Ontology Enrichment by Discovering Multi-Relational Association Rules from Ontological Knowledge Bases

Claudia d'Amato
Computer Science Department
University of Bari, Italy
claudia.damato@uniba.it

Steffen Staab
University of Southampton, UK & University of Koblenz-Landau, Germany
staab@uni-koblenz.de

Andrea G. B. Tettamanzi
Univ. Nice Sophia Antipolis, I3S, France
andrea.tettamanzi@unice.fr

Tran Duc Minh
Univ. Nice Sophia Antipolis, I3S, France
tdminh2110@yahoo.com

Fabien Gandon
INRIA Sophia Antipolis, France
Fabien.Gandon@inria.fr

ABSTRACT

In the Semantic Web context, OWL ontologies represent the conceptualization of domains of interest while the corresponding assertional knowledge is given by the heterogeneous Web resources referring to them. Being strongly decoupled, ontologies and assertion can be out-of-sync. An ontology can be incomplete, noisy and sometimes inconsistent with regard to the actual usage of its conceptual vocabulary in the assertions. Data mining can support the discovery of hidden knowledge patterns in the data, to enrich the ontologies. We present a method for discovering multi-relational association rules, coded in SWRL, from ontological knowledge bases. Unlike state-of-the-art approaches, the method is able to take the intensional knowledge into account. Furthermore, since discovered rules are represented in SWRL, they can be straightforwardly integrated within the ontology, thus (i) enriching its expressive power and (ii) augmenting the assertional knowledge that can be derived. Discovered rules may also suggest new axioms to be added to the ontology. We performed experiments on publicly available ontologies validating the performances of our approach.

CCS Concepts

•**Computing Methodologies** → **Artificial Intelligence;** •**Artificial Intelligence** → *Knowledge Representation and Reasoning;*

Keywords

Description Logics; Pattern Discovery

1. INTRODUCTION

Data, information, and knowledge on the Semantic Web (SW) are connected following best practices and exploiting standard Web technologies, e.g. HTTP, RDF and URIs. This allows to share and link information that can be read automatically by computers [4, 5]. A key aspect is the description of data/resources in terms of controlled vocabularies, namely ontologies, which formally define the meaning of data/resources, and support powerful deductive reasoning capabilities. The Linked Open Data (LOD) cloud[1] is a collection of interlinked open datasets (e.g. DBPedia, GeoNames, FOAF) published according to these best practices. LOD could be seen as a large source of assertional knowledge, whose intensional part is formally defined by existing formal RDFS[2]/OWL[3] ontologies. However, due to the heterogeneous and distributed nature of the SW, ontological knowledge bases (KBs)[4] may turn out to be incomplete and noisy w.r.t. the domain of interest. An ontology is incomplete when it is logically consistent (i.e., it contains no contradiction) but it lacks information (e.g., assertions, disjointness axioms, etc.) w.r.t. the domain of reference; an ontology is noisy when it is consistent but it contains invalid information w.r.t. the reference domain. This may prevent the inference of relevant information or cause incorrect information to be derived. Data mining techniques could be exploited for discovering hidden knowledge patterns from ontological KBs, to be used for enriching an ontology both at terminological (schema) and assertional (facts) level. We present a method for discovering hidden knowledge patterns in the form of multi-relational association rules (ARs) coded in SWRL [12], which can be added to the ontology enriching its expressive power and increasing the assertional knowledge that can be derived. Additionally, discovered rules may suggest new axioms to be added to the ontology, such as transitivity and symmetry of a role, and/or concept/role inclusion axioms. First works for mining hidden knowledge patterns in the SW [14, 13] proposed solutions for discovering DATALOG clauses and conjunctive queries from hybrid sources of knowledge (a rule set and an ontology). These methods are grounded on a notion of *key*, standing for the basic attribute to be used for counting elements for building the frequent patterns. Unlike these methods, the proposed solution focuses on an ontological KB and does not require any notion of *key*. A method for learning ARs from RDF datasets is proposed in [16],

SAC 2016, April 4-8, 2016, Pisa, Italy
ACM. 0-12345-67-8/90/01…$15.00
DOI: http://dx.doi.org/10.1145/2851613.2851842

[1] http://lod-cloud.net/

[2] http://www.w3.org/TR/rdf-schema/

[3] http://www.w3.org/TR/owl-features/

[4] By *ontological knowledge base*, we refer to a populated ontology, that is both schema and instance level are specified. The expression will be interchangeably used with the term ontology.

while a method for inducing new assertional knowledge from RDF datasets is presented in [9]. Differently from our approach, these methods do not consider any background/ontological knowledge and do not exploit any reasoning capabilities. Furthermore, our rules can be directly added to the ontology. Our solution is experimentally evaluated and comparisons with the main state-of-the art system are provided. In the next section, basics are illustrated; the proposed method is presented in Sect. 3 and experimental evaluation is given in Sect. 4. Conclusions are drawn in Sect. 5.

