Semantic Web 0 (2019) 1–0
IOS Press

An Unsupervised Approach to Disjointness Learning based on Terminological Cluster Trees

Giuseppe Rizzo *, Claudia d'Amato **, Nicola Fanizzi ***
LACAM – Dipartimento di Informatica
Università degli Studi di Bari "Aldo Moro"
Via Orabona 4, 70125 Bari, Italy

Abstract. In the context Semantic Web context regarded as a *Web of Data*, research efforts have been devoted to improving the quality of the ontologies that are used as vocabularies to enable complex services based on automated reasoning. From various surveys it emerges that many domains would require better ontologies that include nonnegligible constraints. In this respect, disjointness axioms are representative of this general problem: these axioms are essential for making the negative knowledge about the domain of interest explicit yet they are often overlooked during the modeling process (thus affecting the efficacy of the reasoning services). To tackle this problem, automated methods for discovering these axioms can be used as a tool for supporting knowledge engineers in the task of modeling new ontologies or evolving existing ones. The current solutions, either those based on statistical correlations or those relying on external corpora, often do not fully exploit the terminology of the knowledge base. Stemming from this consideration, we have been investigating on alternative methods to elicit disjointness axioms from existing ontologies based on the induction of *terminological cluster trees*, which are logic trees in which each node stands for a cluster of individuals which emerges as a sub-concept. The growth of such trees relies on a divide-and-conquer procedure that assigns, for the cluster representing the root node, one of the concept descriptions generated via a refinement operator and selected according to a heuristic based on the minimization of the risk of overlap between the candidate sub-clusters (quantified in terms of the distance between two prototypical individuals). Preliminary works have showed some shortcomings that are tackled in this paper. To tackle the task of disjointness axioms discovery we have extended the terminological cluster tree induction framework with various contributions which can be summarized as follows: 1) the adoption of different distance measures for clustering the individuals of a knowledge base; 2) the adoption of different heuristics for selecting the most promising concept descriptions; 3) a modified version of the refinement operator to prevent the introduction of inconsistency during the elicitation of the new axioms; 4) the integration of frameworks for the distributed and efficient in-memory processing, namely *Spark*, for scaling up the set of candidate concepts generated through the refinement operator. A wide empirical evaluation showed the feasibility of the proposed extensions and the improvement with respect to alternative approaches.

Keywords: disjointness learning, conceptual clustering trees, inconsistency prevention, distance measure

1. Introduction

In the perspective of the *Semantic Web* (SW) as a *Web of Data*, a plethora of datasets are constantly published and connected to others in the form of *Linked Data*, along a standard data model and based on schemata formalized as Web ontologies [1].

In this scenario, many important services have been devised and deployed for exploiting and enriching these knowledge bases in a variety of tasks, such as *classification*, *query answering*, *population* and *enrichment*, *reconciliation* (*instance matching*), and *consistency checking*.

*Corresponding author. E-mail: giuseppe.rizzo1@uniba.it.
**Corresponding author. E-mail: claudia.damato@uniba.it.
*** Corresponding author. E-mail: nicola.fanizzi@uniba.it.

The effectiveness of the mentioned complex inference services that can be built upon them is strictly dependent on the quality of the ontologies, namely on how precisely (and exhaustively) their axioms convey the intended semantics of the underlying domains. As the ontologies are represented through standard languages ultimately based on *Description Logics* (DLs) [2], an open-world semantics is generally adopted as suitable for such a Web-scale scenario, opposed to the *complete knowledge assumption* backing the semantics of other contexts (e.g. relational databases and logic programs). Checking *disjointness* is one of the key reasoning tasks for DL knowledge bases together with *satisfiability*, *subsumption*, *equivalence* tests (they can be reduced to one another). Reasoning under open-world semantics, with tasks involving individuals (e.g. *instance checking* and *retrieval queries*) assertions cannot be proven to hold because of the inherent incompleteness of the knowledge bases (for technical details see [2], Ch. 2). This feature affects also the other services mentioned above that are built upon them. For example, in tasks aimed at enriching knowledge bases, detecting the possible introduction of conflicting assertions (e.g. cases of inconsistency) is very important, as this might trigger further *repair*-actions to reconcile the possible causes. Hence *disjointness axioms* are essential to detect such cases.

To clarify this point, let us consider the case of a simple corporate knowledge base fragment (whose hierarchy is depicted in Fig. 1) with two sibling subconcepts, namely `Person` and `Robot`. Suppose also that the fragment includes an assertion `Robot(BotSmith)`, stating that the individual `BotSmith` is a robot (working in one factory of the company, etc.), but later the noisy assertion `Person(BotSmith)` may be added by mistake (e.g. because lexically similar to `Person(BobSmith)`) or it may be inferred from the assumption of other (inaccurate) facts, e.g. an assertion `welded(BotSmith,Piece123)` and a flawed fact like `domain(welded,Worker)` given that `subClassOf(Worker,Person)`. Without an explicit axiom stating that *a robot is not a person*, this inconsistency with respect to the intended interpretation of the knowledge base cannot be detected. Note that such an axiom would extend its effect to the whole subconcepts hierarchies.

However, in the design of various popular ontologies currently in use the introduction of disjointness axioms as required by concept modeling methodologies (see [2], Ch. 10) has been neglected. As a result,

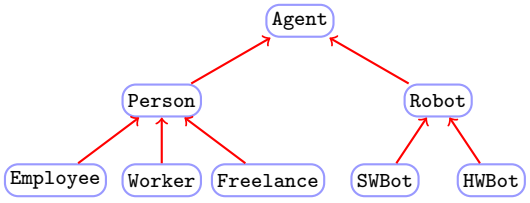

Fig. 1. A simple concept hierarchy modeling the agents in a corporate domain

they provide only a rather approximate representation of the domains, failing to capture all of the underlying constraints or making a complete knowledge assumption that distorts the intended semantics by admitting unintuitive cases. This lack of modeling accuracy was testified by a survey [3] in which only 97 out of a total of 1275 considered ontologies was found to include disjointness axioms. A possible reason for such an issue may be the inexperience with the language constructs, often leading users to overlook disjointness during the definition of domain axiomatizations [4]. As a result, while a growing number of datasets have been published joining the Linked Data cloud over the years, this issue is still ignored: at the time of writing this article, only 7 out of 9960 knowledge bases (0.7%) include this kind of axioms[1].

Another cause for this issue may lie in the context-dependent nature of the notion of disjointness which is consequently perceived in different ways [5]. For instance, considering the previous case, it may be assumed that the same individual cannot be both a `Worker` and a `Freelance` within the same context (but, of course, it may be possible to ascribe him/her to either class in separate contexts). This means that a clear understanding of the domain to be modeled is crucial for its careful axiomatization in an ontology. However, the manual introduction of disjointness axioms may become a discouraging and, easily, also error-prone activity with large ontologies. Nevertheless, the task can be (partially) accomplished *ex post* with the support of statistical models emerging from the data as the result of *machine learning* techniques.

Noticeably, various works have shown how to exploit association rule mining for statistical *schema induction* [6,5]. The proposed methods often depend on the availability of heterogeneous external resources (corpora) for their elicitation for relying on lexical features. Conversely, the interplay between extensional and intensional knowledge (i.e. assertions regarding

[1] This can be checked at LODSTATS: `http://stats.lod2.eu`

the individuals and terminological axioms) was only marginally taken into account. Most of the current approaches move from the assumption that two (or more) concepts may be mutually disjoint when the sets of their known instances, which should be representative for their *extensions* [2], do not *overlap* [7,4].

Moving from these considerations, a data-driven approach could be devised with the goal of finding partitions of similar individuals of the knowledge base according to a criterion that maximize the homogeneity of the individuals in each partition, i.e. emerging concepts, while minimizing their mutual overlap. Evidently, this boils down to a *clustering problem* [8] which is a classic topic in machine learning. It has also been taken into account in the context of ontology learning as a preliminary step for *concept induction* [9] or for automated solutions to *concept drift* or *novelty detection* problems [10]. Such problems have been tackled through extensions of basic clustering methods, such as or (*fuzzy*) K-MEDOIDS [11,12], adapted to work on expressive knowledge bases.

An emerging approach to disjointness axioms discovery, has proposed the employment of *terminological cluster trees* (TCTs) [13,14]. Solving *conceptual clustering* problems [15], finding natural partitions of the individuals to induce intensional definitions of the corresponding classes expressed in the standard representation languages, this framework aims at deriving potential disjointness axioms, that may even involve complex concept descriptions, by leveraging the background knowledge of the underlying schema (the axioms in the knowledge base). Unlike other methods, conceptual clustering produces partitions defined intensionally, with concept descriptions to decide the membership, rather than, extensionally, as a simple list of their individuals.

A TCT (see an example in Fig. 2) is quite similar to a *terminological decision tree* [16,17]: both are grown through *divide-and-conquer* strategy and can be thought to form concept hierarchies exploiting refinement operators for DLs [18,9]. The latter are essentially binary decision trees, induced by *supervised* methods exploiting information-based heuristics; they are meant for the classification of the individuals through the logical tests in their inner nodes and classes associated with the leaves. The former (with a similar structure except for the leaves) are induced through *unsupervised* learning methods aimed at eliciting *types* from the partitions of similar individuals. The concepts to be installed at inner nodes are defined by progressively refining the one at the parent node (or its complement for right-children): possible specializations of such a concept are computed by a refinement operator, and the best one is selected based on the quality of the (bi)partitions of individuals that would be routed to either children. This quality is measured in terms of their membership w.r.t. the candidate refinement. Suitable specific metrics for the underlying representation are required to derive a notion of *cluster prototype* [10]. Noticeably, the method is able to detect dense data regions in the underlying instance space, hence the number of clusters – which has a strong impact on the quality of the clustering structure – need not to be required as a parameter. As in many partitional clustering method [8] the *cohesion* of the clusters (e.g. measured as the similarity of the members w.r.t. the prototype) determines the stop condition: further partitioning of coherent clusters should be prevented to preserve the quality of the structure.

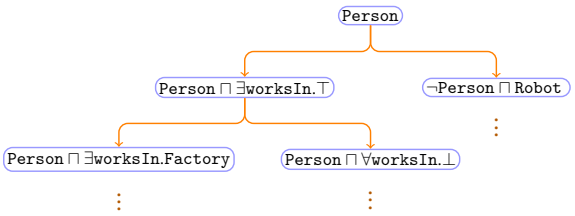

Fig. 2. A fragment of a TCT.

Once the TCT is grown, the concepts in the tree are extracted to form candidate disjointness axioms. These axioms are intended to be validated by a domain expert/ontology engineer and may be involved in a subsequent debugging process for eliciting future cases of inconsistency (i.e. newly available individuals found to belong to disjoint concepts as shown in the previous example).

Despite the potential benefits deriving from the induction of TCTs, there are some issues, that were not investigated in preliminary works [14], and whose solution can further improve the framework.

Firstly, we tackle the issue of possible inconsistency cases that may be introduced by some axioms (out of a potentially large number of candidates) elicited by the data-driven method. Due to the context-dependent nature of disjointness, it may be hard to determine if a case of inconsistency indicates a truly erroneous axiom, or a special case for the domain represented by the ontology. As an example, let us consider the concepts `USPresident` and `Actor`, and two individuals `RONALD_REAGAN` and `DONALD_TRUMP`. Let us suppose that a candidate axiom proposes the disjointness of

these concepts. The actual inconsistency of the two cases depends on the *intended* meaning of `Actor`: if it is meant to denote anyone that participated in a movie, then the case of `DONALD_TRUMP` (who has some appearance as himself credited for some movies) may be considered as conflicting, whereas `RONALD_REAGAN` may be assumed as an exception to a general rule. To prevent the late evaluation of such cases after the induction of entire TCTs, it is important to improve the process of concept generation for the tree nodes. This has been redesigned, anticipating the verification of overlapping concepts.

Secondly, the best candidate partition of a given cluster-node (and the corresponding concept descriptions) was selected adopting the cluster *medoid* as the prototype to measure the cohesion (separation) of the resulting child-clusters [14]. This setting did not consider the fact that there may be outliers or *noisy* individuals in one sub-cluster, that may be really *close* to the sibling cluster (e.g. children nodes with common parents in the TCT of Fig. 2). As a consequence, a perfect homogeneity of each sub-cluster cannot be ensured.

Lastly, we also reconsidered the refinement operators used to generate the concept descriptions for the clusters. They are exploited to traverse the virtual space of refinements (specializations w.r.t. concept subsumption) of a given concept. As this space is huge (and redundant) most of the learning methods based on such operators have to trade *completeness* [18,9] (the ability of computing all the possible refinements) for efficiency, resorting to a stochastic search. In the new version of the method, the width of the *beam* of candidate refinements can be properly tuned so to enable spanning larger regions of the search space.

Summarizing, we further extend our framework for disjointness axiom discovery based on TCTs:

- a new version of the refinement operator is able to prevent cases of inconsistency introduced by concepts installed in the tree nodes;
- the adoption of different versions of the distance measures between the individuals;
- a different heuristic aiming at maximizing the distances between the closest elements of two clusters instead of their medoid;
- *Big Data* technologies have been adopted, as implemented in the *Spark*[2] framework, for parallelizing and distributing the refinement generation workload on more threads/clusters.

Given such extensions, a new and more comprehensive empirical evaluation was designed and carried out aiming at assessing the effectiveness of the method based on the TCTs also in comparison with other related methods.

The paper is organized as follows: in Sect. 2, the disjointness axiom discovery problem is formalized in terms of a clustering problem of individuals of an ontological knowledge base. In Sect. 3, the details of the enhanced methods for inducing TDTs used for solving the targeted problem are illustrated. The comparative experimental evaluation of the new implementation on a testbed of common ontologies is presented in Sect. 4. In Sect. 5 related works are surveyed and discussed. Finally, Sect. 6 concludes this work delineating further research directions.

2. Disjointness Discovery as a Conceptual Clustering Problem

In this section, we formalize the problem of *discovering concept disjointness axioms* from ontological knowledge bases in terms of a *clustering* task.

We will borrow notation and terminology from *Description Logics*, being the theoretical foundation of the standard representation languages for the SW. Hence, we will use the terms *concept* (*description*) and *role* as synonyms of *class* and *property* respectively. DL constructors will be used for defining concept descriptions. Logic *entailment*, *subsumption* and *equivalence* for complex axioms will be denoted with the usual symbols $\models$, $\sqsubseteq$, and $\equiv$, respectively. A *knowledge base* (KB) $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is made up of the *TBox* $\mathcal{T}$, a set of terminological axioms regarding concepts and roles) and the *ABox* $\mathcal{A}$, a set of facts, i.e. concept/role *assertions*, regarding the individuals. $\mathsf{Ind}(\mathcal{A})$ denotes the set of individuals (resource names) occurring in $\mathcal{A}$.

