Parameter Learning for Probabilistic Ontologies

Fabrizio Riguzzi[1], Elena Bellodi[2], Evelina Lamma[2], and Riccardo Zese[2]

[1] Dipartimento di Matematica e Informatica – University of Ferrara
Via Saragat 1, I-44122, Ferrara, Italy
[2] Dipartimento di Ingegneria – University of Ferrara
Via Saragat 1, I-44122, Ferrara, Italy
{fabrizio.riguzzi,elena.bellodi,evelina.lamma,riccardo.zese}@unife.it

Abstract. Recently, the problem of representing uncertainty in Description Logics (DLs) has received an increasing attention. In probabilistic DLs, axioms contain numeric parameters that are often difficult to specify or to tune for a human. In this paper we present an approach for learning and tuning the parameters of probabilistic ontologies from data. The resulting algorithm, called EDGE, is targeted to DLs following the DISPONTE approach, that applies the distribution semantics to DLs.

1 Introduction

Recently, the problem of representing uncertainty in description logics (DLs) has received an increasing attention due to the ubiquity of uncertain information in real world domains. Various authors have studied the use of probabilistic DLs and many proposals have been presented for allowing Description Logics(DLs) to represent uncertainty [8,13,7].

In [9,10] we proposed an approach for the integration of probabilistic information in DLs called DISPONTE for "DIstribution Semantics for Probabilistic ONTologiEs" (Spanish for "get ready"). DISPONTE applies the distribution semantics for probabilistic logic programming [11] to DLs. DISPONTE allows two types of probabilistic axioms: an epistemic type, that represents a degree of belief in the axiom as a whole, and a statistical type, that considers the populations to which the axiom is applied.

However, specifying the values of the probabilities is a difficult task for humans. On the other hand, data is usually available about the domain that can be leveraged for tuning the parameters. In this paper we present a machine learning approach for learning the parameters of probabilistic ontologies from data. The resulting algorithm, called EDGE for "Em over bDds for description loGics paramEter learning", is targeted to DLs following the DISPONTE semantics. EDGE starts from examples of instances and non-instances of concepts and builds Binary Decision Diagrams (BDDs) for representing their explanations from the theory. The parameters are then tuned using an EM algorithm [5] in which the required expectations are computed directly on the BDDs.

We also present preliminary experiments comparing the parameters learned by EDGE with those derived from an association rule learner [13,7] for a real

world dataset. The results show that EDGE achieves significantly higher areas under the Precision Recall and the Receiver Operating Characteristics curves.

The paper is organized as follows. Section 2 introduces Description Logics and the DISPONTE semantics while Section 3 describes EDGE. Section 4 discusses related works and Section 5 shows the results of experiments with EDGE. Section 6 concludes the paper.

2 Description Logics and the DISPONTE semantics

Description Logics (DLs) are knowledge representation formalisms that are particularly useful for representing ontologies. They are usually represented using a syntax based on concepts and roles. A concept corresponds to a set of individuals of the domain while a role corresponds to a set of couples of individuals of the domain. In the following we consider and describe $\mathcal{ALC}$ [12].

Let C and D be concepts, R be a role and a and b be individuals, a *TBox* $\mathcal{T}$ is a finite set of *concept inclusion axioms* $C \sqsubseteq D$, while an *ABox* $\mathcal{A}$ is a finite set of *concept membership axioms* $a : C$, *role membership axioms* $(a, b) : R$, *equality axioms* $a = b$ and *inequality axioms* $a \neq b$. A *knowledge base* (KB) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$.

A KB is usually assigned a semantics using interpretations of the form $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty *domain* and $\cdot^{\mathcal{I}}$ is the *interpretation function* that assigns an element in $\Delta^{\mathcal{I}}$ to each individual a, a subset of $\Delta^{\mathcal{I}}$ to each concept C and a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to each role R. The semantics of DLs can be given equivalently by transforming a DL knowledge base into a predicate logic theory and then using the model-theoretic semantics of the resulting theory.