2. BASICS

We refer to ontological KBs described in Description Logics (DLs) [3], which are the theoretical foundation of OWL. A DL KB $\mathcal{K}$ consists of a set of *axioms*, of two kinds: terminological (TBox) $\mathcal{T}$ and assertional (ABox) $\mathcal{A}$. The formal meaning of DL axioms is given in terms of model-theoretic semantics. Common ABox axioms are *concept assertions* of the form $C(a)$ and *role assertions* e.g. $R(a, b)$, where C is a concept name, R is a role name, and a, b are individual names[5]. DLs are endowed with deductive reasoning capabilities such as: *instance checking*, assessing if an individual is instance of a given concept; and *concept subsumption*. In the following the general definition of relational AR for an ontological KB is given. Hence, the problem we want to address is defined.

DEFINITION 1 (RELATIONAL ASSOCIATION RULE). *Given a populated ontological KB $\mathcal{K}$= ($\mathcal{T}, \mathcal{A}$), a* relational association rule *r for $\mathcal{K}$ is a Horn-like clause of the form: $body \rightarrow head$, where: (a) body is a generalization of a set of assertions in $\mathcal{K}$ co-occurring together; (b) $head$ is a consequent that is induced from $\mathcal{K}$ and body*

DEFINITION 2 (PROBLEM DEFINITION). **Given:**

- *a populated ontological knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$;*
- *a minimum "frequency threshold", θ_f;*
- *a minimum "head coverage threshold", θ_{hc};*
- *a minimum "confidence improvement threshold", θ_{ic};*

Discover: *all frequent hidden patterns w.r.t θ_f, in the form of multi-relational ARs, that may induce new assertions for $\mathcal{K}$.*

Intuitively, a *frequent hidden pattern* is a generalization of a set of concept/role assertions co-occurring reasonably often (w.r.t. a fixed frequency threshold) together, showing an underlying form of correlation that is exploited for obtaining new assertions.

2.1 Representation Language

For representing the rules to be discovered (following Def. 2), we adopt the Semantic Web Rule Language (SWRL) [12], extending the set of OWL axioms of a given ontology with Horn-like rules[6].

DEFINITION 3 (SWRL RULE). *Given a KB $\mathcal{K}$, a SWRL rule is an implication between an antecedent (body) and a consequent (head) of the form: $B_1 \wedge B_2 \wedge \ldots B_n \rightarrow H_1 \wedge \cdots \wedge H_m$, where $B_1 \wedge \cdots \wedge B_n$ is the rule body and $H_1 \wedge \cdots \wedge H_m$ is the rule head. Each $B_1, \ldots, B_n, H_1, \ldots H_m$ is called* atom. *An* atom *is a unary or binary predicate of the form $P_c(s)$, $P_r(s_1, s_2)$, $\mathsf{sameAs}(s_1, s_2)$ or $\mathsf{differentFrom}(s_1, s_2)$, where the predicate symbol P_c is a concept name in $\mathcal{K}$, P_r is a role name in $\mathcal{K}$, s, s_1, s_2 are* terms. *A* term *is either a variable (denoted by x, y, z) or a constant (denoted by a, b, c) standing for an individual name or data value.*

[5]In OWL, concepts and roles are called classes and properties.

[6]The results is a KB with an enriched expressive power. More complex relationships than subsumption can be expressed. For details see [11].

The discovered rules can be generally called *multi-relational* rules since multiple binary predicates $P_r(s_1, s_2)$ with different role names of $\mathcal{K}$ could appear in a rule. The intended meaning of a rule is: whenever the conditions in the antecedent hold, the conditions in the consequent must also hold. A rule having more than one atom in the head can be equivalently transformed, due to the *safety condition* (see Def. 4), into multiple rules, each one having the same body and a single atom in the head. We will consider, w.l.o.g., only SWRL rules (hereafter just "rules") with one atom in the head.

2.2 Fixing the Language Bias

In this section, the adopted *language bias* is specified. We manage rules having only atomic concepts and/or role names of $\mathcal{K}$ as predicate symbols, and individual names as constants. Only *connected* [9] and non-redundant [13] rules satisfying the *safety condition* [11] are considered. Additionally, to guarantee decidability, only *DL-safe rules* are managed [15], that is rules interpreted under the DL-safety condition consisting in binding all variables in a rule only to explicitly named individuals in $\mathcal{K}$[7]. In the following, notations and formal definitions for the listed properties are reported.

Given an atom A, let $T(A)$ denote the set of all the terms occurring in A and let $V(A) \subseteq T(A)$ denote the set of all the variables occurring in A e.g. $V(C(x)) = \{x\}$ and $V(R(x, y)) = \{x, y\}$. Such notation may be extended to rules straightforwardly.

DEFINITION 4 (SAFETY CONDITION). *Given a KB $\mathcal{K}$ and a rule $r = B_1 \wedge B_2 \wedge \ldots B_n \rightarrow H$, r satisfies the* safety condition *if all variables appearing in the rule head also appear in the rule body; formally if: $V(H) \subseteq \bigcup_{i=1}^{n} V(B_i)$,*

DEFINITION 5 (CONNECTED RULE). *Given a KB $\mathcal{K}$ and a rule $r = B_1 \wedge B_2 \wedge \ldots B_n \rightarrow H$, r is* connected *iff every atom in r is transitively connected to every other atom in r.*
Two atoms B_i and B_j in r, with $i \neq j$, are connected *if they share at least a variable or a constant i.e. if $T(B_i) \cap T(B_j) \neq \emptyset$.*
Two atoms B_1 and B_k in r are transitively connected *if there exist in r, atoms $B_2, \ldots, B_{k-1}$, with $k \leq n$, such that, for all $i, j \in \{1, \ldots, k\}$ with $i \neq j$, $T(B_i) \cap T(B_j) \neq \emptyset$.*

