Rule Induction and Reasoning over Knowledge Graphs

Daria Stepanova, Mohamed H. Gad-Elrab, Vinh Thinh Ho

Max Planck Institute for Informatics,
Saarland Informatics Campus, Germany
{dstepano, gadelrab, hvthinh}@mpi-inf.mpg.de

Abstract. Advances in information extraction have enabled the automatic construction of large knowledge graphs (KGs) like DBpedia, Freebase, YAGO and Wikidata. Learning rules from KGs is a crucial task for KG completion, cleaning and curation. This tutorial presents state-of-the-art rule induction methods, recent advances, research opportunities as well as open challenges along this avenue. We put a particular emphasis on the problems of learning exception-enriched rules from highly biased and incomplete data. Finally, we discuss possible extensions of classical rule induction techniques to account for unstructured resources (e.g., text) along with the structured ones.

1 Introduction

Motivation. Recent advances in information extraction have led to huge graph-structured knowledge bases (KBs) also known as knowledge graphs (KGs) such as NELL [47], DBpedia [42], YAGO [72] and Wikidata [77]. These KGs contain millions or billions of relational facts in the form of subject-predicate-object (SPO) triples, e.g., $\langle albert\_einstein\ marriedTo\ mileva\_maric\rangle$ or $\langle albert\_einstein\ type\ phycisist\rangle$. Such triples can be straightforwardly represented by means of positive unary and binary first-order logic facts, e.g. $marriedTo(albert\_einstein, mileva\_maric)$ and $phycisist(albert\_einstein)$.

An important task over KGs is rule learning, which is relevant for a variety of applications ranging from knowledge graph curation (completion, error detection) [55] to data mining and semantic culturomics [73]. Rules over KGs are of the form $head \leftarrow body$, where $head$ is a binary atom and $body$ is a conjunction of (possibly negated) binary and unary atoms.

Traditionally, rule induction has been studied in the context of relational data mining in databases (see e.g., [58] for overview), and has recently been adapted to KGs (e.g., [30,6,79]). The methods from this area can be used to identify prominent patterns from KGs, such as *"Married people live in the same place"*, and cast them in the form of Horn rules (i.e., rules with only positive atoms), such as: $r_1 : livesIn(Y, Z) \leftarrow marriedTo(X, Y), livesIn(X, Z)$.

For the KG curation, this has two-fold benefits. First, since KGs operate under the open world assumption (i.e., absent facts are treated as unknown

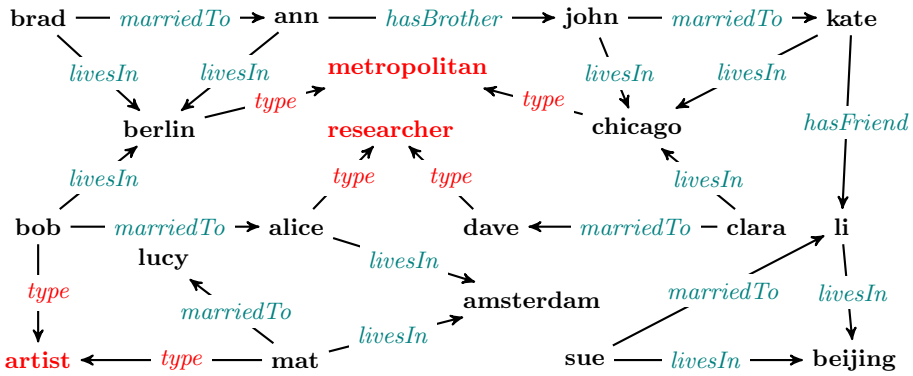

Fig. 1: Example KG: marriage relations and living places [76].

rather than false), the rules can be used to derive additional facts such as missing living places. Second, rules can be used to eliminate erroneous facts in KGs. For example, assuming that *livesIn* is a functional relation, a living place of a person could be questioned if it differs from the spouse's.

When rules are automatically learned, statistical measures like support, confidence and their variations are used to assess the quality of the rules. Most notably, the confidence of a rule is the fraction of facts predicted by the rule that are indeed in the KG. However, this is a meaningful measure for rule quality only when the KG is reasonably complete. For rules learned from incomplete KGs, confidence and other measures may be misleading, as they do not reflect the patterns in the missing facts. This might lead to the extraction of erroneous rules from incomplete and biased KGs. For example, a KG that stores a lot of information about families of popular scientists but lacks data in other domains, would yield a heavily biased rule $r_1' : \mathit{hasChild}(X, Y) \leftarrow \mathit{worksAt}(X, Z), \mathit{educated}(Y, Z)$, stating that workers of certain institutions often have children among the people educated there, as this is frequently the case for scientific families.

To address this issue, several rule measures that are specifically tailored towards incomplete KGs have been proposed [30,82] (see [18,3] for an overview of other measures). Along with KGs themselves, additional background knowledge could be used for better rule assessment. As proposed in [75] this includes metadata about the concrete numbers of facts of certain types that hold in the real world (e.g., *"Einstein has 3 children"*) automatically extracted from Web resources using techniques like [45]. Other types of background knowledge comprise description logic ontologies, e.g., [11] or more general hybrid theories, e.g., [43,36].

Horn rules such as r_1, might not always be sufficiently expressive to capture KG patterns accurately. Indeed, these rules cannot handle exceptions, which often appear in practice. For instance, application of r_1 mined from the KG in Figure 1 results in the facts $\mathit{livesIn}(\mathit{alice}, \mathit{berlin})$, $\mathit{livesIn}(\mathit{dave}, \mathit{chicago})$ and $\mathit{livesIn}(\mathit{lucy}, \mathit{amsterdam})$. Observe that the first two facts might be suspected to

be wrong. Indeed, both *alice* and *dave* are researchers, and the rule r_1 could have researcher as a potential exception resulting in its more accurate version given as $r_2 : livesIn(Y, Z) \leftarrow marriedTo(X, Y), livesIn(X, Z), not\ researcher(Y)$. Exception handling has been faced in inductive logic programming (ILP) by learning nonmonotonic logic programs, i.e., programs with negations from databases (e.g., [35,66,64]), and recently also studied in the context of KGs [29,76].

The aim of this article is to survey the current research on rule learning from knowledge graphs. We present and discuss different techniques with the roots in inductive logic programming and relational data mining as well as their interrelation and applications for KGs.

Tutorial Overview. In Section 2, we briefly introduce knowledge graphs and their key properties. We then provide necessary preliminaries on rule-based deductive reasoning over KGs and discuss the tasks within the area of inductive logic programming in Section 3. Section 4 describes recent research progress in the context of Horn rule induction for KG completion. We present techniques for nonmonotonic rule extraction in Section 5. Finally, in Section 6 we conclude the article with an outlook discussion, where we identify a number of promising directions for future work.

2 Knowledge Graphs

Knowledge graphs have been introduced in the Semantic Web community to create the "Web of data" that is readable by machines. They represent interlinked collections of factual information, and are often encoded using the RDF data model [38]. This data model represents the content of a graph with a set of triples of the form $\langle subject\ predicate\ object \rangle$ corresponding to positive unary and binary first-order logic (FOL) facts.

Formally, we assume countable sets $\mathcal{R}$ of unary and binary predicate names (aka relations) and $\mathcal{C}$ of constants (aka entities). A knowledge graph $\mathcal{G}$ is a finite set of ground atoms of the form $p(s, o)$ and $c(s)$ over $\mathcal{R} \cup \mathcal{C}$. With $\Sigma_{\mathcal{G}} = \langle \mathcal{R}, \mathcal{C} \rangle$, the signature of $\mathcal{G}$, we denote elements of $\mathcal{R} \cup \mathcal{C}$ that occur in $\mathcal{G}$.

Example 1. Figure 1 shows a snippet of a graph about people, family and friendship relations among them as well as their living places and professions. For instance, the upper left part encodes the information that "Ann has a brother John, and lives with her husband Brad in Berlin, which is a metropolitan city" represented by the FOL facts $\{hasBrother(ann, john)$, $livesIn(ann, berlin)$, $livesIn(brad, berlin)$, $metropolitan(berlin)$, $marriedTo(brad, ann)\}$. The set $\mathcal{R}$ of relations appearing in the given KG contains the predicates $livesIn$, $marriedTo$, $hasBrother$, $hasFriend$, $researcher$, $metropolitan$, $artist$, while the set $\mathcal{C}$ of constants comprises of the names of people and locations depicted on Figure 1. □

All approaches for KG construction can be roughly classified into two major groups: manual and (semi-)automatic. The examples of KGs constructed manually include, e.g., WordNet [44] which has been created by a group of experts

Knowledge Graphs	# Entities	# Relations	# Facts
DBpedia (en)	4.8 M	2800	176 M
Freebase	40 M	35000	637 M
YAGO3	3.6 M	76	61 M
Wikidata	46 M	4700	411 M
Google Knowledge Graph	570 M	35000	18000 M

Table 1: Examples of real-world KGs and their statistics [53,55].

or Freebase [1] and Wikidata [77] which are constructed collaboratively by volunteers. Automatic population of KGs from semi-structured resources such as Wikipedia info-boxes using regular expressions and other techniques gave rise to, e.g., YAGO [72] and DBpedia [42]. There are also projects devoted to the extraction of facts from unstructured resources using natural language processing and machine learning methods. For example, NELL [47] and KnowledgeVault [16] belong to this category. Table 1 shows examples of some KGs and their statistics.

2.1 Incompleteness, Bias and Noise of Knowledge Graphs

While the existing KGs contain millions of facts, they are still far from being complete; Therefore they are treated under the open world assumption (OWA), i.e., facts not present in KGs are assumed to be unknown rather than false. For example, given only Germany as the living place of Albert Einstein, we cannot say whether $\mathit{livedIn}(\mathit{einstein}, \mathit{us})$ is true or false. This is opposed to the closed world assumption (CWA) made in databases, under which $\neg\mathit{livedIn}(\mathit{einstein}, \mathit{us})$ would be inferred.

Apart from incompleteness, both manually and semi-automatically created KGs also suffer from the problem of bias in the data. Indeed, manually created KGs such as Wikidata contain crowd-sourced information. While leveraging crowd-sourcing for KG construction, human curators from different countries add factual statements that they find interesting. Due to the difference in the population of contributors obviously facts about some countries are covered better than about others. For example, KGs typically store more facts about Austrians than Ghanians even though there are three times more inhabitants in Ghana than in Austria. Moreover, different contributors find different facts interesting, e.g., Austrians would add detailed information about music composers, while Ghanians about national athletes. This naturally leads to cultural bias in KGs. Likewise KGs that are semi-automatically extracted from Wikipedia infoboxes such as DBPedia and YAGO highly depend on the pre-defined properties that the infoboxes contain [40].

Along with completeness and absence of data bias, there are also other important aspects reflecting KGs' quality including their correctness. Regardless of the KG construction method, the resulting facts are rarely error-free. Indeed, in the case of manually constructed KGs human contributors might bring their own

opinion on the added factual statements (e.g., Catalonia being a part of Spain or an independent country). On the other hand, automatically constructed KGs often contain noisy facts, since information extraction methods are imperfect. We refer the reader to [53,55] for further discussions on the available KGs and their quality.

