An ILP Approach to Learning Inclusion Axioms in Fuzzy Description Logics

F.A. Lisi and U. Straccia

CILC 2011

Motivation

Knowledge is inherently

1. structured
   - description in terms of objects and relations between objects
2. incomplete
   - partial description
3. vague
   - imprecise description

in many real-world domains.

Examples

1. Multimedia information retrieval
   - E.g., “Find top-k image regions about Gabriele D’Annunzio"
2. Database query
   - E.g., “Find top-k cheapest hotels close to Pescara campus"
3. Decision support
   - E.g., the notions of temperature, pulse and respiratory rate in medicine

Objective

We want to learn the conceptual descriptions of a target concept, given

1. the data stored into a relational database as fuzzy sets
2. the background knowledge about the application domain described via a standard ontology language

Fuzzy Description Logics

- Description Logics (DLs)
  - Family of KR formalisms for incomplete structured knowledge
  - Decidable fragments of FOL
  - Expressive power depending on the set of constructors
  - Very expressive DLs at the basis of the W3C OWL 2 standard language for ontologies
- Mathematical Fuzzy Logic
  - Theoretical foundation of KR formalisms for vague knowledge
  - Truth of statements is a matter of degree (*score*) measured on an ordered scale ($[0,1]$)
  - A *fuzzy interpretation* $\mathcal{I}$ maps each basic statement p_i into $[0,1]$ and is then extended inductively to all statements
  - A *fuzzy set* R over a countable crisp set X is a function $R\colon X \to [0,1]$

Uncertainty vs. Vagueness

- Uncertainty: "Emanuela Orlando is in Turkey to degree 0.83"
- Vagueness: "Pescara Dannunziana B&B is close to Pescara campus to degree 0.83"
- Uncertainty + vagueness: "It is possible/probable to degree 0.83 that it will be hot tomorrow"

Fuzzy DL-Lite[2]

- DL-Lite[1]
  - DL behind the *OWL 2 QL* profile
  - Especially aimed at data intensive applications
  - Tractable query answering
- Fuzziness with Gödel logic
  - $a \otimes b = \min(a, b)$
  - $a \oplus b = \max(a, b)$
  - $\ominus a = \begin{cases} 1 & \text{if } a \leqslant b \\ b & \text{otherwise} \end{cases}$
  - $a \Rightarrow b = \begin{cases} 1 & \text{if } a = 0 \\ 0 & \text{otherwise} \end{cases}$
- Ontology-based access to a relational database
- Implemented in the SoftFacts system (http://www.straccia.info/software/SoftFacts/SoftFacts.html)

[1] (Calvanese et al., 2006)
[2] (Straccia, 2010)

SoftFacts: Knowledge base

- $\mathcal{F}$ is a finite set of *expressions* of the form

$$R(c_1, \ldots, c_n)[s] \ , \tag{1}$$

  where:
  - R is an n-ary relation
  - every c_i is a constant
  - s is the score
- $\mathcal{O}$ is a finite set of *inclusion axioms* having the form

$$Rl_1 \sqcap \ldots \sqcap Rl_m \sqsubseteq Rr \ , \tag{2}$$

  where
  - $m \geqslant 1$
  - all Rl_i (*left-hand relation*) and Rr (*right-hand relation*) have the same arity
- $\mathcal{A}$ is a finite set of *statements* of the form

$$R \mapsto (c_1, \ldots, c_n)[c_{score}].sql \ , \tag{3}$$

  where sql is a SQL statement returning n-ary tuples $\langle c_1, \ldots, c_n \rangle$ ($n \leqslant 2$) with score determined by c_{score}.