DEFINITION 6 (NON-REDUNDANT RULE). *Given a KB $\mathcal{K}$ and a rule $r = B_1 \wedge B_2 \wedge \ldots B_n \rightarrow H$, r is a non-redundant rule if no atom in r is entailed by other atoms in r with respect to $\mathcal{K}$, i.e., if, $\forall i \in \{0, 1, \ldots, n\}$, with $B_0 = H$, results: $\bigwedge_{j \neq i} B_j \not\models_{\mathcal{K}} B_i$,*

EXAMPLE 1 (REDUNDANT RULE). *Given $\mathcal{K}$ with $\mathcal{T} = \{\mathsf{Father} \sqsubseteq \mathsf{Parent}\}$ and the rule $r = \mathsf{Father}(x) \wedge \mathsf{Parent}(x) \rightarrow \mathsf{Human}(x)$ where* Human *is a primitive concept, r is redundant since the atom $\mathsf{Parent}(x)$ is entailed by the atom $\mathsf{Father}(x)$ with respect to $\mathcal{K}$.*

2.3 Metrics for Rules Evaluation

For determining the rules of interest for the goal in Def. 2, metrics for assessing the quality of a rule are necessary. In the following, the adopted metrics are summarized.

Given a rule $r = B_1 \wedge \ldots \wedge B_n \rightarrow H$, let us denote:

- $\Sigma_H(r)$ the set of distinct bindings of the variables occurring in the head of r, formally: $\Sigma_H(r) = \{binding\, V(H)\}$
- $E_H(r)$ the set of distinct bindings of the variables occurring in the head of r provided that the body and the head of r are satisfied, formally: $E_H(r) = \{binding\, V(H) \mid \exists\, binding\, V(B_1 \wedge \cdots \wedge B_n) : B_1 \wedge \cdots \wedge B_n \wedge H\}$.
  Since rules are connected, $V(H) \subseteq V(B_1 \wedge \cdots \wedge B_n)$

[7]When added to an ontology, DL-safe rules are decidable and generate sound results but not necessarily complete.

- $M_H(r)$ the set of distinct bindings of the variables occurring in the head of r also appearing as binding for the variables occurring in the body of r, formally: $M_H(r) = \{binding\, V(H) \mid \exists\, binding\, V(B_1 \wedge \cdots \wedge B_n) : B_1 \wedge \cdots \wedge B_n\}$

Following [9], and differently from the classic definitions (as used in [1]) for ensuring monotonicity, they are recalled in the following.

DEFINITION 7 (RULE SUPPORT). *Given a rule $r = B_1 \wedge \ldots \wedge B_n \rightarrow H$, its support is given by the number of distinct bindings of the variables in the head, formally:* $\mathrm{supp}(r) = |E_H(r)|$.

DEFINITION 8 (HEAD COVERAGE FOR A RULE). *Given a rule $r = B_1 \wedge \ldots \wedge B_n \rightarrow H$, its head coverage is given by the proportion of the distinct variable bindings from the head of the rule that are covered by the predictions of the rule:*

$$\mathrm{headCoverage}(r) = |E_H(r)|/|\Sigma_H(r)|$$

DEFINITION 9 (RULE CONFIDENCE). *Given a rule $r = B_1 \wedge \ldots \wedge B_n \rightarrow H$, its confidence is defined as the ratio of the binding of the predicting variables in the rule head and their binding in the rule body:* $\mathrm{conf}(r) = |E_H(r)|/|M_H(r)|$

An issue with these definitions, and particularly Def. 9, is that an implicit closed-world assumption is made, since no distinction between *false* predictions, i.e., bindings σ matching r such that $\mathcal{K} \models \neg H\sigma$, and *unknown* predictions, i.e., bindings σ matching r such that both $\mathcal{K} \models H\sigma$ and $\mathcal{K} \models \neg H\sigma$, is made. On the contrary, reasoning on ontologies is grounded on the open-world assumption. Additionally, our goal is to maximize correct predictions, not just describing the available data. To circumvent this limitation the following metric, generalizing the *PCA Confidence* [9], is introduced.

DEFINITION 10 (RULE PRECISION). *Given a rule $r = B_1 \wedge \ldots \wedge B_n \rightarrow H$, its precision is given by the ratio of the number of correct predictions made by r and the total number of correct and incorrect predictions (predictions logically contradicting $\mathcal{K}$), leaving out the predictions with unknown truth value.*

This metric expresses the ability of a rule to perform correct predictions, but it is not able to take into account the induced knowledge, that is the *unknown* predictions. For this reason, the metrics proposed for this purpose in [8] are also considered for the evaluation in Sect. 4. They are recalled in the following:

- *match rate*: number of predicted assertions in agreement with facts in the complete ontology, out of all predictions;
- *commission error rate*: number of predicted assertions contradicting facts in the full ontology, out of all predictions;
- *induction rate*: number of predicted assertions that are not known (i.e., for which there is no information) in the complete ontology, out of all predictions.