Before formalizing the problem of discovering concept disjointness axioms, for the sake of completeness, we recall some basics of the *clustering* methods.

Clustering is an unsupervised learning task aiming at grouping a collection of objects into subsets, named *clusters*, such that those within each cluster are more closely related/similar to one another than the objects assigned to different clusters [8]. In cluster analysis, the quality of the clusters is assessed using indices that

[2] https://spark.apache.org/

take into account measures of *cohesion*, i.e. total reciprocal similarity, among the objects within a cluster, and of *separation*, i.e. total reciprocal dissimilarity, among different clusters. In a general setting, an object is usually described in terms of *features* from a selected set $\mathcal{F}$; a measure of *similarity* between objects is expressed in terms of a *metric*; for example, in the case of datasets of objects described by tuples of numeric feature values, the *Euclidean* distance, *Cosine* similarity or more complex metrics for vector spaces are typically adopted.

A more complex clustering goal is pursued moving from flat to *hierarchical* structures. Another choice among the various clustering models is related to the form of membership of the objects with respect to the clusters. In the simplest (*crisp*) case, e.g. K-MEANS, cluster membership is *exclusive*: each object is assigned to one cluster. Extensions, such as FUZZY C-MEANS or EM [8], admit overlapping clusters with objects exhibiting a graded membership (*responsibility*) w.r.t. the clusters.

A further interesting class of methods is represented by *conceptual clustering* approaches [15] which generate also an intensional description (defining the membership property) for each cluster (e.g. a conjunction of propositional atoms). Beyond vectorial or, equivalently, propositional representations, more expressive richer logic languages may be adopted, such as the mentioned DLs. Differently from other methods, conceptual clustering algorithms for such representations may exploit available (schema-level) background knowledge for building descriptions for each cluster, i.e. axioms defining new concepts. Expressive representations require the support of suitable metrics for upgrading clustering methods.

A necessary condition for the disjointness of two (or more) concepts to hold is that their extensions do not overlap. Then the task of discovering disjointness axioms may be regarded as an unsupervised conceptual clustering problem aimed at finding separate partitions of individuals in the KB (such that each cluster consists of similar individuals, according to a given criterion) and producing intensional descriptions for them.

The problem is defined as follows:

Definition 1 (disjointness axiom discovery problem)

Given

- *a knowledge base* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
- *a set of individuals* $\mathbf{I} \subseteq \mathsf{Ind}(\mathcal{A})$

Find

- *a partition* Π *of* $\mathbf{I}$ *in a set of pairwise disjoint clusters* $\Pi = \{\mathbf{C}_1, \ldots, \mathbf{C}_{|\Pi|}\}$
- *for each* $i = 1, \ldots, |\Pi|$, *a concept description* D_i *that describes* $\mathbf{C}_i$, *so that:*
  - $\forall a \in \mathbf{C}_i \colon\ \mathcal{K} \models D_i(a)$ *and*
  - $\forall b \in \mathbf{C}_j, i \neq j \colon\ \mathcal{K} \models \neg D_i(b)$.

  Hence $\forall D_i, D_j, i \neq j \colon\ \mathcal{K} \models D_j \sqsubseteq \neg D_i$.

It should be noted that, differently from other settings, the number of clusters (say $k = |\Pi|$) is not a required parameter.

Example 1 *In the context of the corporate domain introduced in Sect. 1, let us consider the agents in a company KB (humans and machines). A concept description should be assigned to each cluster of a partition produced by a suitable method. As a result, a disjointness axiom may be discovered involving e.g. the clusters corresponding to the concepts* `Worker` *and* `Robot`, *namely* $\texttt{Worker} \sqsubseteq \neg \texttt{Robot}$, *provided that the sets of their instances do not overlap.*

Note that the formalization reported in Def. 1 is language-independent and it is aimed at a simple flat partitioning structure, yet it can be generalized to target hierarchical structures. In the next section, a solution to such a more complex form of clustering is presented.

3. Induction of Terminological Cluster Trees for Disjointness Axiom Discovery

The proposed approach is grounded on a two-step process. In the first step, given a knowledge base, clusters and the related concepts that describe them are discovered and organized in a tree structure. In the second step, the induced structure is exploited for learning a set of candidate disjointness axioms.

The model is formally defined as follows:

Definition 2 (terminological cluster tree) *Given a knowledge base* $\mathcal{K}$, *a* terminological cluster tree *(TCT) is a binary logical tree [19] where:*

- *each node, which stands for a cluster* $\mathbf{C}$ *of individuals, contains a concept description* D *(defined over the signature of* $\mathcal{K}$*)*

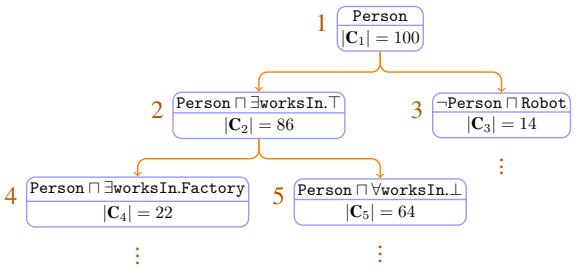

Fig. 3. A fragment of TCT whose nodes are also decorated with the size of the respective cluster of individuals $\mathbf{C}_i$. Intuitively, the concept descriptions D_i in the various nodes could be roughly mapped ($CN_i \equiv D_i$) to the following names, respectively: CN_2 : `Employee`, CN_3 : `Robot`, CN_4 : `Worker`, CN_5 : `Freelance`.

– *each edge departing from an internal node corresponds to a partition of* $\mathbf{C}$ *in (two) sub-clusters*[3].

A tree-node is represented by a quadruple $\langle D, \mathbf{C}, T_{left}, T_{right} \rangle$ *with the left and right subtrees connected by either departing edge.*

Fig. 3 illustrates an example of TCT resulting from the fragment of the corporate KB introduced in previous examples. In this tree, each concept installed into inner nodes is used to split a cluster of individuals $\mathbf{C}_i$ in two sub-clusters. For instance, the cluster $\mathbf{C}_2$ is split into 2 sub-clusters according to the membership of individuals in $\mathbf{C}_2$ w.r.t. the concept description `Person` $\sqcap \exists$ `worksIn`.$\top$ (roughly corresponding to `Employee` $\sqcup$ `Worker` or $\neg$`Freelance`) obtained as described in the sequel.

Note that, similarly to the typical clustering methods [8], an optimal TCT is one that induces a hierarchical partition of the individuals that maximizes both the cohesion within each cluster and the mutual separation between pairs of clusters. Obviously, finding such a tree requires searching a very large space of all possible TCTs based on the various cluster structures resulting from the set of individuals (and all possible concept descriptions used to split them). This turns out to be computationally unfeasible, so a heuristic approach has been advised.

The construction of a TCT combines elements of logical decision trees induction [20,16] (recursive partitioning and refinement operators for specializing concept descriptions) and of *instance-based learning* (a distance measure over the instance space). The details of the algorithms for (*a*) growing a TCT and (*b*) deriving intensional definitions of candidate disjoint concept descriptions are reported in the sequel.

[3] Noticeable difference with concept hierarchies: for each node in the TCT, its cluster, composed by instances of the concept in the parent node (ideally $\top$ for the root), is bi-partitioned according to the membership w.r.t. the concept in the current node.

Algorithm 1 Main routine for growing TCTs and stop condition test

```
1  function INDUCETCT(I, C, CS)
2  input I: set of individuals
3        C: concept description
4  output T: TCT
5  begin
6     T ← new TCT
7     if STOPCONDITION(I, C) then
8        T ← ⟨null, I, null, null⟩
9        CS ← CS ∪ {C} {update the set of concepts}
10    else
11       S ← SPECIALIZE(C, I, CS) {specializations}
12       E* ← SELECTBESTCONCEPT(S, I)
13       ⟨I_left, I_right⟩ ← SPLIT(I, E*)
14       T_left ← INDUCETCT(I_left, E*)
15       T_right ← INDUCETCT(I_right, ¬E*)
16       T ← ⟨E*, I, T_left, T_right⟩
17    return T
18 end
19
20 const ν, δ: stop thresholds {from the configuration}
21 function STOPCONDITION(I, C)
22 input I: set of individuals
23       C: concept description
24 output boolean
25 begin
26    if |I| ≤ δ then {test on the number of individuals}
27       return true
28    if C ≠ ⊤ then {tested to avoid trivial trees}
29       ⟨P, N⟩ ← SPLIT(I, C) {cluster partition w.r.t. C}
30       s ← d(p(P), p(N)) {prototype separation}
31       if s ≤ ν then {cohesion test}
32          return true
33       else
34          return false
35    else
36       return false
37 end
```

3.1. Growing Terminological Cluster Trees

A TCT is induced by a recursive strategy (see Algo. 1), which follows the schema proposed for growing terminological decision trees (TDTs) [16,17] solving the instance classification problem. The ultimate goal is to find a partition of *pure* clusters in terms of *cohesion*.

The main routine INDUCETCT is to be invoked passing $\mathbf{I}$ and $\top$ as parameters. In this recursive function, the base case tests the STOPCONDITION predicate checking whether either the cluster $\mathbf{I}$ is too small to be partitioned or its (measure of) cohesion exceeds a given threshold ν (further details about the heuristics and the stop condition are reported in Sect. 3.1.4). In this case the algorithm updates a set of concepts $\mathbf{CS}$ that is used by the refinement operator to prevent producing specializations that overlap with concept descriptions previously installed in other tree nodes.

In the inductive step, which occurs when the stop condition does not hold, the current (parent) concept description C has to be specialized using a refinement operator (ρ) that spans over a search space of concepts subsumed by C. A set of *candidate* specializations $\mathbf{S} \subseteq \rho(C)$ is obtained via SPECIALIZE$(C, \mathbf{I}, \mathbf{CS})$ such that, for each of them at least a positive and a negative instance can be found.

Then SELECTBESTCONCEPT evaluates each candidate specialization in $\mathbf{S}$ in terms of a measure of *separation* based on the *distance* (see Eqs. 1 and 2 discussed in the following) between the pairs of subclusters $\mathbf{P}$, made up of positive instances w.r.t. the current candidate concept, and $\mathbf{N}$, made up of negative instances w.r.t. the current candidate concept. The membership tests are based on *instance checking* [2]). Hence, SELECTBESTCONCEPT returns the best concept description $E^* \in \mathbf{S}$, that is the one maximizing the mentioned heuristic grounded on the notion of *separation*. Then E^* is installed in the current node and the individuals in $\mathbf{I}$ are partitioned by SPLIT to be routed along the left or the right branch departing from the current node, i.e. positive and negative instances w.r.t. E^*.

This divide-and-conquer strategy is applied recursively by the algorithm until no further branching is advisable, as the sets of instances routed to the leaf-nodes meet the stop condition. As mentioned before, differently from common clustering techniques, the number of the clusters is not required as an input, it depends on the number of branches grown: the algorithm is able to determine it according to the data distribution.

3.1.1. Downward Refinement Operators

The proposed approach relies on a downward refinement operator ρ [16,18] that must be able to generate *satisfiable* concepts – in terms of the models of the KB – performing a specialization process. It can be defined by cases in terms of ancillary sub-functions. Given the a concept description C (or its complement) to be specialized, the operator ρ computes specializations of C in one of the following forms:

- ρ_1 by adding a concept atom (or its complement) as a conjunct: $C' = C \sqcap (\neg)A$;
- ρ_2 by adding a general existential restriction (or its complement) as a conjunct: $C' = C \sqcap (\neg)\exists\, R.\top$;
- ρ_3 by adding a general universal restriction (or its complement) as a conjunct: $C' = C \sqcap (\neg)\forall\, R.\top$;
- ρ_4 by replacing a sub-description C_i in the scope of an existential restriction in C with one of its refinements: $\exists R.C'_i \in \rho(\exists R.C_i) \wedge C'_i \in \rho(C_i)$;
- ρ_5 by replacing a sub-description C_i in the scope of a universal restriction with one of its refinements: $\forall R.C'_i \in \rho(\forall R.C_i) \wedge C'_i \in \rho(C_i)$.

Note that the cases of ρ_4 and ρ_5 are recursive.

Example 2 *Given the TCT in Fig. 3, starting from* `Person`*, the following refinements can be obtained:*

- $\texttt{Person} \sqcap \exists \texttt{worksIn}.\top$*, installed in the node 2, is generated using* ρ_2*;*
- $\texttt{Person} \sqcap \exists \texttt{worksIn.Factory}$ *in node 4 is generated using* ρ_4*.*

The refinement operator ρ performs a sort of *random sampling* over a DL concept space. It is to be remarked that it does not satisfy all of the properties required for *ideality*, i.e. *finiteness* (for any concept the set of specializations is finite), *completeness* (for all concepts C and D, such that $D \sqsubset C$, a concept E, such that $E \equiv D$, can be computed chaining a number of applications of ρ) and *properness* (for all concepts C and D, if $D \in \rho(C)$, $D \sqsubset C$) [18]. Specifically, concerning the *finiteness* property, the refinement operator does not satisfy it because no bounds are imposed to the number of specializations to be generated through the random process. However this problem can be easily solved controlling algorithmically the number of specializations by imposing a finite beam dimension n and/or to the depth of the recursive calls.

Also the *completeness* of the refinement operator is not guaranteed because the constructors employed are evidently limited to those available for the $\mathcal{ALC}$ DL. Besides, the random process needed for generating each specialization may consider some concept descriptions more times while others (potentially useful for the learning problem) may be overlooked.

Lastly, even the *properness* is not ensured, due to the equivalence axioms defined in the TBox. For instance,

given a concept description $(C_1 \sqcap C_2)$ to be refined, the operator (via ρ_1) may add a new concept name B such that $C_1 \equiv B$. Also this property can be enforced by further checks on refinements to be output.

Instead of aiming at the definition of a theoretical operator endowed with these properties, which would be intended for a use with a generic *generate-and-test* algorithm, we decided to devise a more complex yet effective data-driven specialization procedure, capable of leveraging on the situation of the tree under construction. Algo. 2 illustrates the resulting procedure SPECIALIZE, which embeds the refinement operator ρ. Besides the concept description C to be refined, it requires a set of individuals $\mathbf{I}$ and a set of concept description $\mathbf{CS}$ that are employed to drive the traversal of the search space. Note that its behavior is also controlled by the beam dimension n.

The specializations are generated through the following steps:

- generate the specialization $D = C \sqcap E$ adding a conjunct E (selected by ADDCONJUNCT);
- apply subroutine SIMPLIFY to reduce redundancy and *syntactic length*[4] of this D [21];

These steps are repeated to ensure that

- the resulting specialization D is satisfiable w.r.t. $\mathcal{K}$ and if (both negative and positive) instances of D are available in $\mathbf{I}$;
- it does not overlap with the concepts $D' \in \mathbf{CS}$ of the control set (where the concept extensions are approximated using the *retrieval* $r_{\mathcal{K}}$ inference service [2]).

