Towards the Induction of Terminological Decision Trees

Nicola Fanizzi
Computer Science Dept.
University of Bari, Italy
fanizzi@di.uniba.it

Claudia d'Amato
Computer Science Depart.
University of Bari, Italy
claudia.damato@di.uniba.it

Floriana Esposito
Computer Science Dept.
University of Bari, Italy
esposito@di.uniba.it

ABSTRACT

A concept learning framework for terminological representations is introduced. It is grounded on a method for inducing logic decision trees as an adaptation of the classic tree induction methods to the Description Logics representations adopted in the Semantic Web context. Differently from the original setting of logical trees based on clausal representations, tree-nodes contain terminological concept descriptions (corresponding to OWL-DL classes) which makes it appealing for the Semantic Web applications. The method has been implemented in a prototypical system which has been experimentally evaluated on real ontologies.

1. INTRODUCTION

The problems automated knowledge acquisition in the *Semantic Web* (SW) context has recently gained lots of attention: besides of learning *for* the SW, the question is also how to learn *from* the SW. Most of the works on this problem focus on *ontology learning* [13, 6], *ontology matching and alignment* [9] etc. Only few methods on enrichment of existing ontologies have been proposed [2, 8, 10]. In this paper, we propose a method for learning *Description Logics* (DLs) concept definitions. the method is grounded on the induction of decision trees [14] with the interesting side-effect of requiring new intermediate concepts for the knowledge base (KB). Moreover, the induced decision trees can be used for classifying individuals of an ontology w.r.t. a query concept, as proposed in [8], where an inductive instance-based learning method is adopted for classifying individuals.

The induction of *decision trees* [14] is a classic machine learning problem with a plenty of successful solutions proposed. Decision trees have been extended from the classic attribute-value representation to a multi-relational representation (*logical decision trees*) admitting the test of predicates on the tree-nodes. Suitable methods for learning logical trees have been proposed [3]. Our goal is to extend the representation of the logic decision trees to DLs languages and propose solutions for their induction.

SAC'10 March 22-26, 2010, Sierre, Switzerland.

DLs constitute a family of languages underlying the standard ontology languages [1] designed for the SW. The family is composed by (decidable) fragments of First Order Logic (FOL) which differ from the typical clausal languages employed in ILP and related settings, They have a different syntax and especially very different semantics [4]. Early works on learning in DLs essentially focused on *structural* supervised methods for terminological languages like CLASSIC and its successors [7]. More recently, methods based on refinement operators and sequential covering and [11, 10] or genetic programming [12] have been proposed.

In this work we adopt an expressive DL for representing the tests on tree nodes. This allows to describe different concepts w.r.t. the original clausal representation and inherits the open-world semantics characterizing the DL languages. Moreover, learning in DLs requires a different setting w.r.t. the clausal representation [10]. Particularly, a special treatment of the unlabeled individuals is necessary. Like its multi-relational predecessor, the method employs a top-down *divide-and-conquer* strategy [5] which differs from the standard strategies used for learning concept definitions which are based on sequential covering [11, 10] or genetic programming with the use of refinement operators [12].

The proposed algorithm has been implemented in the prototype of the system TERMITIS (*Terminological Tree Induction System*) which was evaluated on the task of individual classification w.r.t. query concepts using real ontologies already employed as testbeds for other related systems [11, 12, 10]. Given that instance checking in DLs yields three possible answers, *membership*, *non-membership*, *unknown-membership*, we had to resort to *ad hoc* performance indices (already employed in [8, 10]) measuring the alignment of the inductive classification decided by the learned model with the deductive classification decided by a DL reasoner (instance checking). This requires measuring the amount of unlabeled instances that may be ascribed to the newly induced concepts (or to their negations), which may constitute a real value added brought by an inductive method. Actually these conclusions should be evaluated by the knowledge engineer; however, this is not always possible.

The remainder of the paper is organized as follows. The next section introduces the representation. In Sect. 3 the formalization of: the learning problem and the method for inducing terminological decision trees is presented. Experiments are discussed in Sect. 4 while possible developments are examined in Sect. 5.