SoftFacts: Query answering

- A *ranking query* is a conjunctive query of the form

$$q(\mathbf{x})[s] \quad \leftarrow \quad \begin{array}{l} \exists \mathbf{y}\ R_1(\mathbf{z}_1)[s_1], \ldots, R_l(\mathbf{z}_l)[s_l], \\ \text{OrderBy}(s = f(s_1, \ldots, s_l, p_1(\mathbf{z}'_1), \ldots, p_h(\mathbf{z}'_h)) \end{array} \tag{4}$$

  where

  - q is an n-ary relation and every R_i is an n_i-ary relation ($1 \leqslant n_i \leqslant 2$).
  - $\mathbf{x}$ and $\mathbf{y}$ are the *distinguished* and the *non-distinguished* variables
  - $\mathbf{z_i}, \mathbf{z'_j}$ are tuples of constants or variables in $\mathbf{x}$ or $\mathbf{y}$;
  - $s, s_1, \ldots, s_l$ are distinct variables and different from those in $\mathbf{x}$ and $\mathbf{y}$;
  - p_j is an n_j-ary *fuzzy predicate* assigning to each n_j-ary tuple $\mathbf{c}_j$ a *score* $p_j(\mathbf{c}_j) \in [0, 1]$.
  - f is a *scoring* function $f : ([0, 1])^{l+h} \to [0, 1]$, which combines the scores of the l relations $R_i(\mathbf{c}'_i)$ and the n fuzzy predicates $p_j(\mathbf{c}''_j)$ into an overall score s to be assigned to the query head $q(\mathbf{c})$.
- The *answer set* $ans_{\mathcal{K}}(q)$ over $\mathcal{K}$ of a query q is the set of tuples $\langle \mathbf{t}, s \rangle$ such that $\mathcal{K} \models q(\mathbf{t})[s]$ with $s > 0$
  - Informally, $\mathbf{t}$ satisfies the query to non-zero degree s) and the score s is as high as possible, *i.e.* if $\langle \mathbf{t}, s \rangle \in ans_{\mathcal{K}}(q)$ then (i) $\mathcal{K} \not\models q(\mathbf{t})[s']$ for any $s' > s$; and (ii) there cannot be another $\langle \mathbf{t}, s' \rangle \in ans_{\mathcal{K}}(q)$ with $s > s'$.

Learning Fuzzy DL-Lite Inclusion Axioms

- the *target concept* H is a DL-Lite atomic concept;
- the *background theory* $\mathcal{K}$ is a fuzzy DL-Lite knowledge base $\langle \mathcal{F}, \mathcal{O}, \mathcal{A} \rangle$
- the *training set* $\mathcal{E}$ is a collection of fuzzy DL-Lite like facts of the form (1) and labeled as either positive or negative examples for H. We assume that $\mathcal{F} \cap \mathcal{E} = \emptyset$;
- the *target theory* $\mathcal{H}$ is a set of inclusion axioms of the form

$$B \sqsubseteq H \quad (5)$$

where H is an atomic concept, $B = C_1 \sqcap \ldots \sqcap C_m$, and each concept C_i has syntax

$$C \longrightarrow A \mid \exists R.A \mid \exists R.\top \ . \quad (6)$$

A FOIL-like algorithm

- The *coverage relation* for a concept $C \neq H$

$$\mathcal{I}_{ILP} \models C(t) \text{ iff } \mathcal{K} \cup \mathcal{E} \models C(t)[s] \text{ and } s > 0 . \qquad (7)$$

- The *confidence degree* of an inclusion axiom is:

$$cf(B \sqsubseteq H) = \frac{\sum_{t \in P} B(t) \Rightarrow H(t)}{|D|} \qquad (8)$$

  where
  - $P = \{t \mid \mathcal{I}_{ILP} \models C_i(t) \text{ and } H(t)[s] \in \mathcal{E}^+\}$;
  - $D = \{t \mid \mathcal{I}_{ILP} \models C_i(t) \text{ and } H(t)[s] \in \mathcal{E}\}$;
  - $B(t) \Rightarrow H(t)$ denotes the degree to which the implication holds for the instance t;
  - $B(t) = \min(s_1, \ldots, s_n)$, with $\mathcal{K} \cup \mathcal{E} \models C_i(t)[s_i]$;
  - $H(t) = s$ with $H(t)[s] \in \mathcal{E}$.
- The *information gain* function uses the above formulas

$$\text{Gain}(cf(r'), cf(r)) = p * (log_2\, cf(r') - log_2\, cf(r)) \ ,$$

  where p is the number of distinct positive examples covered by the inclusion axiom r that are still covered by r'.

