil de couverture de 1% ($minHC = 0.01$) et une longueur maximale de 3 atomes ($l = 3$). Ceci étant dit l'utilisateur peut changer ces paramètres comme il l'entend.

Autres détails d'implémentation. AMIE s'appuie sur une base de données en mémoire pour stocker la base de connaissances. Cette base de données est optimisée pour les types de requêtes requises par les opérateurs susmentionnés et aussi pour le calcul de la confiance de règles. En outre, AMIE exécute Algorithme 6 en parallèle, donc l'accès à la file d'attente est synchronisé de telle sorte que plusieurs règles puissent être traitées en même temps.

Constantes	WARMR	AMIE
non	18h	6.02s
oui	(48h)	1.43min

TABLE A.1 – Temps d'exécution d'AMIE et WARMR sur une échantillon de YAGO2. Pour AMIE "Constantes" implique l'activation du opérateur d'instanciation.

A.4.6 Expérimentation

Dans cette section nous évaluons notre système AMIE par rapport à son performance d'exécution et la qualité de la sortie. Pour l'évaluation du temps d'exécution nous avons comparé AMIE avec deux systèmes d'extraction de règles dans l'état de l'art, à savoir WARMR [28,29] et ALEPH [113]. Pour évaluer la qualité de la sortie, nous avons mesuré la pertinence de la confiance PCA en identifiant des règles de bonne qualité. La qualité d'une règle est définie par la précision de ses prédictions. Nous avons comparé la confiance PCA avec la confiance standard et la fonction "Positives-only" proposé par les auteurs d'ALEPH.

Contexte expérimental. Nos expériences ont été effectuées dans un serveur avec 48GB RAM, 8 CPUs physiques (Intel Xeon at 2.4GHz, 32 fils d'exécution) et en utilisant Fedora 21 comme système d'exploitation. Sauf indication contraire, nous avons testé AMIE avec sa configuration par défaut.

Données. Pour la plupart de nos expériences nous avons testé AMIE sur un jeu de données construit à partir de YAGO2, consistant de 948 mille faits sur 470 mille entités et avec 32 relations. Comme nos concurrents ne supportent pas tel volume de données, nous avons construit un échantillon de YAGO, qui contient 47 mille faits sur 14 mille entités. Nous avons aussi testé AMIE sur une partie de DBpedia ayant 13,7 millions de faits, 1,4M relations entités et 1595 relations.

AMIE vs. WARMR. WARMR [28,29] est un système d'extraction de règles implémenté en Prolog. Contrairement à AMIE, WARMR opère sous l'hypothèse du monde clos, donc il utilise la confiance standard pour évaluer la qualité des règles. De plus, WARMR exige à l'utilisateur de mettre un "language bias", qui consiste dans un ensemble de configurations qui détermine le types de règles a rapporter. En revanche, AMIE n'a pas besoin d'un tel paramètre.

La table A.1 montre les résultats de la comparaison du temps d'exécution entre AMIE et WARMR sur notre échantillon de YAGO2. Pour obtenir une comparaison juste, nous avons adapté AMIE de sorte qu'elle simule la notion de support implémentée par WARMR. Nous avons utilisé un seuil de support de 5 entités pour les deux systèmes. Nos résultats montre qu'AMIE est de trois ordres de grandeur plus rapides que WARMR. Ceci est possible grâce à notre implémentation en parallèle spécialement

Jeu de données	ALEPH	AMIE
YAGO2	4.96s à $>$ 1 jour	4.41min
YAGO2 (échantillon)	0.05s à $>$ 1 jour	5.65s

TABLE A.2 – Temps d'exécution ALEPH vs. AMIE

adaptée aux bases de connaissances RDF et les types de requêtes requises par notre algorithme. Au niveau de la sortie, WARMR a trouvé 41 règles tandis que AMIE à rapporté 75 règles, qui incluent ceux rapportée par WARMR.

AMIE vs. ALEPH. ALEPH est un système ILP qui offre un ensemble de mesures de confiance pour évaluer la qualité des règles. Dans nos expériences nous avons utilisé la fonction "Positives-only" [75, 87] car elle n'a pas besoin de contre-évidence de façon explicite. Cette fonction assume des contre-exemples aléatoirement, donc elle favorise les règles qui concluent des faits connus par la base de connaissance et qui ne concluent aucun des contre-exemples. Cette fonction prend aussi en compte la longueur des règles, donc elle préfère des clauses de Horn avec moins d'atomes.

Pour comparer AMIE et ALEPH, nous avons testé les deux systèmes sur YAGO2 et l'échantillon qui nous a utilisé pour WARMR avec un seuil de support de 2 entités. Pour ALEPH nous avons limité le nombre de contre-exemples générés aléatoirement à 50. Tout comme WARMR, ALEPH ne peut pas être directement utilisé car il requit d'un "language bias". En outre, ALEPH prend en entrée la relation objectif r pour la conséquence des règles, donc l'utilisateur doit démarrer le système pour chaque relation qui lui intéresse. La table A.2 illustre nos constatations. Comme ALEPH traite une seul relation objective par exécution, nous montrons le meilleur et le pire temps d'exécution. Pour quelques relations (e.g., *isPoliticianOf*), ALEPH a finit très vite, alors que pour des autres relations le système n'était pas capable de finir pendant un jour d'exécution - après lequel nous l'avons interrompu. En revanche AMIE a pris moins de 5 minutes sur YAGO2 et quelques secondes sur notre échantillon.