The specializations D are produced by adding a new (complex) description via the auxiliary function ADDCONJUCT. The recursive procedure tests the value of a random variable $X \sim \mathcal{U}(0,1)$ to decide which refinement case (ρ_1 - ρ_5) must be applied. In the base case, a random concept name A is picked from those in the signature of $\mathcal{K}$ to output a concept in one of the possible forms: $C \sqcap A$ (see ρ_1) or $C \sqcap \exists(\forall)R.(D \sqcap A)$ (see ρ_4 - ρ_5). The recursive calls produce sub-descriptions

[4] The *length* of a concept description C, $\text{len}(C)$ is defined inductively:

* $\text{len}(A) = \text{len}(\top) = \text{len}(\bot) = 1$
* $\text{len}(\neg D) = \text{len}(D) + 1$
* $\text{len}(D \sqcap E) = \text{len}(D \sqcup E) = \text{len}(D) + \text{len}(E) + 1$
* $\text{len}(\exists R.D) = \text{len}(\forall R.D) + 1$

Algorithm 2 The specialization routines employed for inducing TCTs

```
1  const n: number of candidates {from the configuration}
2  function SPECIALIZE(C, I, CS)
3  input C: concept description {to be specialized}
4        I: set of individuals
5        CS: set of concept descriptions
6  output S: set of concept descriptions
7  begin
8     for i ← 1 to n
9        E ← ⊤
10       repeat
11          E ← ADDCONJUNCT()
12          D ← SIMPLIFY(C ⊓ E) {reduce complexity}
13    until r_K(D) ∩ I ≠ ∅ and r_K(¬D) ∩ I ≠ ∅ and
        ¬OVERLAP(C, CS)
14       S ← S ∪ {D}
15    return S
16 end
17
18 function ADDCONJUNCT()
19 output concept description
20 begin
21    ⟨CN, RN⟩ ← Signature(K) {atomic concepts, roles}
22    X ∼ U(0, 1)
23    if X ≥ 3/4 then
24       A ← RANDOMPICK(CN) {random concept}
25       return A
26    else
27       C ← ADDCONJUNCT()
28       if X ≥ 1/2 then
29          return ¬C
30       else
31          R ← RANDOMPICK(RN) {random role}
32          if X ≤ 1/4 then
33             return ∃R.C
34          else
35             return ∀R.C
36 end
```

for the complement or the existential and universal restriction (w.r.t a randomly picked role) operators.

After ADDCONJUCT has produced a sub-description E, SIMPLIFY is applied to possibly generate a shorter concept equivalent to $C \sqcap E$. This is aimed at improving the overall interpretability of the resulting TCTs. Essentially, this function checks if $E \sqsubseteq C$ is entailed by the knowledge base, returning the concept description E as an output when this condition holds.

Example 3 *Let us suppose that* ADDCONJUNCT *refines the concept* $\exists$`worksIn.`$\top$ *by adding the conjunct* $\exists$`worksIn.Factory`*. In this case,* $\exists$`worksIn.Factory`

would be returned via SIMPLIFY *rather than the redundant description* $\exists worksIn.\top \sqcap \exists worksIn.Factory$.

As previously mentioned, the specializations returned by ρ are required to be satisfiable. This does not ensure that instances of such concepts are actually represented in the training sets. It is important to avoid the generation of satisfiable concept descriptions for which the training individuals exhibit a neutral membership: installing one of these concepts in a node would end up with all the individuals in the current node to be sorted to a single branch, thus undermining the divisive value of the node tests and increasing the complexity of the trees (in terms of number of nodes) with no evident advantage. To avoid these refinements, the algorithm verifies if a non-null number of positive and negative instances can be found for D, i.e. both the intersection between $\mathbf{I}$ and the instances of D ($r_{\mathcal{K}}(D)$) and the intersection between $\mathbf{I}$ and the complement ($r_{\mathcal{K}}(\neg D)$) are empty. Note that this constraint may be too strict to be satisfied, due to the sparseness of the assertional knowledge (in the ABox) for some of the involved concepts. These situations, causing delays (or even infinite loops) can be easily prevented by adopting some timeout condition to stop the generation of a new refinement (producing a leaf-node, instead).

A further aspect to be considered is the possible overlap between the concept description D and those installed in other clusters (they are supposed to be contained in the control set **CS**, passed as an argument to the procedure). It has been observed that, owing to this overlap, the clustering procedure may install concept descriptions that would introduce inconsistency once the axioms derived from the TCT were added to the KB [14]. Therefore, the refinement operator has been extended to avoid the generation of such concepts: the procedure will return a specialization only if it has no common instances with those contained in **CS** (check operated via OVERLAP).

Example 4 *Let us Suppose that the function* OVERLAP *checks if the control set contains the concept* $Person \sqcap \forall worksIn.\bot$ *(used to describe the individuals in* $\mathbf{C}_5$*). The procedure implementing the refinement operator should avoid the generation of the concept* $Person \sqcap Freelance$.

3.1.2. Distributed Implementation of the Specialization Procedure

Efficiency and scalability are certainly among the most challenging properties required for inductive methods. In the case of TCT induction, the mentioned incompleteness of the refinement operator adopted and limitations on the size of beam of candidates imply that the search algorithm may miss some important features (concepts) that would describe the clusters optimally. Conversely, tuning the algorithm with a large beam sizes makes the approach inefficient. To tackle these issues, we illustrate an approach to reworking the refinement operator benefiting from frameworks for supporting distributed architectures, i.e. *Spark*.

Spark is a distributed processing framework intended for large amounts of heterogeneous data. It provides an API for devising applications that are able to process such kind of data by means a transparent approach with respect to specific file systems and architectures. *Spark* relies on the notion of *resilient distributed datasets* (RDDs), distributed memory abstractions that let programmers perform in-memory computations on large computer clusters. Essentially, an RDD is a read-only, partitioned collection of records. RDDs are created through operations called *transformations* that take an RDD and return a new RDD as their output, and *actions* that return a single value of a given type rather than an RDD. Some examples of such operations are the well known *map & reduce* functions for parallel processing: MAP provides a one-to-one processing of each element contained in the input RDD) and REDUCE processes a RDD containing values of a given type to get a single new value (of the same type).

Algo. 3 reports a new version of the specialization procedure described in Algo. 2 reworked for allowing an implementation on a distributed processing framework. Similarly to the original version, the procedure SPECIALIZE takes as its arguments the concept description C to be refined, the set of individuals $\mathbf{I}$, and the control set **CS** of concept descriptions. The procedure creates an RDD **S**, which is initialized by the transformation MAP. The high-order function requires a function INITIALIZE[5] as an argument. This function returns a new RDD containing the concept description C that will be refined. After the initialization, the RDD **S** is physically distributed among the available processing units (of a cluster or, if running on a single machine, of the various cores decomposed in multiple threads) through the auxiliary procedure PARALLELIZE[6]. Once the initialization is parallelized, the refinement process starts invoking again the function

[5] This is implemented in Java 8 as a *lambda expression*

[6] It can be transparently managed by the underlying framework infrastructure.

Algorithm 3 Distributed specialization procedure for *Spark*

```
const n: number of candidates {from the configuration}
function SPECIALIZE(C, I, CS)
input C: concept description
      I: set of individuals
      CS: set of concept descriptions
output S: set of concept descriptions
begin
  S ← RDD[n] {an RDD of n elements}
  S ← MAP(S, INITIALIZE(S, C)) {RDD init.}
  PARALLELIZE(S)
  S ← MAP(S, GENERATEAREFINEMENT(S, I))
  return S
end

function GENERATEAREFINEMENT(S,I)
input S: RDD
      I: set of individuals
output S′: RDD
begin
  S′ ← RDD[n]
  for each C ∈ S do {RDD transformation}
    repeat
      D ← C ⊓ ADDCONJUNCT()
      D ← SIMPLIFY(C ⊓ E) {reduce complexity}
    until r_K(D) ∩ I ≠ ∅ and r_K(¬D) ∩ I ≠ ∅ and
    ¬OVERLAP(C, CS)
    S′ ← S′ ∪ D
  return S′
end
```

MAP. In this case, the algorithm passes the function GENERATEAREFINEMENT, which generates for each concept in **S** the (properly simplified) specializations (similarly to Algo. 2).

3.1.3. Heuristics

The algorithms for growing TCTs and TDTs share a common structure but differ on the criterion for selecting the test concepts to be installed in the nodes: while the latter adopts a scoring function based on the classic notion of *information gain*, the separation measure to be maximized in the procedure for the TCTs relies on a distance defined over the individuals occurring in the knowledge base. Specifically, the heuristic for selecting the best refinement of the parent concept is defined as follows:

$$E^* = \operatorname*{argmax}_{E \in \rho(C)} d(p(\mathbf{P}_E), p(\mathbf{N}_E)) \tag{1}$$

where $\mathbf{P}_E$ and $\mathbf{N}_E$ are the sub-clusters output by SPLIT, $d(\cdot,\cdot)$ is a distance measure between individuals in a KB and $p(\cdot)$ is a function that maps a cluster of individuals to its *prototype*, such as the *medoid* of the cluster.

However, it was observed that maximizing the distance between the medoids may not guarantee to avoid the overlap between the sub-clusters $\mathbf{P}_E$ and $\mathbf{N}_E$ [14]. Indeed, we observed cases of boundary individuals in one cluster which were on average closer to those in the other cluster (including their medoid) than to the others belonging to the same cluster. To tackle such cases a more sophisticated heuristic can be adopted:

$$E^* = \operatorname*{argmax}_{E \in \rho(C)} \min_{b \in \mathbf{P}_E, c \in \mathbf{N}_E} d(b,c) \tag{2}$$

The score for each candidate E is determined quantifying the risk of overlap between two clusters according to the distance between the closest individuals belonging to $\mathbf{P}_E$ and $\mathbf{N}_E$.

The heuristic resorts to a variation of a language-independent dissimilarity measure proposed in previous works [10,22]. Given the knowledge base $\mathcal{K}$, the idea is to compare the behavior of the individuals w.r.t. a set of concepts $\mathcal{C} = \{F_1, F_2, \ldots, F_m\}$ that is dubbed *context* or *committee* of features. For each $F_i \in \mathcal{C}$, a *projection function* $\pi_i : \mathsf{Ind}(\mathcal{A}) \to [0,1]$ is defined as a simple mapping:

$$\forall a \in \mathsf{Ind}(\mathcal{A}) \quad \pi_i(a) = \begin{cases} 1 & \text{if } \mathcal{K} \models F_i(a) \\ 0 & \text{if } \mathcal{K} \models \neg F_i(a) \\ 0.5 & \text{otherwise} \end{cases} \tag{3}$$

where the third value (0.5) represents a case of maximal uncertainty on the membership.

As an alternative value, the estimate of the likelihood of being an instance of F_i for a generic individual a could be considered. Especially with densely populated ontologies (as those forming the Web of Data) the probability value $\Pr[\mathcal{K} \models F(a)]$ may be estimated by $|r_{\mathcal{K}}(F)|/|\mathsf{Ind}(\mathcal{A})|$, where $r_{\mathcal{K}}()$ denotes the *retrieval* of a concept w.r.t. $\mathcal{K}$, i.e. the set of individuals of $\mathsf{Ind}(\mathcal{A})$ that (can be proven to) belong to F_i [2]. Hence, a family of distance measures $\{d_n^{\mathcal{C}}\}_{n \in \mathbb{N}}$ can be defined as follows:
$d_n^{\mathcal{C}} : \mathsf{Ind}(\mathcal{A}) \times \mathsf{Ind}(\mathcal{A}) \to [0,1]$ with

$$d_n^{\mathcal{C}}(a,b) = \left[\sum_{i=1}^{m} w_i \left[1 - \pi_i(a)\pi_i(b) \right]^n \right]^{1/n} \tag{4}$$

Non uniform values for the *vector of weights* $\vec{w}$ can be considered to reflect the specific importance of each

feature. For example it may be set according to an entropic measure [10,22] based on the average information brought by each concept:

$$\forall i \in \{1,\ldots,m\} \quad w_i = -\sum_{k\in\{-1,0,+1\}} \mu_i(k)\log\mu_i(k) \quad (5)$$

where, given a generic $a \in \mathsf{Ind}(\mathcal{A})$, the following estimates can be used: $\mu_i(+1) \approx \Pr[\mathcal{K} \models F_i(a)]$, $\mu_i(-1) \approx \Pr[\mathcal{K} \models \neg F_i(a)]$ and $\mu_i(0) = 1 - \mu_i(+1) - \mu_i(-1)$.

An alternative distance measure proposed in other works [23] is the following:

$$d_n^{\mathcal{C}}(a,b) = \left[\sum_{i=1}^{m} w_i \left[\pi_i(a) - \pi_i(b)\right]^n\right]^{1/n} \quad (6)$$

Note that the two distance measures reported above exhibit different behaviors. Specifically, the former reaches its maximum when, given two individuals a and b and a feature concept $F_i \in \mathcal{C}$, $\pi_(a) = 0$ and $\pi_(b) = 0$ or $\pi_(a) = 0.5$ and $\pi_(b) = 0.5$ (assuming $w_i = 1$), i.e in the cases of negative and uncertain membership. The latter reaches its maximum value, when the individuals a and b have opposite definite memberships for F_i (i.e. $\pi_(a) = 0$ and $\pi_(a) = 1$ and vice-versa).

3.1.4. Stop Conditions

The growth of a TCT can be stopped if one of the following conditions are satisfied (see Algo. 1):

- *the set of individuals is too small to be partitioned*: this is made testing if $|\mathbf{I}| \leq \delta$
- *The concept C to be specialized is different from* $\top$:in this case the algorithm finds the positive and negative instances of C and exploits a threshold $v \in [0,1]$ for the value of $d(\cdot,\cdot)$. If the value is below the threshold, the branch growth is stopped.

To avoid trivial clusters, the Boolean function STOPCONDITION is forced to return `false` when $C = \top$. Conversely, the growth of TCT would stop after the first call, i.e. when the $\top$ and $\mathbf{I}$ are passed as input and the specialization process would never occur.

3.2. Extraction of Disjointness Axioms from TCTs

The procedure for discovering/extracting disjointness axioms requires a TCT as its input. Its details are reported in Algo. 4.