The problems of KG completion and cleaning are among the central ones. Approaches for addressing them can be roughly divided into two groups: statistics-based, and logic-based. The firsts apply such techniques as tensor factorization, or neural-embedding-based models (see [54] for overview). The second group concentrates on logical rule learning [28]. In this tutorial, we primarily focus on rule-based techniques, and their application for the KG completion task. In the following, we assume that the given KG $\mathcal{G}$ stores only a subset of all true facts.

Suppose we had an *ideal* KG $\mathcal{G}^i$[1] that contains *all* correct and true facts in the world reflecting the relations from $\mathcal{R}$ that hold among the entities in $\mathcal{C}$. The gap between $\mathcal{G}$ and its ideal version $\mathcal{G}^i$ is defined as follows:

Definition 1 (Incomplete Knowledge Graph [12]). *An* incomplete knowledge graph *is a pair* $G = (\mathcal{G}, \mathcal{G}^i)$ *of two KGs, where* $\mathcal{G} \subseteq \mathcal{G}^i$ *and* $\Sigma_{\mathcal{G}} = \Sigma_{\mathcal{G}^i}$. *We call* $\mathcal{G}$ *the available graph and* $\mathcal{G}^i$ *the ideal graph.*

Note that $\mathcal{G}^i$ is an abstract construct, which is normally unavailable. Intuitively, *rule-based KG completion task* concerns the reconstruction of the ideal KG (or its approximation) by means of rules induced from the available KG and possibly other external information sources.

3 Rules and Reasoning

In this section, we briefly review the concepts of rules, logic programming (see [21] for more details) as well as deductive and inductive reasoning.

3.1 Logic Programs

Logic programs consist of a set of rules. Intuitively, a rule is an if-then expression, whose if-part may contain several conditions, some possibly with negation. The then-part has a single atom that has to hold, whenever the if-part holds. In general, the then-part can also contain disjunctions, but in this tutorial we consider only non-disjunctive rules. More formally,

Definition 2 (Rule). *A* rule r *is an expression of the form*

$$h(\boldsymbol{X}) \leftarrow b_1(\boldsymbol{Y_1}), \ldots, b_k(\boldsymbol{Y_k}), not\ \ b_{k+1}(\boldsymbol{Y_{k+1}}), \ldots, not\ \ b_n(\boldsymbol{Y_n}) \tag{1}$$

where $h(\boldsymbol{X}), b_1(\boldsymbol{Y_1}), \ldots, b_n(\boldsymbol{Y_n})$ *are first-order atoms and the right-hand side of the rule is a conjunction of atoms. Moreover,* $\boldsymbol{X}, \boldsymbol{Y_1}, \ldots, \boldsymbol{Y_n}$ *are tuples of either variables or constants whose length corresponds to the arity of the predicates* $h, b_1, \ldots, b_n$ *respectively.*

[1] The superscript i stands for *ideal*.

The left-hand side of a rule r is referred to as its head, denoted by $head(r)$, while the right-hand side is its body $body(r)$. The positive and negative parts of the body are respectively denoted as $body^+(r)$ and $body^-(r)$. A rule r is called *positive* or *Horn* if $body^-(r) = \emptyset$.

For simplicity, in this work we use the shortcut

$$H \leftarrow B, not\ \ E \tag{2}$$

to denote rules with a single negated atom, i.e., here $H = h(\boldsymbol{X}), B = b_1(\boldsymbol{Y_1}), \ldots, b_k(\boldsymbol{Y_k}), E = b_{k+1}(\boldsymbol{Y_{k+1}})$.

Example 2. Consider r_2 from Section 1. We have that $head(r_2) = \{livesIn(Y, Z)\}$, while $body^+(r_2) = \{isMarriedTo(X, Y), livesIn(X, Z)\}$, and moreover it holds that, $body^-(r_2) = \{not\ researcher(Y)\}$. □

A logic program P is *ground* if it consists of only ground rules, i.e. rules without variables.

Example 3. For instance, a possible grounding of the rule r_2 is given as follows
$livesIn(dave, chicago) \leftarrow livesIn(clara, chicago), isMarriedTo(clara, dave),$
$not\ \ researcher(dave).$ □

Ground instantiation $Gr(P)$ of a nonground program P is obtained by substituting variables with constants in all possible ways.

Definition 3 (Herbrand Universe, Base, Interpretation). *A* Herbrand universe $HU(P)$ *is a set of all constants occurring in the given program* P*. A* Herbrand base $HB(P)$ *is a set of all possible ground atoms that can be formed with predicates and constants appearing in* P*. A* Herbrand interpretation *is any subset of* $HB(P)$*.*

We now formally define the satisfaction relation.

Definition 4 (Satisfaction, Model). *An interpretation* I satisfies

- *a ground atom* a*, denoted* $I \models a$*, if* $a \in I$*,*
- *a negated ground atom* $not\ a$*, denoted* $I \models not\ a$*, if* $I \not\models a$*,*
- *a conjunction* $b_1, \ldots, b_n$ *of ground literals, denoted* $I \models b_1, \ldots, b_n$*, if for each* $i \in \{1, \ldots, n\}$ *it holds that* $I \models b_i$*,*
- *a ground rule* r*, denoted* $I \models r$ *if* $I \models body(r)$ *implies* $I \models head(r)$*, i.e., if all literals in the body hold then the literal in the head also holds.*

An interpretation I is a *model* of a ground program P, if $I \models r$ for each rule $r \in P$. A model I is *minimal* if there is no other model $I' \subset I$. By $MM(P)$ we denote the set-inclusion minimal model of a ground positive program P.

The classical definition of answer sets based on the Gelfond-Lifschitz reduct [31] is given as follows.

Definition 5 (Gelfond-Lifschitz Reduct, Answer Set [31]). *An interpretation I of P is an* answer set *(or* stable model*) of P iff $I \in MM(P^I)$, where P^I is the* Gelfond-Lifschitz (GL) reduct *of P, obtained from $Gr(P)$ by removing (i) each rule r such that $Body^-(r) \cap I \neq \emptyset$, and (ii) all the negative atoms from the remaining rules. The set of answer sets of a program P is denoted by $AS(P)$.*

An alternative definition of answer sets relies on the FLP-reduct [25] which might be more intuitive. For the class of programs considered in this work, the GL-reduct and FLP-reduct are equivalent.

Definition 6 (Faber-Leone-Pfeifer Reduct, Answer Set [25]). *An interpretation I of P is an* answer set *(or* stable model*) of P iff $I \in MM(fP^I)$, where fP^I is the* Faber-Leone-Pfeifer (FLP) reduct *of P, obtained from $Gr(P)$ by keeping only rules r, whose bodies are satisfied by I, i.e.,*

$$fP^I = \{r \in P \mid head(r) \leftarrow body(r), I \models body(r)\}.$$

Example 4. Consider the program

$$P = \left\{ \begin{array}{l} (1)\ livesIn(brad, berlin);\ (2)\ isMarriedTo(brad, ann); \\ (3)\ livesIn(Y, Z) \leftarrow isMarriedTo(X, Y), livesIn(X, Z), not\ \ researcher(Y) \end{array} \right\}$$

The relevant part of ground instantiation $Gr(P)$ of P is obtained by substituting X, Y, Z with *brad*, *ann* and *berlin* respectively. For $I = \{isMarriedTo(brad,ann), livesIn(ann,berlin), livesIn(brad, berlin)\}$, both the GL- and FLP-reduct contain $livesIn(ann, berlin) \leftarrow livesIn(brad, berlin), isMarriedTo(brad, ann)$ and the facts (1), (2). As I is a minimal model of these reducts, I is an answer set of P. □

Apart from the model computation, another important task concerns deciding whether a given atom a is entailed from the program P denoted by $P \models a$. Typically, one distinguishes between brave and cautious entailment. We say that an atom a is *cautiously entailed* from a program P ($P \models_c a$) if a is present in all answer sets of P. Conversely, an atom a is *bravely entailed* from a program P ($P \models_b a$) if it is present in at least one answer set of P. If a program has only a single answer set, e.g., it is positive, then obviously cautious and brave entailments coincide, in which case we omit the subscript.

Example 5. For P from Example 4 and $a = livesIn(ann, berlin)$, we have that a is entailed both bravely and cautiously from P, i.e., $P \models a$, while for $a' = marriedTo(brad, brad)$, it holds that $P \not\models a'$. □

The answer set semantics for nonmonotonic logic programs is based on the CWA, under which whatever can not be derived from a program is assumed to be false. Nonmonotonic logic programs are widely applied for formalizing common sense deductive reasoning over incomplete information.

3.2 Inductive Reasoning Tasks

So far we have focused on logic programs and their semantics for deductive reasoning. We now discuss the problem of automatic rule induction, which is a research area commonly referred to as rule learning (see, e.g., [58,27]).

Broadly speaking, rule learning is an important sub-field of machine learning, which focuses on symbolic methods for intelligent data analysis, i.e., methods that employ a certain description language in which the learned knowledge is represented.

First-order learning approaches are also referred to as *inductive logic programming (ILP)*, since the patterns they discover are expressed in relational formalisms of first-order logic (see [58] for overview). The goal of ILP is to generalize individual instances/observations in the presence of background knowledge by building hypotheses about yet unseen instances. The most commonly addressed task in ILP is the task of learning logical definitions of relations. From training examples ILP induces a logic program (predicate definition) corresponding to a view that defines the target relation in terms of other relations that are given as background knowledge.

More formally, the classical inductive logic programming task of learning from positive and negative examples (also known as learning from entailment) is defined as follows:

Definition 7 (Inductive Learning from Examples [49]).

Given:

- *Positive examples E^+ and negative examples E^- over the target n-ary relation p, i.e. sets of facts;*
- *Background knowledge T, i.e. a set of facts over various relations and possibly rules that can be used to induce the definition of p;*
- *Syntactic restrictions on the definition of p.*

Find:

- *A hypothesis Hyp that defines the target relation p, which is (i)* complete*, i.e., $\forall e \in E^+$, it holds that $T \cup Hyp \models e$, and (ii)* consistent*, i.e., $\forall e' \in E^-$: $T \cup Hyp \not\models e'$.*

Example 6. Suppose that you possess information about some of the relationships between people in your family and their genders. However, you do not know what the relationship *fatherOf* actually means. You might have the following beliefs, i.e., background knowledge.

$$T = \left\{ \begin{array}{l} (1)\ parentOf(john, mary); (2)\ male(john); (3)\ parentOf(david, steve); \\ (4)\ male(david); (5)\ parentOf(kathy, ellen); (6)\ female(kathy); \end{array} \right\}$$

Moreover, you are given the following positive and negative examples.