[1] Such as OWL-DL: `http://www.w3.org/TR/owl-ref/`.

2. BACKGROUND

In this section we shortly recall syntax and semantics of the DL representation.

Basic elements are the primitive *concepts* and *roles* that are used to describe restrictions on concepts. In a DL language, primitive *concepts* $N_C = \{C, D, \ldots\}$ are interpreted as subsets of a domain of objects (resources) and primitive *roles* $N_R = \{R, S, \ldots\}$ are interpreted as binary relations on such a domain (properties). The *individuals* represent the objects through names selected from some $N_I = \{a, b, \ldots\}$.

Complex concept descriptions are built using atomic concepts and primitive roles by means of specific constructors. The meaning of the descriptions is defined by an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is the *domain* of the interpretation and the functor $\cdot^{\mathcal{I}}$ stands for the *interpretation function*, mapping each concept C (resp. role R) to its *extension* $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ (resp., $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$).

The *top* concept $\top$ is interpreted as the whole domain $\Delta^{\mathcal{I}}$, while the *bottom* concept $\bot$ corresponds to $\emptyset$. Complex descriptions can be built, using the specific constructors. For example, in $\mathcal{ALC}$ logic: *full negation*: given any concept description C, denoted $\neg C$, it amounts to $\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$; concept *conjunction*, denoted with $C_1 \sqcap C_2$, corresponds to the extension $C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$ and, dually, concept *disjunction*, denoted with $C_1 \sqcup C_2$, interpreted as $C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$. Moreover, there are two restrictions on roles: the *existential restriction*, denoted with $\exists R.C$, and interpreted as the set $\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$ and the *universal restriction*, denoted with $\forall R.C$, whose extension is $\{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}} : (x, y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$.

Further constructors extend the expressiveness of the $\mathcal{ALC}$ language giving a new name to the particular DL language. OWL-DL is based $\mathcal{SHOIQ}(\mathbf{D})$ that, roughly, extends $\mathcal{ALC}$ with *transitive* roles, *role hierarchies*, *individual classes*, *inverse roles* and *qualified number restrictions*. Besides, concrete domains ($\mathbf{D}$) can be dealt with, i.e. data types with their own semantics.

A *knowledge base* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ contains two sets of axioms: a T-box $\mathcal{T}$ and an A-box $\mathcal{A}$. $\mathcal{T}$ is a set of terminological axioms $C \sqsubseteq D$, yet we will consider only definitions $A \equiv D$, where $A \in N_C$ is a concept name and D is a concept description given in terms of the language constructors, meaning $A^{\mathcal{I}} = D^{\mathcal{I}}$. The ABox $\mathcal{A}$ contains extensional assertions (ground facts) on concepts and roles, e.g. $C(a)$ and $R(a, b)$, meaning, respectively, that $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. A model for $\mathcal{K}$ is an interpretation $\mathcal{I}$ satisfying all of its axioms.

Concept (resp. role) hierarchies are induced by the relationship of *subsumption*, given in terms of these models:

DEFINITION 2.1 (SUBSUMPTION). *Given the knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and two concept (role) descriptions D_1 and D_2, we say that D_1* subsumes *D_2, denoted by $D_1 \sqsupseteq D$, iff for every model $\mathcal{I}$ of $\mathcal{K}$ it holds that $D_1^{\mathcal{I}} \supseteq D_2^{\mathcal{I}}$. Hence, $D_1 \equiv D_2$ amounts to $D_1 \sqsupseteq D_2$ and $D_2 \sqsupseteq D_1$.*

A noteworthy inference service, from the inductive point of view, is *instance checking* [1], that amounts to ascertaining concept-membership assertions: $\mathcal{K} \models C(a)$, where a is an individual name and C is a concept description.

An important difference w.r.t. other FOL fragments is the *open-world assumption* (OWA) which makes it more difficult to answer concept-membership queries: an individual that cannot be proved to belong to a certain concept does not necessarily belong to the negation. The OWA just leads to interpreting these situationsas cases of insufficient (incomplete) knowledge.