AMIE pour l'inférence de faits. Pour évaluer la confiance PCA comme mesure de qualité, nous avons testé AMIE en utilisant la confiance PCA et la confiance standard comme mesures de classement pour les règles extraites à partir de YAGO2. De la même façon, nous avons comparé la confiance PCA de AMIE, avec celle d'ALEPH dans notre échantillon de YAGO2. Tous les deux comparaisons s'appuient sur le même protocole expérimental : (1) nous avons extrait des règles, (2) nous avons classé les règles par la mesure de confiance correspondante, (3) nous avons fait des prédictions au-delà de la base de connaissances et (4) nous avons mesuré la précision de ces prédictions en les vérifiant soit dans le web soit dans YAGO2s[3]. La précision agrégée par rapport au nombre de prédictions est montré respectivement pour nos deux cas

3. YAGO2s est la version suivante à YAGO2.

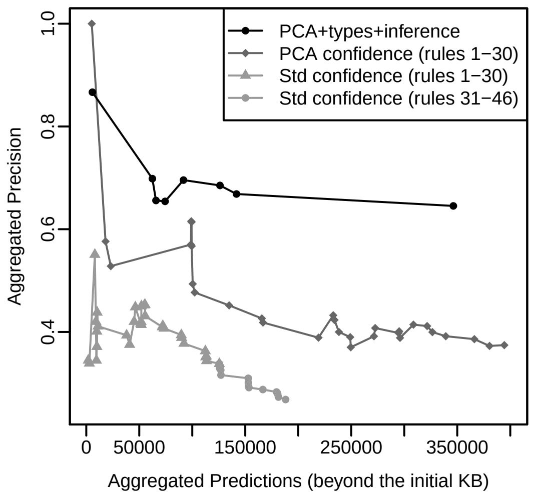


FIGURE A.1 – Confiance standard vs. confiance PCA sur YAGO2

	Top n	Prédictions	Exactitude
Positives-only	7	2997	27%
Confiance PCA	12	2629	62%
Positives-only	9	5031	26%
Confiance PCA	22	4512	46%
Positives-only	17	8457	30%
Confiance PCA	23	13927	43%

TABLE A.3 – Confiance PCA vs. fonction positives-only : précision agrégé pour les règles extraites dans un échantillon de YAGO2.

d'étude dans le graphique A.1 et la table A.3.

Nous constatons qu'AMIE avec la confiance PCA est capable d'identifier de règles prédictives de bonne qualité, de un façon plus efficace que celle de la confiance standard (troisième courbe de haut en bas dans le graphique A.1). Par ailleurs, en utilisant les classes des arguments des prédictions, en combinaison avec un mécanisme simple de prédiction-jointe (courbe en haut dans le graphique A.1), il est possible d'améliorer encore la qualité des prédictions. D'une part les classes de la base de connaissances sont utilisées pour filtrer de fallacieuses prédictions. D'autre part, la prédiction-jointe assigne un score agrégé aux prédictions. Ce score dépend de la confiance de toutes les règles qui font telle prédiction. Il aussi permet de classer les prédictions. Toutes ces considérations nous permettent de prédire 350 mille faits avec une précision de 70%. Nous pouvons aussi constater que la confiance PCA surpasse la fonction "Positives-only" en repérant des règles de nature prédictive.

AMIE avec des configurations différentes. Comme preuve de concept, nous avons

Jeu de données	Temps d'exécution	# de règles
YAGO2	3.62min	138
YAGO2 ($l = 4$)	27.14min	645
YAGO2 const	17.76min	18886
DBpedia 2.0 ($l = 2$)	2.89min	6963

TABLE A.4 – AMIE avec des configurations différentes

testé AMIE avec (1) sa configuration par défaut, (2) l'activation de l'opérateur d'instanciation ($\mathcal{O}_I$) et (2) avec une longueur maximal de 4 ($l = 4$) sur YAGO2 and de 2 ($l = 2$) sur DBpedia 2.0. Les résultats sont présentés dans la table A.4. Dans la table A.5, nous montrons quelques règles rapportés par AMIE dans YAGO et DBpedia.

y :hasAdvisor$(x, y) \wedge$ *y :graduatedFrom*$(x, z) \Rightarrow$ *y :worksAt*(y, z)
y :wasBornIn$(x, y) \wedge$ *y :isLocatedIn*$(y, z) \Rightarrow$ *y :isCitizenOf*(x, z)
y :hasWonPrize$(x, Grammy) \Rightarrow$ *y :hasMusicalRole*$(x, Guitar)$
d :capital$(x, y) \Rightarrow$ *d :largestCity*(x, y)

TABLE A.5 – Quelques règles trouvées par AMIE dans YAGO (y :) et DBpedia (d :)

A.4.7 Conclusion

Dans ce chapitre, nous avons présenté AMIE, une approche pour la fouille de règles d'association (clauses de Horn) dans des bases de connaissances. AMIE s'appuie sur un model d'extraction de règles adaptée pour des bases de connaissances qui opèrent sous l'hypothèse de monde ouvert. Contrairement au systèmes traditionnels d'extraction de règles, AMIE peut être utilisé "out-of-the-box" sans que l'utilisateur doive fournir des configurations ou ajuster les paramètres. AMIE surclasse les autres systèmes de fouille de règles dans l'état de l'art, tant en performance qu'en la qualité des règles rapportées.

A.5 Accélérer l'extraction de règles

A.5.1 Introduction

La section expérimentale présentée dans le dernier chapitre a montré qu'AMIE est capable de trouver des règles d'association dans une base de connaissances d'un million de faits. Dans ce chapitre nous présentons AMIE+, une extension d'AMIE qui implémente un ensemble d'optimisations visant à améliorer la performance du système pour le permettre de traiter de plus grands volumes de données. Nos extensions visent à accélérer deux phases du algorithme 6, à savoir la spécialisation et le calcul de la confiance des règles. Nous décrivons nos optimisations ci-dessous.

A.5.2 Accélérer la phase de spécialisation

Dans cette section nous décrivons les stratégies implémentées par AMIE+ pour accélérer la spécialisation de règles.

Longueur maximal. La longueur maximal l est un paramètre de notre système. La phase de spécialisation consiste en l'application de trois opérateurs qui produisent des nouvelles règles dérivées à partir d'une règle mère. AMIE n'applique pas les opérateurs décrits dans la section A.4.5 aux règles de l atomes. Nous observons cependant que cette stratégie peut en principe produire de règles qui sont inutiles. Par exemple, l'application de l'opérateur O_D à une règle de $l - 1$ atomes produit des règles de longueur l qui ne sont pas fermées et donc ni spécialisées ni rapportées. Pour cette raison AMIE+ n'applique pas cet opérateur aux règles de taille $l - 1$. Un analyse similaire est applicable aux autres opérateurs.

Règles parfaites. Par définition, une règle ne peut pas avoir une confiance plus grande que 100%, donc pour telles règles – qui nous appelons *règles parfaites* – l'addition de nouveaux atomes ne sert à rien. Il en est ainsi parce que des atomes additionnels ne peuvent que baisser le support des règles dérivées alors que la confiance ne peut pas augmenter. Cela veut dire que les spécialisations de règles parfaites sont toujours de pire qualité. Pour cette raison, AMIE+ ne spécialise pas ces règles.