A query over a knowledge base is usually an axiom for which we want to test the entailment from the knowledge base. The entailment test may be reduced to checking the unsatisfiability of a concept in the knowledge base, i.e., the emptiness of the concept.

Given a predicate logic formula F, a *substitution* θ is a set of pairs x/a where x is a variable universally quantified in the outermost quantifier in F and a is an individual. The application of θ to F, indicated by $F\theta$, is called an *instantiation of* F and is obtained by replacing x with a in F and by removing x from the external quantification for every pair x/a in θ.

DISPONTE applies the distribution semantics to probabilistic ontologies [11]. In DISPONTE a *probabilistic knowledge base* $\mathcal{K}$ is a set of certain and probabilistic axioms. *Certain axioms* take the form of regular DL axioms. *Probabilistic axioms* take the form $p ::_{Var} E$, where p is a real number in $[0, 1]$, Var can be empty or the symbol x and specifies the type of probability and E is a DL axiom. In order to give a semantics to such probabilistic knowledge bases, we consider their translation into predicate logic. The idea of DISPONTE is to associate independent Boolean random variables to (instantiations of) the formulas in predicate logic that are obtained from the translation of the axioms. If Var is empty, a single random variable is associated to the axiom E and p represents a degree of belief in E (epistemic probability). If $Var = x$, we associate a random

variable with every instantiation of E. In this case p indicates the probability that a random individual from the subclass belongs to the superclass (statistical probability). This type of probability is allowed only for subclass axioms.

A DISPONTE KB defines a distribution over DL KB called worlds. Each world is obtained by including every formula from a certain axiom. For each probabilistic axiom, we generate all the substitutions for the variables of the equivalent predicate logic formula. For each instantiated formula, we decide whether or not to include it in w. By multiplying the probability of the choices made to obtain a world we can assign it a probability. The probability of a query is then the sum of the probabilities of the worlds where the query holds true.

The system BUNDLE [9,10] computes the probability of a query w.r.t. ontologies that follow DISPONTE semantics by first computing all the explanations for the query and then building a Binary Decision Diagram (BDD) that represents them. A *set of explanations* for a query Q is a set of sets of triples (F_i, θ_j, k) where F_i is the translation of the ith probabilistic axiom into a predicate logic formula, θ_j is a substitution and $k \in \{0, 1\}$; k indicates if the formula is required of forbidden for deriving the query. Explanations are found by using a modified version of Pellet that returns, besides the axioms used to derive the query, also the individuals to which the axioms are applied. The set of explanations is then translated into a BDD, a data structure that allows the computation of the probability with a dynamic programming algorithm in polynomial time in the size of the diagram.

3 EDGE

EDGE is based on the algorithm EMBLEM [3,2] developed for learning the parameters for probabilistic logic programs under the distribution semantics. EDGE adapts EMBLEM to the case of probabilistic DLs under the DISPONTE semantics. EDGE takes as input a DL KB and a number of examples that represent the queries. Tipically, the queries are concept assertions and are divided into positive examples that represent information that we regard as true and for which we would like to get high probability and negative examples that represent information that we regard as false and for which we would like to get low probability.

EDGE first computes, for each example, the BDD encoding its explanations using BUNDLE. For negative examples, EDGE computes the explanations of the query, builds the BDD and then negates it. For example, if the negative example is $a : C$, EDGE executes the query $a : C$, finds the BDD and then negates it.

Then EDGE enters the EM cycle, in which the steps of expectation and maximization are repeated until the log-likelihood (LL) of the examples reaches a local maximum. At each iteration the LL of the example increases, i.e., the probability of positive examples increases and that of negative examples decreases. The EM algorithm is guaranteed to find a local maximum, which however may not be the global maximum.

EDGE's main procedure consists of a cycle in which the procedures Expectation and Maximization are repeatedly called. Procedure Expectation takes as input a list of BDDs, one for each example, and computes $P(X_{ij} = x/a)$ for all variables X_{ij} in the BDD built from explanations. Procedure Maximization computes the parameters values for the next EM iteration by relative frequency. Procedure Expectation returns the LL of the data that is used in the stopping criterion: EDGE stops when the difference between the LL of the current iteration and the one of the previous iteration drops below a threshold ϵ or when this difference is below a fraction δ of the previous LL.