EXAMPLE 2.1 (OWA REASONING). *Given a TBox $\mathcal{T}$ containing the concept definitions*
$\texttt{Mother} \equiv \texttt{Female} \sqcap \exists \texttt{hasChild}.\top$
$\texttt{MotherWithNoDaughter} \equiv \texttt{Mother} \sqcap \forall \texttt{hasChild}.\neg \texttt{Female}$
and the following ABox:

$$\begin{aligned}\mathcal{A} = \{&\texttt{Female}(\texttt{ELISABETH}), \texttt{Female}(\texttt{DIANA}),\\ &\texttt{Male}(\texttt{CHARLES}), \texttt{Male}(\texttt{EDWARD}), \texttt{Male}(\texttt{ANDREW}),\\ &\texttt{MotherWithNoDaughter}(\texttt{DIANA}),\\ &\texttt{hasChild}(\texttt{ELISABETH}, \texttt{CHARLES}),\\ &\texttt{hasChild}(\texttt{ELISABETH}, \texttt{EDWARD}),\\ &\texttt{hasChild}(\texttt{ELISABETH}, \texttt{ANDREW}),\\ &\texttt{hasChild}(\texttt{DIANA}, \texttt{WILLIAM}),\\ &\texttt{hasChild}(\texttt{CHARLES}, \texttt{WILLIAM})\}\end{aligned}$$

the instance check $\texttt{MotherWithNoDaughter}(\texttt{ELISABETH})$? *would be answered* `unknown` *because it may well happen that a* `Female` *individual which is a child of* `ELISABETH` *exists but it is not known yet.*

A related inference service is concept *retrieval* that amounts to find all individuals occurring in the knowledge base that can be proved to belong to a given concept.

3. LEARNING TERMINOLOGICAL TREES

3.1 The Learning Problem in DLs

We formalize the learning problem to be solved inducing TDTs. Given a knowledge base, let us suppose that an expert provides proper ABox assertions to deem some individuals as exemplars (or non-exemplars) w.r.t. a new target concept for which one wants to learn a DL definition [10]:

DEFINITION 3.1 (LEARNING PROBLEM).
Given

- *a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$,*
- *the (new) target concept name C,*
- *the sets of positive and negative examples for C, resp. $\mathsf{Ind}_C^+(\mathcal{A})$ and $\mathsf{Ind}_C^-(\mathcal{A})$*

Build *a new concept definition $C \equiv D$ such that*

- $\mathcal{K} \models D(a) \qquad \forall a \in \mathsf{Ind}_C^+(\mathcal{A})$
- $\mathcal{K} \not\models D(b) \qquad \forall b \in \mathsf{Ind}_C^-(\mathcal{A})$

This is the setting of a generic supervised concept learning problem. Note that, unlike the typical settings for learning logic decision trees [3], or similar ones, were multiple *disjoint* concepts are to be learned, in this case one cannot consider the target concepts as being disjoint unless that is explicitly indicated in the knowledge base. Alternatively, a stronger requirement may be made for the negative examples $b \in \mathsf{Ind}_C^-(\mathcal{A})$ that is $\mathcal{K} \models \neg D(b)$.

This setting may be extended to cover also the case of *refinement problems* in which a definition for the target concept may be inconsistent w.r.t. some positive ($a \in \mathsf{Ind}_C^+(\mathcal{A})$: $\mathcal{K} \not\models D(a)$) or negative ($b \in \mathsf{Ind}_C^-(\mathcal{A})$: $\mathcal{K} \models D(b)$) example [10]. Consequently, it needs to be refined to restore the consistency of the knowledge base.

