# Learning from Linked Data by Markov Logic

Man Zhu, Zhiqiang Gao

School of Computer Science & Engineering,
Southeast University, Nanjing 211189, P.R. China
{mzhu,zqgao}@seu.edu.cn

**Abstract.** As the 'Web of Linked Data' vision of the Semantic Web is coming true, the 'explosion' of Linked Data provides more than sufficient data for Description Logic learning algorithms in terms of quantity. However, noises arise as an inevitable issue to the effectiveness of these algorithms. To cope with *noises* in Linked Data, we propose to learn $\mathcal{ALC}$ concept definitions from Linked Data by Markov logic. In this paper, we describe our approach and report the evaluations using a small illustrative data set, and four larger data sets. Experiment results show that our approach performs well on noisy data, and is applicable for learning $\mathcal{ALC}$ concept definitions.

## 1 Introduction

As the 'Web of Linked Data' vision of the Semantic Web is coming true, the size of Linked Data [1] kept growing over last years. According to the report of the year 2009 [2], there have been over 6 billion RDF triples, and over 148 million links in Linked Data. They provide more than sufficient data for Description Logic learning algorithms in terms of quantity.

Although the RDF triples can provide plenty of examples for learning algorithms, as argued by Auer and Lehmann [1], many data sets on the Linked Data lack rich knowledge representation and contain noises. Moreover, in [3] d'Amato et al. proposed the problem of handling the uncertainties on the Web. Learning from Linked Data is both worth-investigating and challenging.

Several approaches have been proposed for learning from Linked Data. In [12], Völker and Niepert propose a statistical approach, to be specific, association rule mining, for learning OWL 2 $\mathcal{EL}$ from Linked Data. Lehmann J. et al. have done a series of work on learning description logics, which have been implemented in DLs learning tool DL-Learner. DL-Learner includes four class description learning algorithms, namely CELOE, random guesser learning algorithm, ISLE, brute force learning algorithm. These algorithms select class expressions according to heuristics [7]. AutoSPARQL is a most recent work, which makes use of the individual assertions in the ABox, and is able to learn description for individuals [8]. However, they seldom focus on handling noises in Linked Data.

[1] The Linked Data project aims to expose, share and connect related data from diverse sources on the Semantic Web, based on URIs, HTTP and RDF.

Hogan et al. analyzed the types of noises exist in the Linked Data [6]. We are particularly interested in handling two types of noises: *incompleteness* and *error*. Incompleteness means that concept assertions or the relationships between named individuals are actually true but missed, and error means that the RDF triples are not correct. Take a family ontology for example. The declaration of Heinz is a father and Heinz is a male are existed in the RDF triples, then Heinz should have a child, however it is not declared in the ontology. This is an example of incompleteness. On the contrary, if we know Anna has a child, and she is a female, Anna should not be a father, but in the ontology Anna is incorrectly declared to be a father. This illustrates the error case.

We propose to learn $\mathcal{ALC}$ axioms inductively from Linked Data based on Statistical Relational Learning (SRL) models, to be specific, Markov logic [11]. SRL is an emerging area of machine learning, and it attempts to represent, reason, and learn in domains with complex relational and rich probabilistic structure [5]. As a combination of first-order logic and Markov network, Markov logic is able to handle Description Logic $\mathcal{ALC}$ in terms of expressing power, which is subsumed by first-order logic [4]. Markov logic soften the hard constraints made by first-order KB by introducing weights to be associated with formulas indicating the strongness of the constraints. Using Markov logic, two challenges of learning from Linked Data can be easily handled: 1) Linked Data are highly structured due to the relations between entities and the underlying ontology. 2) Linked Data contains noises, here, as described above, we refer particularly to incompleteness and error.