3. DISCOVERING RELATIONAL ASSOCIATION RULES FROM ONTOLOGIES

Given a populated ontological KB, our goal is to discover frequent hidden patterns in the form of multi-relational ARs to be exploited for making predictions of new assertions in the KB. The discovered rules are DL-Safe and expressed in SWRL (see Sect. 2). Hence, they can be integrated with the existing ontology, resulting in a KB with an enriched expressive power [11, 12]. For reaching this goal, we propose an algorithm grounded on the general framework for discovering frequent DATALOG patterns (i.e., conjunctive DATALOG queries) [6, 7, 10], successively adapted in [9] for discovering relational ARs from RDF datasets. To the best of our knowledge, our method represents the first work for discovering multi-relational ARs from an ontological KB that is able to take into account terminological axioms and deductive reasoning capabilities.

Algorithm 1 Discover multi-relational ARs from a populated ontological KB.

Input: $\mathcal{K}$: ontological KB; θ_f: frequency threshold; θ_{hc}: *head coverage* threshold;
Output: *frequent*: set of frequent patterns discovered from $\mathcal{K}$

```
1: infrequent ← ∅;    frequent ← ∅
2: q ← CREATEGENERALPATTERNS(K, θ_f)
3: while ¬q.isEmpty() do
4:     p ← q.dequeue()
5:     specPatternList ← GENERATESPECIALIZEDPATTERNS(p)
6:     specializationAdded ← false
7:     for all p' ∈ specPatternList do
8:         pruned ← EVALUATEPATTERNFORPRUNING(K, p, p', q, infrequent)
9:         if pruned then
10:            infrequent ← infrequent ∪ {p'}
               {Stopping criterion for the specialization process of a single pattern}
11:        else if p'.length() < MAX_LENGTH then
12:            q.enqueue(p')   {p' is enqueued for further specialization steps}
13:            specializationAdded ← true
14:        else if ISSAFE(p'.asRule()) then
15:            frequent ← frequent ∪ {p'} {p' has the max length and is saved as a
               discovered frequent pattern}
16:            specializationAdded ← true
17:    {If all specializations are pruned the parent pattern is saved as frequent}
18:    if (¬specializationAdded) ∧ (p.length() ≥ 2) then
19:        p_safe ← GETSAFEPATTERNORANCESTORPATTERN(p)
20:        if p_safe ≠ null ∧ p_safe ∉ frequent then
21:            frequent ← frequent ∪ {p_safe}
22: return frequent
```

This allows pruning rules that may be inconsistent, when considered jointly with the KB, and taking into account additional (derived) information, which would not been considered otherwise.

3.1 The Algorithm

The process comprises two main phases [1, 2]: (1) discover all possible frequent patterns (the most expensive computation); (2) obtain relational ARs from the discovered frequent patterns. As in [9], a pattern is represented as a list of atoms to be interpreted in conjunctive form. For each discovered frequent pattern, a multi-relational AR is obtained (Phase 2) by considering the first atom in the list as the head and the remaining atoms as the body.

For discovering frequent patterns, we implement a level-wise *generate-and-test* approach. The ground level is given by an initial general pattern consisting of a single atom. Frequent patterns are discovered by successively (level-wise) specializing the pattern by suitable operators, which allow to explore the search space, until a stopping criterion is met. Precisely, at each level, a set of specialized patterns is computed (*generate phase*) and each pattern is then evaluated for possible pruning (*test phase*). Alg. 1 formalizes the frequent patterns discovery phase. As first step, all general patterns are generated (with the function CREATEGENERALPATTERNS) and stored in a queue q. Given all concept and role names in the KB, the function maintains those whose cardinality extensions (approximated with instance retrieval) is higher than a threshold θ_f. Each generated general pattern is dequeued from q and processed for specialization (with the function GENERATESPECIALIZEDPATTERNS) until q is empty. Pattern specialization is performed level-wise by adding a new atom for each level and evaluating the obtained pattern for possible pruning, until the maximum pattern length (stopping criterion) is reached. All possible specializations for a given pattern are generated by applying the following operators:[8]

Add a concept atom: (detailed in Alg. 2) adds an atom whose predicate symbol is a concept name in the ontology and its variable argument already appears in the pattern to be special-

[8]Currently, we focus on rules containing variables only, but operators taking into account constants might similarly be considered.

ized. The predicate symbol can already appear in the pattern, in that case, a different variable name has to be used.

Add a role atom: (Alg. 3 and Alg. 4) adds an atom whose predicate symbol is a role name in the ontology and at least one of its variable arguments is shared with one or more atoms in the pattern while the other could be a shared or a new variable. The predicate symbol could appear in the pattern.

The operators are applied so that, at each step of the specialization process, rules in agreement with the language bias (see Sect. 2) are obtained. Particularly, non-redundancy is ensured by checking that (i) a candidate concept name is never added as a concept atom with the same variable names appearing in a concept atom whose predicate symbol subsumes or is subsumed by the candidate concept name w.r.t. the considered ontology or with variable names appearing in a role atom where domain and/or range of the predicate symbol is subsumed by the candidate concept name w.r.t. the considered ontology (lines 2–8 of Alg. 2); (ii) a candidate role name is never added as a role atom with the same variable names appearing in a concept atom whose predicate symbol subsumes the domain and/or range of the role name w.r.t. the considered ontology (Alg. 3 and Alg. 4). The formal proof that only non-redundant patterns are generated can be built by (similarly to how the Tableaux algorithm works) attempting to build an interpretation for the specialized pattern (regarded as a conjunction of concept and role names) where a concept or role name in it, is assumed to be redundant. This ends up with an empty interpretation, showing that there cannot exist any redundant atom within the specialized pattern.

