Learning Sentences and Assessments in Probabilistic Description Logics

José Eduardo Ochoa Luna[1], Kate Revoredo[2], and Fabio Gagliardi Cozman[1]

[1] Escola Politécnica, Universidade de São Paulo,
Av. Prof. Mello Morais 2231, São Paulo - SP, Brazil
[2] Departamento de Informática Aplicada, Unirio
Av. Pasteur, 458, Rio de Janeiro, RJ, Brazil
eduardo.ol@gmail.com,katerevoredo@uniriotec.br,fgcozman@usp.br

Abstract. The representation of uncertainty in the semantic web can be eased by the use of learning techniques. To completely induce a probabilistic ontology (that is, an ontology encoded through a probabilistic description logic) from data, two basic tasks must be solved: (1) learning concept definitions and (2) learning probabilistic inclusions. In this paper we propose and test an algorithm that learns concept definitions using an inductive logic programming approach and then learns probabilistic inclusions using relational data.

1 Introduction

Probabilistic Description Logics (PDLs) have been extensively investigated in the last few years [5, 8, 19, 7]. The goal is to represent uncertainty in the context of classical description logics. So far probabilistic description logics have been mostly restricted to academic purposes, as caveats in syntax and semantics have prevented them from spreading into several domains. Additionaly, it can be hard to elicit the probability component of a particular set of sentences.

The probabilistic description logic CR$\mathcal{ALC}$ [6, 22, 7] allows one to perform probabilistic reasoning by adding uncertainty capabilities to the logic $\mathcal{ALC}$ [2]. Previous efforts for learning CR$\mathcal{ALC}$ have separately focused on concept definitions [20] and probabilistic inclusions [24]. In this paper, we present an algorithm for learning concept definitions and probabilistic inclusions at once; i.e., we discuss how to construct the whole probabilistic terminology based on CR$\mathcal{ALC}$ from relational data. We expect that learning techniques can accomodate together background knowledge and deterministic and probabilistic concepts, giving each component its due relevance.

The algorithm we propose is mostly based on inductive logic programming (ILP) [9] techniques with a probabilistic twist. A search for the best concept description is performed. At the end of this search a decision is made as to whether to consider the concept description found or to insert a probabilistic inclusion based on this concept.

The paper is organized as follows. Section 2 reviews basic concepts of description logics, probabilistic description logics, CR$\mathcal{ALC}$ and machine learning

in a deterministic setting. Section 3 presents our algorithm for probabilistic description logic learning. Experiments are discussed in Section 4, and Section 5 concludes the paper.

2 Basics

The aim of this paper is to learn probabilistic terminologies from data. In this section we briefly review both deterministic and probabilistic components of probabilistic description logics. In addition, machine learning in a deterministic setting is discussed.

2.1 Description Logics

Description logics (DLs) form a family of representation languages that are typically decidable fragments of first order logic (FOL) [2]. Knowledge is expressed in terms of *individuals*, *concepts*, and *roles*. The semantic of a description is given by a *domain* $\mathcal{D}$ (a set) and an *interpretation* $\cdot^{\mathcal{I}}$ (a functor). Individuals represent objects through names from a set $N_{\mathsf{I}} = \{a, b, \ldots\}$. Each *concept* in the set $N_{\mathsf{C}} = \{C, D, \ldots\}$ is interpreted as a subset of a domain $\mathcal{D}$. Each *role* in the set $N_{\mathsf{R}} = \{r, s, \ldots\}$ is interpreted as a binary relation on the domain.

Concepts and roles are combined to form new concepts using a set of *constructors*. Constructors in the $\mathcal{ALC}$ logic are *conjunction* ($C \sqcap D$), *disjunction* ($C \sqcup D$), *negation* ($\neg C$), *existential* restriction ($\exists r.C$), and *value* restriction ($\forall r.C$). *Concept inclusions/definitions* are denoted respectively by $C \sqsubseteq D$ and $C \equiv D$, where C and D are concepts. Concepts ($C \sqcup \neg C$) and ($C \sqcap \neg C$) are denoted by $\top$ and $\bot$ respectivelly. Information is stored in a *knowledge base* ($\mathcal{K}$) divided in two parts: the TBox (terminology) and the ABox (assertions). The TBox lists concepts and roles and their relationships. A TBox is acyclic if it is a set of concept inclusions/definitions such that no concept in the terminology uses itself. The ABox contains assertions about objects.