## 2 Learning from Linked Data

Similar to the SEQUENTIAL-COVERING algorithm, the $\mathcal{ALC}$ learning algorithm contains a loop. In the loop, two operations are conducted. Firstly, the algorithm generates candidates with $\mathcal{ALC}$ constructors according to the current concept (c.f. Sect. 2.1), named oldconcept . Secondly, according to a function which selects the 'best' one from the candidates according to the performance measure (c.f. Sect. 2.2). The loop breaks when the performance of the new candidate is not better than the previous one.

**Algorithm 1:** Learning Algorithm

```
input  : target concept $T$, $KB$
output: concept $C$ satisfying $T \equiv C$
1 current ← $\top$;
2 do
3     oldconcept ← current;
4     candidates ← GenerateCandidates($T$, oldconcept);
5     current ← BestWeightConcept($T$, $KB$,candidates);
6 while performance(oldconcept)<performance(current);
7 return current;
```

## 2.1 Generating Candidate Specializations

We use the $\mathcal{ALC}$ downward refinement operator proposed by Lehmann J. [9] to generate candidate specializations. Refinement operator is a mapping $S \mapsto 2^S$ on a quasi-ordered space $S$. Subsumption is a quasi-ordering relation. The learning algorithm conducts a general to specific search over the hypotheses space. According to our observations, refinement operator is particularly suitable for generating candidate specializations because it is well-founded, and several good properties of it ensure the learning process will return the result if there is.

The candidates are generated by an iterative procedure, which starts from a base set $\mathcal{B}$. $\mathcal{B}$ includes most general atomic concepts, negated most specific atomic concepts, $\{\exists r.\top | r$ is an atomic role$\}$. The details of the refinement operator can be found from Table 1.

**Table 1.** Downward refinement operator $\rho_{\downarrow}(C)$

| if $C$ is | $\rho_{\downarrow}(C)$ |
|---:|:---|
| $\bot$ | $\emptyset$ |
| $\top$ | $\{C_1 \sqcup \ldots \sqcup C_n \| C_i \in \mathcal{B} (1 \leq i \leq n)\}$ |
| $A(A \in A_C)$ | $\{A' \| A' \in sh_{\downarrow}(A)\} \cup \{A \sqcap D \| D \in \rho'_{\downarrow}(\top)\}$ |
| $\neg A(A \in A_C)$ | $\{\neg A' \| A' \in sh_{\uparrow}(A)\} \cup \{\neg A \sqcap D \| D \in \rho'_{\downarrow}(\top)\}$ |
| $\exists r.D$ | $\{\exists r.E \| E \in \rho'_{\downarrow}(D)\} \cup \{\exists r.D \sqcap E \| E \in \rho'_{\downarrow}(\top)\}$ |
| $\forall r.D$ | $\{\forall r.E \| E \in \rho'_{\downarrow}(D)\} \cup \{\forall r.D \sqcap E \| E \in \rho'_{\downarrow}(\top)\}$ $\cup \{\forall r.\bot \| D = A \in A_C$ and $sh_{\downarrow}(A) = \emptyset\}$ |
| $C_1 \sqcap \ldots \sqcap C_n$ $(n \geq 2)$ | $\{C_1 \sqcap \ldots \sqcap C_{i-1} \sqcap D \sqcap C_{i+1} \sqcap \ldots \sqcap C_n \|$ $D \in \rho'_{\downarrow}(C_i), 1 \leq i \leq n\}$ |
| $C_1 \sqcup \ldots \sqcup C_n$ $(n \geq 2)$ | $\{C_1 \sqcup \ldots \sqcup C_{i-1} \sqcup D \sqcup C_{i+1} \sqcup \ldots \sqcup C_n \|$ $D \in \rho'_{\downarrow}(C_i), 1 \leq i \leq n\}$ $\cup \{(C_1 \sqcup \ldots \sqcup C_n) \sqcap D \| D \in \rho'_{\downarrow}(\top)\}$ |