At each specialization level, the generated pattern is evaluated by the function EVALUATEPATTERNFORPRUNING (see Alg. 5) for possible pruning. Several pruning conditions are verified (see discussion in Sect. 3.2). If the specialized pattern is pruned (line 9), it is added to the list of the infrequent patterns. If not pruned, the pattern is enqueued for a successive specialization step, provided that its pattern length does not exceed the fixed maximum length (lines 11–12). If the pattern is not pruned but the maximum length is reached, it is added to the list of the discovered frequent patterns, provided it satisfies the *safety condition* (lines 14–15). It may happen that all specializations of a given pattern are pruned (line 17). In this case, the pattern from which specializations are computed is added to the list of the discovered frequent patterns, provided it satisfies the safety condition. If not, its first ancestor satisfying the safety condition (GETSAFEPATTERNORANCESTORPATTERN) (if any) is added to the list of the discovered frequent patterns, provided that no equivalent pattern is already present in the list (lines 18–21). Once all the general patterns are processed for specializations, namely q is empty, the set of all discovered frequent patterns is returned. The corresponding rules are straightforwardly obtained and coded in SWRL by considering, for each pattern, the first atom as the head of the rule and the remaining as the rule body.

3.2 On Assessing Pattern Pruning

The EVALUATEPATTERNFORPRUNING function (Alg. 5) checks for different pruning conditions on a given specialized pattern. If one of them is verified, the pattern is pruned. The first pruning condition (line 2) checks if the rule obtained from the discovered pattern is unsatisfiable when considered jointly with the ontology. If so, the rule is pruned since it contradicts the reference KB.[9] Please note that, this condition cannot occur if the ontological KB is consistent and noise-free. Nevertheless, since the proposed method can be also applied to noisy ontologies, it may happen that an unsatisfiable rule/pattern (when considered jointly with the ontology) is extracted, particularly if low *frequency* and *Head Coverage* thresholds (see Sect. 2.3) are considered.

The second pruning condition (line 4) is used for pruning a pattern if its *Head Coverage* is less then a threshold θ_{hc}. This ensures that the generated pattern is an abstraction of a sufficient number of assertions and not of an isolated number of cases. Even though we ensure that the final discovered rules satisfy the *safety condition* (see Def. 4), while building a pattern (see Sect. 3) it may happen that a *dangling* variable occurs, that is a variable occurring in the corresponding rule head but not in the body. This may happen when a binary atom occurs in the head and a unary atom is added as a first atom in the body[10]. In such a case, when computing *support*, *confidence* and *Head Coverage* (see Sect. 2.3), according to the DL-safety condition: a) the dangling variable y has to be bound to any individual in $\mathcal{K}$ i.e., the rule r is treated as if an additional conjunct $\top(y)$ is present in the body; b) a variable $z \in V(r) \setminus V(H)$ is treated as if it is existentially quantified, i.e., a binding of the variables $V(H)$ is counted once and only if there exists an

Algorithm 2 ADDCONCEPTATOMS($r, C', concepts_r, roles_r, vars_r$) : *specializedPatternsGivenAConceptAtom*

Implements the specialization operator "add concept atom" which, given the current pattern r and the candidate concept atom C', returns all possible non-redundant patterns w.r.t. the combination of variables

Input: r: the pattern to be specialized; C': the candidate concept atom; *concepts*$_r$: concept names appearing in the pattern under construction; *roles*$_r$: role names appearing in the pattern under construction; *vars*$_r$: variable names appearing in the pattern under construction;
Output: *specializedPatternsGivenAConceptAtom*: the list of all non-redundant specializations for the input pattern, given the candidate concept atom C'

1: *specializedPatternsGivenAConceptAtom* $\leftarrow \emptyset$
2: *subC*$_r$ $\leftarrow$ *concepts*$_r$.getConceptsSubsumedBy(C')
3: *superC*$_r$ $\leftarrow$ *concepts*$_r$.getConceptsSubsuming(C')
4: *subR*$_r$ $\leftarrow$ *roles*$_r$.getRolesWithDomainOrRangeSubsumedBy(C')
5: {*Avoid rules with semantically redundant atoms*}
6: *used* $\leftarrow$ *subC*$_r$.getVars() $\cup$ *superC*$_r$.getVars() $\cup$ *subR*$_r$.getVars()
7: **for all** $v \in vars_r \setminus used$ **do**
8: *specializedPatternsGivenAConceptAtom*.add($r \wedge C'(v)$)
9: **return** *specializedPatternsGivenAConceptAtom*

Algorithm 3 ADDROLEATOMSWITHFRESHVAR($r, R', concepts_r, roles_r, vars_r$) : *specializedPatternsWithFreshVarGivenARoleAtom*

Implements the operator "add role atom introducing a fresh variable", which, given the current pattern r and the candidate role atom R', returns all non-redundant possible patterns w.r.t. the combination of variables and a fresh variable