Given a knowledge base $\mathcal{K} =< \mathcal{T}, \mathcal{A} >$, the reasoning services typically include (i) consistency problem (to check whether the $\mathcal{A}$ is consistent with respect to the $\mathcal{T}$); (ii) entailment problem (to check whether an assertion is entailed by $\mathcal{K}$; note that this generates class-membership assertions $\mathcal{K} \models C(a)$, where a is an individual and C is a concept); (iii) concept satisfiability problem (to check whether a concept is subsumed by another concept with respect to the $\mathcal{T}$). The latter two reasoning services can be reduced to the consistency problem [2].

2.2 Probabilistic Description Logics and CR$\mathcal{ALC}$

Several probabilistic descriptions logics (PDLs) have appeared in the literature. Heinsohn [12], Jaeger [14] and Sebastiani [25] consider probabilistic inclusion axioms such as $P_{\mathcal{D}}(\mathsf{Professor}) = \alpha$, meaning that a randomly selected object is a $\mathsf{Professor}$ with probability α. This characterizes a *domain-based* semantics: probabilities are assigned to subsets of the domain $\mathcal{D}$. Sebastiani also allows inclusions

such as $P(\mathsf{Professor}(\mathsf{John})) = \alpha$, specifying probabilities over the interpretations themselves. For example, one interprets $P(\mathsf{Professor}(\mathsf{John})) = 0.001$ as assigning 0.001 to be the probability of the set of interpretations where John is a Professor. This characterizes an *interpretation-based* semantics.

The PDL CR$\mathcal{ALC}$ is a probabilistic extension of the DL $\mathcal{ALC}$ that adopts an interpretation-based semantics. It keeps all constructors of $\mathcal{ALC}$, but only allows concept names on the left hand side of inclusions/definitions. Additionally, in CR$\mathcal{ALC}$ one can have probabilistic inclusions such as $P(C|D) = \alpha$ or $P(r) = \beta$ for concepts C and D, and for role r. If the interpretation of D is the whole domain, then we simply write $P(C) = \alpha$. The semantics of these inclusions is roughly (a formal definition can be found in [7]) given by:

$$\forall x \in \mathcal{D} \ : \ P(C(x)|D(x)) = \alpha,$$

$$\forall x \in \mathcal{D}, y \in \mathcal{D} \ : \ P(r(x,y)) = \beta.$$

We assume that every terminology is acyclic; no concept uses itself. This assumption allows one to represent any terminology $\mathcal{T}$ through a directed acyclic graph. Such a graph, denoted by $\mathcal{G}(\mathcal{T})$, has each concept name and role name as a node, and if a concept C directly uses concept D, that is if C and D appear respectively in the left and right hand sides of an inclusion/definition, then D is a *parent* of C in $\mathcal{G}(\mathcal{T})$. Each existential restriction $\exists r.C$ and value restriction $\forall r.C$ is added to the graph $\mathcal{G}(\mathcal{T})$ as nodes, with an edge from r to each restriction directly using it. Each restriction node is a *deterministic* node in that its value is completely determined by its parents.

The semantics of CR$\mathcal{ALC}$ is based on probability measures over the space of interpretations, for a fixed domain. Inferences, such as $P(\mathsf{A}_o(\mathsf{a}_0)|\mathcal{A})$ for an ABox $\mathcal{A}$, can be computed by propositionalization and probabilistic inference (for exact calculations) or by a first order loopy propagation algorithm (for approximate calculations) [7].

2.3 Learning Description Logics

The use of ontologies for knowledge representation has been a key element of proposals for the Semantic Web [1]. However, constructing ontologies from scratch can be a bundersome and time consuming task [10]. Nowadays, mainly due to the availability of data, learning of ontologies has turn out to be an interesting alternative. Indeed, considerable effort is currently invested into developing automated means for the acquisition of ontologies [16].