Simplification de requêtes. Nos mesures de support et couverture sont monotones, donc l'addition de nouveaux atomes ne peut pas augmenter le support des règles dérivées. Bien que dans la plupart de cas le support baisse, il y a de cas où les règles dérivées par l'opérateur O_D ont le même support que leur règle mère. Cela arrive lorsque la nouvelle variable n'impose aucune contrainte additionnelle à la règle mère, donc tous les deux règles conduisent à une requête équivalente dans la base de connaissances. Cela implique que les règles dérivées dans ce cas peuvent être remplacées par leur règle mère pendant leur spécialisation. Cette stratégie accélère la spécialisation car elle produit des requêtes avec moins d'atomes qui s'exécutent plus rapidement.

A.5.3 Accélérer le calcul de la confiance

Un partie importante du temps du algorithme 6 est utilisé dans le calcul de la confiance des règles. Lorsque AMIE calcule la confiance d'une règle, elle connaît déjà son support, donc il ne reste que évaluer la valeur du dénominateur des équations A.1 et A.2. Ce calcul peut être coûteux au niveau de temps d'exécution si l'antécédent d'une règle a de nombreuses instanciations. Pour cette raison nous avons conçu deux méthodes pour atténuer ces problèmes. Nos méthodes sont applicables aux certains types de règles et visent à écarter des règles fallacieuses sans investir du temps dans le calcul de leur confiance. Nos méthodes sont applicables tant à la confiance standard que à la confiance PCA.

Limites supérieures de confiance. Pour des règles de la forme $r(x,z) \wedge r(y,z) \Rightarrow r_h(x,y)$, le dénominateur de l'équation de la confiance standard (équation A.1) est calculé selon la formule suivante : $d_{std} := \#(x,y) : \exists z : r(x,z) \wedge r(y,z)$. Comme tous les deux atomes ont la même relation, nous pouvons réécrire cette formule comme suit :

$$d_{std} \geq \#x : \exists z : r(x,z) \tag{A.3}$$

Cette expression peut être calculée en temps constant. Le même analyse est applicable au dénominateur de la formule de la confiance PCA en produisant l'expression ci-dessous :

$$d_{pca} \geq \#x : \exists\ z, y' : r(x,z) \wedge r_h(x,y') \tag{A.4}$$

Tous les deux inégalités A.3 et A.4 définissent une limite supérieure (facile à calculer) pour la confiance. AMIE+ se sert de ces limites pour écarter en avance des règles dont leur limite supérieure de confiance est moindre que le seuil donné au système.

Approximation de la confiance. Si les variables x, y dans la conséquence d'une règle $\boldsymbol{B} \Rightarrow r(x,y)$ sont connectés par une seule chaîne de variables existentiellement quantifiées $z_1, \ldots, z_{n-1}$, nous pouvons estimer la confiance de façon efficace. Ces règles sont de la forme :

$$r_1(x,z_1) \wedge r_2(z_1,z_2) \wedge \ldots \wedge r_n(z_{n-1},y) \Rightarrow r_h(x,y)$$

Un exemple est la règle qui marrie chaque metteur en scène avec tous les acteurs qui ont joué dans ses films : $directed(x,z) \wedge hasActor(z,y) \Rightarrow marriedTo(x,y)$. Les antécédents de ce type de règles ont de nombreux instanciations qui font le calcul de la confiance très coûteux. Grâce à l'expression suivante, nous pouvons estimer le dénominateur de la confiance PCA d'une règle de manière très vite sans encourir l'exécution de requêtes extrêmement coûteuses.

$$\widehat{d_{pca}}(R) := \frac{ov_{dd}(r_1, r_h)}{fun(r_1)} \times \prod_{i=2}^{n} \frac{ov_{rd}(r_{i-1}, r_i)}{|rng(r_{i-1})|} \frac{ifun(r_i)}{fun(r_i)} \tag{A.5}$$

La formule A.5 utilise de valeurs de fonctionalités pour des relations (fun et $ifun$, voir [116]), ainsi que de statistiques sur la taille des domaines et des images des relations (rng) et sur le nombre d'entités en commun entre les arguments des relations ov. La formule A.5 tend à sous-estimer la valeur de d_{pca}, donc la confiance est normalement sur-estimée. De cette façon AMIE ne écarte pas de règles incorrectement.

A.5.4 Expérimentation

Le but de cette évaluation expérimentale est de montrer les bénéfices menés par les optimisations d'AMIE+. Nous comparons donc le temps d'exécution d'AMIE et AMIE+.

Contexte expérimental. Nos expériences pour ce chapitre utilisent le même contexte

Jeu de données	AMIE	AMIE+					
		Seulement Spécialisation	Seulement Sortie	Sortie +			Tout
				MRL	QRW	PR	
YAGO2	3.17min	29.37s	2.82min	29.03s	38.16s	2.80min	28.19s
YAGO2 (const)	37.57min	11.72min	37.05min	8.90min	12.04min	36.48min	9.93min
YAGO2 ($l = 4$)	27.14min	9.49min	26.48min	8.65min	15.69min	24.20min	8.35min
YAGO2s	$>$ 1 jour	$>$ 1 jour	$>$ 1 jour	1h 7min	1h 12min	$>$ 1 jour	59.38min
DBpedia 2.0	$>$ 1 jour	$>$ 1 jour	$>$ 1 jour	45.11min	46.37min	$>$ 1 jour	46.88min
DBpedia 3.8	$>$ 1 jour	$>$ 1 jour	11h 46min	8h 35min	7h 33min	10h 11min	7h 6min
Wikidata	$>$ 1 jour	$>$ 1 jour	$>$ 1 jour	1h 14min	7h 56min	$>$ 1 jour	25.50min

TABLE A.6 – Comparaison du temps d'exécution entre AMIE et AMIE+ dans des plusieurs jeus de données. YAGO2 (const) : l'operateur d'instanciation est activé.

expérimental de celles du chapitre A.4. Nous avons aussi testé les systèmes sur DBpedia 3.8 (11 millions de faits, plus de 600 relations) et Wikidata[4] (8,4 millions de faits, plus de 400 relations)

Résultats. La table A.6 montre une comparaison du temps d'éxecution d'AMIE et AMIE+. Pour AMIE+ la table détaille l'effet individuel de chacune des optimisations, à savoir l'accéleration de la spécialisation (Seulement spécialisation) et l'accélération du calcul de confiance (Sortie+). Pour la première catégorie nous avons aussi détaillé l'effet isolé de chaque stratégie : MLR (longueur maximal), QRW (simplification de requêtes), PR (règles parfaites). Nous avons aussi constaté le bénéfice de chacune de ces stratégies en combinaison avec l'optimisation du calcul de la confiance. Sous la catégorie "Tout" nous montrons l'effet conjoint de toutes nos stratégies.