Input: r: the pattern to be specialized; R': the candidate role atom; *concepts*$_r$: concept names appearing in the pattern under construction; *roles*$_r$: role names appearing in the pattern under construction; *vars*$_r$: variable names appearing in the pattern under construction;
Output: *specializedPatternsWithFreshVarGivenARoleAtom*: list of specializations

1: *specializedPatternsWithFreshVarGivenARoleAtom* $\leftarrow \emptyset$;
2: $z \leftarrow$ GETFRESHVARIABLE()
3: *supConceptsOfDomain*$_r$ $\leftarrow$ *concepts*$_r$.getConceptsSubsuming(dom(R'))
4: *supConceptsOfRange*$_r$ $\leftarrow$ *concepts*$_r$.getConceptsSubsuming(range(R'))
5: **for all** $v \in vars_r \setminus$ *supConceptsOfDomain*$_r$.getVars() **do**
6: *specializedPatternsWithFreshVarGivenARoleAtom*.add($r \wedge R'_i(v, z)$)
7: **for all** $v \in vars_r \setminus$ *supConceptsOfRange*$_r$.getVars() **do**
8: *specializedPatternsWithFreshVarGivenARoleAtom*.add($r \wedge (R'_i(z, v))$)
9: **return** *specializedPatternsWithFreshVarGivenARoleAtom*

[9] As remarked in [13], the satisfiability check is useful only if disjointness axioms occur in the ontology, otherwise no rules can be pruned. This check can be omitted (saving computational costs) if no disjointness axioms occur.

[10] Please note if such a case is avoided, the search space would result drastically cut with the risk of skipping important/useful patterns.

Algorithm 4 ADDROLEATOMSWITHALLVARSBOUND(r, R', $concepts_r$, $roles_r$, $vars_r$) : *specializedPatternsWithBoundVars*
Implements the specialization operator "add role atom with all variables bound" that given the current pattern r and the candidate role atom R' return all non-redundant patterns w.r.t. the combination of variables

Input: r: the pattern to be specialized; R': the candidate role atom;
$concepts_r$: concept names appearing in the pattern under construction;
$roles_r$: role names appearing in the pattern under construction;
$vars_r$: variable names appearing in the pattern under construction;
Output: *specializedPatternsWithBoundVars*: list of specializations

1: *specializedPatternsWithBoundVars* $\leftarrow \emptyset$
2: *supConceptsOfDomain*$_r$ $\leftarrow$ *concepts*$_r$.getConceptsSubsuming(dom(R'))
3: *supConceptsOfRange*$_r$ $\leftarrow$ *concepts*$_r$.getConceptsSubsuming(range(R'))
4: **if** $R' \notin$ *roles*$_r$ **then**
5: **for all** $v \in vars_r \setminus$ *supConceptsOfDomain*$_r$.getVars() **do**
6: **for all** $w \in vars_r \setminus$ *supConceptsOfRange*$_r$.getVars() **do**
7: *specializedPatternsWithBoundVars*.add($r \wedge R'(v, w)$)
8: **else**
9: *usedDomVars* $\leftarrow$ *roles*$_r$.getElement(R').getListOfVarsForDomain()
10: *usedRangeVars* $\leftarrow$ *roles*$_r$.getElement(R').getListOfVarsForRange()
11: *usedDomVars* $\leftarrow$ *usedDomVars* $\cup$ *supConceptsOfDomain*$_r$.getVars()
12: *usedRangeVars* $\leftarrow$ *usedRangeVars* $\cup$ *supConceptsOfRange*$_r$.getVars()
13: **for all** $v \in vars_r \setminus$ *usedDomVars* **do**
14: **for all** $w \in vars_r \setminus$ *usedRangeVars* **do**
15: *specializedPatternsWithBoundVars*.add($r \wedge R'(v, w)$)
16: **return** *specializedPatternsWithBoundVars*

Algorithm 5 EVALUATEPATTERNFORPRUNING($\mathcal{K}$, p, p', q, *infrequent*) : *pruned*
Determine if a pattern has to be pruned or not.

Input: $\mathcal{K}$: ontological KB; p: parent pattern of the pattern to be evaluated; p': pattern to be evaluated for pruning; q: list of the generated patterns; *infrequent*: set of the patterns that have been pruned;
Output: *pruned*: true if the pattern has to be pruned, false otherwise

1: $r' \leftarrow p'$.asRule(); $r \leftarrow p$.asRule()
2: **if** $\mathcal{K} \cup r' \models \bot$ **then**
3: **return true**
4: **else if** $\mathrm{headCoverage}(r') < \theta_{hc}$ **then**
5: **return true**
6: **else if** $\mathrm{conf}(r') - \mathrm{conf}(r) < \theta_{ic}$ **then**
7: **return true**
8: **else if** ISPATTERNALREADYGENERATED(p', q) **then**
9: **return true**
10: **else**
11: **for all** *infPatt* $\in$ *infrequent* **do**
12: **if** r'.getSupportExtension() $\subseteq$ *infPatt*.getSupportExtension() **then**
13: **return true**
14: **return false**

individual to which z can be bound while satisfying both the head and the body.

The third condition (line 6) prunes a pattern if it does not show a sufficient confidence improvement w.r.t. its parent. A lack of confidence improvement means that the specialized pattern does not contain new information, hence, by adopting the minimal criterion length, the shorter pattern is preferred.