The details of the procedures can be found in [3].

4 Related Work

In [8] the authors used an extension of $\mathcal{ALC}$, called cr$\mathcal{ALC}$ that adopts an interpretation-based semantics. cr$\mathcal{ALC}$ allows statistical axioms of the form $P(C|D) = \alpha$, which means that for any element x in $\mathcal{D}$, the probability that it is in C given that is in D is α, and of the form $P(R) = \beta$, which means that for each couple of elements x and y in $\mathcal{D}$, the probability that x is linked to y by the role R is β. Axioms of the form $P(C|D) = \alpha$ are equivalent to DISPONTE axioms of the form $\alpha ::_x C \sqsubseteq D$, while axioms of the form $P(R) = \beta$ have no equivalent in DISPONTE but can be introduced in principle. On the other hand, cr$\mathcal{ALC}$ does not allow to express a degree of belief in axioms. A cr$\mathcal{ALC}$ KB $\mathcal{K}$ can be represented as a directed acyclic graph $G(\mathcal{K})$ in which a node represents a concept or a role and the edges represent the relations between them.

The algorithm of [8] learns parameters and structure of cr$\mathcal{ALC}$ knowledge bases. It starts from positive and negative examples for a single concept and learns a probabilistic definition for the concept. For a set of candidate definitions, their parameters are learned using an EM algorithm. Differently for us, the expected counts are computed by resorting to inference in the graph, while we exploit the BDD structures.

Another approach is presented in [13,7], where the authors use an algorithm, called GoldMiner, that exploits Association Rules (AR) for building ontologies. GoldMiner extracts information about individuals, named classes and roles using SPARQL queries. Then, starting from this data, it builds two *transaction tables*: one for individuals and one for couples of individuals. The first contains a row for each individual and a column for all named classes and classes of the form $\exists R.C$ for R a role and C a named class. The cells of the table contain 1 if the individual belongs to the class of the column. The second table contains a row for each couple of individuals and a column for each named role. The cells contain 1 if the couple of individual belongs to the role in the column. Finally, the APRIORI algorithm [1] is applied to each table in order to find ARs. These association rules are implications of the form $A \Rightarrow B$ where A and B are conjunctions of columns. Each AR can thus be converted to a subclass or subrole axiom $A \sqsubseteq B$. So from the learned ARs a knowledge base can be obtained. Moreover, each AR $A \Rightarrow B$ is associated with a confidence that can be

interpreted as the statistical probability of the axiom $p ::_x A \sqsubseteq B$. So GoldMiner can be used to obtain a probabilistic knowledge base.

5 Experiments

EDGE has been tested over a real world dataset from the linked open data cloud: `educational.data.gov.uk`. In order to build the ontology for which we want to learn the parameters, we used GoldMiner. We ran GoldMiner using the following parameters for the APRIORI algorithm: 0.1 as the minimum support and 0.05 as the minimum confidence. The number of individuals extracted from `educational.data.gov.uk` is 134149. The resulting ontology contains around 5500 axioms. Each axiom is obtained from an AR with a confidence smaller than 1 so we consider all of them as probabilistic.

Then we selected positive and negative examples by randomly choosing individuals from the extracted ones. For each individual a we identified three random named classes: for the first two classes A and B we randomly chose a class to which a explicitly belongs, for the third class C we randomly selected a class to which a does not explicitly belong but there is at least one explanation for the query $a : C$. Finally we added the axiom $a : A$ to the ontology, $a : B$ to the set of positive examples and $a : C$ to the set of negative examples. In the training phase, we ran EDGE on the ontology obtained by GoldMiner where we consider all the axioms as probabilistic. We randomly set the initial values of the parameters. We used a 5-fold cross validation to test the system. EDGE took about 65000 seconds in average for handling 5000 examples, most of which were spent finding the explanations and building the BDDs, while the execution of the EM iterations took only about 6 seconds. In the testing phase, we computed the probability of the queries using BUNDLE. For a negative example of the form $a : C$ we compute the probability p of $a : C$ and we assign probability $1 - p$ to the example.

