International Journal of Semantic Computing
Vol. 6, No. 4 (2012) 1–18

DOI: 10.1142/S1793351X1200158X

NUMERIC PREDICTION ON OWL KNOWLEDGE BASES THROUGH TERMINOLOGICAL REGRESSION TREES

NICOLA FANIZZI*, CLAUDIA D'AMATO,
FLORIANA ESPOSITO and PASQUALE MINERVINI

Dipartimento di Informatica, Università degli studi di Bari
Campus Universitario, Via Orabona 4, 70125 Bari, Italy
**nicola.fanizzi@uniba.it*
http//lacam.diunibe.it

In the context of semantic knowledge bases, among the possible problems that may be tackled by means of data-driven inductive strategies, one can consider those that require the prediction of the unknown values of existing numeric features or the definition of new features to be derived from the data model. These problems can be cast as regression problems so that suitable solutions can be devised based on those found for multi-relational databases. In this paper, a new framework for the induction of logical regression trees is presented. Differently from the classic logical regression trees and the recent fork of the terminological classification trees, the novel *terminological regression trees* aim at predicting continuous values, while tests at the tree nodes are expressed with Description Logic concepts. They are intended for multiple uses with knowledge bases expressed in the standard ontology languages for the Semantic Web. A top-down method for growing such trees is proposed as well as algorithms for making predictions with the trees and deriving rules. The system that implements these methods is experimentally evaluated on ontologies selected from popular repositories.

Keywords: Regression tree; knowledge bases; prediction.

1. Introduction

Next generation knowledge bases will likely rely on Web-scale distributed repositories of resources. These will be indexed in terms of shared ontologies that will be expressed through standard machine-interpretable representations. Indeed, among its various facets, the *Semantic Web* is a *Web of data*. A growing number of structured data sources are distributed over the Web and comply with the new standards for data integration. Semantic Web technologies provide infrastructures for novel applications to be enabled by the possibility of querying, making inferences, etc. on such knowledge bases.

An interesting problem may concern making predictions on numeric features of the resources contained in such knowledge bases. Learning predictive models it is possible to extend the factual knowledge therein and automatically derive rules to enrich the reasoning capacities, e.g. learning numeric functions (such as ranking functions, reinforcement functions for planning, etc.) that are hard to express analytically.

1.1. *Scenarios*

In this paper, we focus on regression problems in the context of Web ontologies. Such problems naturally emerge in various scenarios. In a possible scenario, suppose a (human/software) agent is looking for the distance to a certain place of interest. It may query a knowledge-based system that contains a repository of spatial annotations about places and may contain a query-answering component which may be also able to perform spatial reasoning. Would the query system be able to provide an answer when required information to answer the query is missing or non-derivable[a] through automated reasoning procedures?

An even more complex scenario is one where the numeric feature to be found for a given resource is not analytically formalized in the knowledge base through a proper relationship. Suppose an organization wants to predict the share of a given TV program given a categorization of TV programs and a history of shares registered in the past for other programs (see also the next Ex. 1, where this scenario is formalized). Scoring a certain share can be hardly modeled with a relationship in the knowledge base, i.e. its definition can be only extensional: an (incomplete) enumeration of cases observed in the past.

Determining or predicting the filler of a real-valued role is not a standard inference service for DL reasoners. Even quantifying the degree of membership with respect to a given query concept may be cast as a regression problem (especially when reasoner is not able to provide a positive or negative answer). An inductive inference service that is able to at least suggest an answer to these problems may enhance the query answering components of semantic repository managers with further reasoning capabilities.

1.2. *Solutions*

The induction of decision trees is among the most well-known machine learning techniques, along with its extensions towards logical representations in clausal form [14, 16, 3]. Together with several adaptations of classical learning algorithms based on refinement operators for inducing DL concepts (e.g. see [17]), the general tree induction framework has recently been extended to cope with the DL languages supporting the Web ontologies [7]: the tests at the internal nodes of a logical decision tree are represented through DL concepts (class expressions). Compared to the other logic representations of the tree tests, ours can naturally comply with the semantics of the data expressed in the standard languages for the Semantic Web

[a] Spatial reasoning systems are generally not able to perform inductive inferences.

context. The resulting *terminological decision trees* have been used as alternative models for classifying individuals and/or for inducing new concept descriptions from examples.

Following [16], the setting is further extended to address regression rather than classification problems: leaf-nodes have to indicate the values for a target function to be learned. These values may be provided as constants (real numbers) or even through more complex parametric models (e.g. linear functions) to be applied to the individuals routed to the given leaf.

We introduce a new type of logical model trees called *Terminological Regression Trees*. We propose a tree-induction algorithm that adopts a classical top-down *divide-and-conquer* strategy with the use of refinement operators for DL concept descriptions [13, 17, 6]. An *ad hoc* evaluation measure is adopted as a heuristic for deciding the test concepts that are installed at the internal nodes.

In an extensive experiment, the implementation of the proposed algorithms was applied to real ontologies selected from popular repositories. Artificial problems were crafted by considering the prediction of the values of a continuous target function through trained regression trees. The experiments show empirically the feasibility of the method as very limited errors were observed on average.

First-order trees may be considered for various applications such as clustering, categorization or numeric prediction. In particular, the induction of regression trees may be considered an alternative way for learning *ranking functions*, a problem that has also been considered in the context of DL knowledge bases [8].

1.3. *Plan*

In the remainder of this paper, we briefly introduce the basics of the underlying representation (Sec. 2). Then, the learning problem is formalized (Sec. 3) and we present a solution based on terminological regression trees presenting the algorithms for growing them, deriving rules and for making predictions (Sec. 4). Experiments proving the effectiveness of the approach are reported in Sec. 5. Finally, possible applications and further developments are discussed in Sec. 6.