## 2.2 Guiding the Search

At each step, the candidate concept descriptions for $Target$ are assumed to be uncertain, and they construct a Markov logic network (MLN) together with their weights $\{(d_i, w_i)\}$. Given the RDF triples in Linked Data, the MLN can be instantiated to be a Markov network $M_{L,C}$, in which each binary node is a grounding of a predicate and each feature $f_{i,j}$ is a grounding of one of the concept descriptions. $M_{L,C}$ therefore encodes the joint probability distribution $P(X = x) = \frac{1}{Z} \exp\left(\sum_i w_i n_i\right)$, where $n_i = \sum_j \mathbf{1}\{f_{i,j}$ is true$\}$. In order to make both inferencing and learning tractable, this joint probability distribution is always approximated by pseudo-log-likelihood:

$$\log P^*_w(X = x) = \sum_{l=1}^{n} \log P_w(X_l = x_l | MB_x(X_l)) \tag{1}$$

After taking derivative of (1), we get

$$\begin{aligned}\frac{\partial}{\partial w_i}\log P_w^*(X=x) = \sum_{l=1}^{n}[n_i(x) - P_w(X_l=0|MB_x(X_l))n_i(x_{[X_l=0]}) \\ -P_w(X_l=1|MB_x(X_l))n_i(x_{[X_l=1]})]\end{aligned} \qquad (2)$$

In MLN, the weight represents the relative strength or importance of the class description (rule) [10]. The higher the weight, the greater the difference in log probability between a world that satisfies the concept description and the one that does not [4]. At each step, the concept description with the highest weight is selected. To select the most promising candidate from the candidates generated at each step, the weight vector is learned by L-BFGS algorithm, a quasi-Newton method suitable for large-scale optimization. Inasmuch as the joint probability distribution is a good indicator of the performance of the selected concept description, the iteration stops when the pseudo-log-likelihood stops to increase when more specific description is selected.

## 3 Experiments

We adopt the measures frequently used in information retrieval domain for the evaluations, namely precision, recall and F1-score. The experiments are conducted on two types of data sets. With the illustrative data sets (which are smaller compared to the other type), we focus on analyzing the capability of the approach on handling noises. As an illustrative example, we choose a small data set from the examples of DL-Learner[2], and do small modifications on it to show the noisy cases, which are difficult to be illustrated on large data sets. As for the real-world ontology, we use four ontologies, named Semantic Bible ontology, Adhesome[3] ontology, financial ontology (obtained from DL-Learner), and SC[4] ontology.

### 3.1 Results

The statistics of the data sets are shown in Table 2.

**Learn Definitions** To show the performance on noisy cases, we modify the ontology and add two positive examples father(anna) and father(heinz) separately to correspond to error and incompleteness cases, and build two ontologies named family_error and family_incomplete [5]. Using DL-Learner GUI (version 2010-08-07) with CELOE algorithm under default settings for class learning problem, the concept description for father can be correctly learned from family.owl to be

[2] http://aksw.org/Projects/DLLearner
[3] http://www.sbcny.org/datasets/adhesome.owl
[4] http://www.mindswap.org/ontologies/SC.owl
[5] http://research.aturstudio.com/family_noisy/

**Table 2.** Statistics of the data sets for evaluation. The statistics include counts for concept, instance, object property, inclusion axiom, and the DL expressivity of the ontology.

| | # concept | | # inst | # object property | # inclusion axiom | # DL expressivity |
|---|---|---|---|---|---|---|
| | 0 inst | >0 inst | | | | |
| Semantic Bible | 1 | 49 | 724 | 29 | 51 | $\mathcal{SHOIN}(\mathcal{D})$ |
| Adhesome | 60 | 22 | 3032 | 66 | 216 | $\mathcal{ALCHN}(\mathcal{D})$ |
| SC | 1 | 29 | 3542 | 8 | 10 | $\mathcal{ALH}+(\mathcal{D})$ |
| financial | 12 | 49 | 17941 | 16 | 55 | $\mathcal{ALCOF}$ |

$male \sqcap \exists hasChild.Thing$, but from family_error.owl and family_incomplete.owl the concept description cannot be learned correctly. However, our learner is able to learn correct concept definition for concept father from all these ontologies.