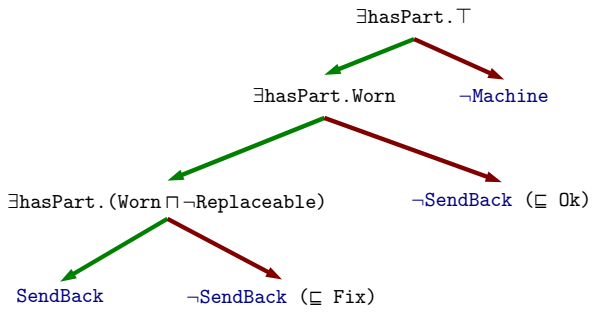

Figure 1: A TDT whose leftmost path corresponds to SendBack ≡ ∃hasPart.(Worn ⊓ ¬Replaceable).

```
function CLASSIFY(a: individual, C: concept, T: TDTree,
                  K: knowledge base): boolean;
begin
N ← ROOT(T);
while ¬LEAF(N, T) do
        ⟨D, T_left, T_right⟩ ← INODE(N);
        if K ⊨ D(a)
        then N ← ROOT(T_left)
        else N ← ROOT(T_right)
⟨D, ·, ·⟩ ← INODE(N);
return (C = D);
end
```

Figure 2: Classification with TDTs.

3.2 Terminological Decision Trees

Blockeel and De Raedt [3] introduced the notion of *first-order logical decision trees* (FOLDTs) as binary decision trees in which (1) the nodes contain tests in the form of conjunctions of literals, (2) left and right branches stand for the truth-value (resp. true and false) determined by the test evaluation and (3) different nodes may share variables, under the following restriction: a variable that is introduced (i.e. it occurs for the first time) in a certain node must not occur in the right branch of that node.

As DLs represent different fragments of FOL with a compact variable-free syntax and the related open-world semantics, an extension of the FOLDTs is proposed where DL descriptions are set as node tests (instead of LP predicates). The result is a *terminological decision tree* (TDT) as the one depicted in Fig. 1. Once that a TDT is obtained, it can be used for classifying individuals w.r.t. a given query concept, according to a simple algorithm in Fig. 2. The auxiliary function LEAF() determines whether a node is a leaf of the input tree; ROOT() returns the root node of the input tree; INODE() retrieves the test concept and the left and right subtrees branching from a given internal node. Given an individual a, a test concept in a node (say D) corresponds to checking $\mathcal{K} \models D(a)$. An individual is sorted to the left or the right branch depending on the test outcome.

Note that in TDTs, each child node may be meant as a specialization of its parent. This can be given either (trivially) by adding a concept description (like in FOLDTs) or by refining a sub-description in the scope of an existential or universal restriction. Note also that expressive DL languages (like $\mathcal{ALC}$) are endowed with full negation, hence the situation is perfectly symmetric. No special care is to be devoted to negated atoms and their variables as with FOLDTs [3]. It is possible to derive a *single* concept definition from a TDT. The algorithm is shown in Fig. 3. One follows all paths leading to success nodes for the target concept building possibly many versions of the concept in the form of conjunctions C_i, $i = 1, \ldots, M$. Then the single resulting concept is merely the union of this finite set of concepts: $D = \bigsqcup_{i=1}^{M} C_i$.

```
function DERIVEDEFINITION(C: class name, T: TDTree):
                          concept description;
begin
S ← ASSOCIATE(C, T, {});
return ⨆_{D∈S} D;      // ⊤ if S is empty
end

function ASSOCIATE(C: class name, T: TDTree,
                   S: set of descriptions): set of descriptions;
begin
N ← ROOT(T);
if LEAF(N, T) then
        ⟨D, ·, ·⟩ ← INODE(N);
        if D = C then return S else return ∅;
else
        ⟨D, T_left, T_right⟩ ← INODE(N);
        S_left ← ASSOCIATE(C, T_left, S);
        S_right ← ASSOCIATE(C, T_right, S);
        S' ← ∅;
        for each E ∈ S_left do
                S' ← S' ∪ {E ⊓ D};
        for each E ∈ S_right do
                S' ← S' ∪ {E ⊓ ¬D};
        return S'
end
```

Figure 3: Mapping a TDT to a concept description.

3.3 Learning Terminological Decision Trees

The subsumption relationship (see Def. 2.1) induces a partial order on the space of all the possible concept descriptions [11, 12]. Hence a solution of the inductive problem stated in the previous section can be cast as a search for the right concept definition (hypothesis) in the induced search space. In this perspective, suitable operators to traverse the search space ought to be defined. As usual in inductive search, we will define two kinds of operators, namely given a starting (incorrect) hypothesis, they must return one (or some) of its generalizations or specializations.