Sauf pour YAGO2, AMIE n'est pas capable de finir pendant un jour d'exécution. Nous remarquons aussi que la seule activation des optimisations pour la spécialisation comporte un bénéfice important, surtout sur YAGO2. Par contre, dans certains cas (par exemple DBpedia 3.8), il faut activer toutes nos optimisations pour que AMIE+ finisse.

A.5.5 Conclusion

Dans ce chapitre nous avons présenté AMIE+, une extension de AMIE qui inclut un ensemble d'optimisations. Grâce à ces optimisations, AMIE est capable d'extraire de règles d'association dans de bases de connaissances de jusqu'à 11 millions de faits.

A.6 Semantification de wikilinks

A.6.1 Introduction

Les bases de connaissances construites à partir de données de Wikipedia telles que DBpedia [7] ou YAGO [117], stockent les hyperliens entre les entités de Wikipedia. Par exemple, si la page Wikipedia de Barack Obama a un lien à la page du prix Nobel,

4. Nous avons testé AMIE et AMIE+ sur un vidage de Wikidata de décembre 2014.

YAGO et DBpedia représentent cela comme un fait de la forme $linksTo$(*BarackObama*, *NobelPeacePrize*). Nous appelons ces faits *wikilinks*. Bien que les wikilinks représentent une partie importante du contenu des bases de connaissances, ils sont rarement utilisés, même si un hyperlien entre deux entités suggère l'existence d'une relation. Dans ce chapitre nous présentons une méthode pour prédire telle relation. Nous appelons cette tâche “sémantification de wikilinks” et nous observons qu'elle est une forme particulière d'inférence de faits.

A.6.2 Méthode

Notre approche de sémantification de wikilinks s'appuie sur les observations suivantes : (a) Parfois, des wikilinks portent une relation implicite entre deux entités, (b) quelques wikilinks sont déjà sémantifiés dans les bases de connaissances, (c) les classes des entités d'un wikilink définissent la signature la relation, (d) les wikilinks sémantifiés peuvent servir à la sémantification des autres. Un wikilink est sémantifié si la base de connaissances contient un autre fait qui connecte les arguments du wikilink.

Nous avons donc évalué notre intuition comme suit. Nous avons construit un jeu de données à partir de DBpedia 3.8 avec 18 millions de faits[5] en incluant 4 millions de wikilinks. Nous profitons des wikilinks semantifiés en extrayant des règles de la forme :

$$linksTo^{*}(x, y) \wedge \boldsymbol{B} \wedge type(x, C) \wedge type(y, C') \Rightarrow r(x, y)$$

Ici, *linksTo* est un alias pour *wikiPageWikiLink* (la relation DBpedia), *linksTo** désigne soit *linksTo* soit *linksTo*$^{-1}$, “*type*” est un synonyme pour *rdf :type*, et $\boldsymbol{B}$ est une conjonction logique de au plus 2 atomes. Nous avons adapté AMIE pour extraire des règles de ce type. Avec un seuil de support de 100 et un seuil de confiance de 0.2, AMIE a trouvé 3546 règles dans notre jeu de données. Un exemple est la règle $linksTo(x, y) \wedge is(x, Town) \wedge is(y, Country) \Rightarrow country(x, y)$.

Une fois ayant trouvé des règles d'association, nous les avons utilisées pour faire de prédictions de la forme $r(a, b)$ selon la méthodologie présentée dans la section A.5.4. Cela a produit 180 mille prédictions. Pour certains wikilinks non-semantifiés $linksTo(a, b)$, notre méthode propose plusieurs prédictions $r(a, b)$. Ces prédictions possèdent un score de confiance. Il s'ensuit que nous pouvons classer ces prédictions et proposer les relations r comme candidats pour la sémantification des entités a, b. Nous avons évalué la précision de ce classement pour les candidats dans le top-1 et top-3 de la liste. Nos résultats sont présentés dans la table A.7.

A.6.3 Conclusion

Nous avons montré comme l'extraction des règles peuvent servir à la sémantification de wikilinks, une forme particulière d'inférence de données. Avec ce travail de recherche nous visons à tourner l'attention aux wikilinks car ils portent des informations précieuses qui peuvent être utilisées pour alimenter les bases de connaissances avec des nouveaux faits.

5. 4,2 millions de faits et 1,7 millions d'entités entre des personnes, des lieux et des organisations

Confiance@1	**Confiance@3**
0.77 ± 0.10	0.67 ± 0.07

TABLE A.7 – Scores MAP@1 et MAP@3 moyen pour la sémantification de wikilinks sur DBpedia.

A.7 Alignement de schémas

A.7.1 Introduction

Les bases de connaissances généralistes telles que DBpedia [7], YAGO [117] ou Wikidata possèdent beaucoup de faits en commun. Il en est ainsi parce qu'elles ont été construites à partir d'une source commune, à savoir Wikipedia. Un travail considérable a été accompli en intégrant les bases de connaissances au niveau d'instances. L'initiative "Linked Open Data" (LOD) offre des liens entre les instances de différentes bases de connaissances sous la forme de faits *sameAs* permettant de connecter des milliards de faits fournis par des sources indépendantes.

Néanmoins, l'alignement des instances ne suffit pas pour une véritable intégration de données dans l'esprit du web sémantique. Une requête lancée sur une base de connaissances $\mathcal{K}$ n'aura pas de réponse dans une autre base de connaissances $\mathcal{K}'$ à moins qu'il n'y ait aussi un alignement au niveau des relations et des classes (schéma). Nous observons aussi que la solution consistant à faire l'alignement manuellement est inefficace pour les milles de base de connaissances qui composent le LOD. Nous proposons donc une méthode automatique pour aligner les schémas de deux bases de connaissances pour lesquelles il existe un alignement au niveau d'instances.