Algorithm 4 Derivation of disjointness axioms from TCTs

```
1  const θ: threshold {from the configuration}
2
3  function DERIVECANDIDATEAXIOMS(T)
4  input T: TCT
5  output A: set of axioms collected
6  begin
7     A ← ∅
8     CCD ← COLLECT(⊤, T)
9     for each C ∈ CCD do
10       for each D ∈ CCD do
11          if (D ⊑ ¬C) ∉ A and |r_K(C ⊓ D)| ≤ θ then
12             A ← A ∪ {D ⊑ ¬C}
13    return A
14 end
15
16 function COLLECT(C, T)
17 input C: concept description
18       T: TCT
19 output CCD: set of collected concept descriptions
20 begin
21    let T = ⟨D, I, T_left, T_right⟩
22    if T_left = T_right = null then    {leaf node}
23       return {C}
24    else
25       CCD_left ← COLLECT(C ⊓ D, T_left)
26       CCD_right ← COLLECT(¬C ⊓ ¬D, T_right)
27       return (CCD_left ∪ CCD_right)
28 end
```

Function DERIVECANDIDATEAXIOMS can be employed to traverse the TCT passed as an argument to collect the concept descriptions that are installed in the parents of the leaf-nodes. In this phase, it generates a set of concept descriptions **CCD**. Then, the function considers all pairs of elements C and D in **CCD** and checks if the number of instances of the concepts $D \sqcap C$ does not exceed the threshold θ (a parameter to be set in the configuration).

The set of collected concept descriptions **CCD** is obtained by traversing the TCT. The COLLECT function is invoked for gathering concepts descriptions for which disjointness axioms may hold by exploring (recursively) the various paths along the (sub)tree from the root to the leaves.

Note that the hierarchical nature of the approach may allow for a further generalization of this function, controlling with a further parameter the maximum depth of the inner nodes to be visited during the traversal. This would likely produce fewer and more general axioms with respect to the specification of the function reported in Algo. 4.

Example 5 *Given the TCT in Fig. 3, the following set of concepts can be built using the routine* COLLECT*:*

$$\begin{aligned}\textbf{CDD} = \{\ & \mathtt{Person}, \neg\mathtt{Person} \sqcap \mathtt{Robot},\\ & \mathtt{Person} \sqcap \exists \mathtt{worksIn}.\top,\\ & \mathtt{Person} \sqcap \forall \mathtt{worksIn}.\bot,\\ & \mathtt{Person} \sqcap \exists \mathtt{worksIn.Factory}\}\end{aligned}$$

The following candidates axioms can be derived:

$$\begin{aligned}\mathbf{A} = \{\ & \mathtt{Person} \sqsubseteq \neg(\neg\mathtt{Person} \sqcap \mathtt{Robot}),\\ & \mathtt{Person} \sqcap \exists \mathtt{worksIn}.\top\\ & \qquad \sqsubseteq \neg(\neg\mathtt{Person} \sqcap \mathtt{Robot}),\\ & \mathtt{Person} \sqcap \exists \mathtt{worksIn.Factory}\\ & \qquad \sqsubseteq \neg(\neg\mathtt{Person} \sqcap \mathtt{Robot}),\\ & \mathtt{Person} \sqcap \exists \mathtt{worksIn.Factory}\\ & \qquad \sqsubseteq \neg(\mathtt{Person} \sqcap \forall \mathtt{worksIn}.\bot)\}\end{aligned}$$

The algorithm to elicit candidate axioms from TCTs provides an approximation of the missing axiom described in Ex 1. Additionally, it was also able to generate axioms involving other concepts, e.g. $\mathtt{Person} \sqcap \exists \mathtt{worksIn.Factory} \sqsubseteq \neg(\mathtt{Person} \sqcap \forall \mathtt{worksIn}.\bot)$*, corresponding to the axiom* $\mathtt{Worker} \sqsubseteq \neg Freelance$*.*

4. Experiments

In this section, the design and the outcomes of a comparative empirical evaluation of the proposed model and related methods are reported. The experiments were aimed at assessing the performance of the revised version[7] of the method based on the TCTs, in comparison with other state-of-the-art statistical methods for discovering disjointness axioms (to be further discussed in the next Sect. 5). We first illustrate the methodology with the experimental design and setup, then we report and discuss the outcomes of the various sessions.

4.1. Methodology

4.1.1. Ontologies

In the experiments, we considered a variety of freely available Web ontologies describing various domains, namely: BIOPAX, NEW TESTAMENT NAMES (NTN), FINANCIAL, GEOSKILLS, MONETARY, and DBPEDIA3.9,MUTAGENESIS and VICODI. The principal characteristics of the selected KBs are summarized in Tab. 1.

BIOPAX is a translation into BioPax format of the glycolysis pathway in the EcoCyc database. NTN describes the characters and places mentioned in the New Testament. FINANCIAL was created for modeling the domain of banking. GEOSKILLS comes from an effort aimed at encoding competencies, topics, and educational levels of the mathematics curriculum standards throughout Europe. MONETARY is an ontology that was intended for modeling information about currencies. DBPEDIA will denote a fragment extracted from the DBPEDIA 3.9 ontology, employing a crawling procedure that traversed the RDF graph for retrieving instances and the related schema information. The other ontologies are MUTAGENESIS, a porting of a well known benchmark for relational learning methods, and VICODI, a part of a larger ontology that formalizes knowledge concerning historical events.

4.1.2. Tasks and Design of the Experiments

The experiments had two main goals:

- assessing the ability of the proposed approach to (re-)discover target axioms originally defined as the result of a knowledge engineering process;
- assessing the number and quality of the new axioms discovered preserving the KB consistency in comparison with related methods.

In order to automate the test of the axioms produced we decided to bypass the intervention of domain experts which is hardly available and may compromise the repeatability of the experiments. To cope with the lack of target disjointness axioms in the considered KBs which would offer a natural *gold standard* (ground truth) for the tests, we considered modified versions of the ontologies reported in Tab. 1, which were produced through the artificial introduction of new disjointness axioms involving sibling concepts in the subsumption hierarchy, according to the SDA, yet preserving the consistency of the ontologies. For each ontology, a fraction f of disjointness axioms was randomly removed. To have performance indices unbiased by the specific selection of axioms, this procedure was repeated 10 times per ontology and also increasing f: 20%, 50%, 70%.

The effectiveness of the methods was evaluated in terms of

- *the average number of original axioms rediscovered* (the larger the better) that can be considered as a sort of *recall* measure

[7] Code and testbed of ontologies are publicly available at: https://github.com/Giuseppe-Rizzo/TCT-new

Table 1
Ontologies employed in the experiments

Ontology	*DL Language*	*#Concepts*	*#Roles*	*#Individuals*	*#Disj.Axioms*
BIOPAX	$\mathcal{ALCIF}(D)$	74	70	323	85
NTN	$\mathcal{SHIF}(D)$	47	27	676	40
FINANCIAL	$\mathcal{ALCIF}(D)$	60	16	1000	113
GEOSKILLS	$\mathcal{ALCHOIN}(D)$	596	23	2567	378
MONETARY	$\mathcal{ALCHIF}(D)$	323	247	2466	236
DBPEDIA	$\mathcal{ALCHI}(D)$	251	132	16606	11
MUTAGENESIS	$\mathcal{AL}(D)$	86	5	14145	0
VICODI	$\mathcal{ALHI}(D)$	196	10	16942	0

– *the average number of inconsistencies in the knowledge base* (the less the better) and *the average number of axioms elicited*, which is an indicator of the *precision* of the tested methods.

4.1.3. Set-up of the Implemented Algorithms

In the evaluation we tested various configurations of the overall discovery process based on the TCT learning algorithm that can be summarized as follows:

– *v.1*: first release of the TCT learning algorithm [14] (with and without the new refinement operator implemented on *Spark*) combined with the heuristics (1) and (2) and the distance measures (4) and (6) (both entropic – denoted by *e* – and uniform weights – denoted by *u* – have been considered);
– *v.2*: new version of TCT learning algorithm exploiting the refinement operator implemented on *Spark*, the heuristics (1) and (2) and the distance measures (4) and (6) (with and without entropic weights)
– *v.3*: same as *v.2* but with the consistency check described in Algo. 2 (and using the parallelized refinement operator).

The distance measure $d_2^{\mathcal{C}}$ was selected from the family, with a context of features $\mathcal{C}$ made up of the atomic concepts in the signature of each KB.

The beam width for controlling the number of specializations was set to 100. In addition, the distributed version of the refinement operator required as a parameter the number of slices of the RDDs containing the specializations which was set to 4. The timeout for generating a refinement, useful in cases when positive and negative instances were hard to find in the cluster, was set to 300ms. In all cases but the first release, for the axiom discovery procedure required the value of threshold θ that was set to 10.

As previously mentioned we tested the various configurations of the procedure based on TCTs against other approaches proposed in the related literature (see Sect. 5), in particular those based on *Pearson's correlation coefficient* (PCC) and *negative association rules* (NAR) [5]. As for the latter, rules were mined using APRIORI, with the required parameters values set as follows: minimum *support rate* 10, minimum *confidence rate* 50%, and maximum *rule length* 3 (also in consideration of the sparseness of the instance distributions w.r.t. the concepts in the specific ontologies).

4.2. Experimental Results: Presentation and Analysis

For the sake of readability Tabs. 2–7 report some outcomes of the empirical evaluation while the complete results are listed in Appendix A. Preliminarily, we have to note that as there was a variation in terms of efficiency (discussed in Sect. 4.2.4) and no sensible variation in terms of rediscovered axioms, number of new axioms and number of inconsistency cases in the experiments configured to use the original or the parallel version of the refinement operator (see Sect. 3.1.1). Thus, for the sake of brevity, the corresponding tables will not be replicated.

4.2.1. Rediscovering Disjointness Axioms

Throughout the experiments, we noted that the algorithm based on the TCTs was able to re-discover most of the disjointness axioms that had been previously removed to test this ability. The limited number of cases where the algorithm did not manage to re-discover the axioms depended on the choice for the threshold v: the lower its values the less recursive calls are required for completing the induction of a TCT. Anticipating the termination, along with the inherent incompleteness of the refinement operator, may be one of the reasons for not getting the exact concepts involved in the original axioms. Further aspects that may have affected the outcomes are the choice of the distance measure and the heuristic adopted to select the concepts to be installed in the nodes. As regards the former, we noted that, adopting function (6), the rate of rediscovered axioms was lower than the one obtained with function (4). The resulting trees showed a less complex structure (less nodes) using (6) instead of (4). Besides, higher val-

Table 2

Average rates (and standard deviations) of original axioms re-discovered. Configurations *v.1* and *v.2* – scoring function 1

Ontology	distance / weights	*f*	TCT – *standard mode*		
			TCT 0.9	TCT 0.8	TCT 0.7
BIOPAX	(4) / u	20%	0.82 ± 0.08	0.82 ± 0.08	0.82 ± 0.08
		50%	0.82 ± 0.09	0.82 ± 0.09	0.82 ± 0.09
		70%	0.81 ± 0.10	0.81 ± 0.10	0.81 ± 0.10
	(4) / e	20%	0.90 ± 0.12	0.76 ± 0.13	0.74 ± 0.13
		50%	0.85 ± 0.13	0.74 ± 0.13	0.74 ± 0.13
		70%	0.85 ± 0.13	0.74 ± 0.12	0.74 ± 0.14
	(6) / u	20%	0.63 ± 0.23	0.67 ± 0.23	0.69 ± 0.29
		50%	0.69 ± 0.23	0.69 ± 0.22	0.69 ± 0.22
		70%	0.69 ± 0.23	0.69 ± 0.22	0.69 ± 0.22
	(6) / e	20%	0.70 ± 0.12	0.73 ± 0.11	0.73 ± 0.12
		50%	0.70 ± 0.12	0.73 ± 0.11	0.73 ± 0.12
		70%	0.70 ± 0.12	0.73 ± 0.11	0.73 ± 0.12
MONETARY	(4) / u	20%	0.99 ± 0.08	1.00 ± 0.00	1.00 ± 0.00
		50%	0.94 ± 0.13	1.00 ± 0.00	1.00 ± 0.00
		70%	0.94 ± 0.13	0.91 ± 0.14	0.91 ± 0.13
	(4) / e	20%	0.99 ± 0.08	1.00 ± 0.00	1.00 ± 0.00
		50%	0.94 ± 0.13	1.00 ± 0.00	1.00 ± 0.00
		70%	0.94 ± 0.13	0.91 ± 0.14	0.91 ± 0.13
	(6) / u	20%	0.89 ± 0.14	0.76 ± 0.14	0.76 ± 0.13
		50%	0.92 ± 0.16	0.90 ± 0.16	0.92 ± 0.16
		70%	0.94 ± 0.13	0.94 ± 0.13	0.94 ± 0.12
	(6) / e	20%	0.97 ± 0.15	0.97 ± 0.15	0.97 ± 0.15
		50%	0.93 ± 0.11	0.93 ± 0.11	1.00 ± 0.00
		70%	0.94 ± 0.13	0.91 ± 0.14	0.91 ± 0.13
MUTAGEN.	(4) / u	20%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		50%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		70%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	(4) / e	20%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		50%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		70%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	(6) / u	20%	0.76 ± 0.14	0.77 ± 0.14	0.77 ± 0.13
		50%	0.82 ± 0.11	0.82 ± 0.11	0.81 ± 0.11
		70%	0.84 ± 0.09	0.84 ± 0.08	0.83 ± 0.10
	(6) / e	20%	0.95 ± 0.05	0.95 ± 0.05	0.95 ± 0.05
		50%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		70%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
VICODI	(4) / u	20%	0.95 ± 0.02	0.90 ± 0.08	0.90 ± 0.08
		50%	0.95 ± 0.02	0.90 ± 0.08	0.90 ± 0.08
		70%	0.95 ± 0.02	0.90 ± 0.08	0.90 ± 0.08
	(4) / e	20%	0.95 ± 0.02	0.90 ± 0.08	0.90 ± 0.08
		50%	0.95 ± 0.02	0.90 ± 0.08	0.90 ± 0.08
		70%	0.95 ± 0.02	0.90 ± 0.08	0.90 ± 0.08
	(6) / u	20%	0.92 ± 0.05	0.89 ± 0.08	0.89 ± 0.08
		50%	0.95 ± 0.04	0.93 ± 0.02	0.92 ± 0.04
		70%	0.95 ± 0.04	0.93 ± 0.02	0.92 ± 0.04
	(6) / e	20%	0.92 ± 0.05	0.89 ± 0.08	0.89 ± 0.08
		50%	0.92 ± 0.05	0.89 ± 0.08	0.89 ± 0.08
		70%	0.90 ± 0.05	0.87 ± 0.03	0.87 ± 0.03

ues were generally returned by the first measure. This makes homogeneous clusters of individuals (that determine the stopping condition for the tree growth) harder to find. In this perspective, also the choice of the vector of weights has had some influence: while the easier choice of uniform weights tended to flatten the distance measures, especially in the cases with large contexts of features, entropic weights resulted in a sort of preliminary feature selection that tended to discard many unrelated concepts of the context.