The solutions of the learning problem are generally found adopting a *separate-and-conquer* strategy [11, 10]. Instead here we opt for a different way to solve the problem, adopting a *divide-and-conquer* strategy [5] which is more typical in decision tree induction. The aim is defining a learning algorithm that can overcome the limitations of some DL learning systems: avoiding the computation of language-dependent descriptions[2] (or their approximations) such as the *most specific concepts* or the *least common subsumers* [1]. The algorithm implemented by DL-LEARNER [12] mitigates these disadvantages. It is essentially based on a genetic programming procedure grounded on operators producing refinements whose fitness depends on their coverage.

[2]Early works [7, 11] present methods that require lifting the instances to the concept level through a suitable approximate operator and then start learning from the resulting (extremely specific) concept descriptions.

```
function inducеTDTree(Ps, Ns, Us: set of individuals;
                      C: concept): concept description;
const   Pr+,Pr−: prior probabilities
        θ: threshold
begin
Initialize new TDTree T;
if |Ps| = 0 and |Ns| = 0 then
        if Pr+ ≥ Pr− then T.root ← C else T.root ← ¬C;
        return T;
if |Ns| = 0 and |Ps|/(|Ps| + |Ns| + 1) > θ then
        T.root ← C; return T;
if |Ps| = 0 and |Ns|/(|Ps| + |Ns| + 1) > θ then
        T.root ← ¬C; return T;
S ← generateNewConcepts(Ps, Ns);
D ← selectBestConcept(S, Ps, Ns, Us);
⟨⟨P^l, N^l, U^l⟩, ⟨P^r, N^r, U^r⟩⟩ ← split(D, Ps, Ns, Us);
T.root ← D;
T.left ← inducеTDTree(P^l, N^l, U^l, C);
T.right ← inducеTDTree(P^r, N^r, U^r, C);
return T;
end
```

Figure 4: The main routine for learning TDTs.

In our TDT induction algorithm (see Fig. 4) the refinement operators play a central role. The algorithm derives from the standard tree induction algorithms [14, 3], with the addition of the treatment of unlabeled training individuals. This is due to the mentioned open-world semantics causing some individuals not to be assigned to the target concept or to its negation.

Essentially the set of all individuals is sorted through the branches of the TDT under construction by choosing suitable test for the inner nodes. The aim of this progressive division is obtaining pure nodes (i.e. containing positive, resp. negative, individuals only) which become leaves with the assigned classification.

The three initial conditional statements take care of the base cases of the recursion: 1) no individual got sorted to the current node then the resulting leaf is decided on the grounds of the priors; 2) no negative individual yet more than a minimum number θ of positive ones got sorted to the current node, then the leaf is labeled accordingly; 3) dual case with no positive individual.

In the second part of the algorithm a set S of new test descriptions are randomly generated: they must be satisfiable w.r.t. $\mathcal{K}$ and cover some of the positive and negative individuals. Then, the *best description* is chosen. For *best description* we mean the one that, among the others randomly generated, is better able to discriminate the positive and negative examples. The quality of each candidate test description is measured in terms of an improvement of the purity of the subsets of individuals resulting from the split based on the given description (a sort of *information gain* [14]). Purity is measured by the *entropy* of the three subsets, taking into account the unlabeled individuals.

Once the best test concept has been selected, it is installed as the current subtree root and the sets of individuals sorted to this node are subdivided according to their classification w.r.t. such a concept (and the knowledge base $\mathcal{K}$). Note that unlabeled individuals must be sorted to both left and right subtree. Finally the recursive calls for the construction of the subtrees are made passing the proper sets of individuals.