A.7.2 Méthode

Notre approche pour l'alignement de schémas de bases de connaissances suppose qu'on dispose d'un alignement entre les instances. Cet alignement est utilisé pour fusionner les bases de connaissances à traiter. Cette fusion produit une nouvelle base de connaissances dans laquelle toutes les relations sont préservées. Par contre les identifiants d'une des bases de connaissances sont remplacés par leurs correspondants dans l'autre base de connaissance.

Sur la base de connaissances fusionnée nous appliquons l'extraction de règles avec AMIE. Nous avons adapté AMIE pour extraire des règles telles que l'antécédent contient des relations d'une base de connaissances, tandis que la conséquence a une relation de l'autre base de connaissances. Nous nous intéressons à certains types de règles, que nous appelons *règles ROSA*[6]. La table A.2 montre les types d'alignements pris en charge par les règles ROSA.

Expérimentation. Les règles ROSA expriment certains alignements intéressants entre

6. ROSA est l'acronyme de Rule for Ontology Schema Alignment

$$
\begin{aligned}
r(x,y) &\Rightarrow r'(x,y) && \text{(R-subsumption)}\\
r(x,y) &\Leftrightarrow r'(x,y) && \text{(R-equivalence)}\\
type(x,C) &\Rightarrow type'(x,C') && \text{(C-subsumption)}\\
r_1(x,y) \wedge r_2(y,z) &\Rightarrow r'(x,z) && \text{(2-Hops alignment)}\\
r(z,x) \wedge r(z,y) &\Rightarrow r'(x,y) && \text{(Triangle alignment)}\\
r_1(x,y) \wedge r_2(x,V) &\Rightarrow r'(x,y) && \text{(Specific R-subsumption)}\\
r(y,V) &\Rightarrow r'(x,V') && \text{(Attr-Value translation)}\\
r_1(x,V_1) \wedge r_2(x,V_2) &\Rightarrow r'(x,V') && \text{(2-Value translation)}
\end{aligned}
$$

FIGURE A.2 – Règles ROSA, $r, r_1, r_2 \in \mathcal{R}_1, r' \in \mathcal{R}_2$.

Type	Example	Confiance
R-subsumption	*D :musicalArtist*$(x,y) \Rightarrow$ *Y :created*(y,x)	90%
R-equivalence	*Y :isCitizenOf*$(x,y) \Leftrightarrow$ *D :nationality*(x,y)	96%
C-subsumption	*Y :type*$(x, Y : Site) \Rightarrow$ *D :type*$(x, D : PopulatedPlace)$	97%
2-Hops alignment	*Y :wasBornIn*$(x,y) \wedge$ *Y :label*$(y,z) \Rightarrow$ *I :bornIn*(x,z)	37%
Triangle alignment	*Y :hasChild*$(y,z) \wedge$ *Y :hasChild*$(y,x) \Rightarrow$ *F :sibling*(x,z)	37%
Specific R-subsumption	*Y :bornIn*$(x,y) \wedge$ *Y :type*$(y, City) \Rightarrow$ *F :birthPlace*(x,y)	97%
Attr-Value translation	*Y :locatedIn*$(x,$ *Italy*$) \Rightarrow$ *D :timeZone*$(x,$CET$)$	100%
2-Value translation	*F :type*$(x, Royal) \wedge$ *F :gender*$(x, female) \Rightarrow$ *Y :type*$(y, Princess)$	55%

TABLE A.8 – Examples de règles ROSA. D : DBpedia, Y : YAGO, I : IMDB, F : Freebase

les schémas de deux bases de connaissances. On trouvera dans la table A.8 des exemples de chaque type de règles ROSA que nous avons trouvés en fusionnant YAGO2 avec DBpedia 3.8, un crawl de données de IMDB (utilisé par [116]) et Freebase. Nous avons utilisé la confiance PCA comme mesure de précision pour les alignements. Comme une *R-equivalence* ($r \Leftrightarrow r'$) n'est que la conjonction de deux règles *R-subsumption* ($r \Rightarrow r'$, $r' \Rightarrow r$), son score est le minimum des scores de ses règles composantes. Nous remarquons aussi que chez les alignements plus complexes comme le *2-Hops subsumption*, l'extraction de règles a fourni de nombreux résultats intéressants qui ne sont pas des alignements, mais plutôt des *règles soft*.

Conclusion. Dans ce chapitre, nous avons argumenté que l'intégration de données sémantiques requiert d'un alignement qu'il soit aussi bien au niveau des instances qu'au niveau des schémas. Nous avons contribué au deuxième problème en définissant les règles ROSA. Ces règles sont facile à extraire en ayant un alignement au niveau des instances. En outre, elles expriment des alignements complexes qui peuvent servir à l'intégration efficace de deux bases de connaissances.

A.8 Mise en forme canonique de bases de connaissances ouvertes

A.8.1 Introduction

Les techniques d'extraction ouverte d'information (open IE [10, 35]), tels que ReVerb [37], produisent des bases de connaissances dont les sujets et objets sont des groupes nominaux arbitraires et les relations des groupes verbaux arbitraires. Par exemple ces techniques pourraient extraire les faits ⟨*DC, is capital of, United States*⟩ and ⟨*Washington, capital city of, USA*⟩. Même si les techniques Open IE entraînent un meilleur rappel par rapport aux techniques IE standard, elles produisent des bases de connaissances que ne peuvent pas être directement utilisées. Il en est ainsi parce que les faits obtenus ne sont pas canoniques ; il faut connaître tous les synonymes d'une entité et d'une relation pour obtenir des réponses complètes à leur propos. Inversement, il n'y a aucune garantie qu'un groupe verbal fasse référence à l'entité à laquelle on s'intéresse.

Contribution. Dans ce chapitre nous proposons une approche qui prend une base de connaissances ouverte et la met en forme canonique. Cette canonisation regroupe tous les groupes nominaux qui font référence à la même entité. De la même façon notre méthode regroupe tous les groupes verbaux qui dénotent la même relation.

A.8.2 Canonisation de groupes nominaux

Mention. Une mention est un triplet $m = (n, u, \mathcal{A})$ où n est un groupe verbal comme *Barack Obama*, u est l'URL de la source de la mention (par exemple *bbc.com*), et $\mathcal{A}$ est un ensemble d'attributs sur n. Chaque attribut est un couple ⟨*relation, object*⟩, par exemple, ⟨*was born in, Honolulu*⟩.