2. DL Knowledge Bases

We are targeting OWL knowledge bases with no commitment to a specific version or dialect. In order to make the paper self-contained, we shortly recall the essentials of the representation and reasoning adopted. More details can be easily found by consulting the reference handbook for DLs [1].

Roughly, in the terminological formalisms *concepts* and *relations* are used to describe classes of resources in a domain and relationships between them. 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

February 20, 2013 9:41:19am WSPC/214-IJSC 00158 ISSN: 1793-351X 1st Reading

on such a domain (properties). *Individuals* represent the objects through names chosen from the set $N_I = \{a, b, \ldots\}$. 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 individual a to some $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ and hence, the *intension* of concepts and roles to their *extension* (respectively, a subset of $\Delta^{\mathcal{I}}$ and a binary relation defined on $\Delta^{\mathcal{I}}$).

Complex concept descriptions are built using atomic concepts (including the *top* and *bottom concepts* denoted, resp., with $\top$ and $\bot$) and primitive roles by means of specific constructors. In $\mathcal{ALC}$ the following constructors are allowed: *full concept negation* ($\neg C$), concept *conjunction* (denoted with $C \sqcap C'$) and, then also *disjunction* (denoted with $C \sqcup C'$), and the *existential restriction* and the *value restriction* on roles, denoted, resp. with $\exists R.C$ and $\forall R.C$, selecting the individuals in the domain related through R to (some, resp. all) individuals that belong to C.

Additional constructors extend the expressiveness of the $\mathcal{ALC}$ language. Besides, concrete domains (**D**) with their specific semantics can be dealt with. They may include basic data types, such as numerical types, strings, etc., but also more complex types, such as tuples of the relational calculus or time intervals. In this paper, the interest is limited to the case of real values, in case one wants to encode the learning problem in the same language.

The notion of *subsumption* between concepts is given in terms of the interpretations: Given two concept descriptions C and D, C *is subsumed* by D, denoted by $C \sqsubseteq D$, iff for every interpretation $\mathcal{I}$ of $\mathcal{T}$ it holds that $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Hence, $C \equiv D$ amounts to $C \sqsubseteq D$ and $D \sqsubseteq C$. The interpretations of interest will be limited to those satisfying the axioms in the knowledge base. A *knowledge base* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ contains two components: a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. $\mathcal{T}$ is a set of terminological axioms $C \sqsubseteq D$, yet we will consider only inclusions $A \sqsubseteq D$, where $A \in N_C$ is a concept name (atomic) and D is a concept description given in terms of the language constructors. 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 $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. An interpretation satisfying all the axioms in the knowledge base is said to be a *model* for it. Hence, the usual notions of satisfiability, consistency, etc. apply also for these logics.

The most important inference service from the inductive point of view is *instance checking* [1] which amounts to ascertain class-membership assertions: $\mathcal{K} \models C(a)$, where $\mathcal{K}$ is the knowledge base a is an individual name and C is a concept definition given in terms of the concepts accounted for in $\mathcal{K}$.

An important difference with clausal logic (multi relational databases), where the logic decision trees have stemmed from, is the *open-world assumption* (OWA) which has consequences on answering to class-membership queries. Thus it may happen that an object that cannot be proved to belong to a certain concept is not necessarily a counterexample for that concept. That would only be interpreted as a case of insufficient (incomplete) knowledge for proving the assertion.

3. Regression Problems with DL Knowledge Bases

Given a DL knowledge base as the source for the background knowledge and a target real-valued function (which may have been encoded as a functional role) whose analytic form is unknown (or too complex to be expressed), one may suppose that domain experts are able to provide the values of such a function for a limited number of individuals, e.g. in the form of role assertions.

In this setting, the objective is then to induce an (approximated) analytic function which can exhibit the same behavior of the target function on the training individuals and predict approximately correct values for further ones. More formally:

Definition 1. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a DL knowledge base, with $\mathsf{Ind}(\mathcal{A})$ denoting the individuals occurring in $\mathcal{A}$.

Given

- a target function $f : \mathsf{Ind}(\mathcal{A}) \to \mathbf{R}$
  (or a functional role R ranging on the concrete domain $\mathbf{R}$)
  whose analytic form is unknown;
- a sample of (training) individuals for which the f-value is known, i.e.
  f (resp. R) may be partially (extensionally) defined:
  $\mathcal{S}(\mathcal{A}) = \{(a, f(a)) \mid a \in \mathsf{Ind}(\mathcal{A})\} \subseteq \mathsf{Ind}(\mathcal{A})$
  (resp. $\mathcal{S}(\mathcal{A}) = \{(a, v) \mid R(a, v) \in \mathcal{A}\}$)
- a small $\varepsilon > 0$

Build a regression model $h : \mathsf{Ind}(\mathcal{A}) \to \mathbf{R}$ so that:

$$|h(a) - f(a)| < \varepsilon, \quad \forall a \in \mathsf{Ind}(\mathcal{A})$$

One may consider the case of relevance feedback where the evaluation of Web resources may be exploited to learn an analytical model which adapts as new resources become available. Preference learning in the context of ontologies is another closely related task.

An example on the domain of *TV Programs* may better illustrate the learning task as follows:

Example 1. (TV programs) Suppose a knowledge base concerning TV Programs is available whose terminology allows for expressing information about the programs, such as their type, production year, broadcast date, etc. Suppose one wants to predict the share, i.e. to learn a share function based on the available terminology and assertions known about the TV programs. The TBox $\mathcal{T}$ may include the background knowledge shown in Fig. 1.