$E^+ = \{fatherOf(john, mary), fatherOf(david, steve)\}$
$E^- = \{fatherOf(kathy, ellen), fatherOf(john, steve)\}$

One of the possible hypothesis that can be induced from the above knowledge reflecting the meaning of the *fatherOf* relation is given as follows:
$Hyp : fatherOf(X, Y) \leftarrow parentOf(X, Y), male(X)$. This hypothesis is consistent with the background theory T, and together with T it entails all of the positive examples, and none of the negative ones. The classical ILP task concerns automatic extraction of such hypothesis. □

System	Input data	Output rules	Multiple predicates	Increment	Interact	Noise handling
FOIL [57]	examples	Horn	no	no	no	yes
GOLEM [51]	examples	Horn	no	no	no	yes
LINUS [19]	examples	Horn	no	no	no	yes
ALEPH [71]	examples	Horn	no	no	yes	yes
MPL [61]	examples	Horn	yes	no	no	no
MIS [69]	examples	Horn	yes	yes	yes	no
MOBAL [48]	examples	Horn	yes	yes	no	no
CIGOL [50]	examples	Horn	yes	yes	yes	no
CLINT [13]	examples	Horn	yes	yes	yes	no
σILP [24]	examples	Horn	no	no	no	yes
DROPS [8]	examples	NM	no	no	no	no
ASPAL [9]	examples	NM	no	no	no	no
XHAIL [64]	examples	NM	no	no	no	no
ILASP [41]	interpretations	NM	yes	no	no	no
ILED [37]	examples	NM	no	yes	no	no

Table 2: Overview of classical ILP systems.

In an alternative setting, known as *learning from interpretations*, instead of positive and negative examples over a certain relation, one is given a Herbrand interpretation I, i.e., a set of facts, and conditions (i) and (ii) in Definition 7 are replaced with the requirement that I is a minimal model of $Hyp \cup T$. Formally,

Definition 8 (Inductive Learning from Interpretations [60]).

Given:

- *An interpretation I, i.e., a set of facts over various relations*
- *Background knowledge T, i.e., a set facts and possibly rules*
- *Syntactic restrictions on the form of rules to be induced*

Find:

- *A hypothesis Hyp, such that I is a minimal Herbrand model of $Hyp \cup T$.*

Several variations of learning from interpretations task have been studied including learning from answer sets [66,41], where given possibly multiple partial interpretations, the goal is to find a logic program, which has extensions of (all of) the provided interpretations as answer sets.

To date, the main tasks considered in the ILP area can be classified based on the following parameters [67,4].

- *type of the data source*, e.g., positive/negative examples, interpretations, text
- *type of the output knowledge*, e.g., Horn/nonmonotonic rules over single or multiple predicates, description logic (DL) class descriptions, class inclusions
- *the way the data is given as input*, e.g., all data at once or incrementally
- *availability of an oracle*, e.g., involvement of a human expert in the loop
- *quality of the data source*, e.g., noisy or clean

- *data (in)completeness assumption*, e.g., OWA, CWA
- *availability and type of background knowledge*, e.g., DL ontology, set of datalog rules, hybrid theories, etc.

An overview of some of the systems for Horn and nonmonotonic (NM) rule induction with their selected properties is presented in Table 2.

4 Rule Learning for Knowledge Graph Completion

The majority of the classical existing rule induction methods mentioned in Section 3 assume that the given data from which the rules are induced is complete, accurate and representative. Therefore, they rely on CWA and aim at extracting rule hypotheses that perfectly satisfy the criteria from Definition 7 or Definition 8. On the other hand, as discussed in Section 2.1, knowledge graphs are highly incomplete, noisy and biased. Moreover, the real world is very complicated, and its exact representation often cannot be acquired from the data, meaning that the task of inducing a perfect rule set from a KG is typically unfeasible. Therefore, in the context of KGs, one normally aims at extracting certain regularities from the data, which are not universally correct, but when seen as rules predict a sufficient portion of true facts.

In other words, the goal of automatic rule-based KG completion is to learn a set R of logic rules from the available graph $\mathcal{G}$, such that their application on $\mathcal{G}$ results in a good approximation of the ideal graph $\mathcal{G}^i$. More formally,

Definition 9 (Rule-based KG Completion). *Let $\mathcal{G}$ be a KG over the signature $\Sigma_\mathcal{G} = \langle \mathcal{R}, \mathcal{C} \rangle$. Let, moreover, R be a set of rules with predicates from $\mathcal{R}$ induced from $\mathcal{G}$. Then* rule-based completion of $\mathcal{G}$ w.r.t. *R is a graph $\mathcal{G}_R$ constructed from any answer set $\mathcal{G}_R \in AS(R \cup \mathcal{G})$.*

Example 7. Given the KG in Figure 1 as $\mathcal{G}$ and the rule set $R = \{r_2\}$, where r_2 is from Section 1 we have $\mathcal{G}_R = \mathcal{G} \cup \{livesIn(lucy, amsterdam)\}$. □

If the ideal graph $\mathcal{G}^i$ was known, then the problem of rule-based KG completion would be essentially the same as the task of learning from interpretations given in Definition 8, with $\mathcal{G}$ playing a role of the background knowledge T and $\mathcal{G}^i$ the interpretation I. However, unfortunately $\mathcal{G}^i$ is unknown, and it cannot be easily reconstructed, since KGs follow OWA.

Moreover, reusing the methods that induce logical theories from a set of positive and negative examples from Definition 7 is likewise not possible due to the following important obstacles [76].

First, the target predicates (e.g. *fatherOf* from Example 6) can not be easily identified, since we do not know which parts of the considered KG need to be completed. A standard way of addressing this issue would be just to learn rules for all the different predicate names occurring in the KG. Unfortunately, this is unfeasible given the huge size of KGs. Second, the negative examples are not available, and they cannot be easily obtained from, e.g., domain experts due to - once again - the huge size of KGs.

To overcome the above obstacles, it is more appropriate to treat the KG completion problem as an unsupervised relational learning task [30]. In this section, we describe approaches that rely on *relational association rule learning* techniques for extraction of *Horn rules* from incomplete KGs. These concern the discovery of frequent patterns from a data set and their subsequent transformation into rules (see, e.g., [15] as the seminal work in this direction).

First, we describe the relational association rules, and how they are traditionally evaluated under CWA. Then, we present the standard relational rule learning techniques, which usually proceed in two steps: rule construction and rule assessment.

4.1 Relational Association Rules

An association rule is a rule where certain properties of the data in the body of the rule are related to other properties in its head. For an example of an association rule, consider a database containing transactional data from a store selling computer equipment. From this data one can extract the association rule stating that 70% of the customers buying a laptop also buy a docking station. The knowledge that such rule reflects can assist in planning store layout or deciding which customers are likely to respond to an offer.

Traditionally, the discovery of association rules has been performed on data stored in a single table. Recently, however, many methods for mining relational, i.e., graph-based data have been proposed [58].

The notion of multi-relational association rules is heavily based on frequent conjunctive queries and query subsumption [32].

Definition 10 (Conjunctive Query). *A* conjunctive query Q *over* $\mathcal{G}$ *is of the form* $p_1(\mathbf{X_1}), \ldots, p_m(\mathbf{X_m})$*, where* $\boldsymbol{X_i}$ *are symbolic variables or constants and* $p_i \in \mathcal{R}$ *are unary or binary predicates. The* answer *of* Q *on* $\mathcal{G}$ *is the set* $Q(\mathcal{G}) = \{(\nu(\mathbf{X_1}), \ldots, \nu(\mathbf{X_m}) \mid \forall i : p_i(\nu(X_i), \nu(Y_i)) \in \mathcal{G}\}$ *where* ν *is a function that maps variables and constants to elements of* $\mathcal{C}$*.*

The *(absolute) support* of a conjunctive query Q in a KG $\mathcal{G}$, is the number of distinct tuples in the answer of Q on $\mathcal{G}$ [15].

Example 8. The support of the query

$$Q(X, Y, Z) \text{ :- } marriedTo(X, Y), livesIn(X, Z)$$

over $\mathcal{G}$ in Figure 1 asking for people, their spouses and living places is 6. □

Definition 11 (Association Rule). *An* association rule *is of the form* $Q_1 \Rightarrow Q_2$*, such that* Q_1 *and* Q_2 *are both conjunctive queries, and the body of* Q_1 *considered as a set of atoms is included in the body of* Q_2*, i.e.,* $Q_1(\mathcal{G}') \subseteq Q_2(\mathcal{G}')$ *for any possible KG* $\mathcal{G}'$*.*

Example 9. For instance, from the above $Q(X, Y, Z)$ and

$$Q'(X, Y, Z) \text{ :- } marriedTo(X, Y),\ livesIn(X, Z), livesIn(Y, Z)$$

we can construct the association rule $Q \Rightarrow Q'$. □

Association rules are sometimes exploited for reasoning purposes, and thus (with some abuse of notation) can be treated as logical rules, i.e., for $Q_1 \Rightarrow Q_2$ we write $Q_2 \backslash Q_1 \leftarrow Q_1$, where $Q_2 \backslash Q_1$ refers to the set difference between Q_2 and Q_1 considered as sets of atoms. For example, $Q \Rightarrow Q'$ from above corresponds to *r1* from Section 1.

A large number of measures for evaluating the quality of association rules and their subsequent ranking have been proposed, e.g., *support, confidence*.

For $r: \ H \leftarrow B, not \ \ E$ (see Equation 2), with $H = h(X, Y)$, B, E involving variables from $\boldsymbol{Z} \supseteq \{X, Y\}$ and a KG $\mathcal{G}$, the *(standard) confidence* is given as:

$$conf(r, \mathcal{G}) = \frac{r\text{-}supp(r, \mathcal{G})}{b\text{-}supp(r, \mathcal{G})}$$

where $r\text{-}supp(r, \mathcal{G})$ and $b\text{-}supp(r, \mathcal{G})$ are the *rule support* and *body support*, respectively, which are defined as follows:

$$r\text{-}supp(r, \mathcal{G}) = \#(x, y) : h(x, y) \in \mathcal{G}, \exists \boldsymbol{z} \ B \in \mathcal{G}, E \notin \mathcal{G}$$
$$b\text{-}supp(r, \mathcal{G}) = \#(x, y) : \exists \boldsymbol{z} \ B \in \mathcal{G}, E \notin \mathcal{G}$$

Example 10. Consider the rules r_1, r_2, and the KG $\mathcal{G}$ in Figure 1, we have $r\text{-}supp(r_1, \mathcal{G}) = r\text{-}supp(r_2, \mathcal{G}) = 3$, $b\text{-}supp(r_1, \mathcal{G}) = 6$ and $b\text{-}supp(r_2, \mathcal{G}) = 4$. Hence, $conf(r_1, \mathcal{G}) = \frac{3}{6}$ and $conf(r_2, \mathcal{G}) = \frac{3}{4}$. □

Another popular metrics, which is shown to guarantee the high predictive power [3] by measuring the intensity of rule's implication [18] is *conviction*, defined as:

$$conv(r, \mathcal{G}) = \frac{1 - rel\text{-}supp(H, \mathcal{G})}{1 - conf(r, \mathcal{G})}$$

where $rel\text{-}supp(H, \mathcal{G})$ is the relative support of the head, measured by:

$$rel\text{-}supp(H, \mathcal{G}) = \frac{\#(x, y) : h(x, y) \in \mathcal{G}}{(\#x : \exists y : h(x, y) \in \mathcal{G}) \times (\#y : \exists x : h(x, y) \in \mathcal{G})}$$

Example 11. For the KG in Figure 1, we have $rel\text{-}supp(livesIn, \mathcal{G}) = \frac{10}{10 \times 4} = \frac{1}{4}$, thus $conv(r_1, \mathcal{G}) = \frac{1 - \frac{1}{4}}{1 - \frac{3}{6}} = \frac{3}{2}$ and $conv(r_2, \mathcal{G}) = \frac{1 - \frac{1}{4}}{1 - \frac{3}{4}} = 3$. □

4.2 Rule Construction

We now briefly summarize some of the state-of-the-art methods for rule construction, most of which extract so-called *closed* rules, i.e., rules, in which every variable appears at least twice. Restriction to closed rules ensures the actual prediction of a fact by a rule, but not just its existence [30].