Table 1: Ontologies employed in the experiments.

ontology	language	#concepts	#roles	#individuals
FSM	$\mathcal{SOF}(\mathcal{D})$	20	10	37
MDM0.73	$\mathcal{ALCHOF}(\mathcal{D})$	196	22	112
Wines	$\mathcal{ALCOF}(\mathcal{D})$	75	12	161
BioPax	$\mathcal{ALCIF}(\mathcal{D})$	74	70	323
hDisease	$\mathcal{ALCIF}(\mathcal{D})$	1498	10	639
NTN	$\mathcal{SHIF}(\mathcal{D})$	47	27	676
Financial	$\mathcal{ALCIF}$	60	16	1000

4. PRELIMINARY EXPERIMENTS

The presented method has been implemented in TermiTIS (*Terminological Tree Induction System*) and has been experimented on a number of real ontologies[3] (see Tab. 1 for details). TermiTIS was applied to retrieval problems solved by using inductive classification of the individuals through terminological trees w.r.t. 50 query concepts per ontology. Queries were randomly generated by composition (intersection, union, universal or existential restriction) of 2 through 8 primitive or defined concepts.

A *.632 bootstrap* strategy was adopted for the design of the experiments. A standard reasoner[4]was employed to decide on the theoretical class-membership (and non-membership) of the individuals w.r.t. the query concepts. The performance was evaluated comparing the inductive classification of the individuals with those found deductively by the reasoner. Due to the OWA, cases were observed when, it could not be (deductively) ascertained whether an individual belongs or not to the given query. Hence, as argued in previous works [8, 10], a ternary classification has been adopted and the following indices have been employed:

match rate: cases of individuals that got the same classification by the reasoner and the inductive model;

omission error rate: cases of individuals for which class-membership could not be determined using the induced TDT, while the reasoner found them to belong (do not belong) to the query concept;

commission error rate: cases of individuals classified as not belonging to the query concept according to the induced TDT, while the reasoner found them to belong to it (and vice-versa);

induction rate: cases of individuals found to belong or not to the query concept according to the TDT, while either membership is not logically derivable.

Tab. 2 reports the outcomes of the experiments. The elapsed time (not reported here) was very limited: about 0.5 hour on a (Quadcore linux box) for a whole experiment including the time consumed by the reasoner.

Preliminarily, note that inductive classification was quite accurate: it made few critical mistakes, especially when the considered concepts have many examples (and counterexamples) in the ontology. However, the commission error was limited but not absent, as in the experiments with other classification methods [8]. The cases in which this measure was more sensible are due to the limited number of examples

[3]Swoogle (`http://swoogle.umbc.edu`) and the Protégé ontology library (`http://protege.stanford.edu/plugins/owl/owl-library`) were used.

[4]Pellet v. 2.0.0rc3 `http://pellet.owldl.com`

Table 2: Results in terms of the proposed evaluation rates: average values ± standard deviations.

ontology	match	commission	omission	induction
FSM	97.72±01.98	00.99±01.35	00.02±00.18	01.27±00.51
MDM0.73	94.87±05.44	00.39±00.61	03.50±04.16	01.24±01.47
Wines	72.21±25.63	00.67±04.63	12.46±14.28	14.28±23,49
BioPax	98.59±06.03	01.30±05.72	00.11±00.51	00.00±00,00
hDisease	72.65±39.79	00.02±00.10	01.50±06.01	25.82±39.17
NTN	90.68±15.89	00.01±00.09	01.36±01.58	07.95±14.60
Financial	97.32±10.48	02.14±10.07	00.16±00.55	00.39±00.16

available (too narrow query concepts). Even few mistakes provoked high error rates. This is also due to the absence of axioms stating explicitly the disjointness of some concepts.

Also the omission error rates are quite low. They are comparable with the amount of inductive conclusions that could be drawn with the trees. Again these figures may vary as a consequence of the availability of knowledge about the disjunction of (sibling) concepts in the subsumption hierarchies. In an ontology population perspective, the cases of induction are interesting because they suggest new assertions which cannot be logically derived by using a deductive reasoner yet they might be used to complete a knowledge base [2], e.g. after being validated by an ontology engineer. Better results were obtained on the same task with different inductive methods (instance-based learning in DLs [8]).