Besides, since some concepts are meant to be disjoint, suitable axioms must be added to $\mathcal{T}$:

$$\begin{aligned} \{\texttt{Film} &\sqsubseteq \neg(\texttt{TalkShow} \sqcup \texttt{Serial}),\\ \texttt{Show} &\sqsubseteq \neg(\texttt{Film} \sqcup \texttt{Serial}),\\ \texttt{SoapOpera} &\sqsubseteq \neg(\texttt{SitCom} \sqcup \texttt{Film}), \ldots\} \end{aligned}$$

$$
\left\{
\begin{aligned}
\mathtt{Program} &\sqsubseteq \forall \mathtt{on.TVNetwork} \sqcap \forall \mathtt{producedIn.Year} \sqcap \forall \mathtt{broadcastOn.Date},\\
\mathtt{News} &\sqsubseteq \mathtt{Program} \sqcap \forall \mathtt{hasAnchor.Person},\\
\mathtt{Entertainment} &\sqsubseteq \mathtt{Program},\\
\mathtt{Cultural} &\sqsubseteq \mathtt{Program},\\
\mathtt{Film} &\sqsubseteq (\mathtt{Movie} \sqcup \mathtt{Documentary}) \sqcap \forall \mathtt{directedBy.Person},\\
\mathtt{Movie} &\sqsubseteq (\mathtt{Entertainment} \sqcup \mathtt{Cultural}) \sqcap \forall \mathtt{starring.Actor},\\
\mathtt{Documentary} &\sqsubseteq \mathtt{Film} \sqcap \mathtt{Cultural} \sqcap \forall \mathtt{starring}.\bot,\\
\mathtt{Serial} &\sqsubseteq \mathtt{Entertainment} \sqcap \neg\mathtt{News} \sqcap \forall \mathtt{starring.Actor},\\
\mathtt{SoapOpera} &\sqsubseteq \mathtt{Serial},\\
\mathtt{SitCom} &\sqsubseteq \mathtt{Serial},\\
\mathtt{Show} &\sqsubseteq \mathtt{Entertainment} \sqcap \forall \mathtt{hostedBy.Person},\\
\mathtt{TalkShow} &\sqsubseteq \mathtt{Show} \sqcap \mathtt{News} \sqcap \forall \mathtt{dealsWith.Topic}
\end{aligned}
\right\} \subseteq \mathcal{T}
$$

Fig. 1. Background knowledge for the TV-rating example.

The ABox may contain the following assertions:

$$
\begin{aligned}
\{\, & \mathtt{SitCom}(\mathtt{SEINFELD}), \mathtt{on}(\mathtt{SEINFELD}, \mathtt{NBC}),\\
& \mathtt{broadcastOn}(\mathtt{SEINFELD}, 20101009),\\
& \mathtt{starring}(\mathtt{SEINFELD}, \mathtt{JL\_DREYFUS}),\\
& \mathtt{TalkShow}(\mathtt{LATESHOW}), \mathtt{on}(\mathtt{LATESHOW}, \mathtt{CBS}),\\
& \mathtt{hostedBy}(\mathtt{LATESHOW}, \mathtt{D\_LETTERMAN}),\\
& \mathtt{broadcastOn}(\mathtt{LATESHOW}, 20100212),\\
& \mathtt{Documentary}(\mathtt{SICKO}), \mathtt{on}(\mathtt{SICKO}, \mathtt{CURRENT}),\\
& \mathtt{directedBy}(\mathtt{SICKO}, \mathtt{M\_MOORE}),\\
& \mathtt{broadcastOn}(\mathtt{SICKO}, 20101111), \ldots \,\} \subseteq \mathcal{A}
\end{aligned}
$$

Suppose the intended function to be predicted is the share of a given program. A sample of this function may contain the following couples-

$$
\begin{aligned}
\mathcal{S}(\mathcal{A}) = \{\, & (\mathtt{SEINFELD}, 21.5\%), (\mathtt{LATESHOW}, 11.2\%),\\
& (\mathtt{SICKO}, 23.1\%), \ldots\}
\end{aligned}
$$

Note that this problem differs from settings where the aim is building a classifier through logical or statistical methods employing, for example, *support vector machines* [4], which may be tackled through more suitable kernel machines (for regression). Further related settings will not be discussed further.

4. Growing and Exploiting Terminological Regression Trees

In the context of the clausal representations adopted in *inductive logic programming* (ILP), *first-order logical trees* (FOLTs) are defined [3] as binary trees in which

(1) the inner nodes contain tests in the form of conjunctions of literals;
(2) left and right branches stand, resp., for the truth-value (resp. true and false) determined by the test evaluation;

February 20, 2013 9:41:22am WSPC/214-IJSC 00158 ISSN: 1793-351X 1st Reading

(3) different nodes may share variables, yet a variable that is introduced in a certain node must not occur in the right branch of that node.

In the case of *regression trees* [14–16], the leaf-nodes contain a local regression model in one of the forms that will be discussed later.

We extend the original definition along various directions:

Definition 1. *Terminological Regression Trees* (TRTs) are FOLTs whose internal nodes contain DL concept descriptions as tests and whose leaf-nodes contain functions (*local models*) to be evaluated to get a predicted value.

In the simplest form leaf-nodes represent constant functions that return average values for all the instances that reach those leaves during the evaluation phase (see below). Figure 2 shows a TRT in the context of the knowledge base proposed in Example 1.

We will consider a generic binary tree structure whose nodes N are tuples containing the following fields:

- test concept description: N.test;
- set of training instances rooted at the node: N.set;
- regression model: N.model;
- references to the subtrees, resp. N.left and N.right.

We now show how to grow TRTs and how to use them for predicting values.

4.1. *Induction of TRTs*

While DL concept induction algorithms generally adopt a separate-and-conquer covering strategy [13, 6], the tree learning procedures adopt a divide-and-conquer strategy.