The condition reported on line 8 is used for pruning patterns that are already generated from other specialization paths, thus avoiding that the same pattern is considered more than once for the specialization process, thus saving computational effort. Checking for an already generated pattern is performed by a heuristic: two patterns are assumed to be semantically equivalent if the support extension of their corresponding rules is the same. For avoiding to compare the support extension of the candidate rule/pattern with those of all patterns/rules already generated, the heuristic selects only the patterns whose corresponding rules have the same *Head Coverage* value of the current pattern/rule and among these checks for the existence of a pattern/rule having the same support extension of the candidate pattern/rule. If found, the candidate pattern is pruned.

Table 1: Key facts about the ontological KBs used.

Ontology	# Concepts	# Roles	# Indiv.	# Declared Assertions	# Decl.+Derived Assertions
Financial	59	16	1000	3359	3814
BioPAX	40	33	323	904	1671
NTMerged	47	27	695	4161	6863

The last condition (line 10) prunes a candidate pattern if it results to be the specialization of an existing infrequent pattern. This is checked by an heuristic: the candidate pattern is assessed as infrequent if the support extension of its corresponding rule is contained in the support extension of a rule/pattern in the infrequent patterns.

4. EXPERIMENTAL EVALUATION

We tested our method on ontologies publicly available: Financial Ontology [11], Biological Pathways Exchange (BioPAX) [12] Level 2 Ontology, New Testament Names Ontology (NTN) [13]. Details on them are reported in Tab. 1. We started by considering small ontologies since we wanted to take under control the role of the reasoner in this preliminary phase. For the same reason, efficiency aspects have not been strongly taken into account. At this stage, we wanted to prove the feasibility of our approach, while directions for tackling scalability and efficiency aspects are drawn in Sect. 5. The first goal of our experiments consisted in assessing the ability of the discovered rules to predict new assertional knowledge for a considered ontological KB. For the purpose, different samples of each ontology have been built for learning multi-relational ARs (as presented in Sect. 3) while the full ontology versions have been used as test-bed. Specifically, for each ontology three samples have been built by randomly removing respectively 20%, 30%, 40% of the concept assertions, according to a stratified sampling procedure.[14] We ran our system by repeating 10 times the sampling procedure for Financial and BioPAX ontologies and 1 times for NTNMerged and using the following parameters setting: MAX_LENGTH $= 3$, $\theta_f = 1$ (frequency threshold for building the queue q of the general patterns), $\theta_{hc} = 0.01$, $\theta_{ic} = 0.001$. As in [9], we applied the discovered rules to the full ontology versions and collected all predictions, that is the head atoms of the instantiated rules. All predictions already contained in the reduced ontology versions have been discarded while the remaining predicted facts have been considered. A prediction is assessed as *correct* if it is contained/entailed by the full ontology version. A prediction is assessed as *incorrect* if it is inconsistent with the full ontology version. Results (see Tab. 2) have been averaged over the different runs of the ontology sampling for each ontology and have been measured in terms of: *precision* (see Def. 10), *match rate*, *commission error rate* and *induction rate* (see Sect. refsec:metrics). Looking at Tab. 2, it is possible to see that very high values of match rate are reached for the considered ontologies. This proves that the discovered rules are actually able to predict new assertional knowledge for the considered ontologies. Additionally, as expected (because of the exploitation of the terminology and the reasoning capabilities) no contradicting knowledge is predicted (because of the null commission rate). The cases of not null induction rate testifies the ability of the developed method to induce new knowledge that is not logically derivable. Specifically, induction rate is not null for the case of ontologies where

[11] http://www.cs.put.poznan.pl/alawrynowicz/financial.owl.
[12] http://www.biopax.org/release/biopax-level2.owl.
[13] http://www.semanticbible.com/ntn/ntn-view.html
[14] The reduced versions of the ontologies will be available online

Table 2: Average performance metrics on each ontology.

Ontology	Sample	Match Rate	Comm. Rate	Ind. Rate	Precision	Tot. nr. Predictions
Financial	20%	0.81	0	0.19	1.0	947
	30%	0.81	0	0.19	1.0	1890
	40%	0.82	0	0.18	1.0	2960
BioPAX	20%	1.0	0	0	1.0	669
	30%	1.0	0	0	1.0	1059
	40%	1.0	0	0	1.0	1618
NTMerged	20%	0.94	0	0.06	1.0	9085
	30%	0.9	0	0.1	1.0	9756
	40%	0.94	0	0.06	1.0	10418

Table 3: Comparison # extracted rules: AMIE vs. our system.

Ontology	Samp.	# Rules		Top		
		Ours	AMIE	n	# Predictions Ours	# Predictions #AMIE
Financial	20%	177	2	2	29	208
	30%	181	2	2	57	197
	40%	180	2	2	85	184
BioPax	20%	298	8	8	25	2
	30%	283	8	8	34	2
	40%	272	0	8	50	0
NTMerged	20%	243	1129	10	620	420
	30%	225	1022	10	623	281
	40%	239	1063	10	625	332

cases of concepts and roles for which a large number of assertions is available while for other concepts and roles, few assertions are available. Note that the value of precision is always the highest one since the induced assertions are not considered for the computation of this metric and any mistake (commission error is null) is made. It is also interesting to note how the number of predicted assertions increases when less knowledge is available (since a higher number of assertions have been removed from the ontologies).

