Tree-based Models for Inductive Classification on the *Web Of Data*

Giuseppe Rizzo*, Claudia d'Amato**, Nicola Fanizzi**, Floriana Esposito

LACAM – Dipartimento di Informatica
Università degli Studi di Bari "Aldo Moro",
Via Orabona 4, 70125 Bari, Italy

Abstract

The Web of Data, which is one of the dimensions of the Semantic Web (SW), represents a tremendous source of information, which motivates the increasing attention to the formalization and application of machine learning methods for solving tasks such as concept learning, link prediction, inductive instance retrieval in this context. However, the Web of Data is also characterized by various forms of uncertainty, owing to its inherent incompleteness (missing information, uneven data distributions) and noise, which may affect open and distributed architectures. In this paper, we focus on the inductive instance retrieval task regarded as a classification problem. The proposed solution is a framework for learning *Terminological Decision Trees* from examples described in an ontological knowledge base, to be used for performing instance classifications. For the purpose, suitable pruning strategies and a new prediction procedure are proposed. Furthermore, in order to tackle the class-imbalance distribution problem, the framework is extended to ensembles of Terminological Decision Trees called *Terminological Random Forest*s. The proposed framework has been evaluated, in comparative experiments, with the main state of the art solutions grounded on a similar approach, showing that: 1) the employment of the formalized pruning strategies can improve the model predictiveness; 2) *Terminological Random Forest*s outperform the usage of a single *Terminological Decision Tree*, particularly when the knowledge base is endowed with a large number of concepts and roles; 3) the framework can be exploited for solving related problems, such as predicting the values of given properties with finite ranges.

Keywords: inductive query answering, membership prediction, Web ontologies, decision tree, random forest, concept learning, imbalance learning

1. Introduction

In the perspective of the Semantic Web (SW) as a WEB OF DATA, the *Linked Data* initiative plays a crucial role. It federates a large number of interlinked datasets in a standard format whose semantics is encoded through formal ontologies for plenty of domains, which are accessible through the Web infrastructure, thus ensuring a convenient form for publishing or accessing open data and a superior (semantic) manageability through suitable tools [1].

In this context, a fundamental service, similarly to database management systems, is relational query answering, as a means for assessing properties of interest of individual resources through suitable endpoints. For example, in an academic scenario, given linked data sources that adopt publicly available Web ontologies for such a domain as their vocabularies, a typical query may require determining a person's research interests, his/her co-authorship with other researchers or his/her affiliation to a specific research group. To this purpose, standard technologies such as SPARQL[1] are generally exploited. These services are essentially grounded on pattern matching capabilities that are often inadequate to cope with the issues posed by forms of uncertainty that are inherently related to the distributed architecture of the data sources. This weakness may affect the quality of the answers in terms of *precision* and *completeness*. Especially when also deductive reasoning capabilities are called into play, cases of conflicting information may originate from the diverse quality of the ontologies involved, depending on the axioms in the terminologies and/or by specific assertions made available upon federated data sources. Furthermore, data are expressed in terms of formal vocabularies defined as Web ontologies whose representation and semantics is established on *Description Logics* (DL) [2], a family of languages characterized by the adoption of the *Open World Assumption* (OWA) in the related reasoning services. As a result, the membership of an individual resource w.r.t. a given class or its value for a given property cannot always be ascertained even with the support of a reasoner. A solution borrowed from other multi-relational contexts, such as (deductive) databases and logic programming, amounts to making the *Closed World Assumption* (CWA), i.e. presuming that the current state of knowledge is complete hence, for example, allowing to deem a fact as false

*Principal corresponding author
**Corresponding author
Email addresses: giuseppe.rizzo1@uniba.it (Giuseppe Rizzo), claudia.damato@uniba.it (Claudia d'Amato), nicola.fanizzi@uniba.it (Nicola Fanizzi), floriana.esposito@uniba.it (Floriana Esposito)

[1] http://www.w3.org/TR/rdf-sparql-query/

DOI: 10.1016/j.websem.2017.05.001
https://doi.org/10.1016/j.websem.2017.05.001

when it is not entailed.

All such issues have called for alternative methods that are based on the induction of efficient *classification models* [3, 4, 5, 6] built by exploiting *statistical regularities* in the data, which can be used for assessing the concept membership of individual resources and for other related tasks. Generally, such methods tackle the mere classification problem, disregarding the *comprehensibility* of the resulting model. Conversely, an interesting class of different methods and models that descend from the well-established fields of *Inductive Logic Programming* (ILP), can be adapted to learn analogous models, based on concept descriptions expressed in DLs and complying their semantics [7, 8, 9, 10, 11].

In line with these two research directions, in this paper we focus on a framework based on the notion of *Terminological Decision Tree* [12, 13], as a trade-off between predictiveness and comprehensibility of the models, to be used for performing class-membership or property value predictions.

Terminological decision trees are an extension of the *First Order Logic Decision Trees* [14] coping with DL languages. A terminological decision tree is a logic decision tree where each internal node contains a DL concept description that is used as a test for routing the individuals to be classified towards the leaves (decision nodes). Each departing edge from a node corresponds to one of the possible outcome of the test. As a result, the model induces a partitioning of the instance space in regions where the individuals share the same classification w.r.t. the target concept. Hence the model elicits and encodes some form of *semantic similarity* among the individuals by combining logic and statistical properties of the data.

Terminological decision trees may have several applications: they can be exploited as alternative logic-based classifiers to predict the membership of a resource w.r.t. a target class, but also to give explanations for specific membership assignments in terms of the involved paths [15]; besides, they can be used for eliciting new candidate concepts, defined on the grounds of those that are already encoded in the underlying terminologies, originated from the actual population of the ontologies. As such, these models are proposed as a support to alternative inductive reasoning services to be employed for determining specific concept-memberships (or a property values), being also able to explain why a resource is relevant to the query concept (or a value is assumed by a given property). Additionally, explanations could be used to support ontology engineers in a validation process of new incoming knowledge, through the comparison of the conclusions that can be induced from the data with the domain knowledge encoded in the ontologies.

Besides the various advantages that may derive from resorting to tree-based models, there are some drawbacks that have justified further extensions that we describe in this paper.

Firstly, regarding the prediction task, in the early versions of the methods for traversing the induced tree models there was a quick and oversimplified treatment of the cases in which the test on internal nodes (based on the entailment of either the test concept or its complement) is not able to route an individual down to either branch, resulting in an early stop of the procedure and failing to predict a definite membership [12, 16]. This situation is similar to the case of missing attribute values with standard learning methods hence, stemming from the approaches proposed for decision trees [17], various solutions to this problem can be considered: one consists in the adoption of a majority-value rule; alternatively, a more complex prediction procedure may be devised, in which the input individual membership is split in fractions along the distribution of the values in the dataset and all the downward paths are followed, making a final decision on the grounds of the leaves that are reached.

Secondly, the induced terminological decision trees may overfit the data, resulting in overly complex models in terms of size. Moving from past solutions for standard decision trees, where this problem has been tackled by resorting to *pruning* steps so to simplify the original model in favor of a better generalization [18], suitable pruning procedures for terminological decision trees have been formalized and implemented.

Finally, the *class-imbalance problem* may affect the quality of the terminological decision trees; this occurs when the number of instances that belong to a target class (or, equivalently, having a certain property value) is smaller than the total number of training instances that are used for building the classification model. In the context of Web ontologies, a strong imbalance in the distributions of the instances may easily occur, due to the abundance of uncertain-membership instances w.r.t. specific concepts as a combined effect of open-world reasoning and the lack of disjointness axioms. A largely adopted solution in the related literature is grounded on the usage of sampling methods. However, this may yield further drawbacks owing to the loss of information when discarding (important) training instances.

In order to mitigate these problems, a *Terminological Random Forests* framework, based on ensemble-models built upon terminological decision trees has been proposed [16]. In *Ensemble Learning* [19] a number of classifiers are trained on portions of the available examples while their predictions are combined by a meta-learner, which generally lessens the *risk of misclassification* [20]. However, the majority vote rule employed in [16] by the resulting classification procedure did not consider specific cases in which the votes for alternative decisions are nearly equal (*too close to call*); experimentally, it has been observed that these cases may even make the ensemble-model more error-prone. For this reason, a different voting mechanism is considered in this work.

Moving from these considerations, in this paper we report on the efforts devoted to evolve terminological decision trees and their extension to terminological random forests – henceforth referred to as *Terminological Tree-based Classifiers* – for addressing the issues mentioned above. We show how they can be used to inductively classify individuals w.r.t. a query concept and how they can be employed to solve the related problem of determining role-fillers with properties with discrete ranges.

In this perspective, the paper is an extended version of the previous works appeared in [12, 16]. The new contributions described throughout the paper are summarized as follows:

- definition of a new framework[2] for the induction of terminological decision trees for DL knowledge bases endowed with improved prediction strategies (with respect to the one proposed in [12]) for solving the problem of intermediate tests with uncertain results and for predicting the filler of a role;
- formalization of *pruning* procedures for terminological decision trees in order to avoid data overfitting;
- extension of terminological random forests framework based on a different prediction strategy (w.r.t. the one described in [16]) that exploits a voting mechanism with a *reject threshold* to avoid misclassification cases;
- new extensive empirical evaluation of the proposed improved frameworks by:
  1. comparing with the previous version of terminological decision tree and random forests [12, 16] and other extensions proposed in [21, 22];
  2. comparing with DL-LEARNER that is the main state of the art system focusing on (as for our work) producing understandable classification models (and slightly considered in the foregoing works);
  3. using, as for DL-LEARNER, standard measure like the F-measure (not considered in the previous experiments);
  4. analyzing efficiency aspects that were only marginally considered in the previous works;
  5. using ontologies regarding various domains and focusing on artificial problems as well as specific class-membership (property value) prediction tasks (previously not investigated) .

The paper is organized as follows: the next section recalls the basics of the Web ontology representation in terms of DLs; Sect. 3 introduces the problem of learning classifiers from examples in DLs, especially in case of imbalanced datasets; Sect. 4 presents the new version of the learning framework based on *Terminological tree-based classifiers* while the empirical evaluation of the proposed framework and comparison with existing learning methods is provided in Sect. 5. Related works on logical decision trees and inductive learning algorithms for DLs are illustrated in Sect. 6 and in Sect. 7, conclusions are drawn jointly with perspectives for further developments.

2. Preliminaries on Description Logics

As the data models of interest are built upon vocabularies described in RDF/OWL2 [23], we will briefly recall the basics of the Description Logics [2], a family of representation languages that constitute the theoretical foundation for the mentioned standard Web ontology languages.

[2]The ultimate implementation of the algorithms is now part of the DL-LEARNER suite (ver. 1.3), see `dl-learner.org`.

The most important building block is a countable set of atomic *concept* names, $\mathsf{N_C}$, where each $C \in \mathsf{N_C}$ stands for a set of objects in the domain of interest, enriched with two further atomic concepts that are the *universal concept* $\top$ and the *bottom concept* $\bot$. A set of atomic *role* names, $\mathsf{N_R}$, is also defined, where elements $R \in \mathsf{N_R}$ denote binary relationships between the domain objects. The objects are endowed with their own names, taken from the set $\mathsf{N_I} = \{a, b, \ldots\}$, and are known as *individuals*[3].

A specific DL language is defined by the set of operators that can be used to build complex concept (resp. role) descriptions and axioms that involve them. The simplest DL language is $\mathcal{AL}$ (*Attributive Language*), where concept descriptions can be formed according to the following operators[4]:

Definition 1 ($\mathcal{AL}$ Operators). *Given $\mathsf{N_C}$ and $\mathsf{N_R}$, a concept description has one of the following forms:*

1. $\neg A$ *(atomic negation)*
   with $A \in \mathsf{N_C} \cup \{\top, \bot\}$;
2. $\exists R.\top$ *(limited existential restriction)*
   with $R \in \mathsf{N_R}$;
3. $\forall R.D$ *(universal restriction)*
   where $R \in \mathsf{N_R}$ and D is a concept description;
4. $D \sqcap E$ *(concept intersection)*
   where D and E are concept descriptions.

More expressive DLs are endowed with further operators[5]. For instance, $\mathcal{ALC}$ (*Attributive Language with Complement*) is the DL obtained by enriching $\mathcal{AL}$ by replacing atomic with *full negation*, extending the applicability of *complement operator* as follows:

5. *given a concept description D, its complement is denoted with $\neg D$.*

This implicitly extends also existential restrictions to the more general *full existential restrictions*, defined as follows:

6. *$\exists R.D$, where $R \in \mathsf{N_R}$ and D is a concept description*

The formal semantics of concept descriptions is defined in terms of interpretations. An *interpretation* is a couple $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is the *domain* of the interpretation and $\cdot^{\mathcal{I}}$ is an interpretation function which assigns $a^{\mathcal{I}} \in \Delta^I$ to each individual $a \in \mathsf{N_I}$, a subset $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ to every atomic concept A and a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to every atomic role R. The interpretation function is extended to concept descriptions by the following inductive definitions:

[3]These names are implemented through URIs (or IRIs in OWL2). They are not necessarily unique by default. We adopt the *Unique Name Assumption* (UNA), hence different individual names refer to different objects [2] since it is crucial for the application of statistical learning techniques to DL knowledge bases [5].

[4]Following the notation proposed in [2], we denote the atomic concept in $\mathsf{N_C}$ with A while complex descriptions are denoted with C, D and E.

[5]For the full set of DL operators see [2].

- $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$
- $\bot^{\mathcal{I}} = \emptyset$
- $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
- $(D \sqcap E)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid a \in D^{\mathcal{I}} \cap E^{\mathcal{I}}\}$
- $(\forall R.D)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \forall b \in \Delta^{\mathcal{I}}, (a,b) \in R^{\mathcal{I}} \rightarrow b \in D^{\mathcal{I}}\}$
- $(\exists R.D)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b \in \Delta^{\mathcal{I}}, (a,b) \in R^{\mathcal{I}} \wedge b \in D^{\mathcal{I}}\}$

Among the various syntactic forms of equivalent concept descriptions we will make use of the *Disjunctive Normal Form* (DNF) [2], i.e. concept descriptions standardized as disjunctions of conjuncts with the negation operator pushed in front of concept names (also recursively in the restrictions).

Sometimes it is useful to integrate knowledge available in terms of specific data types, such as numbers or strings, endowed with their own semantics; hence, DLs have been extended with the so called *concrete domains* [24] so that specific relations can be defined[6] ranging on them. They are used for describing concrete qualities of real world objects such as age, weight, temperature, etc. Generally these domains are indicated in the DLs notation with $\mathbf{D}$.

A DL *knowledge base* is defined as a couple $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, where $\mathcal{T}$ is called *TBox* and contains *inclusion* axioms $D \sqsubseteq C$ (C *subsumes* D) interpreted as $D^{\mathcal{I}} \subseteq C^{\mathcal{I}}$, and *equivalence* axioms $C \equiv D$ as a shorthand for $D \sqsubseteq C$ and $C \sqsubseteq D$, and $\mathcal{A}$ is called *ABox*, and contains factual knowledge about individuals, i.e. *assertions* of kind $C(a)$, interpreted as $a^{\mathcal{I}} \in C^{\mathcal{I}}$, and $R(a,b)$, interpreted as $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a *model* for $\mathcal{K}$ iff it satisfies each axiom/assertion α in $\mathcal{K}$, denoted with $\mathcal{K} \models \alpha$. The set of individuals occuring in $\mathcal{A}$ will be denoted by $\mathsf{Ind}(\mathcal{A})$. Among the inference services that are available in DLs, the most important one for our purposes is *instance-checking* that is the decision procedure assessing if $\mathcal{K} \models C(a)$ holds. As mentioned in Sect. 1, it is important to point out that, differently from the analogous case in different contexts where the CWA is adopted (e.g. databases, logic programming), with open-world reasoning adopted in DL, checking $\mathcal{K} \models C(a)$ and $\mathcal{K} \models \neg C(a)$ may result in a negative answer for both cases. This happens when both assertions are satisfiable by different models of $\mathcal{K}$, i.e. there are $\mathcal{I}$ and $\mathcal{I}'$, such that $\mathcal{I} \models C(a)$ and $\mathcal{I}' \models \neg C(a)$. This fact has important implications on the exact notions of *positive* and *negative example* adopted in the inductive phase, as clarified in Sect. 3.1.