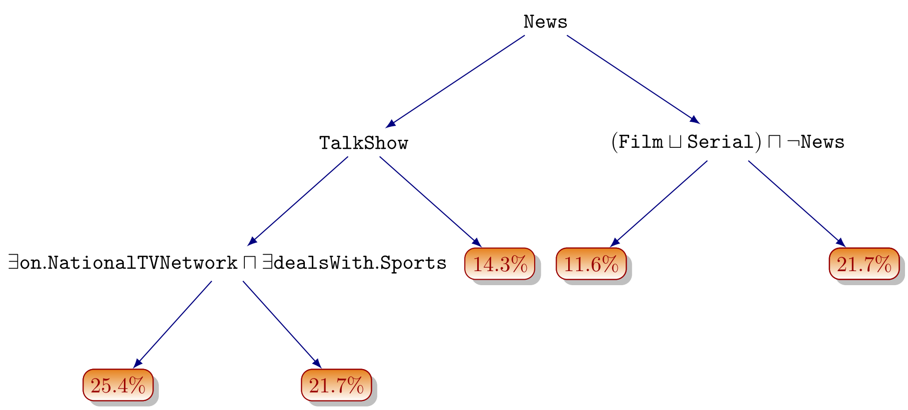

Fig. 2. A simple example of a TRT referring to the problem introduced in Example 1.

February 20, 2013 9:41:22am WSPC/214-IJSC 00158 ISSN: 1793-351X 1st Reading

In the case of TDTs [7], the FOLT induction procedure is adapted to deal with internal nodes that contain DL concepts to be interpreted w.r.t. their semantics. Since the choice of such nodes involves a specialization task this may depend on the properties of the particular space of concept descriptions. The other difference is in the policy for labeling the leaf-nodes. While for decision trees it suffices to indicate the target class for the instances that are routed to those nodes, for regression trees the (regression) model needed for the local regression problem should be decided [18].

As regards the first issue, the subsumption relationship $\sqsubseteq$ induces a partial order on the space of DL concept descriptions. Then, a specialization task can be cast as a search in such a partially ordered space. In such a setting, suitable operators to traverse the search space are required [17]. As this space is very large (especially from a syntactic viewpoint) and may turn out to exhibit a lot of redundancy (equivalent concepts), also depending on the expressiveness of the underlying DL logic, some bias is needed to constrain the search. In [7] only refinement operators working on $\mathcal{ALC}$ constructors are used, even in the context of ontologies represented through more expressive languages, thus trading completeness for efficiency. A further constraint that can be considered leads to prefer candidate specializations for which the training instances tend to exhibit a definite membership, discouraging those concepts for which individuals show an unknown membership (due to the OWA).

The TRT-induction procedure adapts the schema implemented by the extensions of Tilde towards first order regression [3]. A sketch of the main procedure is reported as Algorithm 1. It reflects the standard tree induction algorithms with the addition of

Algorithm 1 The main growing procedure for terminological regression trees
growTRT($\mathcal{K}$, C, LS): T;

Input:
$\mathcal{K}$: knowledge base,
C: concept description,
LS: local set of training individuals

Output: T: TRT

Initialize new tree T
$N.\mathrm{set} \leftarrow LS$
if $|LS| < m$ **or** $\mathrm{sd}(LS)/TSD \leq \theta$ **then**
  $N.\mathrm{model} \leftarrow$ buildRegModel(LS)
  $N.\mathrm{left} \leftarrow \mathrm{nil}$; $N.\mathrm{right} \leftarrow \mathrm{nil}$
else
  $SPC \leftarrow$ generateSpecializations($\mathcal{K}, C, LS$)
  $C^* \leftarrow$ selectBestSDR($\mathcal{K}, SPC, LS$)
  $(LS_+, LS_-) \leftarrow$ split(C^*, LS)
  $N.\mathrm{test} \leftarrow C^*$
  $N.\mathrm{left} \leftarrow$ growTRT($\mathcal{K}$, simplify($C \sqcap C^*$), LS_+)
  $N.\mathrm{right} \leftarrow$ growTRT($\mathcal{K}$, simplify($C \sqcap \neg C^*$), LS_-)
end if
$T.\mathrm{root} \leftarrow N$
return T

the treatment of unlabeled training individuals. The procedure can be invoked passing $\top$ as starting concept and a set of individuals containing all training examples (for which the value of the target function is known).

The initial conditional statement takes care of the base case for the recursion, namely when a limited number of individuals got sorted to the current subtree root-node (w.r.t. some parameter m) or the local standard deviation (sd(LS)) is low w.r.t the global one (TSD), i.e. their ratio is less than a threshold θ, then the resulting leaf value is decided on the grounds of the preferred local regression algorithm, installing the resulting regression model $h(\cdot)$ based on the values in the local set of individuals $LS = \{a_1, \ldots, a_n\}$ [18]:

(1) a constant function, i.e. a real value v that averages the values of the training instances that placed at the leaf node: $h(x) = v = \sum_{i=1}^{n} f(a_i)/|LS|$;
(2) the value that may be determined on-the-fly, for a given instance x, by means of a simple instance-based regression procedure:

$$h(x) = \frac{\sum_{i=1}^{n} w_i f(a_i)}{\sum_{i=1}^{n} w_i} \tag{1}$$

where the weights may be selected as inversely proportional to the distance,[b] e.g. $w_i = [d(x, a_i)]^{-2}$;
(3) extending the idea of the previous point, a regression algorithm that can operate with a limited number of training instances (e.g. a *radial basis function network*), may approximate the value computation as follows:

$$h(x) = \frac{\sum_{i=1}^{n} f(a_i)\kappa(d(x, a_i))}{\sum_{i=1}^{n} \kappa(d(x, a_i))} \tag{2}$$

using a Gaussian kernel κ, for instance.