Most early approaches were only capable of learning simple ontologies such as taxonomic hierarchies. Some recent approaches such as YINYANG [13], DL-FOIL [10] and DL-Learner [18] have focused on learning expressive terminologies (we refer to [20] for a detailed review on learning description logics). To some extent, all these approaches have been inspired by Inductive Logic Programming (ILP) techniques, in that they try to transfer ILP methods to description logic settings. The goal of learning in such deterministic languages is generally to find a correct concept with respect to given examples. A formal definition is:

Definition 1. *Given a knowledge base $\mathcal{K}$, a target concept* Target *such that* Target $\notin \mathcal{K}$*, a set $E = E_p \cup E_n$ of positive and negative examples given as assertions for* Target*, the goal of learning is to find a concept definition C(*Target $\equiv C$*) such that $\mathcal{K} \cup C \models E_p$ and $\mathcal{K} \cup C \not\models E_n$.*

A sound concept definition for Target must cover all positive examples and none of the negative examples. A learning algorithm can be constructed as a combination of (1) a refinement operator, which defines how a search tree can be built, (2) a search algorithm, which controls how the tree is traversed, and (3) a scoring function to evaluate the nodes in the tree defining the best one.

The refinement operator Refinement operators allow us to find candidate concept definitions through two basic tasks: generalization and specialization [17]. Such operators in both ILP and description logic learning rely on θ-subsumption to establish an ordering so as to traverse the search space. If a concept C subsumes a concept $D(D \sqsubseteq C)$, then C covers all examples which are covered by D, which makes subsumption a suitable order. Arguably the best refinement operator for description logic learning is the one available in the DL-Learner system [17, 18], as this operator has been proved to be complete, weakly complete and proper (see [17] for details).

The score function In a deterministic setting a cover relationship simply tests whether, for given candidate concept definition (C), a given example e holds; that is, $\mathcal{K} \cup C \models e$ where $e \in E_p$ or $e \in E_n$. In this sense, a cover relationship $cover(e, \mathcal{K}, C)$ indicates whether a candidate concept covers a given example. A cover relationship is commonly evaluated by instance checking [10].

In description logic learning one often compares candidates through score functions based on the number of positive/negative examples covered. To avoid overfitting on concepts, horizontal expansions[3] are also explored [18]. For instance, in DL-Learner a fitness relationship considers the number of positive examples as well as the length of solutions when expanding candidates in the tree search.

The algorithm to traverse the search space The learning algorithm depends basically on the way we traverse the candidate concepts obtained after applying refinement operators. In a deterministic setting the search for candidate concepts is often based on the FOIL [23] algorithm. There are also different approaches (for instance, DL-Learner, an approach based on genetic algorithms [16], and one that relies on horizontal expansion and redundance checking to traverse search trees [18]).

[3] Given a node in a search tree, the horizontal expansion is its upper bound on the length of child concepts.

3 Learning the PDL CR$\mathcal{ALC}$

A probabilistic terminology consists of both concepts definitions and probabilistic components (probabilistic inclusions in CR$\mathcal{ALC}$). We aim at automatically identifying from data sound deterministic concepts and consistent probabilistic inclusions. A key design choice in learning under a combined approach is to give a due relevance to each component.

It is worth noting that there are well established deterministic concepts such as $\mathsf{Father} \equiv \mathsf{Male} \sqcap \mathsf{hasChild}.\top$ for which it would be unnecessary to find a probabilistic interpretation. On the other hand, there are concepts with natural probabilistic assessments such as $P(\mathsf{FlyingBird}|\mathsf{Bird}) = \alpha$. In principle, a learning algorithm should be able to deal with these subtleties.

We argue that negative and positive examples underlie the choice of either a concept definition or a probabilistic inclusion. In a deterministic setting we expect to find concepts covering all positive examples, which is not always possible. It is common to allow flexible heuristics that deal with these issues. Moreover, there are several examples that cannot be ascribed to candidate hypotheses[4]. Uncertainty stems from such missing information. Therefore, when we are unable to find a concept definition that covers all positive examples we assume such hypothesis as candidates to be a probabilistic inclusion and we begin the search for the best probabilistic inclusion that fits the examples.

As in description logic learning three tasks are important and should be considered: (1) refinement operators, (2) scoring functions and (3) a traverse search space algorithm. The refinement operator described in 2.3 is used for learning the deterministic component of probabilistic terminologies. The other two tasks were adapted for probabilistic description logic learning as follows.

