Learning the Parameters of Probabilistic Description Logics

Fabrizio Riguzzi[1], Elena Bellodi[2], Riccardo Zese[2], and Evelina Lamma[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. Uncertain information is ubiquitous in the Semantic Web, due to methods used for collecting data and to the inherently distributed nature of the data sources. It is thus very important to develop probabilistic Description Logics (DLs) so that the uncertainty is directly represented and managed at the language level. The DISPONTE semantics for probabilistic DLs applies the distribution semantics of probabilistic logic programming to DLs. In DISPONTE, axioms are labeled with numeric parameters representing their probability. These are often difficult to specify or to tune for a human. On the other hand, data is usually available that can be leveraged for setting the parameters. In this paper, we present EDGE that learns the parameters of DLs following the DISPONTE semantics. EDGE is an EM algorithm in which the required expectations are computed directly on the binary decision diagrams that are built for inference. Experiments on two datasets show that EDGE achieves higher areas under the Precision Recall and ROC curves than an association rule learner in a comparable or smaller time.

1 Introduction

Due to the ubiquity of uncertain information, many authors [9, 17, 8] have recently studied approaches to add uncertainty to the Semantic Web. Since Description Logics (DLs) are at the basis of the Semantic Web, in [12] we proposed the DISPONTE ("DIstribution Semantics for Probabilistic ONTologiEs", Spanish for "get ready") semantics. DISPONTE applies the distribution semantics of probabilistic logic programming [15] to DLs.

In [14] we presented an algorithm, called EDGE for "Em over bDds for description loGics paramEter learning", for learning the parameters of probabilistic DLs that follow the DISPONTE semantics. EDGE starts from examples of instances and non-instances of concepts and builds a set of Binary Decision Diagrams (BDDs) that represent their explanations. The parameters are then tuned using an EM algorithm [6] in which the required expectations are computed directly on the BDDs. In [14] the parameters learned by EDGE were compared with those given by the confidence of Association Rules (ARs for short in

the following) on a dataset extracted from `educational.data.gov.uk`. EDGE achieved significantly higher areas under the Precision Recall and the Receiver Operating Characteristics curves (AUCPR and AUCROC).

In this paper we extend the experiments presented in [14] by also considering a dataset extracted from DBPedia and by recording the time required by EDGE and by the computation of ARs' confidence. EDGE achieves again higher AUCPR and AUCROC. Moreover, the time taken by EDGE is comparable to the one required for computing ARs' confidence.

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

2 Description Logics and the DISPONTE semantics

DLs are particularly useful for representing ontologies and have been adopted as the basis of the Semantic Web. 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 $\mathcal{ALC}$ [16].

We use $\mathbf{A}$, $\mathbf{R}$ and $\mathbf{I}$ to indicate *atomic concepts*, *atomic roles* and *individuals*, respectively. A *role* is an atomic role $R \in \mathbf{R}$. *Concepts* are defined as follows. Each $A \in \mathbf{A}$, $\bot$ and $\top$ are concepts. If C, C_1 and C_2 are concepts and $R \in \mathbf{R}$, then $(C_1 \sqcap C_2)$, $(C_1 \sqcup C_2)$ and $\neg C$ are concepts, as well as $\exists R.C$ and $\forall R.C$.

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* $a : C$ and *role membership axioms* $(a, b) : R$. 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 mapping $\cdot^{\mathcal{I}}$ is extended to all concepts (where $R^{\mathcal{I}}(x) = \{y | (x, y) \in R^{\mathcal{I}}\}$ and $\#X$ denotes the cardinality of the set X) as:

$$\top^{\mathcal{I}} = \Delta^{\mathcal{I}} \qquad \bot^{\mathcal{I}} = \emptyset$$
$$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \qquad (C_1 \sqcap C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$$
$$(C_1 \sqcup C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}} \qquad (\forall R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} | R^{\mathcal{I}}(x) \subseteq C^{\mathcal{I}}\}$$
$$(\exists R.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} | R^{\mathcal{I}}(x) \cap C^{\mathcal{I}} \neq \emptyset\}$$

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

DISPONTE applies the distribution semantics to probabilistic ontologies [15]. In DISPONTE [2, 10–13] 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 $p :: E$, where p is a real number in $[0, 1]$ and E is a DL axiom. The idea of DISPONTE is to associate independent Boolean random variables with the axioms. Thus, a single random variable is associated with axiom E and p represents its probability of being true.