The second goal of our experiments consisted in showing the importance and the value added of exploiting the terminological knowledge and the reasoning capabilities when extracting rules from ontological KBs. For this purpose, we compared our system with AMIE [9], which represents the state-of-the-art system in the considered setting, but it is not able to exploit neither terminological information nor reasoning capabilities. Since one of the AMIE key points (as argued in [9]) is its ability to outperform state-of-the-art ILP systems in terms of number of discovered rules, we compared the number of rules discovered by our system with the number of rules discovered by AMIE, using the same samples of the ontologies mentioned above. Averaged results are reported in Tab. 3 where it is possible to see that our system outperformed the number of rules discovered by AMIE for the case of Financial and BioPax ontologies. Indeed, our system is able to output rules not necessarily *closed*[15] and having both concept and role atoms in the head, while AMIE can only output *closed* rules with role atoms in the head. Additionally our system is able to prune redundant rules and rules that are inconsistent when considered jointly with the reference ontology. This is the reason why AMIE registered a larger number of rules then our system for the case of NTNMerged. Our system outperformed AMIE also in terms of number of predictions. For the purpose, we compared the top n rules, ranked w.r.t. their match rate. n was set equal to the number of rules discovered by AMIE when few rules were discovered, and equal to 10 for the other cases. Our system clearly outperformed AMIE for the cases of BioPax and NTNMerged. The same did not happen for Financial ontology. This is because our system outputs as much as possible specific rules. For the case of Financial ontology, AMIE output two rules, each one having just one atom in the body, while our system output several refinements of such two rules, thus preventing the predictions just coming from the general rules. This suggested an improvement of our method consisting in assessing whether more general or refined rules have to be returned.

5. CONCLUSIONS

We presented a method for discovering multi-relational ARs from ontological KBs, to be used for enriching assertional knowledge. The method exploits: 1) terminological axioms; 2) DLs reasoning capabilities; providing a step ahead w.r.t. the state-of-the-art. Discovered rules are represented in SWRL, hence they can be directly added to the considered ontological KB deriving new assertional knowledge. Furthermore, the discovered rules may suggest new axioms at schema level, such as role transitivity, symmetry, role/concept subsumption. The proposed approach has been experimentally evaluated through its application to publicly available ontologies and comparisons with the main state of the art system.

Future works will focus on: enriching the expressiveness of the discovered rules and improving the method performance particularly w.r.t. scalability. For the latter goal, additional heuristics for further cutting the search space will be studied, jointly with indexing methods for caching the results of the reasoner inferences.

[15] A rule is *closed* if every variable in it appears at least twice [9].

6. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proc. of the Int. Conf. on Management of Data*, pages 207–216. ACM Press, 1993.

[2] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proc. of Very Large Data Bases Int. Conf.*, pages 487–499. Morgan Kaufmann, 1994.

[3] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge Univ. Press, 2003.

[4] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 2001.

[5] C. Bizer, T. Heath, and T. Berners-Lee. Linked data – the story so far. *Int. J. on Semantic Web and Information Systems*, 5(3):1–22, 2009.

[6] L. Dehaspe and H. Toivonen. Discovery of frequent DATALOG patterns. *Data Mining and Knowledge Discovery*, 3(1):7–36, 1999.

[7] L. Dehaspe and H. Toivonen. Discovery of relational association rules. In *Relational Data Mining*, pages 189–208. Springer, 2000.

[8] N. Fanizzi, C. d'Amato, and F. Esposito. Learning with kernels in description logics. In F. Zelezný and N. Lavrac, editors, *Proceedings of the 18th Int. Conf. on Inductive Logic Programming (ILP 2008)*, volume 5194 of *LNCS*, pages 210–225. Springer, 2008.

[9] L. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek. AMIE: Association rule mining under incomplete evidence in ontological knowledge bases. In *Proc. of the 22th International Conference on World Wide Web (WWW '13)*, pages 413–422. ACM, 2013.

[10] B. Goethals and J. Van den Bussche. Relational association rules: Getting warmer. In *Int. Works. on Pattern Detection and Discovery Proc.*, volume 2447 of *LNCS*, pages 125–139. Springer, 2002.

[11] I. Horrocks and P. F. Patel-Schneider. A proposal for an OWL rules language. In *Proc. of the Int. Conf. on World Wide Web*, pages 723–731. ACM, 2004.

[12] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A semantic web rule language combining OWL and RuleML, 2004.

[13] J. Józefowska, A. Lawrynowicz, and T. Lukaszewski. The role of semantics in mining frequent patterns from knowledge bases in description logics with rules. *Theory and Practice of Logic Programming*, 10(3):251–289, 2010.

[14] F. A. Lisi. AL-QuIn: An onto-relational learning system for semantic web mining. *Int. J. of Semantic Web and Information Systems.*, 2011.

[15] B. Motik, U. Sattler, and R. Studer. Query answering for OWL-DL with rules. *Web Semantics*, 3(1):41–60, 2005.

[16] J. Völker and M. Niepert. Statistical schema induction. In *ESWC'11 Proc.*, volume 6643 of *LNCS*, pages 124–138. Springer, 2011.