3.1 The Probabilistic Score Function

In our proposal, since we want to learn probabilistic terminologies, we adopt a probabilistic cover relation given in [15]:

$$cover(e, \mathcal{K}, C) = P(e|\mathcal{K}, C).$$

Every candidate hypothesis together with a given example turns out to be a probabilistic random variable which yields true if the example is covered, and false otherwise. To guarantee soundness of the ILP process (that is, to cover positive examples and not to cover negative examples), the following restrictions are needed:

$$P(e_p|\mathcal{K}, C) > 0, \qquad P(e_n|\mathcal{K}, C) = 0.$$

In this way a probabilistic cover relationship is a generalization of the deterministic cover, and is suitable for a combined approach. Probabilities can be

[4] In some cases the Open World Assumption inherent to description logics prevent us for stating membership of concepts.

computed through Bayes' theorem:

$$P(e|\mathcal{K}, C_1, \ldots, C_k) = \frac{P(C_1, C_2, \ldots, C_k|T)P(T)}{P(C_1, \ldots, C_k)},$$

where $C_1, \ldots, C_k$ are candidate concepts definitions, and T denotes the target concept variable. Here are three possibilities for modeling $P(C_1, \ldots, C_k|T)$: (1) a naive Bayes assumption may be adopted [15] (each candidate concept is independent given the target), and then $P(C_1, \ldots, C_k|T) = \prod_i P(C_i|T)$; (2) the noisy-OR function may be used [20]; (3) a less restrictive option based on tree augmented naive Bayes networks (TAN) may be handy [15]. This last possibility has been considered for the probabilistic cover relationship used in this paper. In each case probabilities are estimated by maximum (conditional) likelihood parameters. The candidate concept definition C_i with the highest probability $P(C_i|T)$ is the one chosen as the best candidate.

As we have chosen a probabilistic cover relationship, our probabilistic score is defined accordingly:

$$score(\mathcal{K}|C) = \prod_{e_i \in E_p} P(e_i|\mathcal{K}, C),$$

where C is the best candidate chosen as described before.

In the probabilistic score we have previously defined, a given threshold allow us differentiate between a deterministic and probabilistic inclusion candidate. Further details are given in the next section.

3.2 The Algorithm to Learn Probabilistic Terminologies

Previous efforts for learning the PDL CR$\mathcal{ALC}$ have separately explored concepts definitions [20] and probabilistic inclusions [24]. In this paper, we advocate for a combined approach where we use a classical approach for traversing the space of deterministic concepts and a probabilistic procedure for generating probabilistic inclusions.

The choice between a deterministic or a probabilistic inclusion is based on a probabilistic score. We start by searching a deterministic concept. If after a set of iterations the score of the best candidate is below a given threshold, a search for a probabilistic inclusion is preferred rather than keep searching for a deterministic concept definition. Then, the current best k-candidates are considered as start point for probabilistic inclusion search. The complete learning procedure is shown in Algorithm 1.

The algorithm starts with an overly general concept definition in the root of the search tree (line 1). This node is expanded according to refinement operators and horizontal expansion criterion (line 4), i.e, child nodes obtained by refinement operators are added to the search tree (line 5). The probabilistic parameters of these child nodes are learned (line 6) and then they are evaluated with the best one chosen for a new expansion (line 3) (alternative nodes based

Require: an initial knowledge base $\mathcal{K} =< \mathcal{T}, \mathcal{A} >$ and a training set E.
1: SearchTree with a node $\{C = \top, h = 0\}$
2: **repeat**
3: choose node $N = \{C, h\}$ with highest probabilistic score in SearchTree
4: expand node to length $h + 1$:
5: add all nodes $D \in$ (refinementOperator(C)) with lenght $=h + 1$
6: learn parameters for all nodes D
7: $N = \{C, h + 1\}$
8: expand alternative nodes according to horizontal expansion factor and $h + 1$[18]
9: **until** stopping criterion
10: $N' =$ best node in SearchTree
11: **if** score(N') $>$ threshold **then**
12: return deterministic concept $C' \in N'$
13: **else**
14: call ProbabilisticInclusion(SearchTree)
15: **end if**

Algorithm 1: Algorithm for learning probabilistic terminologies.

on horizontal expansion factor are also considered (line 8)). This process continues until a stopping criterion is attained (difference for scores is insignificant); After that, the best node obtained is evaluated and if it is above a threshold, a deterministic concept definition is found and returned (line 11). Otherwise, a probabilistic inclusion procedure is called (line 13).