3. Inductive Classification Problems in Description Logics

In this paper, tree-based models are induced from data within Web ontologies, to be employed for predicting class and property assertions. The related methods do not merely fit a classification function for making predictions, as in other statistical approaches where logic reasoning may be exploited in the preliminary data-preparation phase (*materialization*) [5], they rather can learn a model, based on logic features that may even make up an intensional definition of the target concept.

However, the quality of the induced models may be affected in case of skewed data distributions (that is when learning in an imbalance setting), which may easily occur when dealing with Web ontologies (see discussion in Sect. 1). In order to cope with this problem, random forests, as ensembles of tree-based classifiers, are considered.

In this section we first formalize the general problem of fitting an instance classification model. Hence, concept membership prediction is regarded as a classification task successively extended to the case of predicting role fillers (i.e. values of object- or data-type properties). Finally, we discuss the case of the class-imbalance learning and some of the most common solutions that are adopted for coping with it. In Sect. 4, we formally present the methods that we propose for solving the issues introduced in this section.

3.1. Concept Membership Classification Problem

One way to fit a classification model from the available data is to produce an intensional description for a set of training examples. This is known as a (supervised) concept learning problem and consists in finding an intensional feature (in the form of a concept description) that is able to explain (cover) positive examples while ruling out negative ones.

As discussed in the previous sections, it is important to note that, owing to open-world semantics of the DL representation, the assessment of negative examples can be controversial. In our setting positive and negative examples are only those for which the membership w.r.t. the target concept C (resp. its complement $\neg C$) is entailed. While in other learning frameworks it is assumed that a negative example is an individual b for which $\mathcal{K} \not\models C(b)$, similarly to the *learning-from-only-positives* settings [25], we intend to enforce the original semantics of the DL knowledge bases, therefore an individual should be considered as a negative example if it actually belongs to the complement, i.e. $\mathcal{K} \models \neg C(b)$. As a consequence, the rest of the individuals e for which $\mathcal{K} \not\models C(e)$ and $\mathcal{K} \not\models \neg C(e)$ should be treated as *unlabeled* examples, i.e. individuals with a non-definite classification, e.g. as in the *semi-supervised* learning framework [26].

Learning methods for DLs [7, 8, 9, 10, 11] generally solve variants of the following basic problem:

Definition 2 (concept learning problem in DL).

Given

- *the knowledge base* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
- *a target concept name* $C \in \mathsf{N_C}$ *which is not in the signature of* $\mathcal{K}$
- *a set of* training examples *for* C: $\mathsf{Tr} \subseteq \mathsf{Ind}(\mathcal{A})$, *i.e. individuals whose intended membership w.r.t.* C *is assigned (by an expert)*
  - *positive examples* $\mathsf{Ps} = \{a \in \mathsf{Tr} \mid C(a) \in \mathcal{A}_C^+\}$

[6] The *datatype properties* in RDF/OWL.

– *negative examples*
$\mathsf{Ns} = \{b \in \mathsf{Tr} \mid \neg C(b) \in \mathcal{A}_C^-\}$
– *individuals with unknown-membership w.r.t. C:* $\mathsf{Us} = \mathsf{Tr} \setminus (\mathsf{Ps} \cup \mathsf{Ns})$.

where $\mathcal{A}_C^+ \subseteq \{C(a) \mid a \in \mathsf{Ind}(\mathcal{A})\}$ *and* $\mathcal{A}_C^- \subseteq \{\neg C(b) \mid b \in \mathsf{Ind}(\mathcal{A})\}$ *such that* $\mathsf{Ps} \cap \mathsf{Ns} = \emptyset$

Find *a new (equivalence) axiom defining C such that, letting $\mathcal{K}' = \mathcal{K} \cup \{C \equiv D\}$:*

- $\forall a \in \mathsf{Ps}: \ \mathcal{K}' \models D(a)$ *(completeness)*
- $\forall b \in \mathsf{Ns}: \ \mathcal{K}' \models \neg D(b)$ *(consistency)*

As fully correct (i.e. complete and consistent) solutions likely overfit the data and show poor predictiveness over unseen incoming individuals, often more general solutions are to be preferred. Moreover, several variants of this setting can be considered adding or relaxing the constraints on $\mathcal{K}$ and/or D, e.g. regarding its size, the consistency of $\mathcal{K}'$ [27, 28, 29], or even adopting the weaker notion of negative example mentioned before. The output of the concept learning problem is not necessarily meant as a fully automated process: a knowledge engineer would have to validate the induced candidate axioms. Often incremental settings are also considered, where a definition for C is already available in $\mathcal{K}$, yet such a definition may turn out to be incomplete or inconsistent w.r.t. the intended classification of new incoming examples, therefore it should be refined.

The *induced* intensional definition D of the target concept can be used to predict *deductively* the membership of other individuals to this concept. Moving towards a statistical setting, we may generalize the problem aiming at building more general predictive models in terms of *classification functions*. Let $\mathcal{L} = \{+1, 0, -1\}$ be a *label set* denoting the cases of positive membership (labeled with $+1$, that is the case $\mathcal{K} \models C(a)$), negative membership (labeled with -1, i.e. the case $\mathcal{K} \models \neg C(a)$) and uncertain membership (labeled with 0, i.e. neither of the previous two cases) w.r.t. a given target concept C. Hence C may be characterized in terms of a function: $f_C: \ \mathsf{N_I} \rightarrow \mathcal{L}$ whose values are known only for a limited set of instances, from which the training set can be sampled. The general problem of learning an approximation for f_C can be stated as follows:

Definition 3 (inductive classification problem in DL).

Given

- *the knowledge base* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$,
- *a target concept name* $C \in \mathsf{N_C}$
- *a set of* training examples *with assigned labels* $\mathsf{Tr} = \{a \in \mathsf{Ind}(\mathcal{A}) \mid \exists l \in \mathcal{L} : f_C(a) = l\}$
- *a set of classification functions (hypotheses)* $\mathcal{H} = \{h : \mathsf{N_I} \rightarrow \mathcal{L}\}$
- *a* loss *function* $L(h, f_C)$ *that measures the quality of* $h \in \mathcal{H}$

Find *a classification function $h_C \in \mathcal{H}$, such that:*

$$h_C = \arg\min_{h \in \mathcal{H}} \mathbb{E}_{\mathbf{P}}[L(h, f_C)] \tag{1}$$

where $\mathbb{E}_{\mathbf{P}}$ denotes the expectation in terms of the unknown distribution of the individuals.

Hence, finding the solution for the inductive classification problem formalized above consists in finding the classification function (in the set of possible classification functions) minimizing the the adopted loss function L. A natural choice for L in concept learning is the *empirical risk*, that may be defined, in this case, as:

$$L_{er}(h_C, f_C) = \frac{1}{|\mathsf{Tr}|} \sum_{a \in \mathsf{Tr}} \mid f_C(a) - h_C(a) \mid \tag{2}$$

Intuitively, L_{er} averages (w.r.t. the cardinality of the training set) the misclassification errors made by the learned function h_C on the training set Tr when comparing its outcomes with the (known) labels returned by f_C.

The null case corresponds to the complete and consistent solutions for the concept learning problem in Def. 2. However, this is known to be error-prone, as it may easily lead to overfit the data, and as the data regarding training examples may be unable to represent the whole underlying distribution. A good solution should be more general (induction) to be able to predict the correct classification over unseen individuals. To this purpose generally the setting may be biased in favor of simpler solutions, e.g. considering a set $\mathcal{H}$ in accordance to the *Minimum Description Length* principle [30], or targeting suboptimal, yet more general (predictive), h's.

3.2. Predicting Assertions on Roles / Properties

The formalization of the learning problem given in Def. 3 follows the *concept-oriented* nature of the ontologies expressed in DLs. However, in the perspective of a more general service, besides concepts also queries involving role fillers (object- and data-type properties), are to be considered. That is, given an individual a and a role R, predicting v such that $\mathcal{K} \models R(a, v)$, where v ranges on a set of individuals or on the values of some concrete domain, depending on R. In this case, given R and a specific filler b, the labels can be determined as follows:
$\forall a \in \mathsf{Tr}$

- $f_{R,b}(a) = +1$, if $\mathcal{K} \models R(a, b)$
- $f_{R,b}(a) = -1$, if $\mathcal{K} \not\models R(a, b)$

As assessing if $\mathcal{K} \models R(a, b)$ amounts equivalently to deciding if $\mathcal{K} \models \exists R.\{b\}(a)$ holds [2], the prediction task that can be solved fitting a specific classifier $\exists h_{R.\{b\}}$.

The problem definition is analogous for the case of datatype properties, that is for the prediction of property values for roles ranging on a concrete domain, say $\mathcal{D} \in \mathbf{D}$. In this case we assume that the domain is discrete[7]. For more general cases

[7] In general, the task of predicting values belonging to infinite and continuous domains (e.g. real numbers) can be regarded as a *regression problem* for which appropriate models exist, e.g. see [31]. In such cases, the *discretization* of the ranges may be taken into account.

concerning non-numeric ranges (e.g. strings), a *one-vs-all* approach, similar to the usual one for solving multi-class classification problems, can be employed.

3.3. Dealing with The Class-Imbalance Problem: Ensemble Learning

The performance of a classifier may be compromised in presence of skewed datasets. Specifically, the *class-imbalance* problem [19] concerns learners facing an imbalanced distribution of the training examples w.r.t. the classes to be predicted by the classification model. In a binary setting, the problem occurs when training instances belonging to the *majority class* outnumber those belonging to the other (*minority class*). This extends to the settings where models for multiple classes (≥ 3) have to be produced.

In our case, the class can be either a target concept or the relation existing with a given filler. The individuals belonging to the target class may be easily underrepresented, particularly with respect to the unlabeled examples, because of the unavailability of specific axioms that would be required to deem explicit negative examples via deductive reasoning. As such a learning method in an ontological setting ought to take seriously the class-imbalance problem into account.

To cope with this problem, sampling techniques are usually adopted. They are employed to re-balance imbalanced datasets so that a classifier can be trained through standard algorithms as if the distribution were less skewed. Various sampling strategies have been proposed. A straightforward strategy is based on a random selection of the instances, usually performed by either *over-* or *undersampling* instances. In the former case (considering a binary setting problem), a selection of the instances of the minority class is replicated until a (more) balanced distribution is reached, in the latter case instances of the majority class are discarded from the training set. While oversampling may lead to overfitting the data owing to the replicated examples, the undersampling strategy might discard very informative training examples, thus compromising the quality of the resulting model. In order to cope with these issues the idea is to combine sampling strategies with ensemble learning approaches [20]. In ensemble learning, two or more models are trained and their predictions are subsequently combined in a certain way. These approaches are typically employed when the learning problem is hard to solve by using a single-classifier approach. Another reason for combining classifiers is to reduce overfitting.

An ensemble learning model is characterized by two components: the *weak learners* and the *meta-learner*. A weak learner is a generic inductive model that is used in the ensemble classification. The meta-learner is basically a rule for combining predictions from the weak learners. Focusing on the algorithms for learning ensemble classifiers, we can distinguish:

- *boosting algorithms* (e.g. ADABOOST [20]), that resort to an iterative procedure where a new weak learner is added to the ensemble and training instances are weighted according to the hardness of their classification (i.e. iteratively the weights are increased for misclassified instances and decreased for the the others);
- *bagging algorithms* (e.g. those learning random forests [32]), which consider weak learners of the same kind (e.g. decision or regression trees) that are trained with different bootstrap samples drawn from Tr and exploit a majority vote to make predictions;
- *stacking approaches*, which resort to training different models as weak learners, on the entire training set, and then perform a further step for training the meta-learning model to combine the predictions made by the weak learners.

While boosting algorithms aim at reducing the bias of the models, bagging algorithms tend to decreasing the variance. In the perspective of dealing with the class-imbalance problem, a bagging-based method is a more suitable solution than the others due to the sole sampling approach. For instance, when an undersampling strategy is employed, an instance may be discarded when training a weak learner whereas it can be considered for the training of another classifier in the ensemble. Additionally, the ability of the bagging approaches of reducing the risk of models that overfit the data is also useful for tackling an imbalance problem in case of oversampling.

4. Tree-based Terminological Classifiers

In this section we introduce models and methods for solving the problem formalized in Def. 3. After recalling some basics on decision tree learning, we present classification models for the DL representation and the related methods for predicting concept and role assertions exploiting such models.

For the sake of simplicity, this section focuses on the class-membership prediction problem since, as argued in Sect. 3.2, predicting role fillers for a given role and a fixed individual can be cast as a concept-membership prediction problem.

4.1. Decision Trees

Decision Trees [33] are well established classifiers in machine learning. Given the set of labels $\mathcal{L} = \{c_1, c_2, \ldots, c_m\}$, standing for the classes to be predicted, and the training set of examples Tr which are tuples described in terms of a *feature set* F, a decision tree is a structure where each internal node corresponds to a test about the value of some $f \in F$ with departing edges corresponding to the possible test outcomes (feature values), and each leaf-node decides the classification of the routed instances. During the training phase, the test to be installed in a node is selected according to a greedy-strategy which assesses the best feature according to a *purity* measure, such as the *information gain* (*KL divergence*). The resulting tree model induces a partition of the instance space according to the test outcomes along each root-leaf path, where each subset region is assigned with one of the classes in $\mathcal{L}$.

It has been shown that the algorithms typically employed for growing decision trees may overfit the training data thus compromising the accuracy of the model. In order to cope with this problem, various procedures for pruning a decision tree have been proposed (see [18] for a survey). They can be

roughly split into pre-pruning and post-pruning methods. The former are typically based on a stop condition to be enforced on the purity of the nodes while building the tree; consequently, the growth of the tree might be stopped prematurely. The latter typically exploit a two-step approach. At first, the complete tree is grown, afterwards subtrees are evaluated and possibly removed for improving the accuracy on a separate partition of the dataset. Focusing on post-pruning, two well-known methods [34] will be recalled in the following as they will be adapted to the new models proposed in this paper.

Reduced-Error Pruning (REP). In this method the training set is split in a *growing set*, used to induce a complete tree, and a *pruning set* used to prune subtrees. REP starts with a complete tree and, for each internal node, it estimates the classification error on the elements of the pruning set both in the case of keeping the subtree and in the case of replacing the subtree with a leaf. Pruning occurs if the estimated error of the pruned tree is lower than the error made by the original one. A subtree cannot be pruned if it contains another subtree whose misclassification error is lower

Pessimistic Error Pruning (PEP). The method is based on the idea of using the training set for both growing and pruning the tree. Due to this overlap, the empirical error rate is an optimistic estimate of the real one. Therefore, the method adopts a continuity correction for the binomial distribution for obtaining a more accurate estimate and to introduce a complexity factor in the pruning step [18]. The pruning condition is based on a comparison between the standard error of a node and the standard error of the subtree: if the first value is lower than the second one then the subtree is pruned.

4.2. Random Forests

Random Forests (RFs) are ensemble models based on decision trees. Learning RFs is based on a bagging strategy that can be combined with various sampling procedures [32]. In a generic multi-class learning problem, where m is the number of classes and $\mathcal{L} = \{c_1, c_2, \ldots, c_m\}$ is the label set and each c_j is the label value assigned to the examples of j-th class, let F be the set of features characterizing the examples and Tr the training set. Growing a random forest amounts to induce a collection of n decision trees. Each tree is trained by considering a subset $\mathsf{Tr}_i \subset \mathsf{Tr}$, $1 \leq i \leq n$ that is generated through a *sampling-with-replacement* procedure. Besides, for each decision tree, only a restricted random subset $A \subseteq F$ of features is considered at each node.

After growing the trees, using a suitable method, the classification of a new instance a amounts to apply a majority vote rule to the classes predicted by the decision trees. Formally, let h_i be the function which maps an instance onto one of the class-labels in $\mathcal{L}$ learned by inducing the i-th decision tree of the forest and $h_i(a) = c_j \in \mathcal{L}$ the prediction assigned to the instance a by the i-th tree of the forest, with $i \in \{1, \ldots, n\}$, the RFs assigns the class label according to the following rule:

$$h^*(a) = \arg\max_{c_j} \sum_{i=1}^{n} I(h_i(a) = c_j) \qquad (3)$$

where h^* is the classification function for the RFs that assigns a c_j to a and I is an *indicator* function, i.e. $I(x) = 1$ if x is true and $I(x) = 0$ otherwise.The rule basically selects the most frequent label predicted by each tree as the final classification for the instance.