Regroupement. Nous utilisons des techniques de regroupement hiérarchique (HAC en anglais) pour regrouper les mentions m dont leurs groupes verbaux n font référence à la même entité. HAC a besoin d'une mesure de similarité pour les mentions. Nous avons étudié les différentes mesures de similarité listées ci-dessous :

1. **Attributes overlap**. Intersection des attributs.
2. **String similarity**. Similarité des groupes nominaux en utilisant la fonction de Jaro-Winkler [127].
3. **String identity**. Identité des groupes nominaux.
4. **IDF tokens overlap**. Intersection des mots dans les groupes nominaux pondérée par IDF (inverse document frequency).
5. **Word overlap**. Intersection des mots dans les sources des mentions (u).
6. **Entity overlap**. Intersection des entités référencées dans les sources des mentions (u).

7. **Types overlap**. Intersection des classes (types) des groupes nominaux.
8. **Full ML**. Mesure combinée qui implémente un modèle de régression logistique à partir de toutes les mesures décrites ci-dessus.

A.8.3 Canonisation de groupes verbaux

Notre méthode de canonisation de groupes verbaux suppose que les sujets et objets de la base de connaissances ont été déjà canonisés. Sur cette base de connaissances demi-canonisée, nous appliquons l'extraction de règles avec AMIE pour trouver des relations de subsomption $r \sqsubset r'$ entre deux groupes verbaux. Ces relations correspondent aux règles de la forme $r(x,y) \Rightarrow r'(x,y)$. Nous construisons des relations d'équivalence $r \Leftrightarrow r'$ entre deux phrases verbales, en combinant les règles $r \Rightarrow r'$ et $r \Rightarrow r'$.

Regroupement. À partir d'un ensemble de règles d'équivalence, nous pouvons regrouper les groupes verbaux en fusionnant les relations d'équivalence qui partagent un groupe verbal (grâce à la transitivité de l'équivalence).

A.8.4 Experimentation

Contexte experimental. Nous avons testé la canonisation de groupes nominaux dans des jeux de données construits à partir de Reverb [37] et ClueWeb[7]. En appliquant Reverb sur Clueweb, nous avons obtenu une base de connaissances ouverte comportant 3 millions de triplets. Nous disposions aussi d'une canonisation pour la moité des triplets sous la forme d'un alignement entre les groupes nominaux et les entités sur Freebase [40]. Cette canonisation nous a servi comme "gold standard" pour l'évaluation de notre méthode de canonisation de groupes nominaux.

Canonisation de groupes nominaux. Dans la table A.9 nous présentons la performance de notre approche par rapport à plusieurs définitions de précision et rappel dans la littérature [76]. Nous avons appliqué notre méthode à un jeu de données de 8,5 milliers de triplets qui parlent de 150 entités différentes référencées au moins avec deux synonymes. La table montre la performance du regroupement en utilisant chacune des fonctions de similarité présentées dans la séction A.8.2.

Canonisation de groupes verbaux. Notre approche de canonisation de groupes verbaux suppose que les sujets et objets des triplets sont canoniques. Nous avons donc mis en forme canonique les sujets et objets de 600 mille triplets (avec 17 mille groupes verbaux) obtenus avec Reverb. Nous appelons ce jeu de données *Clustered KB*. Nous avons construit un autre jeu de données en utilisant le "gold standard" [40], dont nous nous sommes servis pour l'évaluation du regroupement des groupes nominaux. Ce jeu de données, que nous appelons *Linked KB*, contient 1,3 million de faits et 33 mille

7. `http://www.lemurproject.org/clueweb09.php/`

	Macro			Micro			Pairwise		
	Prec.	Rappel	F1	Prec.	Rappel	F1	Prec.	Rappel	F1
String identity	**1.000**	0.436	0.607	**1.000**	0.798	0.888	**1.000**	0.740	0.851
String similarity	0.995	0.658	0.792	0.998	0.844	0.914	0.999	0.768	**0.986**
IDF token overlap	0.994	0.879	0.933	0.996	0.969	0.982	0.999	**0.973**	**0.986**
Attribute overlap	**1.000**	0.05	0.102	**1.000**	0.232	0.377	**1.000**	0.094	0.173
Entity overlap	0.996	0.436	0.607	0.995	0.934	0.964	0.999	0.932	0.964
Type overlap	0.987	**0.926**	**0.956**	0.995	**0.973**	**0.984**	0.999	0.972	0.985
Word overlap	0.988	0.913	0.949	0.995	**0.973**	**0.984**	0.999	**0.973**	**0.986**
Full ML	0.994	0.906	0.948	**1.000**	0.937	0.967	**1.000**	**0.973**	0.869

TABLE A.9 – Précision et rappel de la canonisation de groupes nominaux sur un jeu de données construit à partir de Clueweb09 en utilisant Reverb.

				Précision				
	Conf.	Groupes	Clusters	Macro	Micro	Pairwise	Dans Freebase	Couverture
Linked KB	0.8	522	118	0.900	0.936	0.946	18%	15%
	0.5	967	143	0.896	0.690	0.337	25%	29%
Clustered KB	0.8	826	234	0.940	0.716	0.273	6%	16%
	0.5	1185	264	0.813	0.665	0.292	8%	33%

TABLE A.10 – Quality of relation clusters for two different confidence thresholds.

groupes verbaux. La table A.10 offre un résumé de nos constatations en utilisant des seuils de confiance de 0.5 et 0.8.

A.8.5 Conclusion

Dans ce chapitre nous avons montré que des techniques d'apprentissage automatique – aussi bien que des signaux simples comme l'intersection de mots pondérée par IDF – sont des outils efficaces pour la canonisation de groupes nominaux dans une base de connaissances ouverte construite à partir du web. Ce travail montre que l'extraction de règles est un moyen efficace pour le regroupement de groupes verbaux sémantiquement proches.