The Algorithm 2 learns probabilistic inclusions. It starts retrieving the best k nodes in the search tree and computing the conditional mutual information for every pair of nodes (line 2). Then an undirected graph is built where the vertices are the k nodes and the edges are weighted with the value of the conditional mutual information [21] for each pair of vertices (lines 4 and 5). A maximum weight spanning tree [4] from this graph is built (line 6) and the target concept is added as a parent for each vertice (line 7). The probabilistic parameters are learned (line 8). This learned TAN-based classifier [11] is used to evaluate the possible probabilistic inclusion candidates (line 9) and the best one is returned.

4 Experiments

In order to evaluate the learning algorithm we have divided the analysis in two stages. In a first stage, the algorithm was compared with, arguably, the best description logic learning algorithm available (the DL-Learner system). The second stage evaluated suitability of the algorithm for learning probabilistic terminologies in real world domains.

The aim of the first stage was to investigate whether by introducing a probabilistic setting the algorithm behaves as well as traditional deterministic approaches in description logic learning tasks. In this preliminar evaluation (as a rule, there is a lack of evaluation standards in ontology learning [18]) we have

Require: SearchTree previously computed
1: **for** each pair of candidates C_i, C_j in first k nodes of the SearchTree **do**
2: compute the conditional mutual information $I(C_i, C_j|T)$
3: **end for**
4: build an undirected graph in which vertices are the k candidates
5: annotate the weight of an edge connecting C_i to C_j by the $I(C_i, C_j|T)$
6: build a maximum weight spanning tree from this graph
7: add T as parent for each C_i
8: learn probabilities for $P(C_i|Parents(C_i))$
9: return the highest probabilistic inclusion $P(T|C') = \alpha$

Algorithm 2: Algorithm for learning probabilistic inclusions.

considered five datasets available in the DL-Learner system and reported in [18]. Evaluation results are shown in Table 1.

Table 1. Description logic learning results

Problem	axioms, examples	DL-learner correct (length)	Combined approach correct(length)
trains	252,10	100(5)	100%(5)
arches	47,5	100%(9)	100%(10)
moral	31,43	100%(3)	100%(5)
poker(pair)	35,49	100%(8)	100%(8)
poker (straight)	45,55	100%(5)	100%(5)

The combined approach was able to learn correct concept definitions. However, in some cases produced longer solutions.

In the second stage we focused on learning of probabilistic terminologies from real world data. Wikipedia[5] was used to do so. Wikipedia articles consist mostly of free text, but also contain various types of structured information in the form of Wiki markup. Such information includes infobox templates, categorization information, images geo-coordinates, links to external Web pages, disambiguation pages, redirects between pages, and links across different language editions of Wikipedia.

In the last years, there were several projects aimed at structuring such huge source of knowledge. Examples include, The DBpedia project [3], which extracts structured information from Wikipedia and turns it into a rich knowledge base, and YAGO [26], a semantic knowledge base based on data from Wikipedia and WordNet[6]. Currently, YAGO knows more than 2 million entities (like persons, organizations, cities, etc.). It knows 20 million facts about these

[5] http://www.wikipedia.org/
[6] wordnet.princeton.edu/

entities. Unlike many other automatically assembled knowledge bases, YAGO has a manually confirmed accuracy of 95%. Several domains ranging from films, places, historical events, wines, etc. have been considered in this ontology. Moreover, facts are given as binary relationships that are suitable for our learning settings. There are approximately 92 relationships available. Examples include actedIn, bornIn, created, discovered describes, diedIn, happenedIn, hasAcademicAdvisor, hasChild, hasHDI, hasWonPrize, influences, isMarriedTo, isPartOf, livesIn, politicianOf, worksAt.