RFs can be combined with a sampling strategy. A version of the algorithm coping with the class-imbalance problem is known as *Balanced RF* [35]. It draws a bootstrap sample from the minority class and randomly selects with replacement the same number of instances from the majority class.

4.3. Terminological Decision Trees

First Order Logical Decision Trees (FOLDTs) [14], were introduced in ILP as extensions of the decision trees to clausal representations. They are defined as binary decision trees where the nodes contain tests in the form of conjunctions of literals and left and right branches depend on the truth-value determined by a logic test on given instances in terms of predicates in a clausal theory.

Similarly, *Terminological Decision Trees* (TDTs) can be informally defined as decision trees for the DLs representation. In this case, a different representation (DL concept descriptions) is adopted for the logical tests in the internal nodes [12]. The resulting classification model is formally defined as follows:

Definition 4 (Terminological Decision Tree). *Given a knowledge base $\mathcal{K}$, a* Terminological Decision Tree *for a target concept C is a binary tree such that:*

- *each internal node contains a (conjunctive) concept description D for class-membership tests on individuals;*
- *the two edges departing from an internal node containing D stand for either outcome of the tests[8] w.r.t. D (left branch) and its complement $\neg D$ (right branch);*
- *each leaf-node determines a decision for the individuals to be classified: it contains either C or its complement $\neg C$.*

Fig. 1 proposes an example of a TDT modeling a particular kind of worker, which could be induced from a set of positive and negative examples. The target concept C is used to denote if an example is such a kind of worker. Note that this TDT features also the counts of training instances that have determined the tree structure; they will be explained later, when the learning phase will be presented. The nodes contain concept descriptions, which can be defined upon the signature of the knowledge base. For the sake of generality, we are not targeting a specific DL. In the figure, atomic concept names and $\mathcal{ALC}$ constructors are used. Defined concepts and role names may be employed by adopting a more expressive DL. As previously mentioned, each edge corresponds to the outcome of a test w.r.t. the concept description at the node (instance checking). Considering an individual $a \in \mathsf{Ind}(\mathcal{A})$ to be classified using the TDT, the edges departing from the root depend on

[8] Given $a \in \mathsf{Ind}(\mathcal{A})$, $D(a)$, resp. $\neg D(a)$, is satisfiable w.r.t. $\mathcal{K}$.

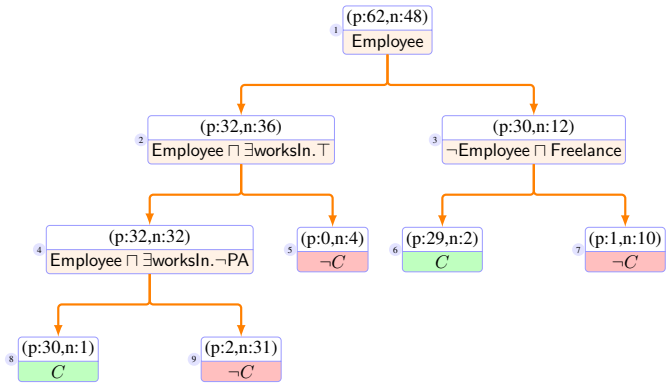

Figure 1: An example of TDT with nodes decorated with the number of positive (p) and negative (n) examples that have reached each node in the training phase. Note that Freelance is supposed to be defined as $\mathsf{Freelance} \equiv \neg(\exists \mathsf{worksIn}.\top)$

the outcomes of the test on the membership to Employee. The left branch is associated to the positive outcome, i.e. a can be an instance of the test concept, that is $\mathsf{Employee}(a)$ is satisfiable w.r.t. $\mathcal{K}$, whereas the right branch corresponds to the case when a can belong to the complement for some model of $\mathcal{K}$, i.e. $\neg\mathsf{Employee}(a)$ is satisfiable. Note that, owing to the open-world semantics, both cases may apply for some individuals, leading them to be routed down both branches. These tests are repeated descending through the branches until leaf nodes are reached which determine the classification to be predicted (as explained later).

Given a TDT, the associated definition of the target concept can be obtained as a disjunction of the conjunctions of the concept descriptions on the paths reaching the leaves labeled with the target concept. Hence, the definition for the target concept C drawn from the TDT in Fig. 1 can be written as follows: $C \equiv (\mathsf{Employee} \sqcap \exists \mathsf{worksIn}.\neg\mathsf{PA}) \sqcup (\neg\mathsf{Employee} \sqcap \mathsf{Freelance})$ where PA is the acronym of *Public Administration*.

Algorithm 1 The routines for inducing TDTs

```
1  const θ: threshold {min. purity rate}
2  function INDUCETDT(Tr, D, C, P̂r): TDT
3  input Tr: training set
4        D: parent concept description
5        C: target concept
6        P̂r: priors
7  begin
8  let Tr = ⟨Ps, Ns, Us⟩ {individuals with positive, negative, uncertain
       membership w.r.t. C}
9  T ← new TDT
10 if |Ps| = 0 and |Ns| = 0 then
11    T.root ← DEFAULTMEMBERSHIP(P̂r, C)
12 else if |Ns| = 0 and |Ps|/(|Ps| + |Ns| + |Us|) > θ then
13    T.root ← C
14 else if |Ps| = 0 and |Ns|/(|Ps| + |Ns| + |Us|) > θ then
15    T.root ← ¬C
16 else
17    S ← ρ(D)
18    { select the best partitioning feature }
19    E* ← SELECTBESTCONCEPT(S, ⟨Ps, Ns, Us⟩)
20    ⟨⟨P^l, N^l, U^l⟩, ⟨P^r, N^r, U^r⟩⟩ ← SPLIT(E*, ⟨Ps, Ns, Us⟩)
21    T.root ← E*
22    T.left ← INDUCETDT(⟨P^l, N^l, U^l⟩, E*, C, P̂r)
23    T.right ← INDUCETDT(⟨P^r, N^r, U^r⟩, ¬E*, C, P̂r)
24
25 return T
26 end
27
28 function INDUCETDT(Tr, C) : TDT
29 input Tr: training set
30       C: target concept
31 begin
32 P̂r ← PRIORDISTRIBUTION(Tr, C)
33 return T ← INDUCETDT(Tr, ⊤, C, P̂r)
34 end
```

4.3.1. Learning TDTs

The algorithm to induce TDTs is based on a *divide-and-conquer* strategy. Given the knowledge base, the target concept, and a training set of examples for the target concept, a recursive strategy is employed to grow the TDT (see Alg. 1).

Preliminarily, two base cases can be distinguished. In the first case, an estimate of the prior probability distribution $\hat{\mathsf{Pr}}$ (computed on the training set by the function PRIORDISTRIBUTION) is used to cope with the special case of a leaf that is lacking of routed (positive and negative) examples ($|\mathsf{Ps}| = 0$ and $|\mathsf{Ns}| = 0$, lines 10-11). The algorithm compares the prior for the positive and negative membership, namely $\hat{\mathsf{Pr}}(+1)$ and $\hat{\mathsf{Pr}}(-1)$, and assigns the label corresponding to the default one. This is made by invoking the function DEFAULTMEMBERSHIP.

In the second case, the condition concerns the homogeneity of the instances w.r.t. the class, i.e. the examples routed to the node should approximately exhibit the same definite membership. The algorithm checks if the label for the leaf node can be determined and set when no negative (resp. positive) example has reached the node, while most of the examples are positive (resp. negative), see lines 12-13 (resp. lines 14-15). This basically means that the purity of the examples goes beyond a given (high) threshold θ.

Note that cases of leaf-nodes reached only by examples with an uncertain-membership may happen. The occurrence of such cases depends on the number of uncertain-membership individuals in the training set that should be lower than the one of those with definite membership. The algorithm determines the label for these nodes according to the prior probabilities.

The third (and recursive) case concerns the availability of both negative and positive examples. In this case, the input concept description D is specialized by means of an operator exploring the search space of downward refinements of D. A set of candidate specializations $\mathbf{S} = \rho(D)$ is obtained. Then, the best description $E^* \in \mathbf{S}$ (according to [12]) is determined and finally installed in the current (internal) node. E^* is one of the admissible specializations of D (chosen according to a purity criterion described in the sequel) obtained by applying a *downward refinement operator* ρ that is a function which maps a concept onto a set of (subsumed) concept descriptions — i.e.

$\rho(D) \subseteq \{E \mid E \sqsubseteq D\}$ — that is required for traversing the (potentially large) space of descriptions, depending on the language adopted (see [36, 37, 8] for more detailed discussions on this specific topic). We will consider a refinement operator that, given D in conjunctive form, computes specializations E in one of the following ways:

ρ_1 adding a concept atom (or its complement) as a conjunct: $E = D \sqcap (\neg)A$;

ρ_2 adding a general existential restriction (or its complement) as a conjunct: $E = D \sqcap (\neg)\exists R.\top$;

ρ_3 adding a general universal restriction (or its complement) as a conjunct: $E = D \sqcap (\neg)\forall R.\top$;

ρ_4 replacing a sub-description D_i in the scope of an existential restriction in D with one of its refinements: $\exists R.E_i$ and $E_i \in \rho(D_i)$;

ρ_5 replacing a sub-description D_i in the scope of a universal restriction with one of its refinements: $\forall R.E_i$ and $E_i \in \rho(D_i)$.

Note that the cases of ρ_4 and ρ_5 are recursive. The implemented operator performs a sort of random sampling from the (very large) space of specializations and it is biased towards small numbers of recursive calls. The employed operator is not *ideal*, i.e. *finite* (for all possible concept descriptions C, $\rho(C)$ is finite), *complete*(for all possible concepts C and D, a concept description E can be obtained by progressively specializing C and $E \equiv D$) and *proper* (for all possible concepts C,D with $D \in \rho(C)$, $C \not\equiv D$ holds) since some constraints cannot be satisfied. For instance, the completeness of the operator cannot be guaranteed due to the fact that the specializations are merely a sampling of all possible refinements (this implies that other refinements might not be returned by ρ).

The best description E^* is the one that maximizes the increment of a *purity* measure, often computed as the difference of the entropy (relative to instances distribution) at a given level w.r.t. the entropy computed at previous level [12]. E^* stands for the refinement that can best separate positive from negative instances. SELECTBESTCONCEPT implements this idea: for each specialized concept $E \in \mathbf{S}$, it estimates the *information gain* deriving from partitioning the individuals routed to the node w.r.t. E. In order to consider the individuals with uncertain-membership, the information gain is computed as:

$$\begin{aligned} \mathsf{Gain}(E, \langle \mathsf{Ps}, \mathsf{Ns}, \mathsf{Us} \rangle) \quad = \quad & H(\mathsf{Ps} \cup \mathsf{Ns} \cup \mathsf{Us}) \\ & -\tfrac{n_l}{n_t} H(\mathsf{P}^l \cup \mathsf{N}^l \cup \mathsf{U}^l) \\ & -\tfrac{n_r}{n_t} H(\mathsf{P}^r \cup \mathsf{N}^r \cup \mathsf{U}^r) \end{aligned} \tag{4}$$

where $H(\mathsf{D}) = -\sum_{i \in \{\mathsf{Ps},\mathsf{Ns},\mathsf{Us}\}} \frac{n_i}{n_t} \log_2(\frac{n_i}{n_t})$ is the entropy function (where n_{Ps}, n_{Ns},n_{Us} are respectively, the number of positive, negative and uncertain instances in D), P^l (resp. P^r), N^l (resp. N^r) and U^l (resp. U^r) stand for the subsets of individuals that routed to the left (resp. right) branch, $n_t = |\mathsf{Ps} \cup \mathsf{Ns} \cup \mathsf{Us}|$ and $n_{l/r} = |\mathsf{P}^{l/r} \cup \mathsf{N}^{l/r} \cup \mathsf{U}^{l/r}|$.

Then the concept description E^* that maximizes this measure is selected.

Example 1 (TDT induction). *Considering the induction of the TDT in Fig. 1, one can observe that the root concept* $\mathsf{Employee}$ *was progressively specialized by means of ρ in the descriptions in the descendant nodes of the leftmost path. Its immediate left-child contains* $\mathsf{Employee} \sqcap \exists \mathsf{worksIn}.\top$ *(refined using ρ_2), further specialized via ρ_4 to* $\mathsf{Employee} \sqcap \exists \mathsf{worksIn}.\neg\mathsf{PA}$*, which describes* employees that do not work in the public administration*, having refined $\top$ in the existential restriction with* $\neg\mathsf{PA} \equiv (\top \sqcap \neg\mathsf{PA}) \in \rho_1(\top)$.

Then the individuals are sorted to the left or right branch by the procedure SPLIT according to the result of the instance-check with respect to E^*, maintaining the same group (i.e. P^l or P^r, N^l or N^r, and U^l or U^r). A training example a is replicated in the left and right partition in case both $E^*(a)$ and $\neg E^*(a)$ are satisfiable w.r.t. $\mathcal{K}$.

The divide-and-conquer strategy is applied recursively until the instances routed to a node satisfy one of the stopping conditions discussed above.

Once a TDT is learned, it can be employed for deriving an intensional definition for the target concept (as briefly discussed earlier). This will not be further discussed since we are focusing on the learning problem in Def. 3.

4.3.2. Concept Membership Prediction with TDTs

The use of a TDT as a classification model is the prediction of the membership of unseen individuals (i.e. further individuals that were not considered in the learning phase) with respect to the target concept.

Given the new individual to be classified, the procedure (see Alg. 2) traverses recursively the TDT starting from the root, performing instance checks with respect to the test concepts contained in each node that is reached: let $a \in \mathsf{Ind}(\mathcal{A})$ and D the concept installed in the current node, if $D(a)$ (resp. $\neg D(a)$) is satisfiable w.r.t. $\mathcal{K}$ then the left (resp. right) branch is followed. Note that, differently from the case of FOLDTs adopting the closed-world setting of clausal theories, both branches may have to be followed in parallel. The final classification is decided by applying a majority criterion to the leaf node labels collected by COLLECTLABELS(), that updates a list $\mathbf{L}$ while traversing recursively the TDT. Cases of ties can be settled either returning a 0 label (no definite membership assigned) as well as the default one (majority class w.r.t. $\widehat{\Pr}$) or by considering also the distributions of positive and negatives examples in the leaves (*weighted* vote). In the experiments, the algorithm assigns the label 0 in such cases.

The service routines ROOT, LEAF and INODE are employed, respectively, to get the root node of a given TDT, to test if a node is a leaf for a tree, and to select the node content.

Example 2 (membership prediction with a TDT). *Considering again the TDT in Fig. 1, suppose a test individual is given such that only the instance-checks in the leftmost path succeed. The algorithm would collect one label (at node 8), hence $\mathbf{L} = [C]$ and the individual would be assigned a positive membership to the target concept.*

Algorithm 2 Concept membership prediction with TDTs

```
function CLASSIFY(a, T, C) : L
input a: test individual
      T: TDT
      C: target concept
begin
 L ← COLLECTLABELS(a, T)
 count[−1, +1] ← {0, 0}

 for each l ∈ L
  if l = C then
      count[+1] ← count[+1] + 1
  else if l = ¬C then
      count[−1] ← count[−1] + 1

 return arg max_{l∈L} count[l] { decision: +1 | −1 }
end

function COLLECTLABELS(a, T) : L
input a: individual
      T: TDT
begin
N ← ROOT(T)
if LEAF(N, T) then
   ⟨D, null, null⟩ ← INODE(N);
   L ← L ∪ {D} { update L }
else
   ⟨D, T.left, T.right⟩ ← INODE(N);
   if D(a) is satisfiable w.r.t. K then
      L ← COLLECTLABELS(a, T.left)
   if ¬D(a) is satisfiable w.r.t. K then
      L ← COLLECTLABELS(a, T.right)

return L
end
```

Conversely, if another individual leads the algorithm to follow multiple paths ending, e.g., in the two leaves 5 and 9, then $\mathbf{L} = [\neg C, \neg C]$ *yielding a negative membership prediction.*

Another case may regard examples for which the test on the root concept leads to follow both branches and eventually with one path ending in node 8, while the others reach nodes 6 and 7, hence $\mathbf{L} = [C, C, \neg C]$*. In this case, the function predicts that the test individual belongs to the target concept.*

4.3.3. Predicting Role Assertions with TDTs

As discussed in Sect. 3.2, assessing if a role assertion holds, e.g. $\mathcal{K} \models R(a, c)$, can be accomplished by deciding if $\mathcal{K} \models \exists R.\{c\}(a)$ holds, which finally amounts to solving a concept membership prediction problem. Specifically, the existential restriction can be used to define the target concept for the learning procedure described above and the corresponding TDT can be used for performing classification. However, differently from the case of concept descriptions, in the case of roles, negative examples can be hardly found in real ontologies, for lack of explicit negative information (disjointness axioms for roles).