A DISPONTE KB defines a distribution over regular DL KB called *worlds*. Each world is obtained by including every certain axiom. For each probabilistic axiom, we decide whether or not to include it in the world. By multiplying the probability of the choices made to obtain a world we can assign a probability to it. The probability of a query is then the sum of the probabilities of the worlds where the query holds true.

3 EDGE

EDGE [14] is based on the algorithm EMBLEM [4, 3] 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 positive and negative examples that represent the queries in the form of concept assertions, i.e., of the form $a : C$ for an individual a and a class C. Positive examples represent information that we regard as true and for which we would like to get high probability while negative examples 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 the reasoner BUNDLE [13]. For a positive example of the form $a : C$, EDGE looks for the explanations of $a : C$ and encodes them in a BDD. For a negative example of the form $a : C$, EDGE looks for the explanations of $a : C$, encodes them in a BDD and negates it with the NOT BDD operator. 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 or until the maximum number of iterations is reached. The EM algorithm is guaranteed to find a local maximum, which however may not be the global maximum. The LL of the examples is guaranteed to increase at each iteration.

Function Expectation takes as input a BDD for each example Q, and computes $P(X_i = x|Q)$ for all the variables X_i in the BDD. Finally, it 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 that of the previous one drops below a threshold ϵ or when this difference is below a fraction δ of the previous LL. Function Maximization computes the parameters' values for the next EM iteration by relative frequency. For more details see [4].

The phase of explanations research for each example has high complexity in the worst case, since the explanations may grow exponentially in number; however, BUNDLE is able to handle domains of significant size. The EM phase has a linear cost in the number of nodes since the E-step requires two traversals of the diagram.

4 Related Work

CR$\mathcal{ALC}$ [9] is an extension of $\mathcal{ALC}$ that adopts an interpretation-based semantics for allowing statistical axioms of the form $P(C|D) = \alpha$ (for each element x in $\mathcal{D}$ that belongs to D, the probability that belongs also to C is α) and of the form $P(R) = \beta$ (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). 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 [9] learns parameters and structure of CR$\mathcal{ALC}$ knowledge bases. It starts from positive and negative examples for a single concept and learns the best probabilistic definition for the concept chosen 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.

GoldMiner [17, 8] is an algorithm that exploits ARs for building ontologies. GoldMiner extracts information about individuals, named classes and roles using SPARQL queries. From these data, it builds two *transaction tables*: one that stores the classes to which each individual belongs and one that stores the roles to which each couple of individuals belongs. Finally, the APRIORI algorithm [1] is applied to each table in order to find ARs. Implications of the form $A \Rightarrow B$ can be converted to subclass axioms of the form $A \sqsubseteq B$. Moreover, the confidence associated with ARs can be interpreted as the probability of the axiom $p :: A \sqsubseteq B$. So GoldMiner can be used to obtain a probabilistic knowledge base.

5 Experiments

EDGE has been compared with ARs over two real world datasets from the Linked Open Data cloud: `educational.data.gov.uk` and an extract of DBPedia. In our experiment, we wanted to simulate the situation in which an expert provides the structure of the ontology together with information on a set of individuals. The ontologies were obtained with GoldMiner: we extracted 10,000 individuals for `educational.data.gov.uk` and 7,200 for DBPedia and we learned ARs from the resulting transaction tables. The ARs were then converted into subclass axioms.

In order to generate a set of examples for EDGE, for each extracted individual a we sampled three named classes: A and B were sampled from the named classes to which a explicitly belonged, while C was sampled from the named classes to which a did not explicitly belong but that exhibited at least one explanation for the query $a : C$. Then, we randomly split individuals into two equally sized sets: the membership assertions regarding the individuals from the first set constituted the training set while the ones in the second set constituted the testing set. The axiom $a : A$ is added to the KB, while $a : B$ is considered as a positive example and $a : C$ as a negative example. The training set contained only the membership assertions for the first set of individuals, while for the testing phase

we removed the membership assertions of the training set from the KB and added the assertions of the second set.

We compared the parameters learned by EDGE with ARs' confidence. For each AR corresponding to the subclass axiom $A \sqsubseteq B$, we computed the confidence by running two SPARQL queries over the training KBs, one for finding all the individuals that belong to $A \sqcap B$ and one for those that belong to A. The confidence is then given by the ratio of the number of individuals in $A \sqcap B$ over those in A. We created 330 different SPARQL queries for `educational.data.gov.uk` and 2,243 for DBPedia.