Example 12. The rules r_1 and r_2 from Section 1 are *closed*. An example of a *non-closed* rule is:

$$\exists Z\ livesIn(Y, Z) \leftarrow isMarriedTo(X, Y)$$

which states that married people live somewhere. This rule cannot infer the exact living place of a person, but merely its existence, and thus it is less interesting in this context. □

The most prominent examples of systems that are specifically tailored towards inducing Horn rules from KGs are AMIE [30] and RDF2Rules [79].

AMIE. AMIE [30] is a state-of-the-art Horn rule mining system. Apart from a KG, it expects the maximum length of the rule and the threshold value for its support. In AMIE, rules are treated as sequences of atoms, where the first atom is the head, and subsequent atoms form the body of the rule. The algorithm maintains a queue of intermediate rules, which initially stores a single rule with an empty body for every KG relation. Rules are removed from the queue and refined by adding literals to the body according to a language bias that specifies allowed rule forms (e.g., based on the user-provided rule length). The system then estimates the support of the rule, and if it exceeds the given threshold, the rule is output to the user and also added to the queue for possible further processing. Refinement of a rule relies on the following set of mining operators used to extend the sequences of atoms in the rule body:

- *add dangling atom*: add a new positive atom with one fresh variable, i.e., variable not appearing elsewhere in the rule;
- *add instantiated atom*: add a positive atom with one argument being a constant and the other one being a shared variable, i.e., variable already present in another rule atom;
- *add closing atom*: add a positive atom with both of its arguments being shared variables.

The implementation of AMIE employs a variety of techniques from the database area, which allow it to achieve high scalability.

RDF2Rules. While AMIE mines a single rule at a time, RDF2Rules [79] parallelizes this process by extracting *frequent predicate cycles* (FPCs) of a certain length k, which have the following form:

$$(X_1, p_1^{d_1}, X_2, p_2^{d_2}, \ldots, X_k, p_k^{d_k}, X_1)$$

where, X_is are variables to appear in the extracted rules, p_is are predicates connecting these variables, and d_is $\in \{0, 1\}$ reflect the direction of the edges in the KG labeled by the respective predicates. To extract FPCs, the RDF2Rules algorithm first mines the *frequent predicate paths* (FPPs) of the form:

$$(X_1, p_1^{d_1}, X_2, p_2^{d_2}, ..., X_k, p_k^{d_k}, X_{k+1})$$

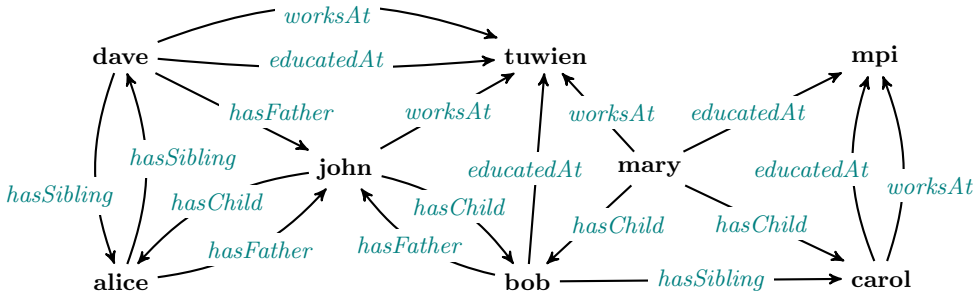

Fig. 2: Example KG: family relations, working and education places [75].

of the length k, which are obtained recursively based on FPPs of the length $k-1$. FPCs are then created from FPPs by merging the last variable X_{k+1} with the first one X_1. After FPCs are mined, rules are extracted from them by choosing a single predicate to be in the rule head, and collecting the rest into it's body.

RDF2Rules is capable of accounting for unary predicates (i.e., types of entities), which are neglected in AMIE for scalability reasons. The unary predicates are added to the constructed rule at the final stage after analyzing the frequent types for FPCs corresponding to a given rule. While RDF2Rules performs the rule extraction faster than AMIE due to an effective pruning strategy used in the process of mining FPCs, the supported rule patterns are more restrictive.

4.3 Rule Evaluation

Most of state-of-the-art KG-based positive rule mining systems differ from each other with respect to the employed rule ranking function. The ranking metrics from data mining such as support and confidence (see e.g., [18] for overview of others) presented in Section 4.1 have been designed for datasets treated under the CWA, and they can be counterintuitive for the KGs, in which facts are largely missing.

Example 13. For instance, consider the KG $\mathcal{G}'$ in Figure 2 [75], which presents information about scientific families. The heavily biased rule from Section 1: r'_1 : $hasChild(X, Y) \leftarrow worksAt(X, Z), educated(Y, Z)$ can be mined from it along with r'_2 : $hasSibling(X, Z) \leftarrow hasFather(X, Y), hasChild(Y, Z)$, which is an accurate rule stating that people with the same father are likely siblings. Since the graph is highly incomplete for the *hasSibling* relation, the standard rule measures such as confidence reflect a counterintuitive rule preference. Indeed, we have $conf(r'_1, \mathcal{G}') = \frac{2}{8}$, while $conf(r'_2, \mathcal{G}') = \frac{1}{6}$. □

Below we present some of other alternative measures, designed to quantify the quality of rules extracted specifically from incomplete data.

PCA Confidence. In [30], a completeness-aware rule scoring based on the partial completeness assumption (PCA) has been introduced. The idea of PCA is that whenever at least one object for a given subject and a predicate is in the KG (e.g., "Eduard is Einstein's child"), then all objects for that subject-predicate pair (Einstein's children) are assumed to be known. PCA relies on a hypothesis that the data is usually added to KGs in batches, i.e., if at least one child for a person has been added, then most probably all person's children are present in the KG. This assumption has turned out to be indeed valid in real-world KGs for some topics [30]. The PCA confidence is defined as follows:

$$conf_{pca}(r, \mathcal{G}) = \frac{r\text{-}supp(r, \mathcal{G})}{\#(x, y) : \exists \boldsymbol{z} : B \in \mathcal{G} \wedge \exists y' : h(x, y') \in \mathcal{G}}$$

However, the effectiveness of the PCA confidence decreases when applied on highly incomplete KGs as experienced in [34].

Example 14. Given the rules r'_1 and r'_2 from Example 13 and the KG from Figure 2, we have that $conf_{pca}(r'_1, \mathcal{G}') = \frac{4}{2}$. Indeed, since *carol* and *dave* are not known to have any children, four existing body substitutions are not counted in the denominator. Meanwhile, we have $conf_{pca}(r'_2, \mathcal{G}') = \frac{1}{6}$, since all people that are predicted to have siblings by r'_2 already have siblings in the available graph.

For the KG in Figure 1 and the earlier introduced rule r_1, it holds that $conf_{pca}(r_1, \mathcal{G}) = \frac{3}{4}$, since we do not know any living places of *lucy* and *dave*. □

Soft Confidence. *Soft confidence* measure introduced in [79], is also designed to work under OWA, and for a rule $r: \ h(X, Y) \leftarrow B$ it is formally defined as follows:

$$conf_{st}(r, \mathcal{G}) = \frac{r\text{-}supp(r, \mathcal{G})}{b\text{-}supp(r, \mathcal{G}) - \sum_{x \in U^r} Pr(x, h, \mathcal{G})}$$

where U^r is the set of entities that have no outgoing h-edges in $\mathcal{G}$, but do have some in $\mathcal{G}_r$, and $Pr(x, h, \mathcal{G})$ is the probability of the entity x having the relation h, approximated using entity *type* information [79]. More specifically,

$$Pr(x, h, \mathcal{G}) = max_{t \in T_x} \frac{|Inst_h(t, \mathcal{G})|}{|Inst(t, \mathcal{G})|}$$

where $T_x = \{t \,|\, t(x) \in \mathcal{G}\}$, $Inst(t, \mathcal{G}) = \{x' \,|\, t(x') \in \mathcal{G}\}$, and, moreover, $Inst_h(t, \mathcal{G}) = \{x' \in Inst(t, \mathcal{G}) \,|\, \exists\, x'' : h(x', x'') \in \mathcal{G}\}$. Intuitively, soft confidence is designed to avoid the under-fitting of standard confidence and over-fitting of PCA confidence by accounting for the probability of entities having certain relations.

Example 15. Consider the KG $\mathcal{G}$ from Figure 1, we have $U^{r_1} = \{lucy, dave\}$. Moreover, $Pr(dave, livesIn, \mathcal{G}) = \frac{1}{2}$, since *dave* has only 1 type *researcher*, and in total the KG stores 2 researchers (*dave*, *alice*) but only *alice*'s living place (*amsterdam*) is known to $\mathcal{G}$. In contrast, $Pr(lucy, livesIn, \mathcal{G}) = 0$, as *lucy* has no *type* information. Based on these numbers, we have $conf_{st}(r_1, \mathcal{G}) = \frac{3}{6 - \frac{1}{2}} = \frac{6}{11}$. □

RC Confidence. The authors of [82] have recently proposed the *RC confidence* as an attempt to rely on assumptions about the tuples not in the KG when evaluating the quality of a potential rule. The intuition behind *RC confidence* is that for computing rule's confidence one does not necessarily have to know which among rule predictions are true; just estimating their number is sufficient. The key assumption for such estimation is that the proportion of positive facts covered by a rule is the same for both true and unknown facts, which is reflected in the following *relationship coverage* defined for $r: \ h(X,Y) \leftarrow B$ as follows

$$\frac{r\text{-}supp(r,\mathcal{G})}{\#(x,y) : h(x,y) \in \mathcal{G}} = \frac{|UP(r,\mathcal{G})|}{\#(x,y) : h(x,y) \in \mathcal{G}^i \backslash \mathcal{G}}$$

where $UP(r,\mathcal{G}) = \mathcal{G}_r \cap \mathcal{G}^i$ (assumed to be 0 in the case of standard confidence). Intuitively, this equation would hold if the instantiations of r that appear in $\mathcal{G}$ were selected completely at random [23] from the set of all true facts for r.

To approximately determine the size of $UP(r,\mathcal{G})$ the authors rely on the proportion β of all the unlabeled examples that are true in $\mathcal{G}^i$, i.e., $UP(r,\mathcal{G}) = \beta * \#(x,y) : h(x,y) \notin \mathcal{G}$, and propose several ways for calculating β both empirically via sampling and theoretically based on properties of $\mathcal{G}$.

