Learning Probabilistic Description Logics

Fabrizio Riguzzi[2] Elena Bellodi[1] Evelina Lamma[1]
Riccardo Zese[1] Giuseppe Cota[1]

Dipartimento di Ingegneria – University of Ferrara

Dipartimento di Matematica e Informatica – University of Ferrara
[fabrizio.riguzzi,elena.bellodi,evelina.lamma,
riccardo.zese,giuseppe.cota]@unife.it

September 4, 2016

Outline

1. Probabilistic Description Logics
   - DISPONTE
2. EDGE: Learning DISPONTE Probabilities
3. Class Expression Learning
   - Definition
   - CELOE
4. Structure Learning
   - Put them all together!
5. And much more...
6. Experiments and Results
   - EDGE
   - LEAP
7. Conclusions

The semantics: DISPONTE

- In the real world the information is often **uncertain**
- The **Distribution Semantics** underlies many probabilistic logic programming languages, such as PRISM, ICL, LPADs, ProbLog
- **DISPONTE applies the Distribution Semantics of probabilistic logic programming to description logics**
- DISPONTE semantics
  - Probabilistic axioms: $p :: E$
    e.g., $p :: C \sqsubseteq D$ represents the fact that we believe in the truth of $C \sqsubseteq D$ with probability p.

Example

$$0.4 :: fluffy : Cat \quad (1)$$
$$0.3 :: tom : Cat \quad (2)$$
$$0.6 :: Cat \sqsubseteq Pet \quad (3)$$

$$\exists hasAnimal.Pet \sqsubseteq NatureLover \quad (4)$$
$$(kevin, fluffy) : hasAnimal \quad (5)$$
$$(kevin, tom) : hasAnimal \quad (6)$$

- $Q = kevin : NatureLover$ has two explanations:

$$\{ (1), (3) \}$$
$$\{ (2), (3) \}$$

- $P(Q) = 0.4 \times 0.6 \times (1 - 0.3) + 0.3 \times 0.6 = 0.348$
- $P(Q)$ can be computed with the inference system **BUNDLE**

EDGE: Em over bDds for description loGics paramEter learning

- **Supervised parameter learning algorithm**
- **Input:** a DISPONTE theory, a set of examples that represent queries
  - The queries are assertional axioms divided into:

    positive examples information that we regard as true for which we would like to get high probability;

    negative examples information that we regard as false for which we would like to get low probability.
- **Build the BDDs** of each example using BUNDLE
- **Enter the EM cycle**
  - **Expectation**
    - computes the expected counts by traversing twice the BDDs
  - **Maximization**
    - computes maximum likelihood parameters from the expected counts

Class Expression Learning

Definition

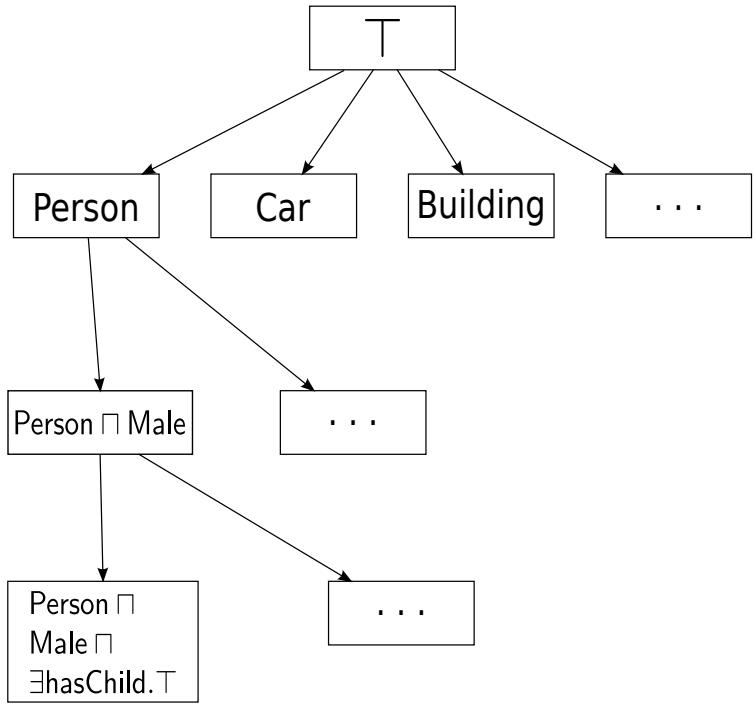
- Given a knowledge base $\mathcal{K}$
- Given a Target class contained in $\mathcal{K}$
- Given $R_{\mathcal{K}}(C)$ a retrieval function of the reasoner that returns all the instances of C

The class learning consists in finding an expression C such that:

$$R_{\mathcal{K}}(\text{Target}) = R_{\mathcal{K}}(C)$$

CELOE: Class Expression Learning for Ontology Engineering

- **C**lass **E**xpression **L**earning for **O**ntology **E**ngineering.
- Generates a set of Class Expressions C_i ordered according to an heuristic.
- Provides a semi-automatic approach to add axioms of the form $Target \equiv C_i$ or $Target \sqsubseteq C_i$.
- Top-down
- Inside the **DL-Learner** project

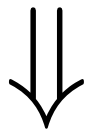


Put them all together!

- Our recipe:
  - BUNDLE: for Inference
  - EDGE: for parameter learning
  - CELOE: for candidate axioms generation

Put them all together!