In this sense, the experiments with BIOPAX reported in Tab. 2 and 3 are particularly illustrative about the effectiveness of the weighting model for the resulting measure: the average rate of discovered axioms noticeably improved when distance (4) was employed, spanning from 0.63 up to 0.85.

Similar improvements were observed in the experiments with MUTAGENESIS and VICODI. In the experiments with the other ontologies, i.e. NTN, FINANCIAL, GEOSKILLS, MONETARY and DBPEDIA (where a rate greater than 0.9 was often observed regardless the specific configuration of TCT learning algorithm), improvements were observed, although to a lesser extent (see Appendix A).

In the evaluation, we also tested the effectiveness of the alternative heuristic (2). In this case very small or no changes were observed in the rate of rediscovered axioms although different tree structures were produced. Furthermore, in the experiments with TCT v.3, we observed an evident decrease of the performance (see the results with BIOPAX in Tab. 3). In such cases, the constraint on consistency made the algorithm based on TCTs more conservative than the other versions. Indeed, introducing this condition as a constraint for generating specializations led to discard lots of concept descriptions.

A further aspect to consider is the availability of individuals that are instances of the concepts involved in disjointness axioms. For eliciting the target axioms, the larger number of individuals, the more likely it is to find sub-clusters whose distance is maximized. Specifically, in the experiments we noted that it was hard to rediscover axioms involving concepts with less than 10-15 available instances: in such cases, the limited number of individuals in the clusters did not allow to maximize the distance of the candidate sub-clusters. Consequently, the scores computed via both heuristics (1) and (2) were quite low and the concepts were ignored. For instance, in the experiments on VICODI a trivial disjointness axiom between the concept Actor and Artefact could not be discovered. These few cases may be treated with specific configurations of the thresholds.

4.2.2. Axiom Discovery and Consistency

As regards the overall number of discovered axioms (see Tabs. 5–7) generally it can be observed that it decreased with larger fractions of axioms removed since the resulting trees showed a less complex structure. Moreover, we noted that, in the case of smallest ontologies (in terms of number of individuals),

Table 3

Average rates (and standard deviations) of removed axioms rediscovered using TCTs v.3 – scoring functions (1) and (2)

Ontology	distance / weights	f	TCT – *standard mode*		
			TCT 0.9	TCT 0.8	TCT 0.7
BIOPAX	(4) / u	20%	0.85 ± 0.03	0.82 ± 0.07	0.82 ± 0.08
		50%	0.86 ± 0.13	0.82 ± 0.13	0.83 ± 0.10
		70%	0.87 ± 0.12	0.87 ± 0.12	0.87 ± 0.13
	(4 / e	20%	0.90 ± 0.12	0.84 ± 0.13	0.81 ± 0.13
		50%	0.92 ± 0.14	0.90 ± 0.11	0.90 ± 0.10
		70%	0.93 ± 0.16	0.91 ± 0.11	0.90 ± 0.11
	(6) / u	20%	0.66 ± 0.20	0.68 ± 0.22	0.67 ± 0.30
		50%	0.68 ± 0.21	0.68 ± 0.21	0.68 ± 0.21
		70%	0.73 ± 0.22	0.70 ± 0.21	0.71 ± 0.21
	(6) / e	20%	0.76 ± 0.12	0.75 ± 0.08	0.74 ± 0.10
		50%	0.78 ± 0.12	0.76 ± 0.14	0.72 ± 0.12
		70%	0.78 ± 0.09	0.73 ± 0.11	0.73 ± 0.12
MUTAG.	(4) / u	20%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		50%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		70%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	(4) / e	20%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		50%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		70%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	(6) / u	20%	0.84 ± 0.07	0.82 ± 0.14	0.77 ± 0.13
		50%	0.82 ± 0.11	0.82 ± 0.11	0.81 ± 0.11
		70%	0.84 ± 0.09	0.84 ± 0.08	0.83 ± 0.10
	(6) / e	20%	0.95 ± 0.05	0.95 ± 0.05	0.95 ± 0.05
		50%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		70%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
VICODI	(4) / u	20%	0.93 ± 0.05	0.92 ± 0.06	0.92 ± 0.06
		50%	0.94 ± 0.01	0.89 ± 0.03	0.90 ± 0.03
		70%	0.94 ± 0.01	0.89 ± 0.03	0.90 ± 0.03
	(4) / e	20%	0.95 ± 0.02	0.90 ± 0.08	0.90 ± 0.08
		50%	0.98 ± 0.04	0.98 ± 0.04	0.98 ± 0.03
		70%	0.98 ± 0.03	0.97 ± 0.05	0.97 ± 0.03
	(6) / u	20%	0.96 ± 0.00	0.93 ± 0.10	0.92 ± 0.11
		50%	0.96 ± 0.13	0.94 ± 0.13	0.94 ± 0.12
		70%	0.96 ± 0.12	0.94 ± 0.14	0.93 ± 0.13
	(6) / e	20%	0.97 ± 0.14	0.96 ± 0.14	0.96 ± 0.14
		50%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		70%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
DBPEDIA	(4) / u	20%	0.90 ± 0.08	0.90 ± 0.08	0.90 ± 0.08
		50%	0.96 ± 0.08	0.96 ± 0.07	0.96 ± 0.09
		70%	0.96 ± 0.08	0.96 ± 0.07	0.96 ± 0.09
	(4) / e	20%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		50%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		70%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	(6) / u	20%	0.96 ± 0.03	0.96 ± 0.03	0.96 ± 0.03
		50%	0.96 ± 0.04	0.96 ± 0.03	0.95 ± 0.06
		70%	0.96 ± 0.04	0.96 ± 0.03	0.95 ± 0.06
	(6) / e	20%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		50%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		70%	0.99 ± 0.03	0.98 ± 0.03	0.99 ± 0.03

there was a non-negligible impact on the effectiveness coming from a proper tuning of the threshold ν. Conversely, the differences among the results are small in the experiments with largest ontologies such as GEOSKILLS, MONETARY, MUTAGENESIS, VICODI, DBPEDIA. This suggests that, for these ontologies, the variations of the numbers of axioms discovered were likely due to the random sampling performed by the refinement operators (both the original and the parallelized versions).

Table 4

Experimental comparison of the various approaches: average numbers of cases of inconsistency (#inc.) and total numbers of discovered axioms (#ax's) for PCC and NAR

Ontology	f	PCC		NAR	
		#inc.	#ax's	#inc.	#ax's
BIOPAX	50%	257	280	352	2990
NTN	50%	32	957	376	3766
FINANCIAL	50%	124	1112	542	5366
GEOSKILLS	50%	456	13384	456	13299
MONETARY	50%	543	13384	423	13456
MUTAGENESIS	50%	20	2264	45	14832
VICODI	50%	475	15518	472	18721
DBPEDIA	50%	1243	30470	1243	30365

Concerning the distance measures used in the experiments, we noted that this was an important factor for the number of the elicited disjointness axioms. Plugging the distance measure (4) in the TCT-based algorithm had as a consequence the induction of taller trees with a larger number of nodes, owing to the selection of concepts that required many splits of the sub-clusters sorted to the (negative) right branches of the trees. Moreover, in some cases, TCTs with clusters containing few individuals were produced. This was due to the distributions of the instances w.r.t. the various concepts: for example concepts with few instances are frequent in FINANCIAL while GEOSKILLS is more densely populated (and the number of empty clusters was more limited). Throughout the evaluation with TCT-v2 and TCT-v3, cases of totally empty clusters were also rarely observed: this result depended on the timeout adopted to avoid time-consuming refinement operations due to the hardness of finding candidate specializations featuring both positive and negative instances .

As regards the choice of the heuristic for selecting the most promising candidates, while this aspect did not affect the ability to rediscover the target axioms, it influenced the ability to induce new axioms avoiding the introduction of inconsistency: the scoring function (2) allowed to select concepts that determined sub-clusters whose distance was larger compared to those produced adopting function (1). As expected, this meant that the new heuristic was able to reduce the problematic cases of individuals of a sub-cluster that were close to those belonging to the sibling sub-cluster (which the heuristic 1 is less sensitive to), improving the homogeneity of the resulting sub-groups (and reducing also the number of the induced axioms). In particular, we noted that no inconsistency cases were introduced in the experiments with Eq. 1: for most candidate axioms involving concepts, say C and D, it seldom occurred that the number of individuals that were

instances of $C \sqcap D$ exceeded the given threshold (10) . This could yield to limit the use of the reasoner in order to check the inconsistency in the phase of the disjointness axiom elicitation. A similar number of axioms was also obtained by checking for inconsistency during the specialization phase. However, anticipating these checks in the generation of the refinements rather than having them discarded through the heuristic (2) made the approach more stable in terms of number of axioms produced with respect to the fraction f of axioms removed.

Comparative Experiments. It is worthwhile to note that throughout the experiments, the number of axioms induced through TCTs was larger than the number of axioms induced through the methods based on PCC and NAR (see Tab. 4). This was due to the fact the, via one of the refinement operators, the TCT-based method performs a search in a larger space than the one considered by the other methods as they focus on the mere combination of concept names selected from the KB signature.

The outcomes reported in Tabs. 4, 5, 6 and 7 show that, in absolute terms, more axioms were generally discovered using the proposed method (for all three choices of threshold ν selected for the experiments) compared with the two other methods and yet the number of inconsistencies introduced (in case of direct addition of the axioms to the KB) was quite limited in proportion to the overall number of axioms produced: for example, with MONETARY and VICODI this rate on average was less than the 3.5% with almost 20,000 discovered axioms. Inspecting sampled TCTs to gain a deeper insight into the outcomes, we could note that, for ontologies with a small number of concepts, such as BIOPAX and NTN, the refinement operator tended to introduce the same concept in more branches. As a consequence, the large number of axioms discovered was likely due to the replication of some sub-trees. This represented also one of the main causes for most of the inconsistency cases. This result improved by using heuristic (2) which turned out to be more robust than (1) (see Tabs. 6 and 7) at determining more suitable concepts describing non-overlapping sub-clusters. However, with such a heuristic the number of cases was lessened but the issue could not be completely prevented: using more complex languages as a trade-off would require equally complex and computationally expensive ref. operators. To get rid of such cases, anticipating the overlap test during the specialization is crucial. Introducing this solution, it was possible to drive the induction of TCTs towards other concept descriptions thus limiting the aforementioned replication problem.

As regards the performance of PCC and NAR, we noted that they had a more stable behavior with respect to the fractions of removed axioms f because, as previously mentioned, they could discover axioms involving exclusively named concepts of the KB signature whose instances are more likely to be available. Moreover, a weak correlation between two concepts is unlikely to depend on the presence of a disjointness ax-

Table 5

Experimental comparison of the various approaches: average numbers of cases of inconsistency (#inc.) and total numbers of discovered axioms (#ax's) using TCT v.1 and v.2 – scoring function (1)

Ontology	distance / weights	f	TCT 0.9		TCT 0.8		TCT 0.7	
			#inc.	#ax's	#inc.	#ax's	#inc.	#ax's
BIOPAX	(4) / u	20%	542	4235	576	4237	589	4237
		50%	345	3773	357	3817	364	3876
		70%	345	3773	357	3817	364	3876
	(4) / e	20%	235	3859	357	4235	365	4256
		50%	125	3576	357	4176	432	4115
		70%	125	3432	235	3875	417	4154
	(6) / u	20%	432	2567	446	2756	578	2757
		50%	236	2578	237	2758	238	2876
		70%	128	2587	128	2587	128	2578
	(4) / e	20%	235	2346	357	2357	365	2458
		50%	125	3576	357	4176	432	4115
		70%	125	3432	235	3675	417	3875
NTN	(4) / u	20%	432	3347	432	3347	432	3347
		50%	415	3256	415	3256	415	3256
		70%	415	3256	415	3256	415	3256
	(4) / e	20%	312	3128	343	3126	354	3124
		50%	234	3023	234	3034	235	3034
		70%	156	2987	176	2679	123	2675
	(6) / u	20%	432	4579	478	4789	478	4783
		50%	356	4321	356	4321	356	4321
		70%	356	4321	356	4321	356	4321
	(6) / e	20%	431	3083	431	3083	431	3083
		50%	345	2987	345	2987	345	2987
		70%	323	2996	324	2993	323	2996
MONETARY	(4) / u	20%	673	13765	673	13765	677	13767
		50%	432	13567	432	13567	432	13567
		70%	247	13127	231	13127	3127	13127
	(4) / e	20%	535	13456	573	13453	623	13460
		50%	315	13236	432	13236	532	13236
		70%	247	13127	231	13127	312	13127
	(6) / u	20%	756	12437	755	12438	847	12589
		50%	643	11357	647	11362	647	11362
		70%	536	10432	536	10432	536	10432
	(6) / e	20%	756	12437	876	12442	876	12321
		50%	643	11386	647	11373	647	11384
		70%	540	10457	540	10458	540	10458
VICODI	(6) / u	20%	431	18231	485	18432	502	18432
		50%	142	18231	345	18432	467	18431
		70%	141	18231	345	18432	312	18432
	(6) / e	20%	34	14753	43	14847	43	14978
		50%	23	14753	31	14753	32	14978
		70%	23	14753	32	14753	32	14978
	(4) / u	20%	431	17176	485	17176	502	17176
		50%	142	17176	142	17176	142	17176
		70%	142	17176	345	17176	467	17176
	(6) / e	20%	431	17176	485	17176	502	17176
		50%	142	17176	142	17176	142	17176
		70%	142	17176	345	171761	467	17176

Table 6

Experimental comparison of the various approaches: average numbers of cases of inconsistency (#inc.) and total numbers of discovered axioms (#ax's) using TCT v.1 and v.2 – scoring function (2)

Ontology	distance / weights	f	TCT 0.9		TCT 0.8		TCT 0.7	
			#inc.	#ax's	#inc./	#ax's	#inc. /	#ax's
BioPax	(4) / u	20%		2123		2124		2145
		50%	0	2123	0	2124	0	2145
		70%		2123		2123		2147
	(4) / e	20%		2145		2145		2145
		50%	0	2346	0	2346	0	2346
		70%		2346		2346		2346
	(6) / u	20%		2126		2126		2126
		50%	0	2098	0	2098	0	2098
		70%		1985		1985		1986
	(6) / e	20%		2145		2145		2145
		50%	0	2346	0	2346	0	2346
		70%		2346		2346		2346
NTN	(4) / u	20%		4123		4123		4123
		50%	0	4113	0	4123	0	4123
		70%		4113		4114		4114
	(4) / e	20%		3083		3083		3083
		50%	0	2987	0	2987	0	2987
		70%		2996		2993		2996
	(6) / u	20%		4123		4123		4123
		50%	0	4113	0	4123	0	4123
		70%		4113		4114		4114
	(6) / e	20%		3083		3083		3083
		50%	0	2987	0	2987	0	2987
		70%		2996		2993		2996
Monetary	(4) / u	20%		10243		10256		10256
		50%	0	10242	0	10257	0	10257
		70%		10243		10258		10258
	(4) / e	20%		10116		10116		10116
		50%	0	10116	0	10117	0	10115
		70%		10115		10116		10116
	(6) / u	20%		10257		10245		10244
		50%	0	10257	0	10245	0	10244
		70%		10257		10242		10257
	(6) / e	20%		10116		10116		10116
		50%	0	10116	0	10116	0	10116
		70%		10116		10116		10116
Vicodi	(6) / u	20%		16432		16432		16432
		50%	0	16239	0	16239	0	16239
		70%		16345		16345		16345
	(4) / e	20%		16456		16576		16579
		50%	0	16453	0	16453	0	16453
		70%		16453		16453		16453
	(6) / u	20%		16432		16432		16432
		50%	0	16239	0	16239	0	16239
		70%		16345		16345		16345
	(6) / e	20%		16456		16576		16579
		50%	0	16453	0	16453	0	16453
		70%		16453		16453		16453

iom involving them. This led them also not to introduce further inconsistencies.