Finally, provided that there is a way to estimate $|UP(r,\mathcal{G})|$, the following formula is used for computing the *RC-confidence*:

$$conf_{rc}(r,\mathcal{G}) = \frac{r\text{-}supp(r,\mathcal{G}) + |UP(r,G)|}{b\text{-}supp(r,\mathcal{G})}$$

(In)completeness Metadata. In the solutions for the rule-based KG completion problem discussed so far, no external meta-information from outside of the KG about potential existence of certain types of facts was exploited. However, this knowledge is obviously useful, and furthermore, it can even be extracted from the Web in the form of cardinality statements, e.g., Brad has three children. If a given KG mentions just a single Brad's child, we could aim at extracting rules that predict the missing one.

Based on this intuition, the recent work on completeness-aware rule learning (CARL) [75] proposed improvements of rule scoring functions by making use of this additional (in)completeness metadata.

In particular, such metadata is presented using cardinality statements by reporting (the numerical restriction on) the absolute number of facts over a certain relation in the ideal graph $\mathcal{G}^i$. More specifically, the partial function *num* is defined that takes as input a predicate p and an entity x and outputs a natural number corresponding to the number of facts in $\mathcal{G}^i$ over p with x as the first argument:

$$num(p,x) = \#y : p(x,y) \in \mathcal{G}^i \tag{3}$$

These cardinality statements can be obtained via Web extraction techniques [46]. It is possible to rewrite cardinalities on the number of subjects for a given predicate and object with such statements provided that inverse relations can be

expressed in a KG. Naturally, the number of missing facts for a given p and x can be obtained as

$$miss(p, x, \mathcal{G}) = num(p, x) - \#y : p(x, y) \in \mathcal{G}$$

In the CARL framework, given a KG and its related cardinality statements of the above form, two indicators are defined for a given rule $r : h(X, Y) \leftarrow B$, reflecting the number of new predictions made by r in incomplete (npi) and, respectively, complete (npc) KG parts:

$$npi(r, \mathcal{G}) = \sum_x min(\#y : h(x, y) \in \mathcal{G}_r \backslash \mathcal{G}, miss(h, x, \mathcal{G}))$$

$$npc(r, \mathcal{G}) = \sum_x max(\#y : h(x, y) \in \mathcal{G}_r \backslash \mathcal{G} - miss(h, x, \mathcal{G}), 0)$$

Using these indicators, a class of **completeness-aware rule measures** have been defined in [75], which we briefly present next.

Completeness Confidence. First, incompleteness information is used to determine whether to consider an instance in the unknown part of the rule as a counterexample or not. Formally, the *completeness confidence* is defined as:

$$conf_{comp}(r, \mathcal{G}) = \frac{r\text{-}supp(r, \mathcal{G})}{b\text{-}supp(r, \mathcal{G}) - npi(r, \mathcal{G})}$$

Example 16. Consider $\mathcal{G}'$ in Figure 2 and cardinality statements for it:
$num(hC, john) = num(hC, mary) = 3$; $num(hC, alice) = 1$;
$num(hC, carol) = num(hC, dave) = 0$;
$num(hS, alice) = num(hS, carol) = num(hS, dave) = 2$;
$num(hS, bob) = 3$;
where hC, hS stand for $hasChild$ and $hasSibling$, respectively. We have:
$miss(hC, mary, \mathcal{G}') = miss(hC, john, \mathcal{G}') = miss(hC, alice, \mathcal{G}') = 1$;
$miss(hC, carol, \mathcal{G}') = miss(hC, dave, \mathcal{G}') = 0$;
$miss(hS, bob, \mathcal{G}') = miss(hS, carol, \mathcal{G}') = 2$;
$miss(hS, alice, \mathcal{G}') = miss(hS, dave, \mathcal{G}') = 1$;
For the rules r'_1 and r'_2 from Example 13 we have $conf_{comp}(r'_1, \mathcal{G}') = \frac{2}{6}$ and $conf_{comp}(r'_2, \mathcal{G}') = \frac{1}{2}$, which establishes the desired rule ranking. □

Completeness Precision and Recall. In the spirit of information retrieval, the notions of *completeness precision* and *completeness recall* are defined to measure the rule quality based on its predictions in complete and incomplete KG parts:

$$precision_{comp}(r, \mathcal{G}) = 1 - \frac{npc(r, \mathcal{G})}{b\text{-}supp(r, \mathcal{G})}, \quad recall_{comp}(r, \mathcal{G}) = \frac{npi(r, \mathcal{G})}{\sum_x miss(h, x, \mathcal{G})}$$

Intuitively, rules having high precision are rules that predict few facts in complete parts, while rules having high recall are rules that predict many facts in incomplete ones. Rule scoring could also be based on any weighted combination of these two metrics.

Example 17. We have $npi(r'_1, \mathcal{G}') = 2$, $npc(r'_1, \mathcal{G}') = 4$, while $npi(r'_2, \mathcal{G}') = 4$, $npc(r'_2, \mathcal{G}') = 1$, resulting in $precision_{comp}(r'_1, \mathcal{G}') = 0.5$, $recall_{comp}(r'_1, \mathcal{G}') \approx 0.67$, and $precision_{comp}(r'_2, \mathcal{G}') \approx 0.83$, $recall_{comp}(r'_2, \mathcal{G}') \approx 0.67$. □

Directional Metric. If rule mining does make use of completeness information, and both do not exhibit any statistical bias, then intuitively the rule predictions and the (in)complete areas should be statistically independent. On the other hand, correlation between the two indicates that the rule-mining is *(in)completeness-aware*. Following this intuition, the *directional metric* has been proposed, which measures the proportion between predictions in complete and incomplete parts as follows:

$$dm(r, \mathcal{G}) = \frac{npi(r, \mathcal{G}) - npc(r, \mathcal{G})}{2 \cdot (npi(r, \mathcal{G}) + npc(r, \mathcal{G}))} + 0.5$$

Since the real-world KGs are often highly incomplete, it might be reasonable to put more weight on predictions in complete parts. This is done via combining any existing association rule measure rm, e.g., standard confidence or conviction, with the directional metric, using a certain dedicated weighting factor $\gamma \in [0..1]$. Formally,

$$weighted\_dm(r, \mathcal{G}) = \gamma \cdot rm(r, \mathcal{G}) + (1 - \gamma) \cdot dm(r, \mathcal{G})$$

Example 18. We have $dm(r'_1, \mathcal{G}') \approx 0.33$ and $dm(r'_2, \mathcal{G}') = 0.8$. With weighted directional metric using standard confidence ($rm = conf$), for $\gamma = 0.5$, we get $weighted\_dm(r'_1, \mathcal{G}') \approx 0.29$ and $weighted\_dm(r'_2, \mathcal{G}') \approx 0.48$. □

Optimized Rule Evaluation. Most of state-of-the-art rule mining systems parallelize only the rule construction phase, while the quality of the rules are determined in a single thread. Huge size of KGs such as YAGO or Freebase prohibits their storage on a single machine. Therefore, the rule quality estimation is usually delegated to some third-party database management system. In practice this might not always be effective.

Unlike other state-of-the-art rule mining systems, the algorithm of *ontological pathfinding* (OP) [6] focuses on the efficient examination of the rule's quality. After candidate rules are constructed relying on a variation of [65], their quality is determined via a sequences of parallelization and optimization methods including KG partitioning, joining and pruning strategies.

5 Nonmonotonic Rule Learning

So far we have considered approaches for constructing Horn rules. However, these are not sufficiently expressive for representing incomplete human knowledge, and they are inadequate for capturing exceptions. Thus, Horn rules extracted from the existing KGs can potentially predict erroneous facts as shown in Section 1.

In this section, we provide an overview of approaches for learning nonmonotonic rules from large and incomplete KGs. First, we describe a method that relies on cross-talk among the extracted rules to guess their exceptions [29,76]. Then we present an approach that exploits embedding-based methods for knowledge graph completion during rule construction [34].

5.1 Revision-based Method

Exception handling has been traditionally faced in ILP by learning non-monotonic logic programs [35,66,64,8,41] (see Section 3). However, most of the existing methods assume that the data from which the rules are induced is complete.

In [29], a revision-based approach for extracting exception-enriched (i.e., nonmonotonic) rules from incomplete KGs has been proposed. It pre-processes a KG by projecting all binary facts into unary ones applying a form of propositionalization technique [39] (e.g., $livesIn(brad, berlin)$ could be translated to $livesInBerlin(brad)$ or $livesInCapital(brad)$ with obvious loss of information) and adapts data mining methods designed for transaction data to extract Horn rules, which are then augmented with negated atoms. In [76], the approach has been extended to KGs in their original relational form. We now briefly summarize the ideas of [29,76].

In these works, the KG completion problem from Definition 9 is treated as a *theory revision* task, where, given a KG and a set of (previously learned) Horn rules, the goal is to revise this set by adding exceptions, such that the obtained rules have higher predictive accuracy than the original ones.

Since normally, the ideal graph $\mathcal{G}^i$ is not available, in order to estimate the quality of a revised ruleset, two generic quality functions q_{rm} and $q_{conflict}$ are devised, which both take as input a ruleset R and a KG $\mathcal{G}$ and output a real value, reflecting the suitability of R for data prediction. More specifically,

$$q_{rm}(R, \mathcal{G}) = \frac{\sum_{r \in R} rm(r, \mathcal{G})}{|R|}, \tag{4}$$

where rm is some standard association rule measure [3]. Conversely, $q_{conflict}$ estimates the number of conflicting predictions that the rules in R generate. To measure $q_{conflict}$ for R, an extended set of rules R^{aux} is created, which contains every revised rule in R together with its auxiliary version. For a rule $r : h(X, Y) \leftarrow B, not\ \ E$ in R, its auxiliary version r^{aux} is constructed by: i) transforming r into a Horn rule by removing not from negated body atoms, and ii) replacing the head predicate h of r with a newly introduced predicate not_h which intuitively should contain instances which are not in h.

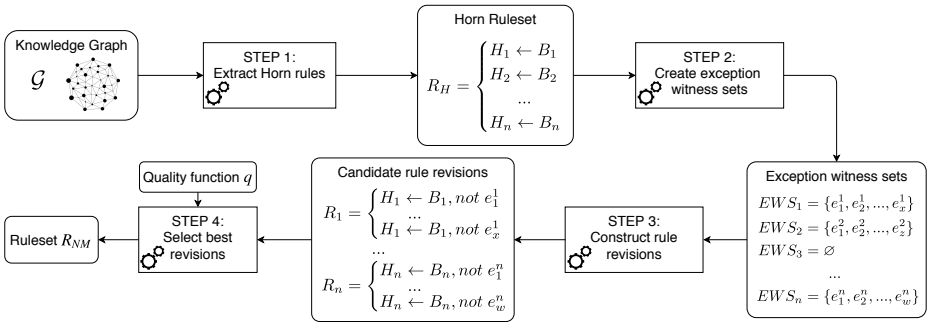

Fig. 3: Rule revision method for nonmonotonic rule learning [29,76].