Next we ran EDGE over the KBs where all the subclass axioms were assigned an initial random probability. We then computed the probability of the examples in the testing set according to the theory learned by EDGE and to the theory composed of the ARs with the confidence as probability. We drew the Precision-Recall and the Receiver Operating Characteristics curves and computed the Area Under the Curve (AUCPR and AUCROC) following the methods of [5, 7]. Table 1 shows the AUCPR, the AUCROC and the execution times (in seconds). Note that the elapsed time for EDGE depends on the number of executed queries and the number of different explanations involved in each query, while the elapsed time for ARs depends on the number of classes in the KB. EDGE achieves much higher areas in a time that is of the same or lower order of magnitude with respect to ARs.

Areas Under Curves		EDGE	ARs
educational.data.gov.uk	AUCPR	0.9702	0.8804
	AUCROC	0.9796	0.9158
	Time (s)	65,170	10,490
DBPedia	AUCPR	0.9784	0.5916
	AUCROC	0.9902	0.4346
	Time (s)	50,800	578,420

Table 1. Resulting AUCPR, AUCROC and execution times.

6 Conclusions

EDGE applies an EM algorithm for learning the parameters of probabilistic knowledge bases under the DISPONTE semantics. It 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`. The experiments over two real world datasets show that EDGE achieves larger areas both under the PR and the ROC curve with respect to an algorithm based on ARs in a comparable or smaller time, thus demonstrating that EDGE is a viable alternative to ARs.

We plan to extend EDGE for learning the structure together with the parameters.

7 Acknowledgements

Elena Bellodi is partially supported by Gruppo Nazionale per il Calcolo Scientifico, Istituto Nazionale di Alta Matematica "F. Severi".

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB 1994. pp. 487–499. Morgan Kaufmann (1994)
2. Bellodi, E., Lamma, E., Riguzzi, F., Albani, S.: A distribution semantics for probabilistic ontologies. In: URSW 2011. CEUR Workshop Proceedings, vol. 778. Sun SITE Central Europe (2011)
3. Bellodi, E., Riguzzi, F.: Experimentation of an expectation maximization algorithm for probabilistic logic programs. Intelligenza Artificiale 8(1), 3–18 (2012)
4. Bellodi, E., Riguzzi, F.: Expectation Maximization over binary decision diagrams for probabilistic logic programs. Intelligent Data Analysis 17(2), 343–363 (2013)
5. Davis, J., Goadrich, M.: The relationship between precision-recall and ROC curves. In: ICML 2006. pp. 233–240. ACM (2006)
6. 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)
7. Fawcett, T.: An introduction to ROC analysis. Pattern Recognition Letters 27(8), 861–874 (2006)
8. Fleischhacker, D., Völker, J.: Inductive learning of disjointness axioms. In: OTM Conferences 2011. LNCS, vol. 7045, pp. 680–697. Springer (2011)
9. Luna, J.E.O., Revoredo, K., Cozman, F.G.: Learning probabilistic description logics: A framework and algorithms. In: MICAI 2011. LNCS, vol. 7094, pp. 28–39. Springer (2011)
10. Riguzzi, F., Bellodi, E., Lamma, E.: Probabilistic Datalog+/- under the distribution semantics. In: DL 2012. CEUR Workshop Proceedings, vol. 846. Sun SITE Central Europe (2012)
11. Riguzzi, F., Bellodi, E., Lamma, E.: Probabilistic ontologies in Datalog+/-. In: CILC 2012. CEUR Workshop Proceedings, vol. 857, pp. 221–235. Sun SITE Central Europe (2012)
12. Riguzzi, F., Bellodi, E., Lamma, E., Zese, R.: Epistemic and statistical probabilistic ontologies. In: URSW 2012. CEUR Workshop Proceedings, vol. 900, pp. 3–14. Sun SITE Central Europe (2012)
13. Riguzzi, F., Bellodi, E., Lamma, E., Zese, R.: BUNDLE: A reasoner for probabilistic ontologies. In: Faber, W., Lembo, D. (eds.) RR 2013. LNCS, vol. 7994, pp. 183–197. Springer (2013)
14. Riguzzi, F., Bellodi, E., Lamma, E., Zese, R.: Parameter learning for probabilistic ontologies. In: Faber, W., Lembo, D. (eds.) RR 2013. LNCS, vol. 7994, pp. 265–270. Springer (2013)
15. Sato, T.: A statistical learning method for logic programs with distribution semantics. In: ICLP 1995. pp. 715–729 (1995)
16. Schmidt-Schauß, M., Smolka, G.: Attributive concept descriptions with complements. Artificial Intelligence 48(1), 1–26 (1991)
17. Völker, J., Niepert, M.: Statistical schema induction. In: ESWC 2011. LNCS, vol. 6643, pp. 124–138. Springer (2011)