4.2.3. *Examples of Discovered Disjointness Axioms*

For a more complete evaluation covering also the qualitative viewpoint, we report some examples of the axioms that could be discovered through the various methods. As previously mentioned, one of the advantages deriving from the employment of the TCTs is related to the kind of axioms that can be elicited. Purely statistical methods focus on the KB signature

Table 7

Experimental comparison of the various approaches: average numbers of cases of inconsistency (#inc.) and total numbers of discovered axioms (#ax's) using TCT v.3 – scoring function (2).

Ontology	distance / weights	f	TCT 0.9		TCT 0.8		TCT 0.7	
			#inc.	#ax's	#inc.	#ax's	#inc.	#ax's
BioPax	(4) / u			2123		2124		2145
	(4) / e	50%	0	2346	0	2346	0	2346
	(6) / u			2095		2100		2095
	(6) / e			2344		2344		2345
NTN	(4) / u			4113		4123		4123
	(4) / e	50%	0	2987	0	2987	0	2987
	(6) / u			4113		4123		4123
	6) / e			2987		2987		2987
Mutagenesis	(4) / u			12456		12326		12326
	(4) / e	50%	0	12217	0	12216	0	12220
	(6) / u			12456		12326		12326
	(6) / e			12217		12217		12217
Vicodi	(6) / u			16239		16239		16239
	(4) / e	50%	0	16453	0	16453	0	16453
	(6) / u			16239		16239		16239
	(6) / e			16453		16453		16453

and merely make pairwise comparisons in order to discover the concepts that are weakly correlated. Conversely, the TCT-based algorithm performs a sort of search that allows to elicit axioms that involve alternative versions of the targeted concepts, i.e. concept descriptions that are candidate to be equivalent to those considered in the target.

For instance, in the case of NTN, PCC and NAR could discover simple axioms like $\texttt{ReligiousOrganization} \sqsubseteq \neg\texttt{Woman}$ and $\texttt{ReligiousOrganization} \sqsubseteq \neg\texttt{Man}$, while the new method allowed to elicit the target axioms $\texttt{GroupofPeople} \sqsubseteq \neg\texttt{Person}$ and $\texttt{Man} \sqsubseteq \neg\texttt{Woman}$.

An interesting (not previously existing) disjointness axiom elicited using the TCTs involved more complex concepts like $(\exists\texttt{spouseOf}(\exists\texttt{visitedPlace}.(\forall\texttt{parentOf}.\top)))$ and $(\exists\texttt{nativePlaceOf}.\neg\texttt{Serial})$.

In the experiments with Financial, PCC and NAR produced the following target axioms: $\texttt{SouthMoravia} \sqsubseteq \neg\texttt{WestBohemia}$, where SouthMoravia and WestBohemia represents two different geographical areas, and the axiom $\texttt{Man} \sqsubseteq \neg\texttt{Woman}$. In the case of TCTs, a similar axiom to the target $\texttt{Man} \sqsubseteq \neg\texttt{Woman}$ was found, i.e. $\exists\texttt{hasSexValue.MaleSex} \sqsubseteq \neg(\exists\texttt{hasSexValue.FemaleSex})$.

In the experiments with GeoSkills, both all methods were able to detect the original disjointness between the concepts Vertex and Volume. Finally, in the case of DBPedia, one of the original axioms that were also elicited by PCC and NAR involved the concepts Mountain and Movie. Instead, the new method

elicited disjointness axioms between a potentially redundant concept description, i.e. $\neg$`Film` $\sqcap$ $\neg$`Person` $\sqcap$ `NaturalPlace`$\sqcap$`Mountain` (where `NaturalPlace` $\sqsupseteq$ `Mountain`) and `Movie` (where `Movie` $\equiv$ `Film`).

4.2.4. Efficiency of the Refinement Operators

One of the extensions proposed in this work concerns the use of a distributed version of the refinement operator in order to speed up the specialization generation task which represents one of prominent bottlenecks of the approach.

We carried out various tests aiming at determining how the solution implemented on the *Spark* framework could improve the efficiency of this task. To this purpose, we considered both the procedure implementing the new version of the refinement operator (henceforth *Distributed Refinement Operator* - DRO) and the original version (*Single-core Refinement Operator* - SRO) [14] also increasing the size of the beam of candidates: 100, 300, 400, 500, 600, 1000. Also, we ran these procedures using the entire set of individuals in each KB to test the stop condition in the procedures (see Algo. 2 and Algo. 3). We repeated the experiments considering both the original ontologies and the versions obtained applying the SDA.

Fig. 4 illustrates the outcomes (*execution time*) using the SDO and the DRO under SDA; similar trends were observed in the experiments on the original KBS.Throughout the experiments, we noted the DRO was significantly faster than the SRO, with differences spanning from less than 500 ms to more than 2,000,000 ms. We noted that using the SRO in most of the cases the time grew linearly w.r.t. the number of specializations, e.g. see the case of the experiments with MONETARY. This depended on two factors: the complexity, in terms of syntactic length, of the generated concept descriptions and the threshold on the number of individuals used to stop the condition. In particular, the generation of the concepts was biased towards the introduction of new concept names as conjuncts rather than the existential and universal restrictions. This means that there was a limited number of recursive calls of the refinement operator and shorter concepts. As a consequence the stop condition was satisfied earlier w.r.t. the case of concepts involving existential and universal restrictions. Indeed, instances of concept descriptions obtained as a conjunction of concept names are generally easier to find than for concepts involving universal and existential restrictions, due to the sparseness of assertional knowledge concerning roles observed in the KBs. As previously mentioned, the DRO was considerably more efficient: the time required for generating refinements increased less than linearly. In particular, with high dimensional beams, the benefits in terms of efficiency obtained with the distributed solution are noticeable whereas the use of low dimensional beams limited this improvement, due to the inherent overhead (e.g. for the transparent management of the RDD distribution).

A noteworthy case was the one related to the experiments with MUTAGENESIS: the time required by the SRO grew less than linearly also in the experiments with SRO. This was likely due to the effect of the SDA used for the limited number of concepts and their organization into a shallow hierarchy. As a consequence, due to the large number of axioms introduced in the KB, the refinement operator could rapidly check which individuals were either positive or negative instances w.r.t. a specialization, satisfying the stop condition early.

A final remark concerns the line of experiments in which the SDA was not made: they showed that the two operators behave similarly to the cases in which the SDA was made. Even if DRO (resp. SRO) showed a less than linear (resp. linear) increase of the time as larger beams were considered, the lack of disjointness axioms of the original KBs makes hard to find individuals with a definite membership for each specialization, thus delaying the satisfaction of the stop condition. However, it should be clarified that this problem depends on the specific reasoner adopted to make the inferences required by the algorithms.

5. Related Work

The problem of discovering the disjointness axioms to enrich and improve the quality of ontological knowledge bases has been receiving a growing attention. In early works the mentioned *strong disjointness assumption* [24], which states that the children of a common parent in the subsumption hierarchy should be considered as disjoint, has been exploited in a pinpointing algorithm for semantic *clarification* (i.e. the process of automatically enriching ontologies with appropriate disjointness statements [25]. Focusing on text and successively on RDF datasets, an unsupervised method for mining axioms, including disjointness axioms, has been proposed [26,4]. The main limitation of this approach is the loose use of any form of background knowledge which, on the contrary, can decisively help

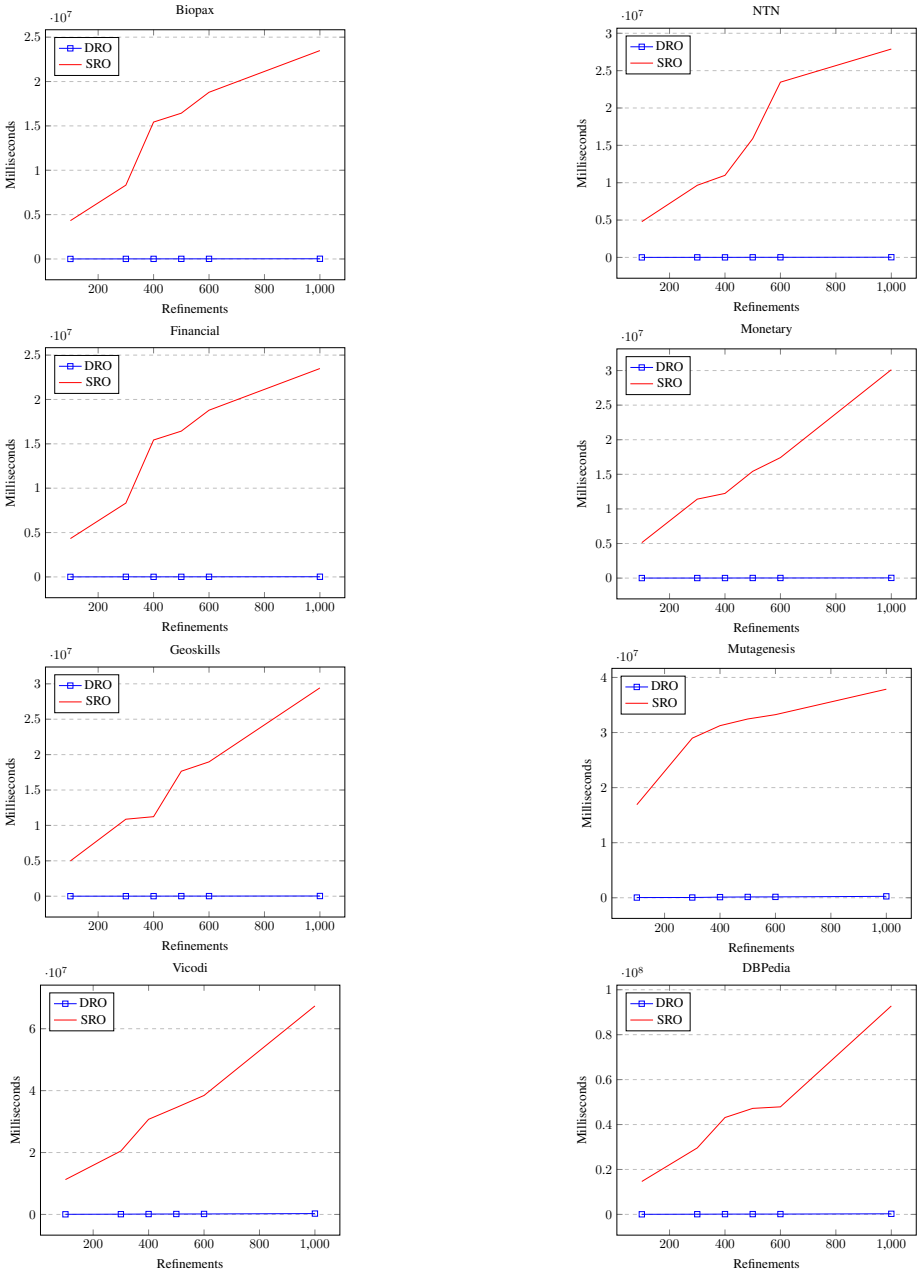

Fig. 4. Efficiency (ms) of the distributed refinement operator compared to the single-core implementation

to increase the number of discovered axioms while preventing unnecessary/wrong axioms.

Moreover, both *relational learning* methods and techniques based on *formal concept analysis* have been proposed to the purpose [27,28], but no specific assessment of the induced axioms quality is made. This limit has been pointed out also by Volker et al. [6], where an approach based on *association rule mining* has been introduced. Additional approaches, relying on association rules have been proposed by Fleischacker et al. [7] and Volker et al. [5]. The focus of these works was studying the correlation between classes comparatively. Specifically, *association rules*, *negative association rules* and *correlation coefficient* have been considered. Also in these cases, background knowledge and reasoning was exploited in a limited

extent. Lehman et al. [27] proposed a tool for *repairing* various types of ontology modeling errors: it uses supervised methods from the DL-LEARNER framework [29] to enrich ontologies with axioms elicited from existing instances.

Our solution is based on an unsupervised approach, deriving from previous works on concept learning and inductive classification [16]. Specifically, we propose a hierarchical conceptual clustering method that, in addition, is able to provide intensional cluster descriptions, and that exploits a novel form of a family of semi-distances over the individuals in an ontological knowledge base [13] which can take into account the available background knowledge. The method is grounded on the notion of medoid as cluster prototypes since clustering algorithms adopting medoids have been introduced to overcome known limits such as the lack of algebraical structure of the representation of the instance space [8]. The hierarchical approach proposed in this paper is related to classic clustering algorithms such as COBWEB [30] with some differences: 1) COBWEB produces directly n-ary cluster hierarchies instead of the binary ones in the TCTs. This allows for eliciting more intermediate concepts with the latter model; 2) the intensional definitions assigned to the clusters adopt a less expressive propositional representation language w.r.t. DLs; 3) a probabilistic cluster membership is modeled (like in the fuzzy clustering approaches) rather than a definite one, that is required to derive disjointness axioms. Related approaches to partitive clustering applied to datasets encoded in DL languages have been proposed, such as the hierarchical BISECTING K-MEDOIDS [11] or the *partition around medoids*, combined with evolutionary programming [10]. They are able to form clusters of individuals occurring in Web ontologies by exploiting metrics that are similar to those adopted in this work. However, these methods generally do not return any intensional cluster description. The derivation of concepts, i.e. intensional cluster definitions (*conceptual clustering* [15]) requires the adoption of additional and suitable concept learning algorithms.