Example 19. The auxiliary rule r_2^{aux} for the rule r_2 from Section 1 is as follows r_2^{aux} : $not\_livesIn(Y, Z) \leftarrow marriedTo(X, Y), livesIn(X, Z), researcher(Y)$, and it informally reflects that married people among whom one is a researcher often do not live together. For r_3 : $livesIn(X, Y) \leftarrow bornIn(X, Y), not\ \ immigrant(X)$ we similarly have r_3^{aux} : $not\_livesIn(X, Y) \leftarrow bornIn(X, Y), immigrant(X)$. If $R = \{r_1, r_2\}$ then $R^{aux} = \{r_1, r_1^{aux}, r_2, r_2^{aux}\}$.

Intuitively, $q_{conflict}(R, \mathcal{G})$ estimates the portion of conflicting predictions $livesIn(s, o), not\_livesIn(s, o)$ made by the rules in R^{aux}. The hypothesis of [29,76] is that for a set R of good rules with reasonable exceptions, the number of conflicting predictions produced by R^{aux} is small. □

Formally, based on statistics of both rules r and r^{aux} in a set of exception-enriched rules R, the measure $q_{conflict}$ is defined as

$$q_{conflict}(R, \mathcal{G}) = \sum_{p \in pred(R)} \frac{|\boldsymbol{c} \,|\, p(\boldsymbol{c}), not\_p(\boldsymbol{c}) \in \mathcal{G}_{R^{aux}}|}{|\boldsymbol{c} \,|\, not\_p(\boldsymbol{c}) \in \mathcal{G}_{R^{aux}}|} \tag{5}$$

where $pred(R)$ stores predicates appearing in R.

Formally, the problem targeted in [29,76] is formulated as follows:

Definition 12 (Quality-based Horn Theory Revision). *Given a KG $\mathcal{G}$, a set of nonground Horn rules R_H mined from $\mathcal{G}$, and a quality function rm, find a set of rules R_{NM} obtained by adding negated atoms to body(r) for some $r \in R_H$ s.t. (i) $q_{rm}(R_{NM}, \mathcal{G})$ is maximal, and (ii) $q_{conflict}(R_{NM}, \mathcal{G})$ is minimal.*

Given the huge size of KGs, finding the best possible revision is infeasible in practice. Thus, in [29,76] a heuristics-based method that computes an approximate solution to the above problem has been proposed (see Figure 3 for overview). It combines standard relational association rule mining techniques with a FOIL-like supervised learning algorithm [57] for computing exceptions. We now briefly discuss the steps of this approach.

Step 1. After mining Horn rules using an off-the-shelf algorithm (e.g., [30]), one computes for each rule the *normal* and *abnormal substitutions*, defined as

Definition 13 (r-(Ab)Normal Substitutions). *Let $\mathcal{G}$ be a KG, r a Horn rule mined from $\mathcal{G}$, and let $\mathbf{V}$ be a set of variables occurring in r. Then*

- $NS(r, \mathcal{G}) = \{\theta \mid head(r)\theta, body(r)\theta \subseteq \mathcal{G}\}$ *is an* r-normal set of substitutions*;*
- $ABS(r, \mathcal{G}) = \{\theta' \mid body(r)\theta' \subseteq \mathcal{G}, head(r)\theta' \notin \mathcal{G}\}$ *is an* r-abnormal set of substitutions*, where* $\theta, \theta' : \mathbf{V} \to \mathcal{C}$.

Example 20. For $\mathcal{G}$ from Figure 1 and r_1, we have $NS(r_1, \mathcal{G}) = \{\theta_1, \theta_2, \theta_3\}$, where $\theta_1 = \{X/brad, Y/ann, Z/berlin\}$, $\theta_2 = \{X/john, Y/kate, Z/chicago\}$ and $\theta_3 = \{X/sui, Y/li, Z/beijing\}$ respectively. Besides, among substitutions in $ABS(r_1, \mathcal{G})$, we have $\theta_4 = \{X/mat, Y/lucy, Z/amsterdam\}$, yet there are others. □

Step 2. Intuitively, if the given data was complete, then the r-normal and r-abnormal substitutions would exactly correspond to instances for which the rule r holds (respectively does not hold) in the real world. Since the KG is potentially incomplete, this is no longer the case and some r-abnormal instances might in fact be classified as such due to data incompleteness. In order to distinguish the "wrongly" and "correctly" classified instances in the r-abnormal set, one constructs *exception witness sets* (EWS), which are defined as follows:

Definition 14 (Exception Witness Set (EWS)). *Let $\mathcal{G}$ be a KG, let r be a rule mined from it, let $\mathbf{V}$ be a set of variables occurring in r and $\boldsymbol{X} \subseteq \mathbf{V}$. Exception witness set for r w.r.t. $\mathcal{G}$ and $\boldsymbol{X}$ is a maximal set of predicates $EWS(r, \mathcal{G}, \boldsymbol{X}) = \{e_1, \ldots, e_k\}$, s.t.*

- $e_i(\boldsymbol{X}\theta_j) \in \mathcal{G}$ *for some* $\theta_j \in ABS(r, \mathcal{G})$, $1 \leq i \leq k$ *and*
- $e_1(\boldsymbol{X}\theta'), \ldots, e_k(\boldsymbol{X}\theta') \notin \mathcal{G}$ *for all* $\theta' \in NS(r, \mathcal{G})$.

Example 21. For $\mathcal{G}$ in Figure 1 and rule r_1, we observe that $EWS(r, \mathcal{G}, Y) = \{researcher\}$ and $EWS(r, \mathcal{G}, X) = \{artist\}$. If *brad* with *ann* and *john* with *kate* did not live in metropolitan cities, then $EWS(r, \mathcal{G}, Z) = \{metropolitan\}$. □

In general, there are exponentially many possible EWSs to construct. Combinations of exception candidates could be an explanation for some missing KG edges, so the search space of solutions to the above problem is large. In [76] a restriction to a single atom as a final exception has been posed; extending exceptions to arbitrary combinations of atoms is left for future research.

Steps 3 and 4. After EWSs are computed for all rules in R_H, they are used to create potential revisions (Step 3), i.e., from every $e_j \in EWS(r_i, \mathcal{G})$ a revision r_i^j of r_i is constructed by adding a negated atom over e_j to the body of r_i. Finally, a concrete revision for every rule is determined, which constitutes a solution to the above problem (Step 4). To find such globally best ruleset revision R_{NM} many candidate combinations have to be checked, which due to the large size of $\mathcal{G}$ and EWSs might be too expensive. Thus, instead, R_{NM} is incrementally built by considering every $r_i \in R_H$ and choosing the locally best revision r_i^j for it.

In order to select r_i^j, four special ranking functions are introduced: a naive one and three more advanced functions, which exploit the concept of *partial*

$r_{e1} : writtenBy(X, Z) \leftarrow hasPredecessor(X, Y), writtenBy(Y, Z),$ **not** $american\_film(X)$
$r_{e2} : actedIn(X, Z) \leftarrow isMarriedTo(X, Y), directed(Y, Z),$ **not** $silent\_film\_actor(X)$
$r_{e3} : isPoliticianOf(X, Z) \leftarrow hasChild(X, Y), isPoliticianOf(Y, Z),$**not** $vicepresidentOfMexico(X)$

Fig. 4: Examples of the revised rules

materialization (***PM***). Intuitively, the idea behind it is to rank candidate revisions not based on $\mathcal{G}$, but rather on its extension with predictions produced by other (selectively chosen) rules (grouped into a set R'), thus ensuring a cross-talk among the rules. We now describe the ranking functions in more details.

- ***Naive*** ranker is the most straightforward ranking function. It prefers the revision r_i^j with the highest value of $rm(r_i^j, \mathcal{G})$ among all revisions of r_i.
- ***PM*** ranking function prefers r_i^j with the highest value of

$$\frac{rm(r_i^j, \mathcal{G}_{R'}) + rm(r_i^{j\,aux}, \mathcal{G}_{R'})}{2} \tag{6}$$

  where R' is the set of rules r'_k, which are rules from $R_H \backslash r_i$ with all exceptions from $EWS(r_k, \mathcal{G})$ incorporated at once. Informally, $\mathcal{G}_{R'}$ contains only facts that can be safely predicted by the rules from $R_H \backslash r_i$, i.e., there is no evident reason (candidate exceptions) to neglect their predictions.
- ***OPM*** is similar to ***PM***, but the selected ruleset R' contains only those rules whose Horn version appears above the considered rule r_i in the ruleset R_H, ordered (***O***) based on some chosen measure (e.g., the same as rm).
- ***OWPM*** is the most advanced ranking function. It differs from ***OPM*** in that the predicted facts in $\mathcal{G}_{R'} \backslash \mathcal{G}$ inherit weights (***W***) from the rules that produced them, and facts in $\mathcal{G}$ get the highest weight. These weights are taken into account when computing the value of (6). If the same fact is derived by multiple rules, the highest weight is stored. To avoid propagating uncertainty through rule chaining when computing weighted partial materialization of $\mathcal{G}$ predicted facts (i.e., derived by applying rules from R') are kept separately from the explicit facts (i.e., those in $\mathcal{G}$), and new facts are inferred using only $\mathcal{G}$.
  For reasoning over weighted facts existing probabilistic deductive tools such as ProbLog [14,26] can be used, but their exploitation is left for future work.

Example Rules. Figure 4 shows examples of rules obtained in [76] by the described method. For instance, r_{e1} extracted from IMDB states that movie plot writers stay the same throughout the sequel unless a movie is American, while r_{e2} reflects that spouses of movie directors often appear on the cast with the exception of actors of old silent movies. Finally, the rule r_{e3} learned from YAGO says that ancestors of politicians are also politicians in the same country with the exception of Mexican vice-presidents.

5.2 Method Guided by Embedding Models

In the work [34] introducing RuLES framework, an alternative method for non-monotonic rule induction has been proposed, which first constructs an approx-

imation of an ideal graph $\mathcal{G}^i$ and then learns rules by relying on it. For building such approximation, representations (i.e. embeddings) of entities and relations are learned from a given KG possibly enriched with additional information sources (e.g., text). We refer the reader to [78] for an overview of KG embedding models.

The RuLES approach [34] establishes a framework to benefit from the advantages of these models by iteratively inducing rules from a KG and collecting statistics about them from a precomputed embedding model to prune unpromising rule candidates.

Let $\mathcal{G}$ be a KG over the signature $\Sigma_{\mathcal{G}} = (\mathcal{R}, \mathcal{C})$. A *probabilistic KG* is a pair $\mathcal{P} = (\mathcal{G}, f)$, where $f : \mathcal{R} \times \mathcal{C} \times \mathcal{C} \to [0, 1]$ is a probability function over the facts over $\Sigma_{\mathcal{G}}$ such that for each atom $a \in \mathcal{G}$ it holds that $f(a) \geq ct$, where ct is a correctness threshold.