A.9 Prédiction de la complétude

A.9.1 Introduction

Les bases de connaissances actuelles souffrent de problèmes d'incomplétude. Par exemple, les auteurs de [83,114] ont constaté qu'entre 69% et 99% des entités dans les bases de connaissances les plus utilisées sont incomplètes, au sens qu'il leur manque au moins un attribut que d'autres entités dans la même classe possèdent. Bien que cette incomplétude soit un fait notoire, la proportion de faits manquants reste inconnue. L'incomplétude est pourtant un problème grave, pour les fournisseurs de données comme pour les utilisateurs. D'un côté, les fournisseurs ne peuvent pas savoir dans quelles parties de la base de connaissances concentrer leurs efforts d'extraction d'information. De l'autre coté, les utilisateurs ne disposent d'aucune garantie que leurs requêtes sur les données produiront tous les résultats qui sont vrais par rapport au monde réel.

Contribution. Dans ce chapitre, nous menons une étude sur la complétude de bases de connaissances, en analysant un ensemble de signaux qui permettent de prédire la complétude. En effet, prédire la complétude permet d'atténuer les problèmes mentionnés ci-dessus.

Défis. L'hypothèse du monde ouvert représente le principal défi pour la prédiction de complétude, car, pour beaucoup de relations, il est difficile de déterminer si un fait absent dans la base de connaissances est faux ou inconnu.

A.9.2 Complétude

Suivant les études existantes [85, 102], nous définissons la complétude à travers une base de connaissances hypothétique $\mathcal{K}^*$ qui contient tous les faits qui sont vrais dans le monde réel. Une base de connaissances $\mathcal{K}$ est complète par rapport à une requête q, si q produit les mêmes résultats dans $\mathcal{K}$ et dans $\mathcal{K}^*$. Nous nous intéressons dans ce chapitre aux requêtes qui demandent quels sont les objets qui correspondent à un certain couple sujet–relation.

Oracles de complétude. Un *oracle de complétude* est un formalisme qui associe une valeur de complétude à chaque couple sujet–relation dans une base de connaissances $\mathcal{K}$. Nous étudions un ensemble d'oracles de complétude listés ci-dessous :

- CWA (Closed World Assumption). L'hypothèse du monde clos ; tous les couples sujet–relation sont complets.
- PCA (Partial Completeness Assumption). Voir section A.4.4.
- Cardinality. Un couple sujet–relation est complet si la base de connaissances connaît au moins k objets qui lui correspondent.
- Popularity. Un couple sujet–relation est complet si le sujet est populaire selon une certaine mesure de popularité.
- No change. Un couple sujet–relation est complet si ses objets n'ont pas changé par rapport à une version antérieure de la base de connaissances.
- Star patterns. Un couple sujet–relation est complet lorsque la base de connaissances connaît certains autres attributs pour le sujet.
- Class information. Les entités de certaines classes spécifiques sont complètes par rapport à certaines relations.

— AMIE. Cet oracle est construit à partir des règles de complétude de la forme $\boldsymbol{B} \Rightarrow c(x, R)$. Ici, c veut dire *complet* ou *incomplet* et R est une relation dont on veut étudier la complétude. Par exemple, la règle $\mathit{lessThan_1}(x, \mathit{isCitizenOf}) \Rightarrow \mathit{incomplete}(x, \mathit{isCitizenOf})$ indique que toute personnes pour laquelle la base de connaissances ne connaît aucune nationalité doit être incomplète par rapport à cette relation. Ces règles ont été extraites en utilisant notre système de fouille de règles AMIE.

A.9.3 Expérimentation

Contexte expérimental. Nous avons évalué nos oracles de complétude sur YAGO3 et Wikidata. Nous avons appliqué les oracles pour prédire la complétude pour des sujets dans les domaines de 11 relations de Wikidata et de 10 relations dans YAGO. Les relations choisies sont celles pour lesquelles la notion de complétude est bien définie – elle peut être évaluée par des évaluateurs humains. Pour chaque oracle, nous avons évalué sa précision et son rappel en utilisant un jeu de données consistant d'annotations de complétude pour 200 couples sujet–relation. Pour certaines relations, notamment les fonctions obligatoires comme le sexe ou le lieu de naissance, nous avons construit ce jeu de données de façon complètement automatique : si le sujet ne dispose pas de l'attribut, il est incomplet, sinon il est complet. Pour d'autres relations, nous avons construit notre "gold standard" en demandant aux évaluateurs humains s'ils ont trouvé des objets supplémentaires sur le Web pour ce couple sujet–relation. À cette fin, nous avons utilisé Crowdflower[8], une plateforme de crowd-sourcing.

Résultats. La table A.11 présente nos résultats portant sur la performance (en termes de la mesure F1) de nos oracles de complétude sur Wikidata (les résultats sont similaires sur YAGO3). Nous observons que les oracles qui s'appuient sur l'extraction de règles ont la meilleure performance dans la plupart de cas.

A.9.4 Conclusion

Dans ce chapitre, nous avons défini et étudié un ensemble de signaux de complétude dans deux bases de connaissances. Nous avons montré aussi que, pour quelques relations les techniques d'extraction de règles peuvent prédire la complétude de manière très précise, en combinant plusieurs signaux individuels.

A.10 Extraction de règles numériques

A.10.1 Introduction

Dans toutes nos expériences avec AMIE, nous avons délibérément exclu tous les faits dont les objets ne sont pas des entités mais des valeurs littérales comme des

8. `https://www.crowdflower.com`

Relation	**CWA**	**PCA**	**card$_2$**	**Pop.**	**Star**	**Class**	**AMIE**
alma_mater	**90%**	14%	5%	1%	87%	87%	87%
brother	93%	1%	—	1%	94%	**96%**	**96%**
child	70%	1%	—	1%	**79%**	72%	73%
country_of_citizenship*	42%	97%	10%	3%	0%	0%	**98%**
director	81%	**100%**	—	3%	94%	89%	**100%**
father	5%	**100%**	6%	9%	89%	8%	**100%**
mother	3%	**100%**	3%	10%	67%*	5%	**100%**
place_of_birth	53%	**100%**	7%	5%	55%	0%	**100%**
place_of_death	89%	35%	1%	2%	81%	81%	**96%**
sex_or_gender	81%	**100%**	6%	3%	92%	91%	**100%**
spouse	**57%**	7%	—	1%	54%	54%	55%

TABLE A.11 – Mesure F1 pour nos oracles de complétude sur Wikidata

chaînes de caractères ou des nombres. En effet, dans la plupart des cas, ces valeurs sont uniques pour les faits, et l'extraction de règles n'obtient donc que des résultats de faible support. Nous observons néanmoins que certaines règles utilisant ces faits peuvent avoir une grande valeur pour les bases de connaissances. Considérons par exemple la règle

$$type(x, MarketEcon) \wedge import(x, y) \wedge export(x, z) \wedge cad(x, w) \Rightarrow w \approx 1.2{\times}(y - z)$$

Cette règle dit que si x est une économie de marché, son déficit (*cad*) est environ 120% de la différence entre le montant de ses exportations et celui de ses importations (exprimé en euros, par exemple). Cette règle est une *règle numérique*. L'extraction de ce type de règles est particulièrement délicate, car l'espace de recherche devient beaucoup plus grand à cause des arguments numériques : ici le facteur aurait pu être différent, par exemple 1.21 ou 1.3.