Therefore, when the problem of predicting role assertions is considered, a form of closed-world reasoning, requiring a binary setting (*negative-as-non positive*), is adopted thus implying that the training sets will be made up of positive and negative instances only, differently from Alg. 1.

The other main difference when learning a TDT in this case, concerns the splitting test of the instances w.r.t. the node tests. Specifically, while growing the TDT, given the current best concept description E^*, the individuals b for which $\mathcal{K} \not\models E^*(b)$ holds, will be routed down the negative (right) branch only.

After the TDT is grown, the strategy for making predictions requires to follow the paths according to the instance check with respect to the test descriptions installed in the nodes. In this simplified case, the algorithm will simply test whether a given individual a is instance of D ($\mathcal{K} \models D(a)$) or not ($\mathcal{K} \not\models D(a)$).

4.3.4. Pruning Procedures for TDTs

As discussed in Sect. 4.1 and already addressed in [16], algorithms for learning tree based models may suffer from overfitting. For this reason, we integrate suitable pruning methods in the proposed approach with the goal of improving the predictiveness of the TDTs through their simplification. To the best of our knowledge, no specific pruning methods for the general FOLDTs have been proposed. Therefore, it is important to understand to which extent pruning methods for propositional representation can be upgraded to the more expressive DL setting. We focus on post-pruning procedures since, differently from the pre-pruning procedures (see Sect. 4.1), they do not suffer from the drawback of premature stops when growing subtrees. Specifically, we adapt both REP and PEP algorithms (briefly presented in Sect. 4.1) to restructure TDTs. A key point is represented by the choice of the strategy for replacing a subtree during the pruning process, reproducing the idea of the majority class exploited with the (propositional) decision trees. In the case of the TDTs, the target concept C or its complement is used to replace the definition installed into a subtree.

The sketch of the REP pruning procedure for TDTs is reported in Alg. 3 (the adaptation of the PEP method, not reported here, is quite similar). It starts by estimating the error w.r.t. a pruning set of examples, invoking the function ESTIMATEERRORS. This function classifies the individuals in the pruning set by estimating the misclassification rate for each node and, consequently, sub-tree. As an output, the algorithm returns an array *Errors* that is then employed as an argument for the procedure PRUNE. This procedure explores the tree in a bottom-up direction, comparing the error of the node with the estimated error of each subtree T. The subtree is cut if its error is greater than the error of the node. In such a case, the concept description D installed in T is replaced with either C or $\neg C$, according to the distribution of positive or negative instances of the pruning set routed to the node during the error estimation phase. The numbers of positive and negative instances are computed by the auxiliary functions #POSITIVES and #NEGATIVES.

4.4. Terminological Random Forests

Performing classification of individuals in DLs representation requires to cope with the class-imbalance problem (see the discussion in Sect. 3.3). To face this problem, a specific kind of random forests is introduced.

Algorithm 3 The REP routines for TDTs

```
function PRUNETDTS(T, PrSet): TDT
input T: TDT
      PrSet: pruning set of examples
begin
 Errors ← new array
 Errors ← ESTIMATEERROR(T, C, PrSet)
 PRUNE(T, Errors)
end

procedure PRUNE(T, Errors)
input T: TDT
      Errors: array of integer
begin
N ← ROOT(T)
if ¬LEAF(N, T) then
   ⟨D, T.left, T.right⟩ ← INODE(N);
   PRUNE(T.left, Errors);
   PRUNE(T.right, Errors);
      if (Errors[T] < Errors[T.left] + Errors[T.right]) then
         if #POSITIVES(T) > #NEGATIVES(T) then
            T.root ← C
         else
            T.root ← ¬C
         T.left ← null
         T.right ← null
      else
         Errors[T] ← Errors[T.left] + Errors[T.right]
end
```

Algorithm 4 The routine for inducing a TRF

```
function INDUCETRF(Tr, C, n): TRF
input Tr: training set
      C: concept
      n: ℕ
begin
P̂r ← PRIORDISTRIBUTION(Tr, C): {C membership est. priors}
F ← ∅
for i ← 1 to n
   {perform undersampling}
   Tr_i ← BALANCEDBOOTSTRAPSAMPLE(Tr)
   T_i ← INDUCETDT(Tr_i, ⊤, C, P̂r); {see Alg. 1}
   F ← F ∪ {T_i}
return F
end
```

A *Terminological Random Forest* (TRF) is an ensemble of TDTs where the predictions of each weak learner (the single TDT) are combined using a majority voting rule. Specifically, TRFs provide a tool for tackling the class-imbalance problem through the integration of a sampling strategy for mitigating the loss of information provoked by the use of the sole sampling (see discussion in Sect. 3.3). In the following, the procedures for learning TRFs (detailed in Alg. 4) and performing classification with TRFs (formalized in Alg. 5) are described.

4.4.1. Learning TRFs

The procedure for learning a TRF is sketched in Alg. 4. Similarly to a TDT, for learning a TRF, the target concept C and a training set $\mathsf{Tr} \subseteq \mathsf{Ind}(\mathcal{A})$ have to be specified. Additionally, the desired number n of trees for building the forest is needed. As for the induction of TDTs, Tr may contain unlabeled instances, besides of positive and negative examples, with respect to the target concept C. This does not occur when a *negative-as-non-positive* approach is employed, as for the case of prediction of roles (as discussed in Sect. 4.3.3).

The training individuals are sampled with replacement for obtaining subsets $\mathsf{Tr}_i \subseteq \mathsf{Tr}$, with $i = 1, \ldots, n$, needed for learning each TDT, acting as a weak learner. More specifically, the two-step procedure BALANCEDBOOTSTRAPSAMPLE routine, is adopted for building each Tr_i. First a *stratified sampling*, w.r.t. the class distribution, is performed with the goal of ensuring the presence of the instances belonging to the minority class. Successively, oversampling or undersampling [19] is performed[9] for obtaining (quasi-)balanced Tr_i sets. This phase considers the initial data distribution. This means that when the undersampling approach is employed the exceeding part of the counterexamples (resp. positive examples) is randomly discarded. In addition, the sampling procedure removes all the possible unlabeled/uncertain instances.

Once the sets $\{\mathsf{Tr}_1, \ldots, \mathsf{Tr}_n\}$ are obtained, for each Tr_i, a TDT T can be built via Alg. 1 or the modified procedure described in Sect. 4.3.3. However, as reported in Sect. 4.2 for the case of RFs, when building a tree, only a restricted random subset of the whole set of features should be considered for determining the actual best feature to be added for the test node. As a consequence, in the case of TRFs, Alg. 1 is modified to introduce a similar random choice. Specifically, following the approach proposed in [38], after the refinements S of the current node description D are generated, only a certain number of them is considered for assessing the best description $E^* \in S$. Precisely, the number of candidates is reduced through a function g applied to the number of specializations returned by the refinement operator. For the experiments presented in Sect. 5, we use $\sqrt{|\rho(D)|}$ as $g(\cdot)$.

4.4.2. Prediction with TRFs

After a TRF is produced, predictions can be made by relying on the resulting ensemble classification model. The related procedure, sketched in Alg. 5, works as follows. Given an individual a to be classified and a forest $\mathbf{F}$, the function CLASSIFYBYTRF initializes counters for positive and negative labels, respectively. Hence, the algorithm iterates on the forest trees and for each TDT T_i of the forest, the returned class label is collected by invoking the procedure CLASSIFY and incrementing the related counter. Specifically, the procedure CLASSIFY traverses recursively the TDT according to Alg. 2 (or the modified prediction procedure for the *negative-as-non-positive* approach). After polling all TDTs and incrementing the class label counters, the majority voting is employed to finally assign

[9] For the experiments reported in Sect. 5 undersampling has been used in order to decrease the number of instances to be considered thus reducing the computational effort particularly required by instance checks to be performed for each individual.

Algorithm 5 Prediction through TRFs

```
const ϵ: real {eq. threshold}

function CLASSIFYBYTRF(a, F, C) : L
input a: individual
      F: TRF { set of TDTs }
      C: target concept
begin
count[ ] ← new array {counters of the votes collected from the trees}
for each T ∈ F
   l ← CLASSIFY(a, T, C)
   count[l] ← count[l] + 1;
if (|count[+1] − count[−1]|/|F| ≤ ϵ) {approximately equal}
   return 0
else
   return arg max_{l∈L} count[l]
end
```

the class label to the test individual a. When the counters value are approximately the same, the method will assign 0 for the individual a. This can be made by comparing the rate of the votes in favor of the positive and negative membership values against a rejection threshold, namely ϵ, according to prior knowledge about the training set. If the difference is lower than ϵ, the algorithm will return 0.

A larger value for ϵ can be set when it is known that the training set is noisy.

4.5. Further extension of Terminological Tree-based models

After introducing TDTs and TRFs and the algorithms for learning and exploiting such models, we shortly describe two further extensions integrating the Dempster-Shafer Theory sharing some common aspects with the aforementioned models [21, 22] that have been considered in the experiments. For brevity, we assume familiarity with the basic notions about the Dempster-Shafer Theory. The motivation for this alternative models is the need to integrate and represent uncertainty aspects for assessing the membership of an individual.

The TDT model has been extended so that each inner node contains two information: the conjunctive concept description D and a *basic belief assignement* (BBA) quantifying the evidence in favor of a definite membership ($+1$ or -1 whilst the evidence in favor of the label 0 is regarded as the ones in favor of the set $\{+1|-1\}$). The induction of the model, called *Evidential Terminological Decision Tree* (ETDT), adopts a divide-and conquer algorithm like the one illustrated for TDTs. For each node that is installed, the algorithm determines both the concept description (for the intermediate nodes) / the label (for the leaf nodes) and the BBA according to the relative frequencies of the positive, negative and uncertain instances routed to the node. The membership of an individual can be determined by using a modified version of the classification procedure described in the previous section: the algorithm traverses the tree for collecting the BBAs assigned to the leaves; they are subsequently combined according to a *combination rule*. The membership value assigned to an unseen individual is the one maximizing the *belief function* obtained from the combined BBAs.

Evidence Terminological Random Forests [22], an ensemble of ETDTs, extends TRFs by allowing each ETDT to return a BBA that is combined to the other ones by using a meta-learner implementing a further combination rule.

5. Empirical Evaluation

In this section, we report the design and outcomes of an empirical comparative evaluation of the proposed approaches. In the first part, we present a comparative test involving TDTs, TRFs, and some related learning methods on a number of artificial classification problems; in the second part, we propose an evaluation on further datasets with specific learning problems that require the adoption of an extended version of the original refinement operator.

5.1. Datasets

In the experiments, we considered datasets drawn from various ontologies regarding heterogeneous domains (see Tab. 1). Some of them are available on the TONES repository[10]: two medical ontologies BCO and HUMAN DISEASE (HD); an ontology included in the EcoCyc database describing the glycolysis pathway, translated into BioPax format (henceforth BIOPAX); and FINANCIAL, an ontology that models the banking and financial domain.

We considered also further and larger ontologies: CARCINOGENESIS, AIFB PORTAL, DBPEDIA 3.9 and VICODI. CARCINOGENESIS is an ontology representing an OWL port of a well-known ILP dataset [39] which is employed to train relational classifiers for predicting the carcinogenicity of chemical compounds. This ontology is available with the DL-LEARNER [40] release[11].

The AIFB PORTAL dataset contains metadata available from the *Semantic Portal* of the AIFB institute[12] in terms of the SWRC ontology [41]: it models key concepts related to the domain of academic and research communities, such as *persons*, *articles*, *technical reports*, *projects* and *courses*.

DBPEDIA 3.9 makes structured information extracted from WIKIPEDIA available as the central data source of the LOD cloud, providing unique identifiers for the described entities that can be dereferenced over the Web: DBPEDIA 3.9, released in September 2013, describes more than 4 million entities. To generate the dataset for our experiments, the DBPEDIA 3.9 RDF graph was traversed starting from resources representing US presidents and vicepresidents: all immediate neighbors were retrieved, together with their related schema information (direct classes and their superclasses, and their hierarchy).

[10] TONES repository available at: `http://owl.cs.manchester.ac.uk/repository/`

[11] In this work we employed the dataset contained in the release 1.0, available at: `http://sourceforge.net/projects/dl-learner/files/DL-Learner/`

[12] The knowledge base is available at: `http://www.aifb.kit.edu/web/Wissensmanagement/Portal`

The VICODI dataset was drawn from an ontology that formalizes knowledge concerning historical events. It is part of VICODI[13], a collaborative RTD project of the 5FP IST EU Programme aiming at developing a novel visual contextualization environment for digital content on the Internet.

5.2. Experiments on Artificial Learning Problems

The learning task was to induce tree-based classifiers for random target concepts from training sets, using the Web ontologies as background knowledge. The effectiveness of the classifiers has been evaluated on class-membership prediction problems for test individuals with respect to the target concepts.

5.2.1. Dataset Preparation for the Learning Problems

For each ontology, 15 target concepts have been randomly generated by combining 2 through 8 (primitive or defined) concepts of the ontology using the conjunction and disjunction operators or universal and existential restriction (involving roles).

In addition, the generation process avoids returning concept definitions for which the dataset was totally composed by positive (resp. negative) instances and no constraint is explicitly introduced about the number of uncertain individuals: the number, reported in Tab. 2 directly depends on the number of disjointness axioms declared in the knowledge base (see Tab. 1).

Another characteristic of the concept generator is its ability to return concepts where the positive instances largely outnumber the negative ones. Specifically, the generator exploits a recursive procedure to return a concept. The base cases of the procedure are reported below:

1. a concept name is randomly chosen from the signature of the knowledge base;
2. the number of random concepts combined through the conjunction and disjunction operators is larger than the maximum number of concept that can used as operands. As reported before, the number is randomly determined and it span from 2 to 8.

The recursive cases of this procedure are:

1. the application of an existential restriction to a randomly generated concept by introducing a role name;
2. the application of an universal restriction to a randomly generated concept;
3. the application of a complement operator to a randomly generated concept

Note that the role name is also randomly selected from the knowledge base. The generator returns the concept as an output only when there are both positive and negative instances in the knowledge base. Conversely, the procedure discards this concept and it will generate the next one.

As regards the distribution of the individuals with respect to the target concepts in the different ontologies, we observed that negative instances outnumbered the positive ones in the problems/datasets drawn from BCO and HD. In the case of BCO this occurred for all concepts but one, with an averaged ratio between positive and negative instances of $1 : 20$. For most problems, the number of uncertain-membership instances represented only the 10% of the individuals of the knowledge base. In the case of HD, this kind of imbalance occurred for all the target concepts. However, in the case of HD, the number of instances with an uncertain membership was very large (about 90%). In the case of FINANCIAL, larger numbers of positive instances were generally available: for most target concepts the averaged ratio between positive and negative instances was $1 : 8$ and few instances with uncertain membership were found (about over 0.3%). A weaker imbalance could be noted with BIOPAX. For most target concepts the ratio between positive and negative instances was $1 : 5$ on average, while the class distribution was balanced for three concepts only. Again, we observed a limited number of the instances with an uncertain membership: for two concepts we observed only 5 individuals without a definite membership; for the other concepts all the individuals adopted as datasets (training/test set) have a definite membership.

For larger datasets (i.e. those with more than 1000 individuals), we observed a stronger imbalance on the distributions, e.g. the datasets from DBPEDIA, the averaged ratios between positive and negative instances w.r.t. the target concepts were about $1.2 : 100$ while the number of uncertain-membership instances was very large: they represented the 82% of the total number of individuals of the knowledge base.