We have used subsets of YAGO facts for learning probabilistic terminologies. Two domais have been mostly explored. The first, related to scientists. The second, related to film directors. In both cases the threshold used was 0.85 and the 20 best candidates were considered in the probabilistic inclusion learning step.

The first dataset consists of 2008 potential scientists for which we have learned concept definitions and probabilistic inclusions. The complete terminology is given below:

$$
\begin{array}{ll}
 & P(\mathsf{Person}) = 0.9 \\
 & P(\mathsf{Topic}) = 0.4 \\
 & P(\mathsf{Year}) = 0.35 \\
 & P(\mathsf{Prize}) = 0.2 \\
 & P(\mathsf{Text}) = 0.25 \\
 & P(\mathsf{EducationalInstitution}) = 0.3 \\
 & P(\mathsf{wrotes}) = 0.4 \\
 & P(\mathsf{hasAcademicAdvisor}) = 0.80 \\
 & P(\mathsf{interestedIn}) = 0.6 \\
 & P(\mathsf{diedOnYear}) = 0.7 \\
 & P(\mathsf{hasWonPrize}) = 0.4 \\
 & P(\mathsf{worksAt}) = 0.85 \\
 & P(\mathsf{influences}) = 0.6 \\
\mathsf{Scientist} \equiv & \mathsf{Person} \\
 & \sqcap(\exists \mathsf{hasAcademicAdvisor.Person} \\
 & \sqcap \exists \mathsf{wrotes.Text} \sqcap \exists \mathsf{worksAt.EducationalInsitution}) \\
P(\mathsf{InfluentialScientist} & \mid \mathsf{Scientist} \sqcap \exists \mathsf{influences.} \\
 & \exists \mathsf{diedOnYear.Year}) = 0.85 \\
P(\mathsf{Musician} & \mid \mathsf{Person} \sqcap \exists \mathsf{hasAcademicAdvisor.} \exists \mathsf{wrote.Text}) = 0.1 \\
\mathsf{HonoredScientist} \equiv & \mathsf{Scientist} \\
 & \sqcap \exists \mathsf{hasWonPrize.Prize}
\end{array}
$$

This resulting CR$\mathcal{ALC}$ terminology can be further investigated by probabilistic inference[7]. The basic task we address is classification. Assume we are interested in classifying a potential scientist given we know he/she has written a book and has an academic advisor:

$$P(\mathsf{Scientist}(0) | \mathsf{Person}(0) \sqcap \exists \mathsf{wrote.Text}(1) \sqcap \mathsf{hasAcademicAdvisor.Person}(2)) = 0.5$$

When further evidence is available the value probability is updated to:

$$
\begin{array}{ll}
P(\mathsf{Scientist}(0) & | \mathsf{Person}(0) \\
 & \sqcap(\exists \mathsf{wrote.Text}(1) \sqcap \exists \mathsf{hasAcademicAdvisor.} \\
 & \exists \mathsf{influences.Person}(3))) = 0.65
\end{array}
$$

[7] Given a domain size, a relational Bayesian network is constructed to do so.

In the second dataset we have collected facts about film directors ranging from classical to contemporary. About 5589 potential directors have been considered. The complete probabilistic terminology is shown below.