Specifically, the method proposed in this paper relies on *logic tree* models [20] which essentially adopt a divide-and-conquer strategy to derive a hierarchical structure. The learning method can work both in supervised and unsupervised mode, depending on the availability of information about the instance classification to be exploited for separating sub-groups of instances. *Terminological decision trees* were derived [16,17] in the former case to classify individuals w.r.t. an unknown target concept (assigning a class label at each leaf node), while for the latter case, *First-order logic clustering trees* [19] were proposed to induce concepts for the clusters expressed in the context of *clausal logic theories*. The C0.5 system, which is integrated in the TILDE framework [20], is able to induce concepts as conjunctions of literals (clause bodies) installed at inner nodes. Almost all these exiting methods are grounded on the exploitation of an heuristic based on the *information gain*, employed in the supervised case, Differently, our approach tends to maximize the separation between cluster medoids according to a semi-distance measure.

6. Conclusions

In this work, we have cast the task of discovering disjointness axioms as a clustering problem that was solved exploiting terminological cluster trees, an extension of terminological decision trees [16] (proposed to solve supervised learning problems). Moving from our previous work [14], we extended the framework for inducing terminological cluster trees along various directions to aim at improving both its effectiveness and efficiency. Specifically, we aimed at improving the quality of the resulting axioms using: 1) different distance measures; 2) different heuristic for selecting the concepts to be installed into tree nodes; 3) a modified version of the refinement operator to generate concepts enabling the elicitation of axioms that do not introduce inconsistency in the KB. Secondly, the efficiency of the proposed solution has been improved integrating distributed computation technologies for processing Big Data, such as the *Spark* framework.

In the empirical evaluation, various experiments have been performed with the goal of assessing the effectiveness of the new methodology. Compared to related unsupervised approaches, ours proved to be able to discover disjointness axioms involving complex concept descriptions exploiting the underlying ontology as a source of background knowledge, unlike the other methods based on the statistical correlation between instances. The evaluation showed also that cases of inconsistency introduced in the KBs by the elicited axioms can be drastically lessened resorting to a different heuristic that selects promising concepts according to the distance between the farthest elements of a cluster w.r.t. the medoid the other cluster resulting from the split of the individuals in the parent cluster. Additionally, the aforementioned cases can be

totally avoided by checking the overlap between the concept installed into the current node and those installed into the tree up to that moment. To assesses the benefits deriving from employing Big Data technologies, we compared the *Spark*-based version of the refinement operator with the one used in he previous version [14]. We noted that, while the time required by the original version of the refinement operator increases linearly w.r.t. the number of specializations, the time required by the new version of the operator was almost constant w.r.t. the number of returned concepts.

Various extensions may be envisaged for this work. The TCT induction algorithm can be further improved by introducing a post-pruning step for better tackling the problem of empty clusters. Besides, it could be interesting to compare the implementation of the refinement operator on *Spark* to one made on similar frameworks. New metrics for evaluating the performance of such methods could also be proposed. Finally, it may be interesting to integrate the methodology within ontology engineering frameworks based on machine learning, such as DL-LEARNER [29], as a service for enriching the terminology of lightweight ontologies.

References

[1] T. Heath and 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 and P. Patel-Schneider (eds), *The Description Logic Handbook*, 2nd edn, Cambridge University Press, 2007.

[3] T.D. Wang, B. Parsia and J. Hendler, A Survey of the Web Ontology Landscape, in: *Proceedings of ISWC 2006*, I. Cruz et al., eds, LNCS, Vol. 4273, Springer, 2006.

[4] J. Völker, D. Vrandečić, Y. Sure and A. Hotho, Learning Disjointness, in: *Proceedings of ESWC 2007*, E. Franconi et al., eds, LNCS, Vol. 4519, Springer, 2007.

[5] J. Völker, D. Fleischhacker and H. Stuckenschmidt, Automatic Acquisition of Class Disjointness, *Journal of Web Semantics* **35**(P2) (2015), 124–139.

[6] J. Völker and M. Niepert, Statistical Schema Induction, in: *Proceedings of ESWC 2011*, G. Antoniou et al., eds, LNCS, Vol. 6643, Springer, 2011, pp. 124–138.

[7] D. Fleischhacker and J. Völker, Inductive Learning of Disjointness Axioms, in: *Proceedings of OTM 2011*, R. Meersman et al., eds, LNCS, Vol. 7045, Springer, 2011, pp. 680–697.

[8] C.C. Aggarwal and C.K. Reddy, *Data Clustering: Algorithms and Applications*, 1st edn, Chapman & Hall/CRC, 2013.

[9] J. Lehmann, N. Fanizzi, L. Bühmann and C. d'Amato, *Concept Learning*, in: *Perspectives on Ontology Learning*, J. Lehmann and J. Voelker, eds, AKA/IOS Press, 2014, pp. 71–91.

[10] N. Fanizzi, C. d'Amato and F. Esposito, Evolutionary Conceptual Clustering Based on Induced Pseudo-Metrics, *Int. J. Semantic Web Inf. Syst.* **4**(3) (2008), 44–67.

[11] N. Fanizzi, C. d'Amato and F. Esposito, Conceptual Clustering and its Application to Concept Drift and Novelty Detection, in: *Proceedings of ESWC 2008*, S. Bechhofer et al., eds, LNCS, Vol. 5021, Springer, 2008, pp. 318–332.

[12] F. Esposito, C. d'Amato and N. Fanizzi, Fuzzy Clustering for Semantic Knowledge Bases, *Fundam. Inform.* **99**(2) (2010), 187–205.

[13] G. Rizzo, C. d'Amato, N. Fanizzi and F. Esposito, Induction of Terminological Cluster Trees, in: *Proceedings of URSW 2016*, F. Bobillo et al., eds, CEUR Workshop Proceedings, Vol. 1665, CEUR-WS.org, 2016, pp. 49–60.

[14] G. Rizzo, C. d'Amato, N. Fanizzi and F. Esposito, Terminological Cluster Trees for Disjointness Axiom Discovery, in: *Proceedings of ESWC 2017, Part I*, E. Blomqvist et al., eds, LNCS, Vol. 10249, 2017, pp. 184–201.

[15] R.E. Stepp and R.S. Michalski, Conceptual Clustering of Structured Objects: A Goal-Oriented Approach, *Artif. Intell.* **28**(1) (1986), 43–69.

[16] N. Fanizzi, C. d'Amato and F. Esposito, Induction of Concepts in Web Ontologies through Terminological Decision Trees, in: *Proceedings of ECML/PKDD2010*, J.L. Balcázar et al., eds, LNAI, Vol. 6321, Springer, 2010, pp. 442–457.

[17] G. Rizzo, C. d'Amato, N. Fanizzi and F. Esposito, Tree-based models for inductive classification on the Web Of Data, *J. Web Sem.* **45** (2017), 1–22.

[18] J. Lehmann and P. Hitzler, A Refinement Operator Based Learning Algorithm for the *ALC* Description Logic, in: *ILP 2007, Revised Selected Papers*, H. Blockeel et al., eds, LNAI, Vol. 4894, Springer, 2007, pp. 147–160.

[19] L. De Raedt and H. Blockeel, Using logical decision trees for clustering, in: *Proceedings of ILP 1997*, N. Lavrač and S. Džeroski, eds, LNAI, Vol. 1297, Springer, 1997, pp. 133–140.

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

[21] J. Lehmann and P. Hitzler, Concept learning in description logics using refinement operators, *Machine Learning* **78**(1) (2009), 203.

[22] C. d'Amato, N. Fanizzi and F. Esposito, Query Answering and Ontology Population: An Inductive Approach, in: *Proceedings of ESWC 2008*, S. Bechhofer et al., eds, LNCS, Vol. 5021, Springer, 2008, pp. 288–302.

[23] C. d'Amato, N. Fanizzi and F. Esposito, Analogical Reasoning in Description Logics, in: *Uncertainty Reasoning for the Semantic Web I*, P.C.G. Costa et al., eds, LNAI, Vol. 5327, Springer Berlin Heidelberg, 2008, pp. 330–347.

[24] R. Cornet and A. Abu-Hanna, Usability of Expressive Description Logics – A Case Study in UMLS, in: *Proceedings of AMIA 2002*, I. Kohane, ed., AMIA, 2002, pp. 180–184.

[25] S. Schlobach, Debugging and semantic clarification by pinpointing, in: *Proceedings of ESWC 2005*, A. Gómez-Pérez and J. Euzenat, eds, LNCS, Vol. 3532, Springer, 2005, pp. 226–240.

[26] P. Haase and J. Völker, Ontology Learning and Reasoning: Dealing with Uncertainty and Inconsistency, in: *Uncertainty Reasoning for the Semantic Web I*, P. da Costa et al., eds, LNAI, Vol. 5327, Springer, 2008, pp. 366–384.

[27] J. Lehmann and L. Bühmann, ORE - A Tool for Repairing and Enriching Knowledge Bases, in: *Proceedings of ISWC 2010*, P. Patel-Schneider et al., eds, LNCS, Vol. 6497, Springer, 2010,

pp. 177–193.
[28] F. Baader, B. Ganter, B. Sertkaya and U. Sattler, Completing Description Logic Knowledge Bases using Formal Concept Analysis, in: *Proceedings of IJCAI 2007*, M. Veloso, ed., AAAI Press, 2007, pp. 230–235.
[29] S. Hellmann, J. Lehmann and S. Auer, Learning of OWL Class Descriptions on Very Large Knowledge Bases, *Int. J. Semant. Web Inf.* **5**(2) (2009), 25–48.
[30] D.H. Fisher, Knowledge Acquisition Via Incremental Conceptual Clustering, *Machine Learning* **2**(2) (1987), 139–172.

Appendix

A. Detailed Results of the Experiments with TCTs

We report in the following further outcomes of the experiments with the TCTs produced by the various versions of the algorithm.

Table 8

Average rates (and standard deviations) of original axioms rediscovered. Configurations *v.1* and *v.2* – scoring function 1

Ontology	distance / weights	*f*	TCT – *standard mode*		
			TCT 0.9	TCT 0.8	TCT 0.7
BioPax	(4) / u	20%	0.82 ± 0.08	0.82 ± 0.08	0.82 ± 0.08
		50%	0.82 ± 0.09	0.82 ± 0.09	0.82 ± 0.09
		70%	0.81 ± 0.10	0.81 ± 0.10	0.81 ± 0.10
	(4) / e	20%	0.90 ± 0.12	0.76 ± 0.13	0.74 ± 0.13
		50%	0.85 ± 0.13	0.74 ± 0.13	0.74 ± 0.13
		70%	0.85 ± 0.13	0.74 ± 0.12	0.74 ± 0.14
	(6) / u	20%	0.63 ± 0.23	0.67 ± 0.23	0.69 ± 0.29
		50%	0.69 ± 0.23	0.69 ± 0.22	0.69 ± 0.22
		70%	0.69 ± 0.23	0.69 ± 0.22	0.69 ± 0.22
	(6) / e	20%	0.70 ± 0.12	0.73 ± 0.11	0.73 ± 0.12
		50%	0.70 ± 0.12	0.73 ± 0.11	0.73 ± 0.12
		70%	0.70 ± 0.12	0.73 ± 0.11	0.73 ± 0.12
NTN	(4) / u	20%	0.95 ± 0.13	0.96 ± 0.12	0.96 ± 0.12
		50%	0.92 ± 0.13	0.93 ± 0.10	0.93 ± 0.10
		70%	0.90 ± 0.10	0.89 ± 0.11	0.89 ± 0.10
	(4) / e	20%	0.99 ± 0.08	0.95 ± 0.06	0.95 ± 0.08
		50%	0.97 ± 0.03	0.93 ± 0.10	0.93 ± 0.01
		70%	0.90 ± 0.10	0.89 ± 0.11	0.89 ± 0.10
	(6) / u	20%	0.83 ± 0.08	0.83 ± 0.08	0.83 ± 0.08
		50%	0.87 ± 0.15	0.87 ± 0.15	0.87 ± 0.15
		70%	0.88 ± 0.15	0.88 ± 0.15	0.88 ± 0.15
	(6) / e	20%	0.95 ± 0.12	0.95 ± 0.12	0.95 ± 0.12
		50%	0.93 ± 0.16	0.92 ± 0.10	0.92 ± 0.13
		70%	0.90 ± 0.10	0.91 ± 0.13	0.91 ± 0.11
Financial	(4) / u	20%	0.95 ± 0.14	0.92 ± 0.15	0.92 ± 0.15
		50%	0.95 ± 0.14	0.92 ± 0.15	0.92 ± 0.15
		70%	0.95 ± 0.05	0.95 ± 0.05	0.95 ± 0.05
	(4) / e	20%	0.99 ± 0.08	0.99 ± 0.08	0.99 ± 0.08
		50%	0.97 ± 0.03	0.97 ± 0.03	0.97 ± 0.03
		70%	0.95 ± 0.05	0.95 ± 0.05	0.95 ± 0.05
	(6) / u	20%	0.79 ± 0.21	0.79 ± 0.21	0.79 ± 0.21
		50%	0.68 ± 0.11	0.68 ± 0.11	0.68 ± 011
		70%	0.68 ± 0.11	0.68 ± 0.11	0.68 ± 011
	(6) / e	20%	0.82 ± 0.13	0.81 ± 0.13	0.81 ± 0.13
		50%	0.80 ± 0.15	0.80 ± 0.15	0.79 ± 0.16
		70%	0.80 ± 0.15	0.80 ± 0.15	0.79 ± 0.16
GeoSkills	(4) / u	20%	0.99 ± 0.08	0.99 ± 0.08	0.99 ± 0.08
		50%	0.92 ± 0.10	1.00 ± 0.00	1.00 ± 0.00
		70%	0.92 ± 0.10	0.92 ± 0.10	0.92 ± 0.10
	(4) / e	20%	0.99 ± 0.08	0.99 ± 0.08	0.99 ± 0.08
		50%	0.92 ± 0.10	1.00 ± 0.00	1.00 ± 0.00
		70%	0.92 ± 0.10	0.92 ± 0.10	0.92 ± 0.10
	(6) / u	20%	0.83 ± 0.23	0.83 ± 0.24	0.83 ± 0.24
		50%	0.85 ± 0.24	0.85 ± 0.25	0.85 ± 0.25
		70%	0.84 ± 0.25	0.84 ± 0.25	0.83 ± 0.24
	(6) / e	20%	0.94 ± 0.14	0.93 ± 0.10	0.93 ± 0.15
		50%	0.92 ± 0.11	0.92 ± 0.11	0.93 ± 0.10
		70%	0.92 ± 0.11	0.92 ± 0.11	0.93 ± 0.10

Table 9

Average rates (and standard deviations) of original axioms rediscovered. Configurations *v.1* and *v.2* – scoring function 1