In the datasets extracted from with CARCINOGENESIS, the ratio between the examples with a definite membership is $100 : 0.02$ (on average) while the examples with uncertain-membership represent almost the 60% of the individual of the knowledge base.

In the datasets extracted from VICODI, the ratio between positive and negative instances w.r.t. the target concepts was again in favor of the positive instances and we found a large number of individuals with an uncertain-membership representing the 95% of the individuals of the knowledge base). Likewise, similar ratio in favor of positive instances and percentages of individuals with uncertain membership were observed also in the datasets from AIFB.

5.2.2. Algorithm Setup and Experiment Design

We ran the TDT induction algorithm[14], both without pruning and with the employment of the REP and PEP procedures. As regards TRFs, they were built by setting the required number of trees parameter to 10, 20, and 30, respectively. The experiments were run by producing TRFs with and without the stratified sampling. When the stratified sampling was used, the

[13] The knowledge base is available at: `http://www.vicodi.org`

[14] The source code as well as the full experimental setting (datasets, target concepts and composition of the folds) and empirical results are available at: `https://github.com/Giuseppe-Rizzo/SWMLAlgorithms`.

Table 1: The ontologies involved in the experiments

Ontology	*DL Language*	*Classes*	*Properties*	*Ind.*	Disj. Axioms
BCO	$\mathcal{ALCHOF}(\mathcal{D})$	196	22	112	279
BIOPAX	$\mathcal{ALCIF}(D)$	74	70	323	85
HUMAN DISEASE (HD)	$\mathcal{ALCIF}(D)$	1498	10	639	0
FINANCIAL	$\mathcal{ALCIF}(D)$	60	16	1000	113
DBPEDIA 3.9	$\mathcal{ALCH}$	251	132	16606	11
VICODI	$\mathcal{ALHI}$	196	10	16942	0
CARCINOGENESIS	$\mathcal{ALCH}(D)$	142	19	22753	6
AIFB PORTAL	$\mathcal{ALEO}$	49	285	44328	0

Table 2: Average rate of uncertain individuals w.r.t. the target concepts

Ontology	*% uncertain individuals*
BCO	10
BIOPAX	1.41
HUMAN DISEASE (HD)	90
FINANCIAL	0.3
DBPEDIA 3.9	82
VICODI	95
CARCINOGENESIS	60
AIFB PORTAL	60

adopted rates were set to 50%, 70%, and 80%. Then, undersampling was performed to re-balance the dataset.

We compared TDT-based and TRF-based classifiers with similar models. Specifically, we compared the extensions against the previous releases of TDT and TRF classifiers performing ternary classification without any strategy for dealing with missing values, proposed in [12] and [16]. We considered also the extensions of such classifiers endowed with evidential reasoning operators, proposed in [42, 22]. In the experiment, we adopted the same sampling rates and the same forest sizes as adopted in those papers. A further parameter that should be set for all tree-based classifiers (and their evidential version) is represented by the threshold θ for tolerating a certain rate of positive/negative instances coverage when a leaf is set in the training phase. In the experiment, we adopted the value $\theta = .95$. The parameter ϵ for controlling the decision in favor of a class-membership values was set[15] to $\epsilon = 0.1$.

The comparison involved also two algorithms implemented in DL-LEARNER: *Class Expression Learning for Ontology Engineering* (CELOE) and *Parallel Class Expression Learning* (PARCEL). CELOE is an extension of OCEL proposed in [9], a learning algorithm that performs an accuracy-driven heuristic search. The main difference between CELOE and OCEL is that the former is more biased towards short concept descriptions. The algorithm can tolerate a certain rate of false negatives, which can be set as a parameter. We tested CELOE at the values 10%, 15%, and 20%. PARCEL [10] is another extension of OCEL that combines top-down and bottom-up refinements in the search space. Top-down search is required in order to solve sub-problems in specific regions of the instance space, while partial solutions are then combined in a bottom-up fashion.

We compared the predictions made using the inductive classification models against the ground truth assessed by a reasoner[16]. The 10-fold *cross validation* technique was employed for estimating the performance of the various algorithms. As an evaluation metric, *macro-averaged F-measure* (i.e. the measure is averaged per membership value) was employed in order to compare our methods to those taken from DL-LEARNER.

5.2.3. Outcomes and Discussion

The results of the first experiment are reported in Tab. 3 and Tab. 4 (average values and standard deviations). They show that the new methods outperform the previous releases of the tree-based classifiers (TDTs and TRFs) and their performance is similar to the one observed for ETDTs and ETRFs (although the trees induced through such approaches are more complex, as we will describe in the sequel). This was likely due to the new discipline in the presence of missing values implemented in the new releases. Conversely, with the BIOPAX problems, as the KB contained disjointness axioms so that individuals tended to have a definite membership w.r.t. the target concepts, the performance of the various classifiers was quite comparable. These results show also that our new methods can outperform PARCEL and CELOE with problems built on all datasets but HD. TDTs seem to work also in case of small datasets (in terms of individual population) while the extended models, i.e. the pruned versions of TDTs, seem to work better when more instances are available, likely owing to their need for a validation set. A similar consideration also holds for TRFs. It can be also noted that, when REP and PEP were employed, the F-measure improved or, at least, did not change. In addition, for most small ontologies employed in the experiments, even when the performance of the various methods seems to be comparable, a lower standard deviation than the ones of PARCEL and CELOE can be observed (e.g. see the BCO column): the proposed solutions could learn better approximations for some target concepts than those obtained through DL-LEARNER methods.

Concerning the learning problems on HD, the tree-based methods preformed sensibly worse than PARCEL and CELOE. These results were likely due to the fact that, owing to the nature of the underlying ontology, the classification procedure with trees was inclined to assign a definite membership that could not be determined by the DL reasoner to test individuals, and this counted as a mistake. These cases could be judged more

[15] Lower values θ and ϵ were also tested but they lead to poorly predictive classifiers

[16] The PELLET reasoner was used: `http://clarkparsia.com/pellet/`.

Table 3: Results of the first experiment on artificial problems (for the group of smaller datasets): TDTs vs. other methods

Algorithm		*F-measure*			
		BCO	BIOPAX	HD	FINANCIAL
TDT (old)		50.35 ± 13.45	97.62 ± 04.55	66.49 ± 06.43	76.14 ± 05.71
TRF 50% (old)	20 trees	51.45 ± 13.45	98.84 ± 02.16	70.43 ± 03.77	90.34 ± 06.57
TDT	(no pruning)	75.34 ± 07.35	97.63 ± 04.54	75.88 ± 07.85	87.14 ± 08.29
	REP	87.46 ± 03.48	96.11 ± 03.56	35.76 ± 14.71	79.15 ± 07.84
	PEP	87.47 ± 03.36	97.23 ± 04.54	35.76 ± 14.71	75.88 ± 07.85
TRF 50%	20 trees	92.35 ± 03.46	98.84 ± 02.16	77.52 ± 04.85	96.61 ± 06.57
ETDT		75.39 ± 07.46	97.63 ± 04.54	75.85 ± 07.53	87.14 ± 08.29
ETRF 50%	20 trees	90.27 ± 05.37	98.84 ± 02.16	77.52 ± 04.36	96.61 ± 06.57
CELOE	noise 10%	88.85 ± 12.61	92.72 ± 11.19	90.59 ± 26.00	71.52 ± 34.53
	noise 15%	88.86 ± 12.61	92.72 ± 11.19	90.58 ± 26.00	74.56 ± 27.32
	noise 20%	88.86 ± 12.60	92.72 ± 11.19	90.58 ± 26.00	75.68 ± 27.56
PARCEL		90.86 ± 13.45	93.33 ± 11.44	90.60 ± 26.01	80.43 ± 12.05

Table 4: Results of the experiments on artificial problems for the group of larger datasets

Algorithm		*F-measure*			
		CARCINOGENESIS	AIFB	DBPEDIA 3.9	VICODI
TDT (old)		60.33 ± 03.46	97.63 ± 04.54	65.88 ± 07.85	85.14 ± 08.29
TRF 50% (old)	20 trees	64.32 ± 09.45	98.84 ± 02.16	70.43 ± 04.43	95.51 ± 04.65
TDT	no pruning	61.45 ± 02.25	87.24 ± 01.45	73.21 ± 02.03	88.14 ± 01.45
	REP	62.05 ± 02.03	87.22 ± 01.75	76.23 ± 03.43	90.14 ± 03.45
	PEP	62.03 ± 02.03	87.24 ± 01.46	77.21 ± 03.43	90.14 ± 03.45
TRF 50%	20 trees	70.35 ± 03.15	87.23 ± 01.45	87.23 ± 01.24	98.21 ± 02.95
ETDT		62.05 ± 02.03	97.63 ± 04.54	75.88 ± 07.85	87.14 ± 08.29
ETRF 50%	20 trees	70.34 ± 03.15	98.84 ± 02.16	87.23 ± 01.45	98.20 ± 02.96
CELOE	10% noise	70.67 ± 05.15	80.34 ± 03.27	84.45 ± 05.09	96.15 ± 07.45
PARCEL		70.67 ± 05.26	80.25 ± 02.17	86.45 ± 04.28	98.27 ± 03.25

appropriately by a domain expert. Moreover, a *small disjunct problem* may have affected the TDTs resulting in the induction of error-prone models [43]. The issue concerns the fact that TDTs induced during the training phase had several branches covering few individuals. This was due to limitations in the choice of the candidate test concept descriptions that led to a nearly null information gain, hence the installed test concepts tended to route most of the individuals down to one of the branches departing from the nodes. These structures resulted in a large number of misclassification cases.

It is worthwhile to note that no sensible difference was observed when using TRFs induced with the various re-sampling rates. Indeed, the application of TRFs was not as successful as awaited (we expected a further improvement of the F-measure). Likely, this could be due to a strong overlap existing among TDTs of the forests. As a result of this limited diversification in the ensemble, wrong predictions returned using a single TDT (related to the small disjuncts problem) were further confirmed by others. Besides the method performance did not change sensibly with varying sampling rates and numbers of trees due to the presence of uninformative examples, i.e. examples that did not alter the choice of the best test concept description. These results suggest that one could choose the simplest combination of parameters in order to reduce the time required for building the models (or enlarge the beam of candidates).

The experiments with larger datasets seem to confirm the feasibility of the proposed extensions, although the performance is not as good as the one of the small datasets. For instance, in the case of CARCINOGENESIS, the poorer performance was likely due to a strong class-overlap resulting into very frequent misclassification cases. Similar consideration holds also for AIFB PORTAL, DBPEDIA and VICODI.

Model Quality and Training Efficiency. Regarding the complexity of the induced models, the details are reported in Tab. 5–6, namely: the number of nodes, the number of paths leading to positive/negative leaves , the time required for the training phase (including the time required for pruning the trees, when such a strategy was applied) and the time required for classifying test individuals. The values are averages both w.r.t. the number of concepts and the number of folds.

A first consideration concerns the comparison of the size of the TDTs w.r.t. ETDTs. We noticed that TDTs were shorter than ETDTs. This was likely due to the heuristic employed for growing the models: growing ETDTs requires to select the best concept description corresponding to the one with the most definite membership among the candidate concepts. This means that the heuristic does not focus on the positive/negative instances, resulting into a purity condition on the training instances routed to the node that could be satisfied after more splits. For TDTs and TRFs, the total number of nodes and the paths which lead to leaves labeled with the positive/negative class depend on various factors, like the imbalance ratio, and especially the ability of the refinement operator to lead to a good test concept which can split the training instances into almost pure subsets. Trees with longer paths were due to the aforementioned small disjunct problem that affected various datasets like HD. In this perspective, resorting to pruning methods reduced the number of paths, in particular the ones leading to the leaves assigning a negative membership (e.g. in the case of BIOPAX or CARCINOGENESIS).

As regards the efficiency of classification adopting the induced models, we noticed that, despite the fact the methods proposed in this paper were generally slower than the training and test performed through CELOE and PARCEL (see Tab

Table 5: Details regarding the TDTs induced for the learning problems considered in the first experiment, with and without the application of pruning algorithms, for the group of smaller datasets

Dataset		TDT	TDT+REP	TDT+PEP	ETDT	TDT (old)
BCO	nodes	07.47	04.59	07.09	19.00	07.47
	paths for C	02.00	01.00	02.00	04.50	02.00
	paths for $\neg C$	02.00	02.00	02.00	05.50	02.00
	training time	$20s$	$30s$	$30s$	$60s$	$20s$
	classification	$20s$	$15s$	$15s$	$20s$	$3s$
BIOPAX	nodes	07.77	03.88	07.37	14.30	07.77
	paths for C	02.50	01.00	02.51	03.50	02.50
	paths for $\neg C$	07.47	02.59	01.40	05.50	07.47
	training time	60.3s	65s	65.5s	78.5s	60.3s
	testing time	15s	3s	15s	35.6s	15s
HD	nodes	07.60	04.50	07.60	14.30	07.60
	paths for C	02.50	01.25	02.51	03.50	02.50
	paths for $\neg C$	02.43	03.24	02.50	05.50	02.43
	training time	$180s$	$200s$	200s	430s	$180s$
	testing time	10.60s	5.30s	5.30	20s	07.54s
FINANCIAL	nodes	25.40	03.84	20.40	40.20	25.40
	for C	08.50	01.00	05.10	14.50	08.50
	paths for $\neg C$	06.43	02.00	11.23	24.50	06.43
	training time	35s	75s	75s	70s	35s
	testing time	$15s$	25s	25s	25s	$13s$

Table 6: Details regarding the TDTs induced for the learning problems considered in the first experiment, with and without the application of pruning algorithms, for the group of larger datasets

Dataset		TDT	TDT+REP	TDT+PEP	ETDT	TDT (old)
CARCINOGENESIS	nodes	40.50	23.40	23.40	40.50	40.50
	paths for C	05.60	05.60	05.60	10.70	05.60
	paths for $\neg C$	10.50	05.70	05.70	10.50	10.50
	training time	713s	786s	786s	712s	715s
	testing time	15.7s	8s	15s	16.6s	7.5s
AIFB	nodes	25.60	13.40	13.40	25.60	25.60
	paths for C	07.00	02.89	03.56	07.00	07.00
	paths for $\neg C$	08.50	04.60	03.67	08.50	08.50
	training time	1877s	2042s	$2042s$	1889s	1873s
	testing time	56s	24.5s	24.5s	76.6s	43.6s
DBPEDIA 3.9	number of nodes	25.75	13.25	13.25	25.75	25.75
	number of paths for C	07.47	02.45	02.45	07.47	07.47
	number of paths for $\neg C$	08.53	04.75	04.75	08.53	08.53
	training time	67s	92.5s	92.5s	67s	43s
	testing time	15s	7.5s	07.5s	25s	8.5s
VICODI	number of nodes	25.67	13.94	13.94	25.73	25.67
	number of paths for C	06.35	02.45	02.45	07.32	06.35
	number of paths for $\neg C$	09.49	04.75	04.75	07.43	09.50
	training time	73s	85s	85s	72s	73s
	testing time	17s	10s	10s	25s	9s

5 - 7), the training/test times required by the new releases of TDT and TRF classifiers are quite limited (although the current implementation can be further optimized) and, as expected, they are better than the cases of the ETDTs (and ETRFs, respectively). Specifically, the times required for learning TDTs (without pruning) span from less than 10s (as in the case of BCO) to 1877s (AIFB). In addition, the time for classifying individuals through TDTs spans from 2.5s (BCO) to 56s (AIFB). Training and testing times are considerably larger in the case of TRFs. The induction of a TRF required from 352s (in the experiments with BCO) to 8462s (in the experiments with AIFB).

While the training time for such models is strictly related to their complexity, the testing time is affected by the strategy exploited for pooling the labels: the rules exploited by ETDTs and ETRFs (based on evidential operators) required more time than simpler majority voting rules. In particular, the training time required by ETDTs induction algorithm spans from 60s (in the case of BCO) to 1889s (in the case of AIFB) while the testing time spans from 3s to 43.6 s. In the case of ETRFs, the training time spans from 425s to 9453s and the testing time spans from 65.6 (FINANCIAL) to 389s (CARCINOGENESIS).