$$
\begin{array}{ll}
 & P(\mathsf{Person}) = 0.9 \\
 & P(\mathsf{Prize}) = 0.1 \\
 & P(\mathsf{Year}) = 0.25 \\
 & P(\mathsf{Film}) = 0.3 \\
 & P(\mathsf{isMarriedTo}) = 0.1 \\
 & P(\mathsf{influences}) = 0.35 \\
 & P(\mathsf{hasWonPrize}) = 0.28 \\
 & P(\mathsf{hasChild}) = 0.05 \\
 & P(\mathsf{diedOnYear}) = 0.5 \\
 & P(\mathsf{directed}) = 0.8 \\
 & P(\mathsf{actedIn}) = 0.4 \\
\mathsf{Actor} \equiv & \mathsf{Person} \sqcap \forall \mathsf{actedIn}.\mathsf{Film} \\
P(\mathsf{Director} & \mid \mathsf{Person} \sqcap (\exists \mathsf{directed}.\mathsf{Film} \sqcap \exists \mathsf{influences}. \\
 & \exists \mathsf{actedIn}.\mathsf{Film}) = 0.75 \\
P(\mathsf{FomerActor} & \mid \mathsf{Director} \sqcap \exists \mathsf{actedIn}.\mathsf{Film}) = 0.6 \\
\mathsf{HonoredDirector} \equiv & \mathsf{Director} \sqcap \exists \mathsf{hasWonPrize}.\mathsf{Prize} \\
\mathsf{FamilyDirector} \equiv & \mathsf{Director} \sqcap (\exists \mathsf{isMarriedTo}.\mathsf{Director} \sqcup \exists \mathsf{hasChild}.\mathsf{Director}) \\
P(\mathsf{InfluentialDirector} & \mid \mathsf{Director} \sqcap \exists \mathsf{hasWonPrize}.\mathsf{Prize} \sqcap \exists \mathsf{influences}. \\
 & \exists \mathsf{isMarriedTo}.\mathsf{Director}) = 0.7 \\
P(\mathsf{MostInfluentialDirector} & \mid \mathsf{Director} \sqcap \exists \mathsf{diedOnYear}.\mathsf{Year} \sqcap \exists \mathsf{influences}. \\
 & \exists \mathsf{hasWonPrize}.\mathsf{Prize}) = 0.8
\end{array}
$$

Learned components range from basic concept definitions such as Actor to probabilistic inclusions for describing most influential directors. Assume we are interested in classifying a person given we know that he/she has acted and directed. According to evidence available:

$$P(\mathsf{Actor}(0) | \mathsf{Person}(0) \sqcap \exists \mathsf{actedIn}.\mathsf{Film}(1) \sqcap \exists \mathsf{directed}.\mathsf{Film}(2)) = 0.4$$

$$P(\mathsf{Director}(0) | \mathsf{Person}(0) \sqcap \exists \mathsf{actedIn}.\mathsf{Film}(1) \sqcap \exists \mathsf{directed}.\mathsf{Film}(2)) = 0.55$$

As further evidence is given, probability value changes to:

$$
\begin{array}{rl}
P(\mathsf{Actor}(0) & \mid \mathsf{Person}(0) \\
 & \sqcap (\exists \mathsf{actedIn}.\mathsf{Film}(1) \sqcap \exists \mathsf{directed}.\mathsf{Film}(2) \\
 & \sqcap \exists \mathsf{influences}.\mathsf{Person}(3))) = 0.3
\end{array}
$$

5 Conclusion

We have proposed a method for learning deterministic/probabilistic components of terminologies expressed in CR$\mathcal{ALC}$. Differently from previous approaches, we have produced a combined scheme, where both the deterministic and probabilistic components receive due attention.

This unified learning scheme has the following components: (1) a refinement operator for traversing the search space, (2) probabilistic cover and score relationships for evaluating candidates, (3) a mixed search procedure. Initially, the search aims at finding deterministic concepts. If the score obtained is below a given threshold, a probabilistic inclusion search is conducted (a probabilistic classifier is produced). Experiments with probabilistic terminology in a real-world domain suggest that probabilistic inclusions do lead to improved likelihoods.

Probabilistic description logics offer expressive languages in which to conduct learning, while charging a relatively low cost for inference. The present contribution offers novel ideas for this sort of learning task; we note that the current literature on this topic is rather scarce. Our future work is to investigate the scalability of our learning methods.

Acknowledgements

The first author is supported by CAPES and the third author is partially supported by CNPq. The work reported here has received substantial support through FAPESP grant 2008/03995-5.