Dans ce chapitre, nous faisons le premier pas vers l'extraction de règles numériques en définissant un langage pour écrire ce type de règles. Notre langage est capable d'exprimer tous les types de règles numériques qui ont été étudiées dans la littérature.

A.10.2 Le langage

Contraintes numériques. Une contrainte numérique est un atome de la forme $x \circ \phi$, où x est une variable et ϕ est une expression numérique comme $1.2{\times}(y - z)$, qui peuvent contenir aussi des variables de la règle.

Notation fonctionnelle. Les relations numériques comme $population$ ou $hasHeight$ ont dans la plupart de cas un comportement fonctionnel[9] dans les bases de connaissances. Nous proposons en conséquence une notation fonctionnelle pour des règles

9. Si on ignore la dimension du temps

numériques, selon laquelle la règle

$$type(x, MarketEcon) \land import(x, y) \land export(x, z) \land cad(x, w) \Rightarrow w \approx 1.2 \times (y - z)$$

devient

$$type(x, MarketEcon) \Rightarrow cad(x) \approx 1.2 \times (import(x) - export(x))$$

Règles prédictives et descriptives. Considérons les règles

$$gdp(x, y) \ \land \ natDebt(x, z) \ \Rightarrow \ z = 0.9 \times y$$

et

$$gdp(x, y) \ \Rightarrow \ \exists z : natDebt(x, z) \ \land z = 0.9 \times y$$

La première règle décrit un invariant des données – il s'agit donc d'une *règle descriptive* – alors que la deuxième fait des prédictions pour la relation $natDebt$. Elle est donc une *règle prédictive*. Nous observons que notre règle prédictive n'est pas une clause de Horn, donc nous introduisons l'opérateur d'affectation $:=$ pour réécrire les règles prédictives comme des clauses de Horn. Notre exemple devient donc (en utilisant la notation fonctionnelle) :

$$\Rightarrow \ natDebt(x) := 0.9 \times gdp(x)$$

A.10.3 Conclusion

Dans ce chapitre, nous avons présenté un langage pour écrire des règles numériques. Nous croyons que cette contribution est le premier pas vers une solution complète pour l'extraction de règles numériques.

A.11 Conclusion

À travers ce travail, nous avons fait le premier pas vers un web encore plus sémantique, en identifiant – de façon automatique – des tendances dans les données. Ces tendances constituent des outils pour rendre les ordinateurs plus proactifs et “intelligents”. Les règles trouvées par les méthodes que nous avons décrites permettent ainsi aux ordinateurs de prédire des faits et de décrire un certain domaine de connaissances. Nous avons montré dans ce travail la valeur que ces règles peuvent apporter à de nombreuses tâches liées aux données : par exemple, l'alignement de schémas, la mise en forme canonique de bases de connaissances et la prédiction de complétude. Nous espérons que ce travail pourra motiver d'autres recherches dans ce domaine. En particulier, nous croyons que l'étude de langages de règles plus expressifs est une direction qui mérite d'être explorée plus en détail.

Rule Mining in Knowledge Bases

Luis Galárraga

RESUME : Le développement rapide des techniques d'extraction d'information a permis de construire de vastes bases de connaissances généralistes. Ces bases de connaissances contiennent des millions de faits portant sur des entités du monde réel, comme des personnes, des lieux, ou des organisations. Ces faits sont accessibles aux ordinateurs, et leur permettent ainsi de "comprendre" le monde réel. Ces bases trouvent donc de nombreuses applications, notamment pour la recherche d'information, le traitement de requêtes, et le raisonnement automatique.

Les nombreuses informations contenues dans les bases de connaissances peuvent également être utilisées pour découvrir des motifs intéressants et fréquents dans les données. Cette tâche, l'*extraction de règles d'association*, permet de comprendre la structure des données ; les règles ainsi obtenues peuvent être employées pour l'analyse de données, la prédiction, et la maintenance de données, entre autres applications.

Cette thèse présente deux contributions principales. En premier lieu, nous proposons une nouvelle méthode pour l'extraction de règles d'association dans les bases de connaissances. Cette méthode s'appuie sur un modèle d'extraction qui convient particulièrement aux bases de connaissances potentiellement incomplètes, comme celles qui sont extraites à partir des données du Web. En second lieu, nous montrons que l'extraction de règles peut être utilisée sur les bases de connaissances pour effectuer de nombreuses tâches orientées vers les données. Nous étudions notamment la prédiction de faits, l'alignement de schémas, la mise en forme canonique de bases de connaissances ouvertes, et la prédiction d'annotations de complétude.

MOTS-CLEFS : Extraction de règles, bases de connaissances, RDF

ABSTRACT : The continuous progress of information extraction (IE) techniques has led to the construction of large general-purpose *knowledge bases* (KBs). These KBs contain millions of computer-readable facts about real-world entities such as people, organizations and places. KBs are important nowadays because they allow computers to "understand" the real world. They are used in multiple applications in Information Retrieval, Query Answering and Automatic Reasoning, among other fields. Furthermore, the plethora of information available in today's KBs allows for the discovery of frequent patterns in the data, a task known as *rule mining*. Such patterns or rules convey useful insights about the data. These rules can be used in several applications ranging from data analytics and prediction to data maintenance tasks.

The contribution of this thesis is twofold : First, it proposes a method to mine rules on KBs. The method relies on a mining model tailored for potentially incomplete web-extracted KBs. Second, the thesis shows the applicability of rule mining in several data-oriented tasks in KBs, namely facts prediction, schema alignment, canonicalization of (open) KBs and prediction of completeness.

KEY-WORDS : Rule Mining, Knowledge Bases, RDF