Comparing TDTs' classification under OWA and CWA. In order to determine the extent to which the incompleteness of a knowledge base affects the quality of both the models and the predictions made through them, we shortly illustrate the results of a further comparison where, given the settings for learning problems and the algorithms adopted so far, a binary settings is used, according to the Closed World Assumption. The results suggested that the performance was drastically lower than the one obtained by solving a ternary classification problem. We omitted to report the results into a table because, for most datasets, the F-measure was about 0.3. Only for two ontologies, i.e. BIOPAX, FINANCIAL, the F-measure was quite similar to the cases where the problem is solved under OWA. For BIOPAX the F-measure was 97.61 for TDTs without employing pruning, 96.10 in the experiments with TDT and the employment REP pruning and 97.24 in the case of TDT with PEP. In the experiments with FINANCIAL, the F-measure was 87.20 for TDTs without pruning, 79.16 and 75.92 by applying REP and PEP pruning. This was likely due to the presence of disjointness axioms involved in the refinement process.

For most datasets, the decrease was likely due to very sim-

Table 7: Execution times for the others models involved in the first experiment

Dataset		TRF	ETRF	TRF (old)	Celoe	Parcel
BCO	training	352s	425s	352s	12.6s	12.7s
	test	134s	167s	117s	20s	3s
BIOPAX	training	256s	325s	325s	78.5s	60.3s
	test	95.5s	125s	95.6s	35.6s	25.6s
HD	training	363s	435s	363s	83.2s	93s
	test	123.4s	173.5s	78.5s	78.5s	78.7s
FINANCIAL	training	123.4s	154.3s	123.4s	35.7s	60.6s
	test	65.6s	65.6s	45.3s	25s	13s
CARC.	training	713s	786s	786s	712s	715s
	test	358.3s	389.6s	355.3s	433.8s	425.7s
AIFB	training	8422s	9453s	8422s	457s	459s
	test	56s	76.7s	24.5s	257s	258s
DBPEDIA 3.9	training	430.7s	432.4s	430.7s	235.7s	365.6s
	test	127.4s	143.6s	116.76	76.35s	86.76s
VICODI	training	125.65s	125.67s	125.67s	85.67s	96.67
	test	65.7s	86.76s	54.1s	43.8s	43.8s

Table 8: TDTs induced in the first experiment under CWA (small datasets)

Dataset		TDT	TDT+REP	TDT+PEP
BCO	nodes	03.00	03.00	03.00
	paths for C	01.00	01.00	01.00
	paths for $\neg C$	01.00	01.00	01.00
	training time	5s	8s	8s
	classification	2.5s	2.5s	2.5s
BIOPAX	nodes	07.77	03.88	07.37
	paths for C	02.50	01.00	02.51
	paths for $\neg C$	07.47	02.59	01.40
	training time	60.3s	65s	65.5s
	testing time	15s	3s	15s
HD	nodes	03.00	03.00	03.00
	paths for C	01.00	01.00	01.00
	paths for $\neg C$	01.00	01.00	01.00
	training time	10s	15s	15s
	testing time	2s	2s	2
FINANCIAL	nodes	25.40	03.84	20.40
	for C	08.50	01.00	05.10
	paths for $\neg C$	06.43	02.00	11.23
	training time	35s	75s	75s
	testing time	15s	25s	25s

Table 9: TDTs induced in the first experiment under CWA (large datasets)

Dataset		TDT	TDT+REP	TDT+PEP
CARCINOGENESIS	nodes	03.00	03.00	03.00
	paths for C	01.00	01.00	01.00
	paths for $\neg C$	01.00	01.00	01.00
	training time	35s	76s	76s
	testing time	15.7s	8s	15s
AIFB	nodes	03.00	03.00	03.00
	paths for C	01.00	01.00	01.00
	paths for $\neg C$	01.00	01.00	01.00
	training time	5s	10s	10s
	testing time	2s	2s	2s
DBPEDIA 3.9	number of nodes	03.00	03.00	03.00
	paths for C	01.00	01.00	01.00
	paths for $\neg C$	01.00	01.00	01.00
	training time	2s	10s	10s
	testing time	2.5s	2.5s	2.5s
VICODI	number of nodes	03.00	03.00	03.00
	paths for C	01.00	01.00	01.00
	paths for $\neg C$	01.00	01.00	01.00
	training time	2.5s	11.5s	11.5s
	testing time	2s	2s	2s

ple and poor predictive models produced through the binary setting (see Tab. 8–9) while, solving a ternary classification problem, the instances with uncertain-membership w.r.t. the intermediate tests (replicated in the recursive calls) helped to induce models describing better the available instances. In most cases the trees were made up of a test node while the other ones contained predictions. As a result, the application of the pruning was almost useless. Moreover, for such problems, the binary settings affected also the performance of TDTs when they were used to make predictions through a TRF. In fact, the overlap between the trees seemed to be increased, in comparison with the trees induced adopting the ternary settings.

5.3. Second Experiment

In the second experiment, we investigated the effectiveness of the proposed approach when applied to some specific learning problems related with the datasets rather than artificial ones.

5.3.1. Learning Problems

Considering the OWL ontologies used in the previous experiment (see Tab. 1), we briefly describe the specific target concepts on which tree-based classifiers were trained and tested. The learning problems have been chosen by considering the availability of a sufficient number of assertions for the target role with a moderate sparsity of the fillers. When possible, properties involving at least 100 different individuals in their assertions have been selected (please note this means that, for each property, the number of assertions may be more than 100).

In the case of BCO, the task was to predict the filler for the target property refersdesease ranging in the set of 12 individuals that are instance of the concept DiseaseType or SyntomsType. The learning problem has been tackled through a *one-vs-all approach*. Due to the limited number of instances available , we used all the individuals of the knowledge base. In this case, the ratio between the positive and negative instances was around $1:4$ (on average w.r.t. the various role fillers).

The learning problem to be solved with BIOPAX ontology was to predict the value of the target property controlType, a datatype property whose range is an enumerative type. In this work, we focused on the problem to decide if an individual assumes one of the values $\{\mathsf{activation}, \mathsf{inibition}\}$. The ratio between the positive and negative examples was around $1:3$.

In the experiments with HD, we tried to use TDTs and TRFs for predicting the values of a boolean datatype property isCIE (CIE stands for *Center for International Education*). In this case we considered a subset of 436 individuals that were evenly distributed with respect to the property values.

In the case of FINANCIAL, the goal was to predict the value of an object property hasSexValue, whose range is the set of individuals SVSEXMALE and SVSEXFEMALE. The learning problem considered 420 individuals, with a ratio between the instances with the value SVSEXMALE and those with the value SVSEXFEMALE of around $1:3$.

In the case of CARCINOGENESIS, the task was to predict whether a compound was *mutagenic*. Note that, the prediction problem concerned a boolean datatype property (see Sec. 3) isMutagenic, which encodes the targeted information. The domain of this property is the class Compound. Hence the learning problem can be considered as a subclass-membership prediction problem: the training set was composed exclusively of instances of such a class for which the property value (true or false) is known. The distribution of the examples was balanced.

In the experiments with the AIFB PORTAL the goal was to predict the affiliation of a *person* to a *research group* (property Forchunsgruppe). A *one-vs-all* approach was employed in the assessment of the affiliation to a specific group. This means that, given a research group representing the filler for the considered role, the individuals belonging to other research groups were considered as counterexamples. For this dataset, the imbalance ratio was around $1.4:10$ (on average w.r.t. the various research groups). This means that for each group 12% of the instances represented positive examples while the rest was considered as negative examples.

For the DBPEDIA 3.9 ontology, the task was to predict if a generic politician is also a congressman. The training set was composed by both positive and negative examples (i.e. politicians that are not congressmen) and, unlike the other cases, also instances with an uncertain-membership (i.e. politicians for which it is not possible to determine if they are congressmen given the knowledge base). The imbalance ratio between positive and non-positive instances was around 1:4.

Finally, in the case of VICODI, the aim was to predict if an historical event concerned politics or not. In the ontology, this is modeled via an object property hasCategory whose domain is the class Time-Dependent and the range is the class of Category (with only 4 individuals), including POLITICS. In this learning problem, we considered a dataset composed of positive (individuals having POLITICS as role filler) and negative instances with an imbalance ratio of $1.6:4$.

5.3.2. Setup of The Algorithms

In this second experiment, again we assessed the performance of the TDTs and TRFs and included also PARCEL and CELOE. Initially, we had adopted exactly the same settings described in the previous experiment for all of the algorithms. However, differently from the previous experiment, the preliminary attempt to tackle learning problems with the new datasets fell short in most cases and the reason was the adoption of the same refinement operator. Indeed it turned out that these ontologies required operators which are able to involve roles ranging on concrete domains (i.e. datatype properties, see Sect. 2) and the related restrictions. In the next section we illustrate the extensions operated per dataset.

It is also worthwhile to note that in the experiments with AIFB we could not obtain results because of API exceptions occurred for the reasoner employed for determining the instance check-test results [17]. We could not circumvent the problem even by trying to replace the instance-checking service provided by the DL reasoner with the fast instance checker available with DL-LEARNER [40]. As the fast instance checker precomputes and stores in memory the instances of named classes and the property relationship occurring in the AIFB ontology, it tended to rapidly exhaust the available memory.

5.3.3. Outcomes

The average results of the second experiment for the related learning problems are reported in Tab. 10 and Tab. 11. More detailed comments per dataset follow.

BCO, BIOPAX, HD, FINANCIAL. For these datasets, the performance was quite good in terms of F-measure (see Tab. 10).

In the learning problems considered in the experiments, the (macro-averaged) F-measure obtained through the adoption of binary tests was higher than the one obtained through the previous release of TDT-based and TRF-based classifiers, where ternary tests both on an intermediate concept and its complement were performed. In the case of ternary tests, the refinement operator could not add candidates that allow to discern well the examples, ending up with a decrease of the model predictiveness that was likely due to employment of concepts for which disjointness axioms were not available. In the case of binary tests, the resulting models turned out to be simpler than those obtained in the ternary setting due to the fact that no replication of the training individuals was performed. Similarly to the previous experiment, we also observed that the models were affected by the small-disjunct problem, but pruning could mitigate this phenomenon by cutting some branches thus leading to an improvement of the F-measure. In the case of BIOPAX, no sensible difference was observed between the new and the old release of the tree-based classifiers. In all experiments with these datasets the performance was comparable to the one obtained through ETDTs and ETRFs. However, ETDTs and ETRFs were more time-consuming owing to the more complex rule for pooling the decisions on the labels to be assigned. Note that in this case no adaptation of the original refinement operator was necessary.

CARCINOGENESIS. The results on this dataset show that this turned out a difficult problem for TDTs (also using pruning procedures) compared to TRFs, PARCEL and CELOE (see Tab. 11).

In particular, the performance of TDTs was far from being comparable to the one of PARCEL and CELOE. But the results slightly improved with an extension of the operator (explained below), with a very limited improvement when pruning algorithms were employed (due to the scarce complexity of the unpruned models). Conversely, TRFs exhibited a comparable

[17] The exception occurred during the experiments was due to an unsupported operation of the reasoner employed. It was related to a bug that could not be fixed without affecting the behavior of the reasoning system.

Table 10: Results of the second experiment. TDT(old), TRFs (old), ETDT and ETRFs classification procedures are the ones presented in [12, 16, 42]

Algorithm		*F-measure*			
		BCO	BIOPAX	HD	FINANCIAL
TDT	old	73.45 ± 13.45	98.33 ± 02.55	63.15 ± 07.35	76.14 ± 05.71
TRF 50%	old | 20 trees	85.34 ± 04.35	98.84 ± 02.16	63.15 ± 07.35	90.34 ± 06.57
TDT	(no pruning)	91.00 ± 11.46	98.34 ± 02.54	80.43 ± 03.42	87.14 ± 08.29
	REP	90.27 ± 07.48	98.11 ± 03.56	83.45 ± 14.71	90.45 ± 09.65
	PEP	90.28 ± 11.46	98.23 ± 04.54	83.14 ± 13.69	89.62 ± 07.48
TRF 50%	20 trees	95.80 ± 02.35	98.84 ± 02.16	90.43 ± 03.47	96.61 ± 06.57
ETDT		91.00 ± 11.47	97.63 ± 04.54	75.85 ± 07.53	87.14 ± 08.29
ETRF 50%	20 trees	95.76 ± 02.76	98.84 ± 02.16	90.35 ± 03.36	96.60 ± 06.60
CELOE	noise 10%	92.35 ± 12.61	100.0 ± 00.00	90.59 ± 26.00	71.52 ± 34.53
PARCEL		90.86 ± 13.45	100.0 ± 00.00	90.60 ± 26.01	80.43 ± 12.05

Table 11: Results of the second empirical evaluation on larger datasets

Algorithm		*F-measure*			
		CARCINOGENESIS	AIFB	DBPEDIA 3.9	VICODI
TDT	old	54.45 ± 03.25	50.34 ± 03.45	50.56 ± 02.47	71.43 ± 03.26
TRF 50%	old | 20 trees	54.45 ± 03.25	50.35 ± 01.45	50.73 ± 02.36	71.43 ± 03.26
TDT	no pruning	54.45 ± 03.25	87.24 ± 01.45	87.21 ± 04.08	71.43 ± 03.26
	REP	55.67 ± 02.30	87.22 ± 01.75	87.22 ± 04.09	76.45 ± 02.35
	PEP	54.43 ± 03.25	87.24 ± 01.46	87.21 ± 04.08	76.57 ± 02.47
TRF 50%	20 trees	73.06 ± 04.25	87.23 ± 01.45	87.23 ± 04.08	98.21 ± 02.95
ETDT		54.45 ± 03.25	87.24 ± 01.45	87.21 ± 04.08	71.43 ± 03.26
ETRF	20 trees	54.45 ± 03.25	87.24 ± 01.45	87.21 ± 04.08	71.43 ± 03.26
CELOE	noise 10%	73.66 ± 11.15	not available	87.22 ± 04.09	97.15 ± 07.45
PARCEL		73.67 ± 11.14	not available	87.22 ± 04.08	98.25 ± 03.15

performance due to a good diversification among the various induced trees, which allowed to take into account various features that were not considered by single trees. Similarly to PARCEL and CELOE, ensemble models yielded a .73 F-measure.

Interestingly, TRFs (but also TDTs) turned out to be more stable than the competitors (note the lower standard deviation) likely because different models are averaged beforehand to make the single prediction.

In the evaluation, we examined the (features in the) learned concept descriptions. Generally, in the case of the experiments with CARCINOGENESIS, the quality of the induced concepts was quite good. Specifically, using CELOE, concept descriptions such as Compound and hasAtom only not (Sulfur-77) were learned. It is worthwhile to note that the refinement operator employed by PARCEL and CELOE can exploit datatype properties, unlike the refinement operator reported in Sect. 4.3 that involves only object properties. However, it was sufficient to extend the refinement operator by adding a further case *allowing restrictions on boolean data properties* (i.e. those selecting individuals for which the filler is true). Rerunning the experiments with this extension of the ref. operator we noticed an improvement of the performance.

AIFB Portal Ontology. Regarding the experiments with AIFB, we can notice the large number of individuals available, the preponderance of roles and the expressiveness of the ontology. Specifically, its terminology is really shallow and some constraints are not adequately represented. For instance, for most roles the domain and range information is missing. Hence, we tried to extend the refinement operator by allowing the introduction of *existential/universal restrictions over a randomly selected set of nominals*.

Adopting this extension of the refinement operator we observed a significant improvement of the performance. This may suggest that, for a shallow ontology like AIFB, a refinement operator exploiting relationships w.r.t. individuals may find good features for the classification models compared to others which merely use the available TBox. However, the resulting models may be very specific w.r.t. the training individuals. In terms of number of nodes, the trees sizes were comparable to the sizes of those built with the original refinement operator. With the employment of the pruning methods, we did not observe further improvements of the F-measure (similarly to the experiments with the original refinement operator). As regards the employment of TRFs, we observed that performance did not change by increasing the forest size. Again, this was likely due to a weak diversification among the various trees.

DBpedia 3.9. In the case of DBPEDIA 3.9, owing to the amount of classes available in the ontology, the refinement operator for our methods *has been extended by introducing a seed class as a starting point for the specialization process*. In this case the seed was set to Person.