References

1. G. Antoniou and F. van Harmelen. *Semantic Web Primer*. MIT Press, 2008.
2. F. Baader and W. Nutt. Basic description logics. In *Description Logic Handbook*, pages 47–100. Cambridge University Press, 2002.
3. C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia - a crystallization point for the web of data. *Web Semant.*, 7(3):154–165, 2009.
4. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2001.
5. P. C. G. Costa and K. B. Laskey. PR-OWL: A framework for probabilistic ontologies. In *Proceeding of the 2006 conference on Formal Ontology in Information Systems*, pages 237–249, Amsterdam, The Netherlands, The Netherlands, 2006. IOS Press.
6. F. G. Cozman and R. B. Polastro. Loopy propagation in a probabilistic description logic. In Sergio Greco and Thomas Lukasiewicz, editors, *Second International Conference on Scalable Uncertainty Management*, Lecture Notes in Artificial Intelligence (LNAI 5291), pages 120–133. Springer, 2008.
7. F. G. Cozman and R. B. Polastro. Complexity analysis and variational inference for interpretation-based probabilistic description logics. In *Conference on Uncertainty in Artificial Intelligence*, 2009.
8. C. D'Amato, N. Fanizzi, and T. Lukasiewicz. Tractable reasoning with Bayesian description logics. In *SUM '08: Proceedings of the 2nd international conference on Scalable Uncertainty Management*, pages 146–159, Berlin, Heidelberg, 2008. Springer-Verlag.
9. L. De Raedt, editor. *Advances in Inductive Logic Programming*. IOS Press, 1996.
10. N. Fanizzi, C. D'Amato, and F. Esposito. DL-FOIL concept learning in description logics. In *ILP '08: Proceedings of the 18th International Conference on Inductive Logic Programming*, pages 107–121, Berlin, Heidelberg, 2008. Springer-Verlag.
11. N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.
12. J. Heinsohn. Probabilistic description logics. In *International Conf. on Uncertainty in Artificial Intelligence*, pages 311–318, 1994.
13. L. Iannone, I. Palmisano, and N. Fanizzi. An algorithm based on counterfactuals for concept learning in the semantic web. *Applied Intelligence*, 26(2):139–159, 2007.

14. M. Jaeger. Probabilistic reasoning in terminological logics. In *Principals of Knowledge Representation (KR)*, pages 461–472, 1994.
15. N. Landwehr, K. Kersting, and L. DeRaedt. Integrating Naïve Bayes and FOIL. *J. Mach. Learn. Res.*, 8:481–507, 2007.
16. J. Lehmann. Hybrid learning of ontology classes. In *Proceedings of the 5th International Conference on Machine Learning and Data Mining*, volume 4571 of *Lecture Notes in Computer Science*, pages 883–898. Springer, 2007.
17. J. Lehmann and P. Hitzler. Foundations of refinement operators for description logics. In Hendrick Blockeel, Jude W. Shavlik, and Prasad Tadepalli, editors, *ILP '07: Proceedings of the 17th International Conference on Inductive Logic Programming*, volume 4894 of *Lecture Notes in Computer Science*, pages 161–174. Springer, 2007.
18. J. Lehmann and P. Hitzler. A refinement operator based learning algorithm for the $\mathcal{ALC}$ description logic. In Hendrick Blockeel, Jude W. Shavlik, and Prasad Tadepalli, editors, *ILP '07: Proceedings of the 17th International Conference on Inductive Logic Programming*, volume 4894 of *Lecture Notes in Computer Science*, pages 147–160. Springer, 2007.
19. T. Lukasiewicz. Expressive probabilistic description logics. *Artif. Intell.*, 172(6-7):852–883, 2008.
20. J. E. Ochoa-Luna and F. G. Cozman. An algorithm for learning with probabilistic description logics. In *5th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW) at the 8th International Semantic Web Conference (ISWC)*, pages 63–74, Chantilly, USA, 2009.
21. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: networks of plausible inference*. Morgan Kaufman, 1988.
22. R. B. Polastro and F. G. Cozman. Inference in probabilistic ontologies with attributive concept descriptions and nominals. In *4th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW) at the 7th International Semantic Web Conference (ISWC)*, Karlsruhe, Germany, 2008.
23. J. R. Quinlan and R. M. Cameron-Jones. FOIL: A midterm report. In *Proceedings of the European Conference on Machine Learning*, pages 3–20. Springer-Verlag, 1993.
24. K. Revoredo, J. Ochoa-Luna, and F.G. Cozman. Learning terminologies in probabilistic description logics. In *Proceedings of the 20th Brazilian Symposium on Artificial Intelligence*. To appear, 2010.
25. F. Sebastiani. A probabilistic terminological logic for modelling information retrieval. In *ACM Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 122–130, 1994.
26. F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 697–706, New York, NY, USA, 2007. ACM.