In the following recursive part of the algorithm (lines 7–12) a list SPC of (satisfiable) candidate concept description are randomly generated (line 7), that can specialize the current description C. As mentioned above, the generation is constrained to produce concepts that can really discriminate the individuals in LS, so that they can contribute as good node tests.

Then (line 8), the best one (C^*) is selected (invoking SELECTBESTSDR) in terms of a purity criterion based on the subsets of individuals resulting from the membership/non-membership determined by the given test concept description. The measure is derived from *information gain*. In the DL setting the problem is made more complex by the presence of instances which cannot be labeled as members or non-members w.r.t. the given concept. Their contribution may be considered as proportional to the prior distribution of positive and negative examples. We propose the following

[b] See similarity or distance functions surveyed in [5], for example.

standard deviation reduction measure:

$$\mathrm{SDR}(C, S) = c_0 \left(\mathrm{std\_dev}(S) - \sum_{S' \in \{S_+, S_-\}} \frac{|S'|}{|S|} \mathrm{std\_dev}(S') \right) \tag{3}$$

where S is divided into S_+, S_- and S_0 depending on the instance check of the individuals w.r.t. C (resp. membership, non-membership, uncertain membership subsets) and c_0 is a discount factor for the missing values, $c_0 = |S \setminus S_0|/|S|$.

Once the best description C^* has been selected, it is installed (line 10) as the current subtree root and the sets of individuals sorted to this node are subdivided (line 9) according to their classification w.r.t. such a concept. Note that unlabeled individuals must be sorted to both subtrees.

Finally the construction of the left and right subtrees is requested recursively (lines 11−12), passing the sets of individuals resulting from the split and the concept descriptions $C \sqcap C^*$ and $C \sqcap \neg C^*$ in their simplified version. Function SIMPLIFY is meant to apply normalization and simplification rules reducing the structural complexity of the concepts.

4.2. *Prediction*

The TRTs are to be used for predicting the values of numeric functions for given individuals. Algorithm 2 illustrates the related procedure that is supported by some auxiliary functions: LEAF() to determine whether a node is a leaf of the argument tree, ROOT() which returns the root node of the tree passed as argument, and EVAL() which returns evaluation w.r.t. the given model.

Algorithm 2 Numeric prediction with TRTs:
PREDICT(a,T,$\mathcal{K}$): $\mathbf{R}$

Input:
a: individual,
T: TRT,
$\mathcal{K}$: knowledge base
Output: $v : \mathbf{R}$

```
N ← ROOT(T);
while not LEAF(N) do
    C ← N.test;
    if K ⊨ C(a) then {follow left-branch}
        N ← ROOT(N.left)
    else if K ⊨ ¬C(a) then {follow right-branch}
        N ← ROOT(N.right)
    else {special case}
        return EVAL(default_model, N.set)
    end if
end while
return EVAL(N.model, N.set)
```

Given an individual a and a TRT T, the algorithm, starting from its root node, simply searches for the node containing the local model for evaluation a. In order to find such a leaf-node, it checks the class-membership w.r.t. the test concept in the current node, say C, i.e. if $\mathcal{K} \models C(a)$, sorting a to the left branch if the test is successful while the right branch is chosen if $\mathcal{K} \models \neg C(a)$. Eventually the local model is found in a leaf-node.

Note that the open-world semantics may cause the failure of both left and right branch tests. This special case may be treated by evaluating the individual w.r.t. a default model (e.g. by averaging) based on the set of training instances that were rooted at the internal node. Alternatively, this case can be avoided by assimilating it into the non-membership case or, equivalently, by considering $\mathcal{K} \not\models C(a)$ as right-branch test (i.e. a single else-branch of the conditional statement).

4.3. *Rule derivation from TRTs*

Similarly to decision trees and, in particular, to FOLTs, each node in a path from the root to a leaf-node may be used to build a concept description through consecutive refinements. Specialization is performed in various ways, for example: (1) by adding a concept description as a new conjunct, (2) by refining a sub-description in the scope of an existential or universal restriction or (3) by narrowing the interval in a number restriction (which may be allowed by the adopted DL language, e.g. $\mathcal{ALN}$ or $\mathcal{ALCQ}$, see [1]).

It is possible to derive a list of rules from a TRT. The procedure (see Algorithm 3), to be invoked with getRules$(T, \top)$, unravels all the paths leading to leaf-nodes which will constitute the consequents; for each of them, an antecedent is formed by tracking back from the leaf along the path and collecting the intermediate test concepts. In this way, each path yields a different rule with a conjunctive antecedent that represents a local (partial) definition of the target function $D \to h(a)$. The final

Algorithm 3 Mapping a TRT onto a list of rules
getRules(T,D): L

Input: T: TRT, D: current concept
Output: L: list of rules

```
N ← root(T)
if leaf(N) then
    r ← (D → N.model)                                  {new rule}
    L ← [r]
else
    C ← N.test
    L_left ← getRules(N.left, D ⊓ C)
    L_right ← getRules(N.right, D ⊓ ¬C)
    L ← L_left + L_right
end if
return L
```

consequent depends on the type of model adopted: it may be based on the local set of training individuals and then it can be evaluated on-the-fly or it may be a parametric function whose coefficients vary from leaf-node to leaf-node.

As an example, looking back at the TRT depicted in Fig. 2, a rule definition that would be extracted for the leftmost path is: $\texttt{News} \sqcap \texttt{TalkShow} \sqcap \exists\texttt{on.NationalTVNetwork} \sqcap \exists\texttt{dealsWith.Sports} \rightarrow 25.4\%$.

A rule language may be the natural candidate for encoding such list of rules in a DL program, so that a suitable reasoner may be employed for the prediction task instead of an *ad hoc* one dealing with TRTs.