Similarly to the previous cases, the performance for all the methods employed in the experiments is good enough with an F-measure around 87%. However, the resulting concept descriptions induced with CELOE and PARCEL were generally slightly shorter yet semantically equivalent to ones encoded by TDTs, denoting a certain redundancy in the latter. The performance of all methods was also quite stable due to the limited variations in the output models.

In general in the experiments with TDTs, simple models were output that could not be further reduced via pruning methods (differently from other experiments, such as those with CARCINOGENESIS).

From a qualitative viewpoint, in the various replications, the concept learned by the DL-LEARNER methods was not (Governor) and not(Senator), which seems to be consistent with the notion of congressman. The corresponding TDTs can be

translated to a very similar yet slightly more redundant concept description: Person and not (Governor) and not(Senator).

Vicodi. Finally, for the VICODI dataset we adopted the same extension of the refinement operator as in the experiments with DBPEDIA 3.9. We set the concept name Flavor as a seed.

In this case, a lower F-measure of the plain TDTs was observed than the ones obtained via PARCEL and CELOE. However the performance could be improved via both REP and PEP pruning, yielding similar F-measure values.

In the experiments with TRFs, we noticed a further improvement which led to a similar performance w.r.t. PARCEL and CELOE. In this case, the training instances came from various regions of the space of the instances which were covered by the refinement operator thanks to the frequent introduction of new concept names. Therefore, the random feature selection produced a good diversification among trees which had a positive effect on the ensemble models.

Quality and Efficiency of Tree Models. As regards the complexity of the classification models, expressed in terms of number of nodes, the induced trees turned out to be quite complex (in particular for group of smaller datasets).

In general, the models were simpler than those induced by the previous versions of the algorithms for both TDTs and ETDTs. Similarly to the previous experiments, the new versions of TDTs were simpler than ETDTs also due to the heuristic exploited for growing the trees (which does not consider the positive and negative examples).

Regarding the comparison w.r.t. the previous release of TDT learning algorithms, we noticed that adopting a binary setting allowed to avoid the training instances replication, which affected the size (and therefore also the efficiency) of the models in the previous experiments. This basically implied that the purity condition was frequently satisfied earlier than with the previous version of TDTs [12, 13].

For BCO and FINANCIAL we observed that the REP pruning was quite effective in terms of trees size. In particular for BCO, the algorithms tended to overprune the trees resulting in a (quite limited) decrease of the F-measure.

In the other cases, the application of pruning strategy was less effective to simplify the trees. Despite this, an improvement in terms of F-measure was observed. Indeed, as reported in Tab. 14, the average number of nodes was much lower for CARCINOGENESIS and DBPEDIA 3.9 whereas the complexity of the trees was much higher for the other datasets/problems. This may explain why the models obtained via pruning methods did not improve considerably the performance. On the other hand the employment of the pruning methods did not affect the performance for AIFB. This was due to the installed intermediate concept descriptions that did not produce more error-prone models than the corresponding pruned versions.

For the VICODI dataset we observed that the pruning methods had some impact on the size of the models, although this was rather limited. Both the PEP and REP procedures allowed for reducing the complexity of the models, but there is no significant difference between the proposed methods.

These results, together with those of the previous experiment, suggest that other factors should be considered to choose a pruning algorithm over another, such as the number and distribution of the training instances and the computational costs.

Concerning the efficiency of the methods, TDT-based and TRF-based classifiers are more time-consuming than PARCEL and CELOE. Even though the current prototypical implementation demands an optimization, efficiency seems to be acceptable for all datasets but AIFB. This was due to the constraints that are not adequately represented in this dataset, so that a lot of time is required to the underlying reasoning services, e.g. for deciding if the membership of an individual is negative.

5.4. Comparing the effectiveness of the extended refinement operators

The experiments with large ontologies required the extension of the refinement operator in order to exploit different information in the knowledge base. These extensions have been tested also in the experiments with the small ontologies for completeness. Tab. 15-17 illustrate the outcomes of this evaluation.

As regards the extension adopted for exploiting data properties (see Tab. 15), we noted for BCO and BIOPAX that the outcomes were exactly the same obtained employing the original refinement operator, due to the lack of boolean properties to be considered for generating specializations. In the experiments with HD and FINANCIAL, a further improvement in terms of F-measure was observed for all the tree-based models. This was due to the introduction of useful tests that allowed for inducing less error-prone models than those produced adopting the former version of the refinement operator. However, the performance did not considerably change likely because of the limited number of informative boolean properties. The F-measure increased by about 2%.

In the experiments that aimed at assessing the effectiveness the refinement operator extended for supporting the usage of nominals (see Tab. 16), we noticed that the performance of the resulting models was worse than the one exhibited by the models induced via all the learning algorithms considered in the evaluation (Tab.10). In general, these results were due to concept-oriented nature of the small ontologies. While the AIFB ontology is characterized by plenty of role assertions and a shallow schema, assertional knowledge about roles contained in ontologies like BCO,HD, FINANCIAL is usually sparse: for various roles they contain only few assertions with different fillers. The extended refinement operator could not cope with this sparseness. As a result, the algorithms tended to introduce non-discriminant tests in the models causing them to be affected by the small disjunct problem more easily than the cases in which the original version of the operator was employed. In this perspective, the pruning procedures improved the overall performance simplifying the trees through the removal of the tests involving nominals.

Finally, Tab. 17 illustrates the outcomes of the experiments with the new refinement operator extended by introducing seed-concepts. The outcomes show a further improvement of the F-measure compared to the results obtained adopting the original

Table 12: Execution times for TDTs / ETDTs w.r.t. the learning problems considered in the second experiment

Dataset		TDT	TDT+REP	TDT+PEP	ETDT	TDT (old)
BCO	training	8.5*s*	17.6*s*	17.6*s*	75*s*	15*s*
	classification	6.5*s*	6.5*s*	6.5*s*	20.4*s*	3.4*s*
BIOPAX	training	34.6s	60.6s	60.6s	85.6s	36.7s
	classification	7.5s	3.6s	3.6*s*	35.6s	8.4s
HD	training	73*s*	104*s*	104s	186.9s	87.2*s*
	classification	10.60s	5.30s	5.30	20.5s	8.54s
FINANCIAL	training	35s	75s	75s	70s	35s
	classification	15*s*	25s	25s	25s	13*s*
CARCINOGENESIS	training	357s	467.45s	467.8s	645s	489.40s
	classification	8.7s	8.7s	8.7s	16.6s	7.5s
AIFB	training	183.63s	215.70s	216.69s	457.70s	563.86s
	classification	34.7s	34.7s	34.8s	76.6s	43.6s
DBPEDIA 3.9	training	34.7s	47.3s	47.4s	75.7s	75.7s
	classification	4.3s	4.3s	4.2s	7.5s	6.43s
VICODI	training	43.76s	50.70s	50.70s	89.45s	50.76s
	classification	12.7s	12.8s	12.8s	23.67s	9.7s

Table 13: Execution times for the others models involved in the second experiment

Dataset		TRF	ETRF	TRF (old)	Celoe	Parcel
BCO	training time	47*s*	65s	75*s*	14.6*s*	16.8*s*
	classification	35.4s	47.3s	43.7*s*	5.6*s*	5.6*s*
BIOPAX	training time	34.7s	95s	95s	78.5s	60.3s
	testing time	95.5s	125s	95.6s	35.6s	25.6s
HD	training time	75.8*s*	86.8s	87.8s	34.7s	93.4s
	testing time	12.15s	25.43s	25.43s	78.50s	78.70s
FINANCIAL	training time	67.45s	154.3s	123.4s	35.7s	43.7s
	testing time	65.6*s*	65.6s	45.3s	25s	13*s*
CARCINOGENESIS	training time	438.75s	657.87s	539.43s	397.87s	398.43s
	testing time	67.43s	75.85s	56.17s	18.36s	18.43s
AIFB	training time	8422s	567.32s	568.43s	not available	not available
	testing time	56s	76.7s	24.5s	not available	not available
DBPEDIA 3.9	training time	107.43s	107.43s	145s	35.35s	35.43s
	testing time	15.7s	43.8s	43.9	4.3s	4.3s
VICODI	training time	117.28s	165.85s	125.67s	25.67*s*	26.73
	testing time	35.82s	45.7s	40.81s	7.6s	7.8s

refinement operator. This was due to the fact that, thanks to the introduction of a concept as a seed, the tree-induction algorithms could discard *a priori* useless tests to be installed into inner nodes as they could lead to branches supported by no individual.

6. Related Works

Various machine learning methods have been proposed in the literature for knowledge discovery tasks on Semantic Web knowledge bases – see [5] for a survey – and, more specifically, for *concept learning*, *clustering* and *assertion prediction* [44].

We focused on methods inspired by *Inductive Logic Programming*. In particular, we targeted extensions of the standard decision trees towards multi-relational representations; the *First-Order Logic Trees* [14], proposed for clausal spaces have been extended to DL languages, i.e. *Terminological Decision Trees* [12, 13]. Therefore, in this section, we consider as strictly close the classification models based on logic features. A number of related methods deriving from ILP have been proposed. YINYANG [7] and its successor DL-FOIL [8] are prototypical algorithms that adopt a *separate-and-conquer* learning strategy, unlike the divide-and-conquer methods for inducing TDTs.

Further related methods are CELOE performs an accuracy-driven search through the concept space [9] by means of a complex refinement operator which exploits class, role and datatype property hierarchies, and PARCEL, a learning algorithm that combines top-down and bottom-up refinements in the search space [10]. Top-down search is required in order to solve subproblems related to the specific regions of the instance space, while partial solutions are then combined in a bottom-up fashion. These methods are biased towards shorter concept descriptions. These characteristics are related to the pruning procedures employed with tree-based models.

Both algorithms are currently implemented in DL-LEARNER [40], a framework for concept learning in DLs that takes into account scalability issues and adopts a specific bias, such as the *Partial Closed World Assumption*, for an efficient computation of the underlying heuristics based on approximate inferences: in a two-step approach, basic inferences are computed by a reasoner and stored so that, later, the instance checker can exploit the stored inferences for determining the concept coverage over the instances.

Learning TDTs is also strictly related to the *bisimulation* approach [45, 46]. It exploits the notion of *standard model* which is a finite interpretation of a knowledge base. Besides of the Unique Names Assumption, differently from the presented TDT learning method, also the CWA is made. Another difference regards the employment of a set of pre-computed selectors, i.e. tests to partition the set of individuals. The bisimulation method has been used also in the perspective of the *roughification* of the target concepts [47] to address the problem of defining *imprecise* concepts (uncertainty as *vagueness*). Throughout these works, a modified version of DL-FOIL has

Table 14: Average size (number of nodes) of TDTs induced for the learning problems considered in the experiments, with and without the application of pruning

Learning Problem	TDT	TDT+REP	TDT+PEP	ETDT	TDT (old)
BCO	16.50	07.47	07.37	25.70	25.70
BIOPAX	07.00	07.00	07.00	07.00	16.50
HD	07.60	04.50	07.60	14.30	14.30
FINANCIAL	25.40	03.84	20.40	40.20	40.20
CARCINOGENESIS	07.00	04.30	07.00	07.00	07.00
AIFB	51.60	51.60	51.60	51.60	51.60
DBPEDIA 3.9	07.00	07.00	07.00	07.00	07.00
VICODI	54.62	49.78	49.76	54.62	54.62

Table 15: Results of the second experiment. TDT(old), TRFs (old), ETDT and ETRFs classification procedures with the extended refinement operator–use of datatype properties

Algorithm		*F-measure*			
		BCO	BIOPAX	HD	FINANCIAL
TDT	old	73.45 ± 13.45	not applicable	67.25 ± 05.46	77.34 ± 05.64
TRF 50%	old | 20 trees	85.34 ± 04.35	98.84 ± 02.16	67.15 ± 05.37	93.43 ± 06.43
TDT	(no pruning)	91.00 ± 11.46	98.34 ± 02.54	82.45 ± 08.43	88.45 ± 08.27
	REP	90.27 ± 07.48	98.11 ± 03.56	84.36 ± 09.84	91.23 ± 09.63
	PEP	90.28 ± 11.46	98.23 ± 04.54	84.35 ± 09.86	90.43 ± 07.34
TRF 50%	20 trees	95.80 ± 02.35	98.84 ± 02.16	92.53 ± 02.56	98.43 ± 05.43
ETDT		91.00 ± 11.47	97.63 ± 04.54	76.97 ± 07.42	88.45 ± 08.32
ETRF 50%	20 trees	95.76 ± 02.76	98.84 ± 02.16	92.43 ± 03.42	98.42 ± 06.47

been exploited and the fuzzification of the properties ranging in the numerical domain [27, 28, 29].

Also TDT models have been extended for tackling aspects of uncertainty related to the incompleteness of the knowledge in the SW. The extended TDT models that integrate aspects of the *Dempster-Shafer Theory* (DST) have been compared against other learning methods in DL-LEARNER [21, 42]. Unlike the TDT model proposed in this paper, the new model employs a different heuristic and is characterized by a different representation. In addition, the model can provide measures of uncertainty that can exploited by other algorithms, as shown in [22].

Like various other types of classification models, also those based on decision trees are known to suffer from the problem of *overfitting* the training examples. The related literature describes various solutions for the problem. The underlying idea of such approaches is to restructure a tree in order to improve the ability of predicting the correct class for unseen instances: most of them are based on a simplification of the tree by using *pruning methods*. As described in Section 4, these algorithms may be generally ascribed to either the pre-pruning or the post-pruning class of methods. Most of the research has focused on the latter with various algorithms proposed [48, 17].

Empirically it was shown how the most common methods allow to improve the accuracy although the performance is often comparable. These methods mainly differ in terms of number of nodes. Indeed, REP pruning tends to *overprune* decision trees. An early detecting outliers that may determine overfitting models has also been tackled in the case of DL representation through suitable distance-based clustering methods [49].

The class-imbalance problem is quite well-known in machine learning [19]but it has been often disregarded in the context of learning for the Semantic Web [50] and recently addressed in [16]. Various techniques have been devised in *machine learning* to deal with the skewness in the data. The most common ones are based on sampling methods, as described in Sect. 3. These techniques span from the random undersampling and oversampling approach to more sophisticated approaches as SMOTE [51, 19].

Further methods basically represent modified versions of well-known algorithms for making the resulting models less sensitive w.r.t. the problem. Finally, other approaches combine *ensemble learning* and *sampling* strategies. Among them, one may mention the *Balanced Random Forests* [35] that extend the original ones [32] by combining sampling strategy and ensemble learning. Specifically, this ensemble model is based on a randomized version of the decision trees.

An upgrade of this model to integrate multi-relational logic representations was proposed in [38]. Our TRFs have been firstly introduced in the context of SW knowledge bases in order to improve approaches to approximate query answering [16]. Recently, a further extension of the TRFs has been proposed [22] that integrates evidence combination operators, borrowed from the mentioned DST approach, in the majority vote classification procedure of the TRFs.

However, only the class-membership prediction task has been considered so far. In this perspective, this work represents an extension of both [16] and [12], tackling a more general problem of assertion prediction. In order to do this, we followed an approach that is similar to [6], where an instance-based learning method was exploited instead.

As regards the task of datatype property prediction, in this paper we focused on properties with a discrete range. Conversely, the related problem of predicting numerical values of a function was tackled in [31] where a method based on regression tree models was proposed, namely *Terminological Regression Trees*, whose leaves contain regression functions (based on the routed training instances) and the intermediate nodes are conjunctive DL concept descriptions generated through the same refinement operators employed for TDTs and TRFs.