The goal of RuLES is to learn rules that do not only describe the available graph $\mathcal{G}$ well, but also predict highly probable facts based on the function f which relies on embeddings of the KG. For that, it utilizes a hybrid rule quality function:

$$\mu(r, \mathcal{P}) = (1 - \lambda) \times \mu_1(r, \mathcal{G}) + \lambda \times \mu_2(\mathcal{G}_r, \mathcal{P}).$$

where λ is a weight coefficient and μ_1 is any classical quality measure of r over $\mathcal{G}$ such that $\mu_1 : (r, \mathcal{G}) \mapsto \alpha \in [0, 1]$ (*e.g.*, *standard confidence* or *PCA confidence* [30]). μ_2 measures the quality of $\mathcal{G}_r$ (*i.e.*, the extension of $\mathcal{G}$ resulting from executing the rule r) based on $\mathcal{P}$ given that $\mu_2 : (\mathcal{G}_r, \mathcal{P}) \mapsto \alpha \in [0, 1]$. To this end, μ_2 is defined as the average probability of the newly predicted facts in $\mathcal{G}_r$:

$$\mu_2(\mathcal{G}_r, \mathcal{P}) = \frac{\Sigma_{a \in \mathcal{G}_r \setminus \mathcal{G}} f(a)}{|\mathcal{G}_r \setminus \mathcal{G}|}.$$

RuLES takes as input a KG, possibly a text corpus, and a set of user-specified parameters that are used to terminate rule construction. These parameters include an embedding weight λ, a minimum threshold for μ_1, a minimum rule support *r-supp* and other *rule-related* parameters such as a maximum number of positive and negative atoms allowed in $body(r)$. As illustrated in the overview of of the RuLES system in Figure 5, the KG and text corpus are used to train the embedding model that in turn is utilized to construct the probabilistic function f. The *Rule Learning* component computes the rules similar to [30] in an iterative fashion by applying refinement operators, starting from the head, by adding atoms to its body one after another until at least one of the termination criteria (that depend on f) is met. The set of refinement operators from [30] is extended by the following two to support negations in rule bodies:

- *add an exception instantiated atom:* add a binary negated atom with one of its arguments being a constant, and the other one being a shared variable.
- *add an exception closing atom:* add a binary negated atom to the rule with both of its arguments being shared variables.

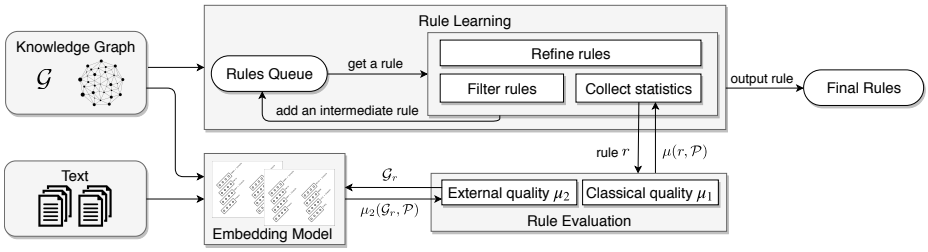

Fig. 5: Embedding-based method for nonmonotonic rule learning.

r_{e4} : $nationality(X, Y) \leftarrow graduatedFrom(X, Z), inCountry(Z, Y)$, **not** $researchUni(Z)$
r_{e5} : $nobleFamily(X, Y) \leftarrow spouse(X, Z), nobleFamily(Z, Y)$, **not** $chineseDynasties(Y)$

Fig. 6: Examples of the revised rules

During the construction of a rule r, the quality $\mu(r)$ is computed by the *Rule Evaluation* component, based on which a decision about the next action is made. Finally as an output, the RuLES system produces a set of nonmonotonic rules suitable for KG completion.

Note that the exploitation of the embedding feedback helps to distinguish exceptions from noise. Consider the rule r_1 stating that married people live together. This rule can have several possible exceptions, e.g., either one of the spouses is a researcher or he/she works at a company, which has headquarter in the US. Whenever the rule is enriched with an exception, naturally, the support of its body decreases, i.e., the size of $\mathcal{G}_r$ goes down. Ideally, such negated atoms should be added to the body of the rule that the average quality of $\mathcal{G}_r$ increases, as this witnesses that the addition of negated atoms to the rule body reduces unlikely predictions.

Example Rules. Figure 6 presents examples of rules learned by the RuLES system from the Wikidata KG. The first rule r_{e4} says that a person is a citizen of the country where his alma mater is located, unless it is a research institution, since most researchers in universities are foreigners. Additionally, r_{e5} encodes that someone belongs to a noble family if his/her spouse is also from the same noble family, excluding the Chinese dynasties.

6 Discussion and Outlook

In this tutorial, we have presented a brief overview of the current techniques for rule induction from knowledge graphs and have demonstrated how reasoning over KGs using the learned rules can be exploited for KG completion.

While the problem of rule-based KG completion has recently gained a lot of attention, several promising research directions are still left unexplored.

Learning Other Rule Forms. The majority of available methods focus on extracting Horn or nonmonotonic rules, yet inducing rules of other, more complex, forms would be beneficial. These include disjunctive rules (e.g., *"Having a sibling implies having a sister or a brother", "Korean speakers are normally either from*

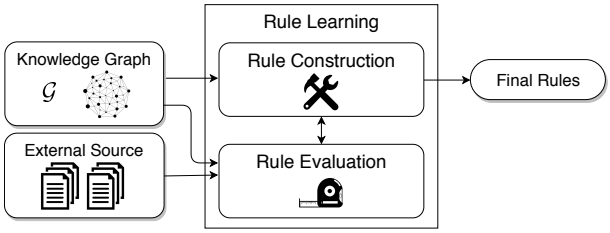

Fig. 7: Rule learning with external sources.

South or North Korea") or rules with existential quantifiers in the head (e.g., *"Being a musicians in a band, implies playing some musical instrument"*). Several recent works on mining keys in KGs [74,40], detecting mandatory relations [40] and learning SHACL constraints [56] are relevant, but they do not directly address the mentioned rule forms. A combination of techniques from relational rule learning [58] and propositionalization approaches [39] can be utilized for learning rules with existentials, yet it is unclear how to detect KG parts that are worth propositionalizing and combine the outputs of both methods.

Rules that reflect correlations between edge counts in KGs such as *"If a person has two siblings then his/her parents are likely to have 3 children"* have been studied in [75]. Inducing more general rules, encoding mathematical functions on edge counts (e.g., *"If a person has k siblings then his/her parents are likely to have k+1 children"*) and other numerical rules is still an open problem, which is particularly challenging due to large search space of possible hypothesis.

Learning temporal rules or constraints such as *"A person cannot graduate from a university before being born"* is another promising future work direction. Deductive reasoning over temporal KGs has been recently considered in [5]; however, the inductive setting has not yet been studied in full details to the best of our knowledge. A framework for learning hard boolean constraints has been described in [62], but its extension to KGs and soft constraints is still missing.

Learning Rules from Probabilistic Data. The majority of existing rule learning approaches over knowledge graphs model KGs as sets of true facts thus totally ignoring possible inaccuracies. Since KGs are usually constructed using (semi-)automatic methods from noisy textual resources, obviously not all of the extracted facts should have the same weight. Learning rules from such noisy KGs by treating all facts equally might naturally lead to problematic rules, which when being applied may propagate faulty facts.

A recent ILP approach that accounts for noise has been proposed in [24]; but it relies on CWA and neglects weights on the facts. Learning rules from KGs treated as uncertain data sources has been considered in [63,59,10]; however, these works neglect negation, disjunction or existential variables in the head. Extending the techniques to more advanced rule forms is beneficial.

Rule Learning with External Sources. Another interesting research stream is to consider pieces of evidence from hybrid external sources while inducing rules

from KGs (see Figure 7). Similar to [22], where external functions are utilized during deductive reasoning, various heterogeneous information sources can be used to guide rule induction. These range from a human expert giving feedback about the correctness of a given rule (similar as done in [20] for pattern mining), to dedicated fact-checking engines (e.g., Defacto [70], FactChecker [52]) that given a fact such as $bornIn(einstein, ulm)$ rely on Web documents to estimate its truthfulness.

Neural-based Rule Learning. Utilizing embedding models for rule learning is a new research direction that has recently gained attention [81,80]. Most of the existing methods are purely statistics-based, i.e., they reduce the rule learning problem to algebraic operations on neural-embedding-based representations of a given KG. The approach [80] constructs rules by modeling relation composition as multiplication or addition of two relation embeddings. The authors of [81] propose a differentiable system for learning models defined by sets of first-order rules that exploits a connection between inference and sparse matrix multiplication [7]. These techniques pose strong restrictions on target rule patterns, which often prohibits learning interesting rules, e.g., non-chain-like or exception-aware ones. Combining neural methods with symbolic ones in a similar way as in [34] but also accounting for rich background knowledge in the form of logical theories is expected to be advantageous for obtaining surprising insights from the data.

Another ambitious direction is to mimic the active learning framework of Angluin *et al.* [2] for hypothesis discovery by issuing dedicated queries to possibly text-enhanced KG embedding models instead of a human expert.

Extracting Rules Jointly from KGs and Text. While modern KGs are rich in facts and typically rather clean, they contain a limited set of encyclopedic relations (e.g., *bornIn*, *marriedTo*). On the other hand, textual resources certainly cover a richer set of predicates (e.g., *gotAquintedWith*, *celebratedWedding*), but suffer from noise. A natural way to address the above issues is to combine text-based rule extraction relying on natural language processing (NLP) and textual entailment techniques [68,33,17] with inductive rule learning from KGs. This interesting research direction comes with many challenging due to the heterogeneity of the input sources.

References

1. Freebase: an open, shared database of the world's knowledge. http://www.freebase.com/
2. Angluin, D.: Queries and concept learning. Machine Learning **2**(4), 319–342 (1987)
3. Azevedo, P.J., Jorge, A.M.: Comparing Rule Measures for Predictive Association Rules, pp. 510–517. Springer (2007)
4. Boytcheva, S.: Overview of inductive logic programming (ilp) systems (2007)
5. Chekol, M.W., Pirrò, G., Schoenfisch, J., Stuckenschmidt, H.: Marrying uncertainty and time in knowledge graphs. In: AAAI. pp. 88–94 (2017)
6. Chen, Y., Goldberg, S.L., Wang, D.Z., Johri, S.S.: Ontological pathfinding. In: SIGMOD. pp. 835–846. ACM (2016)