5. Experimental Evaluation

Preliminary experiments with the implementation of the algorithms presented showed encouraging results. However specific standard datasets to test them are still hard to find (i.e. ontologies that are rich with numeric datatype assertions). In order to provide more than a mere proof of concept, artificial prediction problems on real ontologies to be solved by inducing TRTs have been automatically generated.

5.1. *Setup*

A number of OWL ontologies on various domains have been selected[c]: MDM0.73, Surface Water Model (SWM), Wine, Units, IMdb, Family Tree (Family) Transportation (Trans), New Testament Names (NTN), the *BioPax Glycolysis* ontology (BioPaxG), the Financial ontology and one large ontology generated by the *Lehigh University BenchMark* (LUBM). Table 1 summarizes important details concerning these ontologies, in terms of the numbers of concepts (classes), object and datatype properties and individuals.

For each ontology, artificial learning problems were created on-the-fly as follows: 20 class expressions were randomly generated by composition of 2 through 8 classes using[d] the $\mathcal{ALC}$ constructors in OWL2. For each class expression C, the target function f_C to be approximated assesses the likelihood of membership of the given individual to C. The value of $f_C : \mathsf{Ind}(\mathcal{A}) \rightarrow [0,1]$ for each individual x was assigned by recurring to a simple procedure based on density estimation that returned the likelihood measure:

$$\Pr(h_C(x) = +1 \mid x) = \frac{\pi^C_{+1}\hat{D}_{+1}(x)}{\pi^C_{-1}\hat{D}_{-1}(x) + \pi^C_{+1}\hat{D}_{+1}(x)} \tag{4}$$

[c] They are publicly available in well-known ontology repositories, the Protégé library (http://protege.stanford.edu/plugins/owl/owl-library) and the TONES repository (http://owl.cs.manchester.ac.uk/repository).

[d] Note that since the ontologies employed in the experiment are expressed in various DL languages, see Table 1, these class expressions are represented in a more expressive language than $\mathcal{ALC}$.

Table 1. Facts concerning the ontologies employed in the experiments.

Ontology	DL language	#concepts	#object prop's	#datatype prop's	#individuals
Units	$\mathcal{ALUOF}(\mathbf{D})$	12	3	5	103
MDM0.73	$\mathcal{ALCHOF}(\mathbf{D})$	196	22	3	112
SWM	$\mathcal{ALOF}$	19	9	0	115
Wine	$\mathcal{ALCOF}(\mathbf{D})$	75	12	1	161
Trans	$\mathcal{ALCH}(\mathbf{D})$	445	89	4	183
IMDB	$\mathcal{ALIN}(\mathbf{D})$	7	5	13	302
BioPaxG	$\mathcal{ALCIF}(\mathbf{D})$	74	70	40	323
Family	$\mathcal{SHIOF}(\mathbf{D})$	23	56	6	436
LUBM	$\mathcal{ALR}_+\mathcal{HI}(\mathbf{D})$	55	36	11	555
NTN	$\mathcal{SHIF}(\mathbf{D})$	47	27	8	676
Financial	$\mathcal{ALCIF}$	60	16	0	1000

where h_C is a binary classification function (induced through an auxiliary method, the k-nearest neighbors [11], with k set to $\sqrt{N}$, N being the number of training instances for the problem), the π_v^C's stand for prior probabilities of getting the respective classification value in $\{-1, +1\}$ and the $\hat{D}_v$'s are estimates of the density function for the given classification value, around the input individual x. Hence every individual in $\mathsf{Ind}(\mathcal{A})$ was assigned a value $f_C + \varepsilon$, for each C, where $\varepsilon < .001$ is a small random perturbation. The prior distribution of positive and negative instances were computed for each ontology. A collection of couples $\langle x_i, f_C(x_i)\rangle$ constituted the dataset for a learning problem.[e]

In the experiment design a 10-fold cross validation strategy was adopted. The performance was evaluated measuring the average error the value predicted for the test individuals w.r.t. the various random class expressions using the induced trees $h_C(x)$ compared to $f_C(x)$. The default settings (threshold $\theta = .05$ and $m = 5$) were considered (see Algorithm 1). The Pellet reasoner (v. 2.2.2) was employed for the required reasoning services.

5.2. *Results*

Due to space limitation, we can only report aggregate results of the learning problems. Table 2 shows, for each ontology, the outcomes in terms of the average error and the related standard deviation over the 10 folds averaged also over the various datasets for the class expressions generated.

Preliminarily, we found that the search procedure was quite accurate: it made few critical mistakes, especially when the concepts considered are known to have many instances in the ontology. Indeed, the overall average error is 1.39E-2 in an interval of [1.00E-6, 2.99E-2] whose extrema correspond, resp., to the values observed for the ontologies Trans and LUBM. The standard deviations are also very limited (overall average 1.10E-2) suggesting the method was quite stable over the various learning

[e] A snippet of the code for random concept generation and some of the generated datasets will be available at: http://lacam.di.uniba.it/nico/research/ontologymining.html.

Table 2. Results of the experiments on TRTs: average errors ± standard deviations.

Ontology	Average error	Std. dev.
Units	2.22E-2	2.21E-2
MDM0.73	1.07E-2	9.93E-3
SWM	1.30E-2	1.00E-2
IMDB	1.33E-2	1.31E-2
Wine	1.01E-2	4.97E-3
Trans	1.00E-6	1.00E-9
BioPaxG	1.89E-3	2.83E-3
LUBM	2.99E-2	2.19E-2
Family	2.37E-2	2.28E-2
NTN	1.62E-2	7.86E-3
Financial	1.10E-2	5.66E-3

problems and ontologies. The cases of ontologies for which this measure was higher are probably due to the limited number of available examples.