**Learn inclusions** The evaluation results are shown in Table 3. We use positive examples for target concept to learn inclusions. On Adhesome, the recall is only 0.1852, which can be explained from two views: on the one hand, Adhesome ontology contains more concepts with zero instances, thus data are not sufficient for the learner to go. On the other hand, the Adhesome ontology is expressive and contains number restrictions, which cannot be expressed by $\mathcal{ALC}$. Since SC ontology contain large amount of instances compared with the counts of concepts, our learner tend to select restrictions prior to atomic concepts, which have lead to a very low precision.

**Table 3.** Learning results

| data sets | precision | recall | F1-score |
|---|---|---|---|
| Adhesome | 0.8333 | 0.1852 | 0.3030 |
| Semantic Bible | 0.8958 | 0.9302 | 0.9127 |
| SC | 0.2121 | 0.7000 | 0.3256 |
| financial | 0.7895 | 0.5455 | 0.6452 |

## 4 Conclusion and Future Works

To handle the noises generally exist in Semantic Web data, to be specific, Linked Data, we propose a naive but simple approach for learning Description Logic $\mathcal{ALC}$ inductively from noisy data by Markov logic. Our method conducts a general-to-specific search, which generates progressively more specific concepts until a sufficiently 'accurate' axiom is found. During the candidate selecting process, we transform the selection problem into finding the candidate with the highest weight, which can be viewed as an indicator of the degree of consistency between the candidates and the facts in the knowledge base. Compared to other works of learning description logics, our approach is more insensitive to noises, thus can handle noisy cases quite well. We evaluate the approach on an illustrative data set. The result shows that our approach is able to work well under noises. Furthermore experiments on four real-world data sets are also conducted, and results demonstrate the effectiveness of our method.

Since this is an exploratory work on adopting statistical relational learning approaches to Description Logic learning problems, in-depth studies are still in need. We find the following issues worth investigating:

– In practice, not all examples are needed by the learning algorithms. To improve efficiency without losing accuracy, sampling methods are worth to be explored and will be used by our algorithm in the future.
– Semantic Web makes open world assumption, which means we are unable to know the truth value of a proposition if it is not inferred to be true. However, inductive learning algorithms, like our algorithm, always make closed world assumption. This issue will also be studied in the future.

## References

1. S. Auer and J. Lehmann. Making the web a data washing machine - creating knowledge out of interlinked data. *Semantic Web Journal*, 1(1):97–104, 2010.
2. C. Bizer. The emerging web of linked data. *IEEE Intelligent Systems*, pages 87–92, 2009.
3. d'Amato C., Fanizzi N., and Esposito F. Inductive learning for the semantic web: What does it buy? *Semantic Web*, 1(1-2):53–59, 2010.
4. P. Domingos and D. Lowd. Markov logic: An interface layer for artificial intelligence. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 3(1):1–155, 2009.
5. L. Getoor and B. Taskar. *Introduction to statistical relational learning.* The MIT Press, 2007.
6. A. Hogan, A. Harth, A. Passant, S. Decker, and A. Polleres. Weaving the pedantic web. In *3rd International Workshop on Linked Data on the Web (LDOW2010), in conjunction with 19th International World Wide Web Conference, CEUR*, 2010.
7. J. Lehmann, S. Auer, et al. Class expression learning for ontology engineering. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2011.
8. J. Lehmann and L. Bühmann. Autosparql: Let users query your knowledge base. *The Semantic Web: Research and Applications*, pages 63–79, 2011.
9. J. Lehmann and P. Hitzler. Concept learning in description logics using refinement operator. *Machine Learning*, 78(1):203–250, 2010.
10. D. Lowd and P. Domingos. Efficient weight learning for markov logic networks. *Knowledge Discovery in Databases: PKDD 2007*, pages 200–211, 2007.
11. M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1):107–136, 2006.
12. J. Völker and M. Niepert. Statistical schema induction. *The Semantic Web: Research and Applications*, pages 124–138, 2011.