Learning Set of Inclusion Axioms

function FOIL-Learn-Set-of-Axioms(H, $\mathcal{E}^+$, $\mathcal{E}^-$, $\mathcal{K}$): $\mathcal{H}$
begin
1. $\mathcal{H} \leftarrow \emptyset$;
2. **while** $\mathcal{E}^+ \neq \emptyset$ **do**
3. $\quad r \leftarrow$ FOIL-Learn-One-Axiom(H, $\mathcal{E}^+$, $\mathcal{E}^-$, $\mathcal{K}$);
4. $\quad \mathcal{H} \leftarrow \mathcal{H} \cup \{r\}$;
5. $\quad \mathcal{E}^+_r \leftarrow \{e \in \mathcal{E}^+ | \mathcal{K} \cup r \models e\}$;
6. $\quad \mathcal{E}^+ \leftarrow \mathcal{E}^+ \setminus \mathcal{E}^+_r$;
7. **endwhile**
8. **return** $\mathcal{H}$
end

Learning One Inclusion Axiom

function FOIL-Learn-One-Axiom(H, $\mathcal{E}^+$, $\mathcal{E}^-$, $\mathcal{K}$): r
begin
$B(\mathbf{x}) \leftarrow \top$;
$r \leftarrow \{B(\mathbf{x}) \to H(\mathbf{x})\}$;
$\mathcal{E}_r^- \leftarrow \mathcal{E}^-$;
while $cf(r) < \theta$ **and** $\mathcal{E}_r^- \neq \emptyset$ **do**
    $B_{best}(\mathbf{x}) \leftarrow B(\mathbf{x})$;
    $maxgain \leftarrow 0$;
    foreach $l \in \mathcal{K}$ **do**
        $gain \leftarrow \mathrm{Gain}(cf(B(\mathbf{x}) \wedge l(\mathbf{x}) \to H(\mathbf{x})), cf(B(\mathbf{x}) \to H(\mathbf{x})))$;
        if $gain \geqslant maxgain$ **then**
            $maxgain \leftarrow gain$;
            $B_{best}(\mathbf{x}) \leftarrow B(\mathbf{x}) \wedge l(\mathbf{x})$;
        endif
    endforeach
    $r \leftarrow \{B_{best}(\mathbf{x}) \to H(\mathbf{x})\}$;
    $\mathcal{E}_r^- \leftarrow \mathcal{E}_r^- \setminus \{e \in \mathcal{E}^- | \mathcal{K} \cup r \models e\}$;
endwhile
return r
end

A Refinement Operator

1. Add atomic concept (A)
2. Add complex concept by existential role restriction ($\exists R.\top$)
3. Add complex concept by qualified existential role restriction ($\exists R.A$)
4. Replace atomic concept (A replaced by A' if $A' \sqsubseteq A$)
5. Replace complex concept ($\exists R.A$ replaced by $\exists R.A'$ if $A' \sqsubseteq A$)

Example of Hotel Classification: The database

HotelTable		
id	rank	noRooms
h1	3	21
h2	5	123
h3	4	95

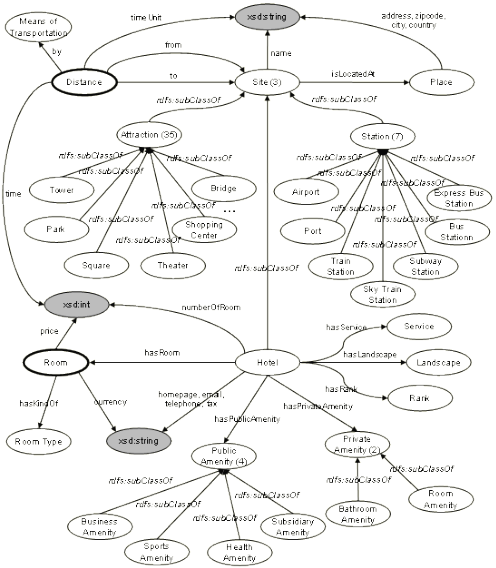
RoomTable			
id	price	roomType	hotel
r1	60	single	h1
r2	90	double	h1
r3	80	single	h2
r4	120	double	h2
r5	70	single	h3
r6	90	double	h3

Tower
id
t1

Park
id
p1
p2

DistanceTable			
id	from	to	time
d1	h1	t1	10
d2	h2	p1	15
d3	h3	p2	5

Example of Hotel Classification: The ontology



$Park \sqsubseteq Attraction$
$Tower \sqsubseteq Attraction$
$Attraction \sqsubseteq Site$
$Hotel \sqsubseteq Site$