The elapsed time was quite short (considered the current prototypical stage of the implementation, which lacks possible optimizations): for all the replications of the experiment (training+test) on the entire set of generated class expressions, they range from a few minutes to about 1.5 hours for a whole experiment on a medium sized ontology (in terms of number of individuals) including the time consumed by the reasoner for the deductive instance checks.

From a qualitative viewpoint, it must be noticed that the class expressions that may derived from the tree branches generally tend to be more complex than the original ones that were generated for the related learning problems.

The height of the resulting TRTs varies a lot among the various learning problems. This size can be controlled by tweaking the two parameters m and θ. Further experiments (not reported here) showed that lower values have the effect of increasing the height of the TRTs and, as expected, their precision (lower average error). However, since the training phase is generally more computationally expensive than predicting with the obtained TRTs a tradeoff may be made setting higher values for these parameters. This may be further extended to work in an interactive mode letting a user decide on whether to further expand a node or transform it to a leaf.

In an ontology population perspective, the predicted values 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, e.g. after being validated by ontology engineers and domain experts.

5.3. *Rank prediction in the linked user feedback LOD dataset*

For verifying the applicability of the proposed TRT induction method in a real world context, it was evaluated on a real knowledge base, namely the *Linked User Feedback*

February 20, 2013 9:41:29am WSPC/214-IJSC 00158 ISSN: 1793-351X 1st Reading

(LUF) dataset.[f] LUF is part of an effort to semantically publishing and retrieving user-generated feedbacks (such as ratings, comments and tags); as part of this effort, ratings from the *Linked Movie Data Base*[g] (LinkedMDB) were processed and integrated into the CKAN[h] dataset, which is part of the *Linked Open Data* (LOD) cloud.[i]

To evaluate the effectiveness and feasibility of our approach, it was applied to a film ranking prediction task: given a sample of ratings provided by users, the system induces a ranking rule to predict ratings for unranked movies. In order to leverage the large amount of structured knowledge available through the LOD cloud, we extracted a fragment of the DBpedia [2] knowledge base related to movies ranked in the LUF dataset.

For this task, we followed the procedure described in [12]: starting from resources representing movies, a search was performed in the RDF graph (with recursion depth 1), and up to 1000 superclasses were extracted for each reached object. Such an extraction process resulted in an OWL 2 EL fragment containing 4789 concepts, 59 object properties and 3082 individuals.

To fit to the RDF(S)'s lack of negation and disjunction [9], we used a "LOD-friendly" variant of TRT — during the TRT growth and prediction processes we assumed that, given a splitting node associated to a concept C and a generic individual a in $\mathcal{K}$, $\mathcal{K} \not\models C(a)$ implies that $\mathcal{K} \models \neg C(a)$ (thus assuming a consistent completion of the knowledge base). An effect of this approach is that e.g. during the PREDICT procedure, given a generic individual a and a splitting node associated to a concept C, we follow the left-branch if $\mathcal{K} \models C(a)$ and the right-branch otherwise, and evaluate only at leaf nodes. At each step of the TRT growth process, the set of candidate concepts to be associated to a split node was the set of all atomic concepts in the ontology.

Figure 3 shows an example of an induced TRT, modelling the rankings given by an user to movies in the LUF dataset.[j] From preliminary empirical evaluations on the LUF dataset we observed that, in induced TRTs, the concepts near to the root tended to be meaningful (e.g. `y : English-languAgeFilms`, `y : AmericanFilms`, `y : AmericanBiographicalFilms`, `y : 1990sRomanticComedyFilms`); while, increasing the distance from the root, there was an higher tendency in adding nodes splitting apart very few individuals with a divergent ranking from the others in that node, and the corresponding concepts were hardly meaningful (e.g. `y : FilmsBasedOnDarkHorseComics`, `y : FilmsBasedOnPlays`). A possible cause for this phenomenon is the *curse of dimensionality*: some concepts could improve the SDR only because of statistical fluctuations within the data, instead of statistical regularities. In future works, this could justify the use of feature selection as a

[f] http://thedatahub.org/dataset/linked-user-feedback
[g] http://www.linkedmdb.org/
[h] http://ckan.org/
[i] http://lamboratory.com/2011/12/17/linked-user-feedback/
[j] http://soa4all.isoco.net/luf/users/movies303. For brevity, we will use y as a prefix corresponding to the http://dbpedia.org/classes/yago/ namespace

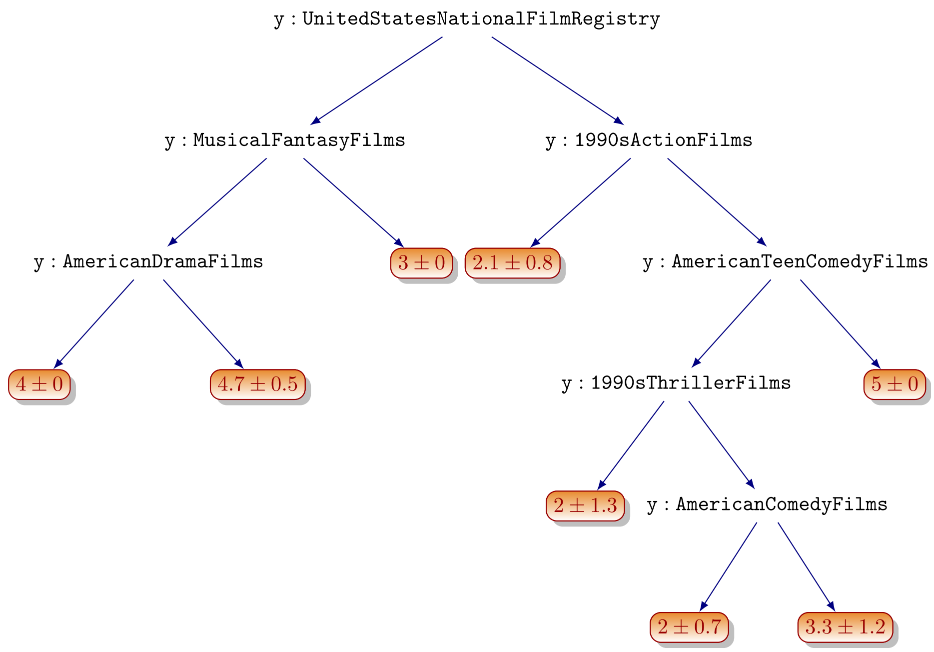

Fig. 3. Portion of an induced TRT modelling user preferences within the Linked User Feedback (LUF) dataset.