We compared the results of EDGE with those obtained from an ontology in which the parameters are the ARs' confidences. Given the probability of the examples in the test set, we drew the Precision-Recall and the Receiver Operating Characteristics curves and computed the Area Under the Curve (AUCPR and AUCROC), a standard way of evaluating machine learning algorithms, following [4,6]. Table 1 shows the AUCPR and AUCROC averaged over the five folds for EDGE and the association rules (ARs) together with the standard deviation. On these results we executed a two-tailed paired t-test and the resulting p-value is 0.000305 for the AUCPR and 0.000449 for the AUCROC: for both areas, the differences are statistically significant at the 5% level.

6 Conclusions

EDGE applies an EM algorithm for learning the parameters of probabilistic knowledge bases under the DISPONTE semantics. Work is currently under way for learning at the same time the structure and the parameters. EDGE

educational.data.gov.uk	Average Areas with Standard Deviation	
	EDGE	ARs
AUCPR	0.9740 ± 0.011	0.8602 ± 0.024
AUCROC	0.9848 ± 0.005	0.9184 ± 0.016

Table 1. Results of the experiments in terms of AUCPR and AUCROC.

exploits the BDDs that are built during inference to efficiently compute the expectations for hidden variables. EDGE is available for download from `http://sites.unife.it/ml/edge` together with the datasets used in the experiments. The experiments over a real world dataset extracted from `gov.uk` show that EDGE achieves larger areas both under the PR and the ROC curve with respect to a simple algorithm based on association rules.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: International Conference on Very Large Data Bases. pp. 487–499. Morgan Kaufmann (1994)
2. Bellodi, E., Riguzzi, F.: Experimentation of an expectation maximization algorithm for probabilistic logic programs. Intelligenza Artificiale 8(1), 3–18 (2012)
3. Bellodi, E., Riguzzi, F.: Expectation Maximization over binary decision diagrams for probabilistic logic programs. Intel. Data Anal. 17(2), 343–363 (2013)
4. Davis, J., Goadrich, M.: The relationship between precision-recall and ROC curves. In: International Conference on Machine Learning. pp. 233–240. ACM (2006)
5. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. J. Royal Statistical Society. Series B 39(1), 1–38 (1977)
6. Fawcett, T.: An introduction to ROC analysis. Pattern Recognition Letters 27(8), 861–874 (2006)
7. Fleischhacker, D., Völker, J.: Inductive learning of disjointness axioms. In: Confederated International Conferences On the Move to Meaningful Internet Systems. LNCS, vol. 7045, pp. 680–697. Springer (2011)
8. Luna, J.E.O., Revoredo, K., Cozman, F.G.: Learning probabilistic description logics: A framework and algorithms. In: Mexican International Conference on Artificial Intelligence. LNCS, vol. 7094, pp. 28–39. Springer (2011)
9. Riguzzi, F., Lamma, E., Bellodi, E., Zese, R.: Epistemic and statistical probabilistic ontologies. In: Uncertainty Reasoning for the Semantic Web. CEUR Workshop Proceedings, vol. 900, pp. 3–14. Sun SITE Central Europe (2012)
10. Riguzzi, F., Lamma, E., Bellodi, E., Zese, R.: Bundle: A reasoner for probabilistic ontologies. In: International Conference on Web Reasoning and Rule Systems. LNCS, Springer (2013)
11. Sato, T.: A statistical learning method for logic programs with distribution semantics. In: International Conference on Logic Programming. pp. 715–729 (1995)
12. Schmidt-Schauß, M., Smolka, G.: Attributive concept descriptions with complements. Artificial Intelligence 48(1), 1–26 (1991)
13. Völker, J., Niepert, M.: Statistical schema induction. In: Extended Semantic Web Conference. LNCS, vol. 6643, pp. 124–138. Springer (2011)