In the discussion of the experiments, we mentioned how important the diversification of weak learners is to improve the predictiveness of ensemble models. A recent work [52] focuses exactly on this point: classifiers are combined for maximizing accuracy and diversity through the *kernelization* of the output

Table 16: Results of the second experiment. TDT(old), TRFs (old), ETDT and ETRFs classification procedures with the extended refinement operator (using nominals)

Algorithm		*F-measure*			
		BCO	BIOPAX	HD	FINANCIAL
TDT	old	70.35 ± 12.30	95.46 ± 02.45	60.08 ± 07.47	74.15 ± 05.54
TRF 50%	old | 20 trees	84.15 ± 04.65	95.48 ± 02.45	69.15 ± 07.35	89.76 ± 06.43
TDT	(no pruning)	89.15 ± 10.23	94.15 ± 02.15	78.87 ± 03.40	85.32 ± 04.26
	REP	84.26 ± 07.76	96.10 ± 04.78	81.65 ± 13.04	88.23 ± 09.74
	PEP	87.95 ± 08.43	96.15 ± 04.23	81.98 ± 11.14	87.43 ± 08.23
TRF 50%	20 trees	91.54 ± 03.34	94.54 ± 02.87	89.45 ± 03.98	95.54 ± 03.43
ETDT		89.15 ± 10.23	94.15 ± 02.15	78.87 ± 03.40	85.32 ± 04.26
ETRF 50%	20 trees	84.15 ± 04.65	95.48 ± 02.45	69.15 ± 07.35	89.76 ± 06.43

Table 17: Results of the second experiment. TDT(old), TRFs (old), ETDT and ETRFs classification with the extended refinement operator (using seeds)

Algorithm		*F-measure*			
		BCO	BIOPAX	HD	FINANCIAL
TDT	old	76.34 ± 12.15	99.46 ± 02.55	68.25 ± 06.16	78.24 ± 06.13
TRF 50%	old | 20 trees	87.24 ± 03.47	99.57 ± 03.21	67.25 ± 07.35	92.36 ± 04.24
TDT	(no pruning)	92.42 ± 10.23	99.46 ± 02.78	83.21 ± 02.89	85.14 ± 08.29
	REP	93.75 ± 07.98	98.87 ± 03.56	86.45 ± 14.71	94.45 ± 08.23
	PEP	93.46 ± 09.47	98.89 ± 03.54	88.37 ± 11.70	92.62 ± 03.46
TRF 50%	20 trees	96.02 ± 01.23	99.15 ± 03.24	92.23 ± 02.14	97.10 ± 05.43
ETDT		92.43 ± 10.21	99.47 ± 02.79	83.20 ± 02.90	85.17 ± 08.32
ETRF 50%	20 trees	96.03 ± 01.22	99.14 ± 03.25	92.22 ± 02.15	97.11 ± 05.42

of the involved classifiers. Another related aspect (not considered in this work) concerns the optimization of an ensemble classifier through a pruning criterion that aims at maximizing the diversity of weak learners [53].

7. Conclusion and Extensions

We have provided comprehensive overview of our learning framework tackling the inductive classification problem of individual resources in the context of DL knowledge bases. We targeted predictive models which are ultimately based on logic features, as a trade-off between accuracy of the predicted membership and comprehensibility of the resulting classifier. The solution is based on the notion of *terminological decision tree* which combines the divide-and-conquer learning strategy with the adoption of logic features for the node tests.

We summarized various novel technical extensions related to such a classification model, both regarding its induction and its usage for classification, that were proposed to address possible issues with the datasets. In particular, the refinement operator has been extended as to perform a random traversal of the space of refinements. Also, the original classification procedure has been extended to follow multiple paths down the TDTs and the decision is taken on account of all reached leaves. With this procedure the classifier is more likely to assign a definite membership. Besides, we have shown that these models could be also exploited to determine the filler of given roles.

We have addressed also the possible issues with the training data distribution (class-imbalance), which affects the induction of predictive classification models, resorting to a combination of sampling methods with ensemble learning techniques. As a result, the terminological random forest model has been introduced. We also devised a strategy for simplifying terminological decision trees and, hence, for improving the predictiveness of the model. Our solution relies on a modification of pruning algorithms adapted to deal with the underlying representations.

An extensive empirical evaluation of the latest release of the implementation of the presented algorithms has been performed on datasets extracted from publicly available ontologies. We considered datasets for both artificial and well-defined learning problems with ontologies of varying complexity in terms of terminology and number of examples. The experiments were useful to elicit strong and weak points of the various tree-based models. A comparison with related concept learning algorithms from the DL-Learner suite allows also for a qualitative evaluation of the logic features (concept descriptions) discovered by the algorithms, which suggests the direction for further extensions of the models and the related algorithms.

The experiments have also shown that exploiting an ensemble learning model like the terminological random forest allows for better results compared to a single model approach when a sufficient diversification of training sets is allowed by a large number of instances (and an expressive terminology).

In specific cases the employment of pruning algorithms has proven its effectiveness to reduce the overfitting, especially for ontologies with a large number of primitive concepts. Experiments have shown that a REP and PEP pruning can be used for simplifying terminological decision tree models. A drawback of REP concerns the fact that it requires a separate pruning set for estimating the error rate. However, in the context of SW, this should not be a problem as the Linked Data Cloud[18] may represent an accessible source of training instances to be sampled for learning purposes. Another lesson learned is that using a refinement operator regardless of the specific ontology may fail to capture important features for discerning the membership of the individual resources. Conversely, the operator should be tuned to better address the particular learning problem to be solved, improving the performance of the resulting model.

In the future, we plan to extend the methods along various directions. One direction regards the choice of refinement op-

[18] `http://lod-cloud.net`

erators that may be applied in order to generate more discriminative intermediate test.

Further ensemble techniques and novel rules for combining the answers of the weak learners could be also employed. We plan also to investigate the effectiveness of strategies aiming at optimizing the ensemble to improve the diversification of the classifiers, which is an important characteristic of ensemble learning methods [53, 52]. This also suggests additional investigations for the experimental evaluation of the TRFs. Specifically, further comparisons could consider: the employment of boosting methods, the employment of oversampling strategy for generating the bootstrap samples and different heuristics for choosing the candidate refinements. The method could be parallelized in order to be deployed as a non-standard reasoning service through an engine for large-scale data processing, e.g. *Apache Spark*[19].

Finally, the idea of using TDTs for determining role fillers, which may be also related to link discovery and regression tasks, deserves further investigations.

[1] T. Heath, C. Bizer, Linked Data: Evolving the Web into a Global Data Space, Synthesis Lectures on the Semantic Web, Morgan & Claypool Publishers, 2011.

[2] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider (Eds.), The Description Logic Handbook, 2nd Edition, Cambridge University Press, 2007.

[3] C. d'Amato, N. Fanizzi, F. Esposito, Query answering and ontology population: An inductive approach, in: Proceedings of ESWC'08, Springer, 2008, pp. 288–302.

[4] N. Fanizzi, C. d'Amato, F. Esposito, ReduCE: A Reduced Coulomb Energy network method for approximate classification, in: L. Aroyo, et al. (Eds.), Proceedings of ESWC 2009, Vol. 5554 of LNCS, Springer, 2009, pp. 323–337.

[5] A. Rettinger, U. Lösch, V. Tresp, C. d'Amato, N. Fanizzi, Mining the Semantic Web - Statistical learning for next generation knowledge bases, Data Min. Knowl. Discov. 24 (3) (2012) 613–662.

[6] G. Rizzo, C. d'Amato, N. Fanizzi, F. Esposito, Assertion prediction with ontologies through evidence combination, in: F. Bobillo, et al. (Eds.), Uncertainty Reasoning for the Semantic Web II, Vol. 7123 of LNAI, Springer, 2013, pp. 282–299.

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

[8] N. Fanizzi, C. d'Amato, F. Esposito, DL-FOIL, Concept learning in description logics, in: F. Zelezný, N. Lavrac (Eds.), Proceedings of ILP 2008, Vol. 5194 of LNAI, Springer, 2008, pp. 107–121.

[9] J. Lehmann, S. Auer, L. Bühmann, S. Tramp, Class expression learning for ontology engineering., J. Web Sem. (2011) 71–81.

[10] A. C. Tran, J. Dietrich, H. W. Guesgen, S. Marsland, Two-way parallel class expression learning, in: Proc. ACML 2012, 2012, pp. 443–458.

[11] J. Lehmann, N. Fanizzi, L. Bühmann, C. d'Amato, Concept learning, in: Lehmann and Voelker [44], pp. 71–91, ch. 2.

[12] N. Fanizzi, C. d'Amato, F. Esposito, Induction of concepts in web ontologies through terminological decision trees, in: J. Balcázar, et al. (Eds.), Proceedings of ECML/PKDD 2010, Vol. 6321 of LNAI, Springer, 2010, pp. 442–457.

[13] N. Fanizzi, C. d'Amato, F. Esposito, Towards the induction of terminological decision trees, in: S. Y. Shin, et al. (Eds.), Proceedings of ACM-SAC 2010, ACM, 2010, pp. 1423–1427.

[14] H. Blockeel, L. De Raedt, Top-down induction of first-order logical decision trees, Artif. Intell. 101 (1-2) (1998) 285–297.

[15] Y. Dimopoulos, A. Kakas, Abduction and inductive learning, in: L. DeRaedt (Ed.), Advances in Inductive Logic Programming, IOS Press, 1996, pp. 144–171.

[16] G. Rizzo, C. d'Amato, N. Fanizzi, F. Esposito, Tackling the class-imbalance learning problem in semantic web knowledge bases, in: K. Janowicz, et al. (Eds.), Proceedings of EKAW 2014, Vol. 8876 of LNCS, Springer, Heidelberg – Germany, 2014, pp. 453–468.

[17] J. R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, San Francisco, CA, USA, 1993.

[18] F. Esposito, D. Malerba, G. Semeraro, A comparative analysis of methods for pruning decision trees., IEEE Trans. Pattern Anal. Mach. Intell. 19 (5) (1997) 476–491.

[19] H. He, Y. Ma, Imbalanced Learning: Foundations, Algorithms, and Applications, 1st Edition, Wiley-IEEE Press, 2013.

[20] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, 2nd Edition, Springer, 2009.

[21] G. Rizzo, C. d'Amato, N. Fanizzi, On the effectiveness of evidence-based terminological decision trees, in: F. Esposito, et al. (Eds.), Proceedings of ISMIS 2015, Vol. 9384 of LNAI, Springer, 2015, pp. 139–149.

[22] G. Rizzo, C. d'Amato, N. Fanizzi, F. Esposito, Inductive classification through evidence-based models and their ensembles, in: F. Gandon, et al. (Eds.), Proceedings of ESWC 2015, Vol. 9088 of LNCS, Springer, 2015, pp. 418–433.

[23] OWL 2 Web ontology language document overview, W3C Recommendation (2009).
URL `http://www.w3.org/TR/owl2-overview/`

[24] C. Lutz, Description logics with concrete domains-a survey, in: P. Balbiani, et al. (Eds.), Advances in Modal Logic 4, papers from the fourth conference on "Advances in Modal logic", King's College Publications, 2002, pp. 265–296.

[25] C. Elkan, K. Noto, Learning classifiers from only positive and unlabeled data, in: Proceedings of KDD'08, ACM, 2008, pp. 213–220.

[26] X. Zhu, A. B. Goldberg, Introduction to Semi-Supervised Learning, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan and Claypool Publishers, 2009.

[27] U. Straccia, M. Mucci, pFOIL-DL: learning (fuzzy) EL concept descriptions from crisp OWL data using a probabilistic ensemble estimation, in: R. L. Wainwright, et al. (Eds.), Proceedings of the 30th Annual ACM Symposium on Applied Computing, Salamanca, Spain, April 13-17, 2015, 2015, pp. 345–352.

[28] F. A. Lisi, U. Straccia, Learning in description logics with fuzzy concrete domains, Fundam. Inform. 140 (3-4) (2015) 373–391.

[29] F. A. Lisi, U. Straccia, An inductive logic programming approach to learning inclusion axioms in fuzzy description logics, in: F. Fioravanti (Ed.), Proceedings of CILC 2011, Vol. 810 of CEUR Workshop Proceedings, CEUR-WS.org, 2011, pp. 57–71.
URL `http://ceur-ws.org/Vol-810/paper-l04.pdf`

[30] P. D. Grünwald, The Minimum Description Length Principle, MIT Press, 2007.

[31] N. Fanizzi, C. d'Amato, F. Esposito, P. Minervini, Numeric prediction on OWL knowledge bases through terminological regression trees, Int. J. Semantic Computing 6 (4) (2012) 429–446.

[32] L. Breiman, Random forests, Machine Learning 45 (1) (2001) 5–32.

[33] J. R. Quinlan, Induction of decision trees, Mach. Learn. 1 (1) (1986) 81–106.

[34] J. R. Quinlan, Simplifying decision trees, Int. J. Man-Mach. Stud. 27 (3) (1987) 221–234.

[35] C. Chen, A. Liaw, L. Breiman, Using random forest to learn imbalanced data, Tech. rep., Department of Statistics, University of Berkeley (2004).
URL `www.stat.berkeley.edu/users/chenchao/666.pdf`

[36] J. Lehmann, P. Hitzler, Foundations of refinement operators for description logics, in: H. Blockeel, et al. (Eds.), Proceedings of ILP 2007, Revised Selected Papers, Vol. 4894 of LNCS, Springer, 2008, pp. 161–174.

[37] J. Lehmann, P. Hitzler, A refinement operator based learning algorithm for the *ALC* description logic, in: H. Blockeel, et al. (Eds.), Proceedings of ILP 2007, Revised Selected Papers, Vol. 4894 of LNCS, Springer, 2008, pp. 147–160.

[38] A. V. Assche, C. Vens, H. Blockeel, S. Dzeroski, First order random forests: Learning relational classifiers with complex aggregates, Machine Learning 64 (1-3) (2006) 149–182.

[39] A. Srinivasan, R. D. King, S. Muggleton, M. J. E. Sternberg, Carcinogenesis predictions using ILP., in: Proc. of ILP 1997, 1997, pp. 273–287.

[40] J. Lehmann, DL-learner: Learning concepts in description logics, Journal of Machine Learning Research 10 (2009) 2639–2642.

[19] `http://spark.apache.org`

[41] Y. Sure, S. Bloehdorn, P. Haase, J. Hartmann, D. Oberle, The swrc ontology – semantic web for research communities, in: C. Bento, et al. (Eds.), Progress in Artificial Intelligence, Vol. 3808 of LNCS, Springer, 2005, pp. 218–231.

[42] G. Rizzo, C. d'Amato, N. Fanizzi, F. Esposito, Towards evidence-based terminological decision trees, in: A. Laurent, et al. (Eds.), Proceedings of IPMU 2014, Vol. 442 of CCIS, Springer, 2014, pp. 36–45.

[43] G. Weiss, The impact of small disjuncts on classifier learning, in: R. Stahlbock, S. F. Crone, S. Lessmann (Eds.), Data Mining, Vol. 8 of Annals of Information Systems, Springer US, 2010, pp. 193–226.

[44] J. Lehmann, J. Voelker (Eds.), Perspectives on Ontology Learning, AKA/IOS Press, 2014.

[45] T. Tran, Q. Ha, T. Hoang, L. A. Nguyen, H. S. Nguyen, Bisimulation-based concept learning in description logics, Fundam. Inform. 133 (2-3) (2014) 287–303.

[46] T. Tran, L. A. Nguyen, T. Hoang, Bisimulation-based concept learning for information systems in description logics, Vietnam J. Computer Science 2 (3) (2015) 149–167.

[47] L. A. Nguyen, A. Szalas, Logic-based roughification, in: A. Skowron, Z. Suraj (Eds.), Rough Sets and Intelligent Systems (1), Vol. 42 of Intelligent Systems Reference Library, Springer, 2013, pp. 517–543.
URL `http://dblp.uni-trier.de/db/series/isrl/isrl42.html#NguyenS13a`

[48] T. Niblett, I. Bratko, Learning decision rules in noisy domains, in: Proc. of Expert Systems '86, Cambridge University Press, 1987, pp. 25–34.

[49] N. Fanizzi, C. d'Amato, F. Esposito, Metric-based stochastic conceptual clustering for ontologies, Information Systems 34 (8) (2009) 792–806.

[50] C. d'Amato, N. Fanizzi, F. Esposito, Inductive learning for the Semantic Web: What does it buy?, Semant. Web 1 (1,2) (2010) 53–59.

[51] K. W. Bowyer, N. V. Chawla, L. O. Hall, W. P. Kegelmeyer, SMOTE: Synthetic minority over-sampling technique, CoRR.

[52] X.-C. Yin, C. Yang, H.-W. Hao, Learning to diversify via weighted kernels for classifier ensemble, CoRR abs/1406.1167.

[53] B. Fu, Z. Wang, R. Pan, G. Xu, P. Dolog, An integrated pruning criterion for ensemble learning based on classification accuracy and diversity, in: L. Uden, et al. (Eds.), Proceedings of KMO 2013, Vol. 172 of AISC, Springer, 2013, pp. 47–58.