5. CONCLUSIONS AND OUTLOOK

In this work, we investigated new methods for learning DLs descriptions. We proposed the terminological decision trees as alternative structure for learning DL concept descriptions and a method based on standard top-down tree induction algorithms for their construction. The method essentially makes use of an ad hoc gain function which takes into account the classification of individuals w.r.t. the open-world assumption. Namely many instances may be available which cannot be ascribed to the target concept nor to its negation.

The method was implemented in the TermiTIS system which allowed for a preliminary experimentation presented in this work, applying the system to learning from individuals in real ontologies. Specifically, we measured the alignment of the classifications decided by the induced TDTs with the classification decided deductively by a DL reasoner. This allowed measuring the amount of unlabeled instances that may be ascribed to the newly induced concepts (or to their negations), which constituted a real added value brought by an inductive method. The experiments showed that the method is quite effective, and its performance depends on the number (and distribution) of the available training individuals. Besides, the procedure appears robust to noise since commission errors were limited in the experiments carried out.

We plan to extend this work along various directions, starting from the underlying DL language to be adopted. The method can already manage ontologies represented in more expressive languages than $\mathcal{ALC}$ but use the concepts therein as atoms and building new ones exclusively thorough $\mathcal{ALC}$ concept constructors. More impurity indices have to be explored especially to better take into account the uncertainty related to the unlabeled individuals. Then a complete framework for abductive reasoning in DLs may be conceived. Finally, the presented method may be the basis for alternative hierarchical clustering algorithms where clusters would be formed grouping individuals on the grounds of the invented subconcept instead of their similarity, as this may be hardly defined with such complex representations [8].

6. REFERENCES

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[2] F. Baader, B. Ganter, B. Sertkaya, and U. Sattler. Completing description logic knowledge bases using formal concept analysis. In M. Veloso, editor, *Proc. of the 20th International Joint Conference on Artificial Intelligence, IJCAI2007*, pages 230–235, 2007.

[3] H. Blockeel and L. De Raedt. Top-down induction of first-order logical decision trees. *Artificial Intelligence*, 101(1-2):285–297, 1998.

[4] A. Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82(1–2):353–367, 1996.

[5] H. Boström. Covering vs. divide-and-conquer for top-down induction of logic programs. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence, IJCAI95*, pages 1194–1200. Morgan Kaufmann, 1995.

[6] P. Buitelaar, P. Cimiano, and B. Magnini, editors. *Ontology Learning from Text: Methods, Evaluation and Applications*. IOS Press, 2005.

[7] W. Cohen and H. Hirsh. Learning the CLASSIC description logic. In P. Torasso and et al., editors, *Proceedings of the 4th International Conference on the Principles of Knowledge Representation and Reasoning*, pages 121–133. Morgan Kaufmann, 1994.

[8] C. d'Amato, N. Fanizzi, and F. Esposito. Query answering and ontology population: An inductive approach. In S. Bechhofer and et al., editors, *Proc. of the European Semantic Web Conference, ESWC2008*, volume 5021 of *LNCS*, pages 288–302. Springer, 2008.

[9] J. Euzenat and P. Shvaiko. *Ontology Matching*. Springer, 2007.

[10] N. Fanizzi, C. d'Amato, and F. Esposito. DL-Foil: Concept learning in Description Logics. In F. Železný and N. Lavrač, editors, *Proc. of the 18th Int. Conference on Inductive Logic Programming, ILP2008*, volume 5194 of *LNAI*, pages 107–121. Springer, 2008.

[11] L. Iannone, I. Palmisano, and N. Fanizzi. An algorithm based on counterfactuals for concept learning in the semantic web. *Applied Intelligence*, 26(2):139–159, 2007.

[12] J. Lehmann and P. Hitzler. A refinement operator based learning algorithm for the $\mathcal{ALC}$ description logic. In H. Blockeel and et al., editors, *Proc. of the 17th Int. Conference on Inductive Logic Programming, ILP2007*, volume 4894 of *LNCS*. Springer, 2008.

[13] A. Maedche and S. Staab. Learning ontologies for the semantic web. *Intelligent Systems, IEEE*, 16(2):72–79, 2001.

[14] R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.