Example of Hotel Classification: The abstraction statements

$Hotel \mapsto (h.id)$. SELECT h.id
FROM HotelTable h

$hasRank \mapsto (h.id, h.rank)$. SELECT h.id, h.rank
FROM HotelTable h

$cheapPrice \mapsto (h.id, r.price)[score]$. SELECT h.id, r.price, *cheap*(r.price) AS score
FROM HotelTable h, RoomTable r
WHERE h.id = r.hotel
ORDER BY score

$closeTo \mapsto (from, to)[score]$. SELECT d.from, d.to *closedistance*(d.time) AS score
FROM DistanceTable d
ORDER BY score

$cheap(p) = leftshoulder(p; 50, 100)$
$closedistance(d) = leftshoulder(d; 5, 25)$

Example of Hotel Classification: The learning problem

- $H = GoodHotel$
- $\mathcal{E}^+ = \{GoodHotel(h1)[0.6], GoodHotel(h2)[0.8]\}$
- $\mathcal{E}^- = \{GoodHotel(h3)[0.4]\}$.
- $r_0 : \top \sqsubseteq GoodHotel$
  $r_1 : Hotel \sqsubseteq GoodHotel$
  $r_2 : Hotel \sqcap \exists cheapPrice.\top \sqsubseteq GoodHotel$
  $r_3 : Hotel \sqcap \exists cheapPrice.\top \sqcap \exists closeTo.Attraction \sqsubseteq GoodHotel$
  $r_4 : Hotel \sqcap \exists cheapPrice.\top \sqcap \exists closeTo.Park \sqsubseteq GoodHotel$
  $r_5 : Hotel \sqcap \exists cheapPrice.\top \sqcap \exists closeTo.Tower \sqsubseteq GoodHotel$
- Consequence:
  - $cf(r_3) = \frac{0.75 \Rightarrow 0.6 + 0.4 \Rightarrow 0.8}{3} = \frac{0.6+1.0}{3} = 0.5333$.
  - $cf(r_4) = \frac{0.4 \Rightarrow 0.8}{2} = \frac{0.4}{2} = 0.2$.
  - $cf(r_5) = \frac{0.8 \Rightarrow 0.6}{2} = \frac{0.6}{2} = 0.3$.
  - $\mathrm{Gain}(r_4, r_3) = 1 * (log_2(0.2) - log_2(0.5333)) = (-2.3219 + 0.907) = -1.4149$
  - $\mathrm{Gain}(r_5, r_3) = 1 * (log_2(0.3) - log_2(0.5333)) = (-1.7369 + 0.907) = -0.8299$
  - r_5 preferred to r_4 as refinement of r_3
  - r_5 turns out to be consistent w.r.t. $\mathcal{E}$
  - r_5 becomes part of $\mathcal{H}$

Related work

- (Shibata et al., 1999; Drobics et al., 2003; Serrurier and Prade, 2007) propose FOIL-like algorithms to learn fuzzy rules.
- (Horváth and Vojtás, 2006) provides a formal study of fuzzy ILP
  - Less promising than our proposal from the practical side.
- (Hellmann et al., 2009) faces the problem of inducing equivalence axioms in a fragment of OWL corresponding to the $\mathcal{ALC}$ DL.
- (Konstantopoulos and Charalambidis, 2010) is based on an ad-hoc translation of fuzzy Łukasiewicz $\mathcal{ALC}$ DL constructs into LP and then uses a conventional ILP method to lean rules.
  - The method is not sound as it has been recently shown that the traduction from fuzzy DLs to LP is incomplete (Motik and Rosati, 2007) and entailment in Łukasiewicz $\mathcal{ALC}$ is undecidable (Cerami and Straccia, 2011).

Conclusions

- Method for inducing fuzzy DL-Lite inclusion axioms
- Extension of FOIL in a twofold direction
  - from crisp to fuzzy
  - from rules to inclusion axioms

Ongoing work

- To investigate the formal properties of the refinement operator
- To investigate the impact of OWA on the proposed ILP approach

Future work

- To implement and experiment our method
- To analyse the effect of the different implication functions and other parameters in the learning process