- Our recipe:
  - BUNDLE: for Inference
  - EDGE: for parameter learning
  - CELOE: for candidate axioms generation



LEAP

Structure and parameter learner for probabilistic KBs under DISPONTE

LEAP: algorithm

1. Input: a knowledge base $\mathcal{K}$, a set of examples that represent **queries**

LEAP: algorithm

1. Input: a knowledge base $\mathcal{K}$, a set of examples that represent **queries**
2. Run CELOE to obtain a set C of candidate probabilistic subclass-of axioms

LEAP: algorithm

1. Input: a knowledge base $\mathcal{K}$, a set of examples that represent **queries**
2. Run CELOE to obtain a set C of candidate probabilistic subclass-of axioms
3. Run EDGE to compute the initial log-likelihood LL of $\mathcal{K}$

LEAP: algorithm

1. Input: a knowledge base $\mathcal{K}$, a set of examples that represent **queries**
2. Run CELOE to obtain a set C of candidate probabilistic subclass-of axioms
3. Run EDGE to compute the initial log-likelihood LL of $\mathcal{K}$
4. For each $ax \in C$

LEAP: algorithm

1. Input: a knowledge base $\mathcal{K}$, a set of examples that represent **queries**
2. Run CELOE to obtain a set C of candidate probabilistic subclass-of axioms
3. Run EDGE to compute the initial log-likelihood LL of $\mathcal{K}$
4. For each $ax \in C$
   1. Add the axiom ax to the knowledge base $\mathcal{K}$

LEAP: algorithm

1. Input: a knowledge base $\mathcal{K}$, a set of examples that represent **queries**
2. Run CELOE to obtain a set C of candidate probabilistic subclass-of axioms
3. Run EDGE to compute the initial log-likelihood LL of $\mathcal{K}$
4. For each $ax \in C$
   1. Add the axiom ax to the knowledge base $\mathcal{K}$
   2. Run EDGE on the new KB to compute the new parameters and the new log-likelihood LL'

LEAP: algorithm

1. Input: a knowledge base $\mathcal{K}$, a set of examples that represent **queries**
2. Run CELOE to obtain a set C of candidate probabilistic subclass-of axioms
3. Run EDGE to compute the initial log-likelihood LL of $\mathcal{K}$
4. For each $ax \in C$
   1. Add the axiom ax to the knowledge base $\mathcal{K}$
   2. Run EDGE on the new KB to compute the new parameters and the new log-likelihood LL'
   3. If $LL' > LL$ keep the axiom ax

LEAP: algorithm

1. Input: a knowledge base $\mathcal{K}$, a set of examples that represent **queries**
2. Run CELOE to obtain a set C of candidate probabilistic subclass-of axioms
3. Run EDGE to compute the initial log-likelihood LL of $\mathcal{K}$
4. For each $ax \in C$
   1. Add the axiom ax to the knowledge base $\mathcal{K}$
   2. Run EDGE on the new KB to compute the new parameters and the new log-likelihood LL'
   3. If $LL' > LL$ keep the axiom ax
   4. Else remove it

Systems and Algorithms

1. Parameter Learning
   - **EDGE^{MR}**: distributed version of EDGE
     - Examples divided among workers
     - Expectation step executed in parallel
     - Maximization phase executed by master process
2. Structure Learning
   - **LEAP** is now part of the DL-Learner Framework
   - **LEAP^{MR}**: distributed version of LEAP
     - Makes use of **EDGE^{MR}**

EDGE: Experiments

- Comparison with GoldMiner
- We used GoldMiner for generating ontologies from the linked open data cloud

educational.data.gov.uk

DBPedia

- We selected positive and negative examples by randomly choosing individuals from the extracted ones
- We ran EDGE on the ontologies obtained by GoldMiner where we consider all the subclass axioms as probabilistic

EDGE: Experiments

- We compared the results of EDGE with those obtained from an ontology in which the parameters are the confidences of the ARs
- We computed the probability of the queries using BUNDLE
- 5-fold cross validation
- Given the probability of examples, we drew the Precision-Recall and the Receiver Operating Characteristics curves and computed the Area Under the Curve

EDGE: Results

	edu.gov.uk		DBPedia	
	EDGE	ARs	EDGE	ARs
AUCPR	0.9702 ± 0.029	0.8804 ± 0.016	0.9784 ± 0.048	0.5916 ± 0.099
AUCROC	0.9796 ± 0.017	0.9158 ± 0.017	0.9902 ± 0.022	0.4346 ± 0.132

EDGE can achieve better areas under both the PR and ROC curves than association rules

LEAP: Experiments

- We used Carcinogenesis ontology which contains 22,372 individuals and 74,405 axioms
- We selected positive and negative examples by randomly choosing individuals for the class *Compound*
- We ran EDGE on the original KB for learning the parameters associated with the probabilistic axioms
- We separately ran LEAP on the original KB for learning probabilistic subsumption axioms for the class Compound and the associated parameters
  - Learned 1 probabilistic subsumption axiom

LEAP: Experiments

- We compared the results of EDGE with those obtained from LEAP.
- We computed the probability of the queries using BUNDLE.
- 5-fold cross validation
- Given the probability of examples, we drew the Precision-Recall and the Receiver Operating Characteristics curves and computed the Area Under the Curve

LEAP: Results

	EDGE	LEAP	p-value
AUCPR	0.5340 ± 0.108	0.8006 ± 0.240	0.0603
AUCROC	0.4452 ± 0.051	0.7980 ± 0.246	0.0360

LEAP can achieve better areas under both the PR and ROC curves, with the difference in AUCROC being statistically significant at the 5% significance level

Conclusions and Future Work

- Conclusion
  - We have presented two learning algorithms: EDGE and LEAP
  - They are at the basis of EDGE^{MR} and LEAP^{MR}: distributed versions of EDGE and LEAP
- Future Work
  - In LEAP^{MR}, the execution of CELOE is **not distributied**
  - Future plan: distributing the scoring of the refinements generated during the execution of CELOE

THANKS FOR LISTENING AND ANY QUESTIONS ?