7. Cohen, W.W.: Tensorlog: A differentiable deductive database. CoRR **abs/1605.06523** (2016)
8. Corapi, D., Russo, A., Lupu, E.: Inductive logic programming as abductive search. In: ICLP. pp. 54–63 (2010)
9. Corapi, D., Russo, A., Lupu, E.: Inductive logic programming in answer set programming. In: Inductive Logic Programming. pp. 91–97. Springer (2012)
10. Corapi, D., Sykes, D., Inoue, K., Russo, A.: Probabilistic rule learning in nonmonotonic domains. In: CLIMA. LNCS (2011)
11. d'Amato, C., Staab, S., Tettamanzi, A.G., Minh, T.D., Gandon, F.: Ontology enrichment by discovering multi-relational association rules from ontological knowledge bases. In: SAC. pp. 333–338 (2016)
12. Darari, F., Nutt, W., Pirrò, G., Razniewski, S.: Completeness statements about RDF data sources and their use for query answering. In: ISWC. pp. 66–83 (2013)
13. De Raedt, L., Bruynooghe, M.: CLINT : a multi-strategy interactive concept-learner and theory revision system. In: Michalski, R., Tecuci, G. (eds.) Proceedings of the Multi-Strategy Learning Workshop. pp. 175–191 (1991)
14. De Raedt, L., Kimmig, A., Toivonen, H.: Problog: A probabilistic prolog and its application in link discovery. In: IJCAI. pp. 2468–2473 (2007)
15. Dehaspe, L., De Raedt, L.: Mining association rules in multiple relations. In: Lavrac, N., Dzeroski, S. (eds.) Inductive Logic Programming. vol. 1297, pp. 125–132 (1997)
16. Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., Zhang, W.: Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 601–610. ACM (2014)
17. Dragoni, M., Villata, S., Rizzi, W., Governatori, G.: Combining nlp approaches for rule extraction from legal documents. In: 1st Workshop on MIning and REasoning with Legal texts (MIREL) (2016)
18. Duc Tran, M., d'Amato, C., Nguyen, B.T., Tettamanzi, A.G.B.: Comparing rule evaluation metrics for the evolutionary discovery of multi-relational association rules in the semantic web. In: Genetic Programming. pp. 289–305 (2018)
19. Dzeroski, S., Lavrac, N.: Learning relations from noisy examples: An empirical comparison of linus and foil. In: ML (1991)
20. Dzyuba, V., van Leeuwen, M.: Learning what matters – sampling interesting patterns. In: Advances in Knowledge Discovery and Data Mining. pp. 534–546 (2017)
21. Eiter, T., Ianni, G., Krennwallner, T.: Answer set programming: A primer. In: Reasoning Web. vol. 5689, pp. 40–110 (2009)
22. Eiter, T., Kaminski, T., Redl, C., Schüller, P., Weinzierl, A.: Answer set programming with external source access. In: Reasoning Web Summer School. pp. 204–275 (2017)
23. Elkan, C., Noto, K.: Learning classifiers from only positive and unlabeled data. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 213–220 (2008)
24. Evans, R., Grefenstette, E.: Learning explanatory rules from noisy data. J. Artif. Intell. Res. **61**, 1–64 (2018)
25. Faber, W., Pfeifer, G., Leone, N.: Semantics and complexity of recursive aggregates in answer set programming. Artif. Intell. **175**(1), 278–298 (2011)
26. Fierens, D., den Broeck, G.V., Renkens, J., Shterionov, D.S., Gutmann, B., Thon, I., Janssens, G., Raedt, L.D.: Inference and learning in probabilistic logic programs using weighted boolean formulas. TPLP **15**(3), 358–401 (2015)

27. Fürnkranz, J., Gamberger, D., Lavrac, N.: Foundations of Rule Learning. Cognitive Technologies, Springer (2012)
28. Fürnkranz, J., Kliegr, T.: A brief overview of rule learning. In: RuleML. pp. 54–69 (2015)
29. Gad-Elrab, M.H., Stepanova, D., Urbani, J., Weikum, G.: Exception-enriched rule learning from knowledge graphs. In: ISWC. pp. 234–251 (2016)
30. Galarraga, L., Teflioudi, C., Hose, K., Suchanek, F.M.: Fast rule mining in ontological knowledge bases with AMIE+. In: VLDB. vol. 24, pp. 707–730 (2015)
31. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Proc. of 5th Int. Conf. and Symp. on Logic Programming, ICLP 1988. pp. 1070–1080 (1988)
32. Goethals, B., den Bussche, J.V.: Relational association rules: Getting warmer. In: PDD (2002)
33. Gordon, J., Schubert, L.K.: Discovering commonsense entailment rules implicit in sentences. In: TextInfer Workshop on Textual Entailment. pp. 59–63. TIWTE '11 (2011)
34. Ho, V.T., Stepanova, D., Gad-Elrab, M.H., Kharlamov, E., Weikum, G.: Rule learning from knowledge graphs guided by embedding models. ISWC2018 (2018), in print
35. Inoue, K., Kudoh, Y.: Learning extended logic programs. In: IJCAI. pp. 176–181. Morgan Kaufmann (1997)
36. Józefowska, J., Lawrynowicz, A., Lukaszewski, T.: The role of semantics in mining frequent patterns from knowledge bases in description logics with rules. TPLP **10**(3), 251–289 (2010)
37. Katzouris, N., Artikis, A., Paliouras, G.: Incremental learning of event definitions with inductive logic programming. Machine Learning **100**(2-3), 555–585 (2015)
38. Klyne, G., Carroll, J.J.: Resource description framework (rdf): Concepts and abstract syntax. W3C Recommendation (2004)
39. Krogel, M., Rawles, S.A., Železný, F., Flach, P.A., Lavrac, N., Wrobel, S.: Comparative evaluation of approaches to propositionalization. In: ILP. pp. 197–214 (2003)
40. Lajus, J., Suchanek, F.M.: Are all people married?: Determining obligatory attributes in knowledge bases. In: WWW. pp. 1115–1124. ACM (2018)
41. Law, M., Russo, A., Broda, K.: The ILASP system for learning answer set programs. https://www.doc.ic.ac.uk/~ml1909/ILASP (2015)
42. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web **6**, 167–195 (2015)
43. Lisi, F.A.: Inductive Logic Programming in Databases: From Datalog to DL+log. TPLP **10**(3), 331–359 (2010)
44. Miller, G.A.: Wordnet: A lexical database for english. Commun. ACM **38**(11), 39–41 (Nov 1995)
45. Mirza, P., Razniewski, S., Darari, F., Weikum, G.: Cardinal virtues: Extracting relation cardinalities from text. ACL (2017)
46. Mirza, P., Razniewski, S., Nutt, W.: Expanding wikidata's parenthood information by 178%, or how to mine relation cardinality information. In: ISWC 2016 Posters & Demos (2016)
47. Mitchell, T., Cohen, W., Hruschka, E., Talukdar, P., Betteridge, J., Carlson, A., Dalvi, B., Gardner, M., Kisiel, B., Krishnamurthy, J., Lao, N., Mazaitis, K., Mohamed, T., Nakashole, N., Platanios, E., Ritter, A., Samadi, M., Settles, B., Wang,

R., Wijaya, D., Gupta, A., Chen, X., Saparov, A., Greaves, M., Welling, J.: Never-Ending Learning. In: AAAI. pp. 2302–2310 (2015)
48. Morik, K.: Balanced cooperative modeling. Machine Learning **11**(2), 217–235 (May 1993)
49. Muggleton, S.: Inductive logic programming. New Generation Comput. **8**(4), 295–318 (1991)
50. Muggleton, S., Buntine, W.L.: Machine invention of first order predicates by inverting resolution. In: International Conference on Machine Learning. pp. 339–352 (1988)
51. Muggleton, S., Feng, C.: Efficient induction of logic programs. In: Algorithmic Learning Theory Workshop. pp. 368–381 (1990)
52. Nakashole, N., Mitchell, T.M.: Language-aware truth assessment of fact candidates. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). vol. 1, pp. 1009–1019 (2014)
53. Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs: From multi-relational link prediction to automated knowledge graph construction. CoRR (2015)
54. Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs. IEEE **104**(1), 11–33 (2016)
55. Paulheim, H.: Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic Web **8**(3), 489–508 (2017)
56. Paulheim, H.: Learning shacl constraints for validation of relation assertions in knowledge graphs. In: ESWC. p. to appear (2018)
57. Quinlan, J.R.: Learning logical definitions from relations. Machine Learning **5**, 239–266 (1990)
58. Raedt, L.D.: Logical and relational learning. Cognitive Technologies, Springer (2008)
59. Raedt, L.D., Dries, A., Thon, I., den Broeck, G.V., Verbeke, M.: Inducing probabilistic relational rules from probabilistic examples. In: IJCAI. pp. 1835–1843. AAAI Press (2015)
60. Raedt, L.D., Dzeroski, S.: First-order jk-clausal theories are pac-learnable. Artif. Intell. **70**(1-2), 375–392 (1994)
61. Raedt, L.D., Lavrac, N., Dzeroski, S.: Multiple predicate learning. In: Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France, August 28 - September 3, 1993. pp. 1037–1043 (1993)
62. Raedt, L.D., Passerini, A., Teso, S.: Learning constraints from examples. In: AAAI (2018)
63. Raedt, L.D., Thon, I.: Probabilistic rule learning. In: ILP (2010)
64. Ray, O.: Nonmonotonic abductive inductive learning. Journal of Applied Logic **7**(3), 329 – 340 (2009), special Issue: Abduction and Induction in Artificial Intelligence
65. Richards, B.L., Mooney, R.J.: Learning relations by pathfinding. In: Proceedings of the 10th National Conference on Artificial Intelligence. pp. 50–55 (1992)
66. Sakama, C.: Induction from answer sets in nonmonotonic logic programs. ACM Trans. Comput. Log. **6**(2), 203–231 (2005)
67. Sazonau, V., Sattler, U.: Mining hypotheses from data in OWL: advanced evaluation and complete construction. In: International Semantic Web Conference (1). vol. 10587, pp. 577–593 (2017)
68. Schoenmackers, S., Etzioni, O., Weld, D.S., Davis, J.: Learning first-order horn clauses from web text. In: EMNLP. pp. 1088–1098 (2010)

69. Shapiro, E.Y.: Algorithmic Program DeBugging. Cambridge, MA, USA (1983)
70. Speck, R., Esteves, D., Lehmann, J., Ngonga Ngomo, A.C.: Defacto - a multilingual fact validation interface. In: ISWC (2015)
71. Srinivasan, A.: The aleph manual. http://www.cs.ox.ac.uk/activities/machlearn/Aleph/aleph.html
72. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: A Core of Semantic Knowledge. In: Proceedings of WWW. pp. 697–706 (2007)
73. Suchanek, F.M., Preda, N.: Semantic culturomics. VLDB **7**(12), 1215–1218 (2014)
74. Symeonidou, D., Galárraga, L., Pernelle, N., Saïs, F., Suchanek, F.M.: VICKEY: mining conditional keys on knowledge bases. In: ISWC. pp. 661–677 (2017)
75. Tanon, T.P., Stepanova, D., Razniewski, S., Mirza, P., Weikum, G.: Completeness-aware rule learning from knowledge graphs. In: ISWC. pp. 507–525 (2017)
76. Tran, H.D., Stepanova, D., Gad-Elrab, M.H., Lisi, F.A., Weikum, G.: Towards non-monotonic relational learning from knowledge graphs. In: ILP. pp. 94–107 (2016)
77. Vrandecic, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. CACM **57**(10), 78–85 (2014)
78. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: A survey of approaches and applications pp. 2724–2743 (2017)
79. Wang, Z., Li, J.: Rdf2rules: Learning rules from RDF knowledge bases by mining frequent predicate cycles. CoRR **abs/1512.07734** (2015)
80. Yang, B., Yih, W., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. CoRR **abs/1412.6575** (2014)
81. Yang, F., Yang, Z., Cohen, W.W.: Differentiable learning of logical rules for knowledge base reasoning. In: NIPS. pp. 2316–2325 (2017)
82. Zupanc, K., Davis, J.: Estimating rule quality for knowledge base completion with the relationship between coverage assumption. In: WWW. pp. 1073–1081 (2018)