pre-processing step prior to TRT induction (e.g. a restriction of the space of possible candidate concepts for splitting nodes); the application of growth stopping criteria and pruning procedures; and the use of scoring function different from the simple SDR [11, 10].

6. Conclusions and Developments

Methods for predicting numerical values in the context of DL knowledge bases were investigated. Introducing the notion of terminological regression trees, which stem from first order trees, we proposed induction algorithms as an adaptation of standard top-down growing methods complying with the issues related to the different representation. We have shown how to exploit such models to predict numerical values, such as those belonging to the range of a datatype or, generally, a complex function that is hard to define analytically. Moreover, the regression trees may be converted into rules which may be easily expressed in Semantic Web rule languages. The experiments made on various ontologies showed that the method is quite effective, its performance depending on the number (and distribution) of the available training individuals.

We plan to extend this work in various directions. At this stage, the adopted refinement operators search the potential candidate refinements incompletely. The methods can manage ontologies represented in more expressive languages than $\mathcal{ALC}$

February 20, 2013 9:41:30am WSPC/214-IJSC 00158 ISSN: 1793-351X 1st Reading

but reuse the concepts therein as atoms and building new ones exclusively thorough $\mathcal{ALC}$ concept constructors. The problem of the dimensions of the resulting trees suggest the adoption of pruning strategies after the induction phase or interleaving with it which could result in more compact and better predicting trees, even in the presence of large number of training instances. Better indices for the standard deviations shall be explored, especially to better take into account the concept simplicity or the uncertainty related to the unlabeled individuals (and missing values).

Possible applications may include ranking queried resources and the enrichment of an ontology with inductively annotated assertions for roles ranging on concrete domains (numeric datatypes). Finally, the presented trees 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.

References

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

[2] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak and S. Hellmann, DBpedia — a crystallization point for the web of data, *Web Semant.* **7**(3) (2009) 154–165.

[3] H. Blockeel, L. De Raedt and J. Ramon, Top-down induction of clustering trees, in J. W. Shavlik (eds.), *Proceedings of the 15th International Conference on Machine Learning* (Morgan Kaufmann, 1998), pp. 55–63.

[4] S. Bloehdorn and Y. Sure, Kernel methods for mining instance data in ontologies, in K. Aberer *et al.*, (eds.), *Proceedings of the 6th International Semantic Web Conference*, LNCS Vol. 4825, 2007, pp. 58–71.

[5] C. d'Amato, S. Staab and N. Fanizzi, On the influence of description logics ontologies on conceptual similarity, in A. Gangemi and J. Euzenat (eds.), *Proceedings of 16th International Conference on Knowledge Engineering: Practice and Patterns, EKAW2008*, LNCS Vol. 5268, 2008, pp. 48–63.

[6] N. Fanizzi, Concept induction in description logics using information-theoretic heuristics, *Int. J. Semantic Web Inf. Syst.* **7**(2) (2011) 23–44.

[7] N. Fanizzi, C. d'Amato and F. Esposito, Induction of concepts in web ontologies through terminological decision trees, in José L. Balcázar *et al.* (eds.), *Proceedings of European Conference on Machine Learning and Knowledge Discovery in Databases, ECML/PKDD2010*, Part I, LNAI Vol. 6321, 2010, pp. 442–457.

[8] N. Fanizzi, C. d'Amato and F. Esposito, Learning to rank individuals in description logics using kernel perceptrons, in P. Hitzler and T. Lukasiewicz (eds.), *Proceedings of 4th International Conference on Web Reasoning and Rule Systems*, LNCS Vol. 6333 (Springer 2010), pp. 173–181.

[9] Ramanathan V. Guha and D. Brickley, RDF vocabulary description language 1.0: RDF schema, W3C recommendation, W3C, February 2004, http://www.w3.org/TR/2004/REC-rdf-schema-20040210/.

[10] J. Han, *Data Mining: Concepts and Techniques* (Morgan Kaufmann, 2005).

[11] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning – Data Mining, Inference, and Prediction* (Springer, 2001).

[12] S. Hellmann, J. Lehmann and S. Auer, Learning of OWL class descriptions on very large knowledge bases, *Int. J. Semantic Web Inf. Syst.* **5**(2) (2009) 25−48.

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

[14] A. Karalic and I. Bratko, First order regression, *Machine Learning* **26** (1997) 147−176.

[15] S. Kramer, Structural regression trees, in *Proceedings of 13th National Conference on Artificial Intelligence* (AAAI Press/MIT Press) 1996, pp. 812−819.

[16] S. Kramer and G. Widmer, Inducing classification and regression trees in first order logic, in S. Džeroski and N. Lavrač (eds.), *Relational Data Mining* (Springer, 2001), pp. 140−156.

[17] J. Lehmann and P. Hitzler, Concept learning in description logics using refinement operators, *Machine Learning* **78**(1−2) (2010) 203−250.

[18] L. Torgo, Functional models for regression tree leaves, in D. H. Fisher (eds.), *Proceedings of 14th International Conference on Machine Learning* (Morgan Kaufmann, 1997), pp. 385−393.