Ontology	distance / weights	f	TCT – *standard mode*		
			TCT 0.9	TCT 0.8	TCT 0.7
MONETARY	(4) / u	20%	0.99 ± 0.08	1.00 ± 0.00	1.00 ± 0.00
		50%	0.94 ± 0.13	1.00 ± 0.00	1.00 ± 0.00
		70%	0.94 ± 0.13	0.91 ± 0.14	0.91 ± 0.13
	(4) / e	20%	0.99 ± 0.08	1.00 ± 0.00	1.00 ± 0.00
		50%	0.94 ± 0.13	1.00 ± 0.00	1.00 ± 0.00
		70%	0.94 ± 0.13	0.91 ± 0.14	0.91 ± 0.13
	(6) / u	20%	0.89 ± 0.14	0.76 ± 0.14	0.76 ± 0.13
		50%	0.92 ± 0.16	0.90 ± 0.16	0.92 ± 0.16
		70%	0.94 ± 0.13	0.94 ± 0.13	0.94 ± 0.12
	(6) / e	20%	0.97 ± 0.15	0.97 ± 0.15	0.97 ± 0.15
		50%	0.93 ± 0.11	0.93 ± 0.11	1.00 ± 0.00
		70%	0.94 ± 0.13	0.91 ± 0.14	0.91 ± 0.13
MUTAGEN.	(4) / u	20%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		50%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		70%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	(4) / e	20%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		50%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		70%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	(6) / u	20%	0.76 ± 0.14	0.77 ± 0.14	0.77 ± 0.13
		50%	0.82 ± 0.11	0.82 ± 0.11	0.81 ± 0.11
		70%	0.84 ± 0.09	0.84 ± 0.08	0.83 ± 0.10
	(6) / e	20%	0.95 ± 0.05	0.95 ± 0.05	0.95 ± 0.05
		50%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		70%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
VICODI	(4) / u	20%	0.95 ± 0.02	0.90 ± 0.08	0.90 ± 0.08
		50%	0.95 ± 0.02	0.90 ± 0.08	0.90 ± 0.08
		70%	0.95 ± 0.02	0.90 ± 0.08	0.90 ± 0.08
	(4) / e	20%	0.95 ± 0.02	0.90 ± 0.08	0.90 ± 0.08
		50%	0.95 ± 0.02	0.90 ± 0.08	0.90 ± 0.08
		70%	0.95 ± 0.02	0.90 ± 0.08	0.90 ± 0.08
	(6) / u	20%	0.92 ± 0.05	0.89 ± 0.08	0.89 ± 0.08
		50%	0.95 ± 0.04	0.93 ± 0.02	0.92 ± 0.04
		70%	0.95 ± 0.04	0.93 ± 0.02	0.92 ± 0.04
	(6) / e	20%	0.92 ± 0.05	0.89 ± 0.08	0.89 ± 0.08
		50%	0.92 ± 0.05	0.89 ± 0.08	0.89 ± 0.08
		70%	0.90 ± 0.05	0.87 ± 0.03	0.87 ± 0.03
DBPEDIA	(4) / u	20%	0.88 ± 0.15	0.88 ± 0.15	0.87 ± 0.16
		50%	0.96 ± 0.08	0.93 ± 0.07	0.91 ± 0.09
		70%	0.96 ± 0.08	0.90 ± 0.08	0.90 ± 0.08
	(4) / e	20%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		50%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		70%	0.96 ± 0.08	0.90 ± 0.08	0.90 ± 0.08
	(6) / u	20%	0.96 ± 0.03	0.96 ± 0.03	0.96 ± 0.03
		50%	0.94 ± 0.05	0.94 ± 0.05	0.94 ± 0.05
		70%	0.96 ± 0.08	0.90 ± 0.08	0.90 ± 0.08
	(6) / e	20%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		50%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		70%	0.96 ± 0.08	0.90 ± 0.08	0.90 ± 0.08

Table 10

Average rates (and standard deviations) of removed axioms rediscovered using TCTs v.3 – scoring functions (1) and (2)

Ontology	distance / weights	f	TCT – *standard mode*		
			TCT 0.9	TCT 0.8	TCT 0.7
BIOPAX	(4) / u	20%	0.85 ± 0.03	0.82 ± 0.07	0.82 ± 0.08
		50%	0.86 ± 0.13	0.82 ± 0.13	0.83 ± 0.10
		70%	0.87 ± 0.12	0.87 ± 0.12	0.87 ± 0.13
	(4 / e	20%	0.90 ± 0.12	0.84 ± 0.13	0.81 ± 0.13
		50%	0.92 ± 0.14	0.90 ± 0.11	0.90 ± 0.10
		70%	0.93 ± 0.16	0.91 ± 0.11	0.90 ± 0.11
	(6) / u	20%	0.66 ± 0.20	0.68 ± 0.22	0.67 ± 0.30
		50%	0.68 ± 0.21	0.68 ± 0.21	0.68 ± 0.21
		70%	0.73 ± 0.22	0.70 ± 0.21	0.71 ± 0.21
	(6) / e	20%	0.76 ± 0.12	0.75 ± 0.08	0.74 ± 0.10
		50%	0.78 ± 0.12	0.76 ± 0.14	0.72 ± 0.12
		70%	0.78 ± 0.09	0.73 ± 0.11	0.73 ± 0.12
NTN	(4) / u	20%	0.93 ± 0.10	0.91 ± 0.11	0.90 ± 0.12
		50%	0.92 ± 0.11	0.90 ± 0.10	0.88 ± 0.09
		70%	0.90 ± 0.10	0.89 ± 0.11	0.89 ± 0.10
	(4) / e	20%	0.99 ± 0.08	0.95 ± 0.06	0.95 ± 0.08
		50%	0.98 ± 0.03	0.98 ± 0.10	0.96 ± 0.01
		70%	0.98 ± 0.07	0.97 ± 0.07	0.97 ± 0.07
	(6) / u	20%	0.90 ± 0.14	0.87 ± 0.13	0.86 ± 0.12
		50%	0.90 ± 0.14	0.90 ± 0.16	0.90 ± 0.16
		70%	0.90 ± 0.13	0.90 ± 0.13	0.89 ± 0.16
	(6) / e	20%	0.95 ± 0.11	0.95 ± 0.13	0.95 ± 0.13
		50%	0.94 ± 0.11	0.91 ± 0.13	0.89 ± 0.12
		70%	0.96 ± 0.15	0.97 ± 0.14	0.96 ± 0.13
FINANCIAL	(4) / u	20%	0.97 ± 0.08	0.93 ± 0.23	0.93 ± 0.23
		50%	0.97 ± 0.08	0.93 ± 0.12	0.92 ± 0.16
		70%	0.95 ± 0.05	0.95 ± 0.05	0.95 ± 0.05
	(4) / e	20%	0.99 ± 0.08	0.99 ± 0.08	0.99 ± 0.08
		50%	0.97 ± 0.03	0.97 ± 0.03	0.97 ± 0.03
		70%	0.95 ± 0.05	0.95 ± 0.05	0.95 ± 0.05
	(6) / u	20%	0.83 ± 0.12	0.82 ± 0.11	0.82 ± 0.11
		50%	0.84 ± 0.11	0.81 ± 0.15	0.81 ± 0.15
		70%	0.85 ± 0.14	0.85 ± 0.14	0.83 ± 0.10
	(6) / e	20%	0.87 ± 0.09	0.84 ± 0.08	0.82 ± 0.06
		50%	0.87 ± 0.09	0.81 ± 0.14	0.80 ± 0.20
		70%	0.91 ± 0.12	0.88 ± 0.23	0.88 ± 0.24
GEOSKILLS	(4) / u	20%	1.00 ± 0.00	0.94 ± 0.12	0.92 ± 0.15
		50%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		70%	1.00 ± 0.00	0.93 ± 0.15	0.93 ± 0.16
	(4) / e	20%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		50%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		70%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	(4) / u	20%	0.81 ± 0.23	0.81 ± 0.24	0.81 ± 0.24
		50%	0.84 ± 0.23	0.84 ± 0.23	0.84 ± 0.23
		70%	0.85 ± 0.27	0.85 ± 0.27	0.85 ± 0.27
	(6) / e	20%	0.96 ± 0.15	0.92 ± 0.130	0.92 ± 0.12
		50%	0.97 ± 0.09	0.94 ± 0.09	0.94 ± 0.10
		70%	0.92 ± 0.11	0.91 ± 0.11	0.90 ± 0.10

Table 11

Average rates (and standard deviations) of removed axioms rediscovered using TCTs v.3 – scoring function (1) and (2)

Ontology	distance / weights	f	TCT		
			TCT 0.9	TCT 0.8	TCT 0.7
MONETARY	(4) / u	20%	0.99 ± 0.08	0.97 ± 0.05	0.97 ± 0.05
		50%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		70%	1.00 ± 0.00	0.96 ± 0.04	0.96 ± 0.03
	(4) / e	20%	0.99 ± 0.08	1.00 ± 0.00	1.00 ± 0.00
		50%	0.99 ± 0.08	1.00 ± 0.00	1.00 ± 0.00
		70%	0.99 ± 0.08	1.00 ± 0.00	1.00 ± 0.00
	(6) / u	20%	0.93 ± 0.15	0.87 ± 0.11	0.86 ± 0.14
		50%	0.94 ± 0.09	0.91 ± 0.08	0.91 ± 0.08
		70%	0.94 ± 0.09	0.91 ± 0.08	0.91 ± 0.08
	(6) / e	20%	0.97 ± 0.08	0.96 ± 0.07	0.97 ± 0.06
		50%	0.99 ± 0.03	0.97 ± 0.03	0.97 ± 0.03
		70%	0.97 ± 0.04	0.97 ± 0.04	0.97 ± 0.04
MUTAG.	(4) / u	20%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		50%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		70%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	(4) / e	20%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		50%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		70%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	(6) / u	20%	0.84 ± 0.07	0.82 ± 0.14	0.77 ± 0.13
		50%	0.82 ± 0.11	0.82 ± 0.11	0.81 ± 0.11
		70%	0.84 ± 0.09	0.84 ± 0.08	0.83 ± 0.10
	(6) / e	20%	0.95 ± 0.05	0.95 ± 0.05	0.95 ± 0.05
		50%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		70%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
VICODI	(4) / u	20%	0.93 ± 0.05	0.92 ± 0.06	0.92 ± 0.06
		50%	0.94 ± 0.01	0.89 ± 0.03	0.90 ± 0.03
		70%	0.94 ± 0.01	0.89 ± 0.03	0.90 ± 0.03
	(4) / e	20%	0.95 ± 0.02	0.90 ± 0.08	0.90 ± 0.08
		50%	0.98 ± 0.04	0.98 ± 0.04	0.98 ± 0.03
		70%	0.98 ± 0.03	0.97 ± 0.05	0.97 ± 0.03
	(6) / u	20%	0.96 ± 0.00	0.93 ± 0.10	0.92 ± 0.11
		50%	0.96 ± 0.13	0.94 ± 0.13	0.94 ± 0.12
		70%	0.96 ± 0.12	0.94 ± 0.14	0.93 ± 0.13
	(6) / e	20%	0.97 ± 0.14	0.96 ± 0.14	0.96 ± 0.14
		50%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		70%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
DBPEDIA	(4) / u	20%	0.90 ± 0.08	0.90 ± 0.08	0.90 ± 0.08
		50%	0.96 ± 0.08	0.96 ± 0.07	0.96 ± 0.09
		70%	0.96 ± 0.08	0.96 ± 0.07	0.96 ± 0.09
	(4) / e	20%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		50%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		70%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	(6) / u	20%	0.96 ± 0.03	0.96 ± 0.03	0.96 ± 0.03
		50%	0.96 ± 0.04	0.96 ± 0.03	0.95 ± 0.06
		70%	0.96 ± 0.04	0.96 ± 0.03	0.95 ± 0.06
	(6) / e	20%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		50%	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
		70%	0.99 ± 0.03	0.98 ± 0.03	0.99 ± 0.03

Table 12

Experimental comparison of the various approaches: average numbers of cases of inconsistency (#inc.) and total numbers of discovered axioms (#ax's) using TCT v.1 and v.2 – scoring function (1)

Ontology	Distance / weights	f	TCT 0.9		TCT 0.8		TCT 0.7	
			#inc.	#ax's	#inc.	#ax's	#inc.	#ax's
BIOPAX	(4) / u	20%	542	4235	576	4237	589	4237
		50%	345	3773	357	3817	364	3876
		70%	345	3773	357	3817	364	3876
	(4) / e	20%	235	3859	357	4235	365	4256
		50%	125	3576	357	4176	432	4115
		70%	125	3432	235	3875	417	4154
	(6) / u	20%	432	2567	446	2756	578	2757
		50%	236	2578	237	2758	238	2876
		70%	128	2587	128	2587	128	2578
	(4) / e	20%	235	2346	357	2357	365	2458
		50%	125	3576	357	4176	432	4115
		70%	125	3432	235	3675	417	3875
NTN	(4) / u	20%	432	3347	432	3347	432	3347
		50%	415	3256	415	3256	415	3256
		70%	415	3256	415	3256	415	3256
	(4) / e	20%	312	3128	343	3126	354	3124
		50%	234	3023	234	3034	235	3034
		70%	156	2987	176	2679	123	2675
	(6) / u	20%	432	4579	478	4789	478	4783
		50%	356	4321	356	4321	356	4321
		70%	356	4321	356	4321	356	4321
	(6) / e	20%	431	3083	431	3083	431	3083
		50%	345	2987	345	2987	345	2987
		70%	323	2996	324	2993	323	2996
FINANCIAL	(4) / u	20%	76	165	87	325	96	276
		50%	37	143	56	307	53	259
		70%	33	143	43	276	40	221
	(4) / e	20%	43	157	45	176	45	187
		50%	32	126	32	126	32	126
		70%	33	143	34	146	34	146
	(6) / u	20%	87	165	123	256	145	243
		50%	45	142	65	321	97	278
		70%	45	146	78	356	103	278
	(6) / e	20%	87	165	123	256	145	243
		50%	45	142	65	321	97	278
		70%	45	146	78	356	103	278
GEOSKILLS	(4) / u	20%	234	14289	357	14297	432	14345
		50%	231	14123	356	14154	417	14256
		70%	234	14122	358	14154	377	14187
	(4) / e	20%	234	14345	234	14356	235	14367
		50%	231	14234	356	14245	417	14256
		70%	216	14122	222	14154	231	14156
	(6) / u	20%	567	13126	568	13147	567	13187
		50%	432	13121	432	13121	432	13121
		70%	319	13098	319	13098	319	13098
	(6) / e	20%	357	13789	357	13789	357	13457
		50%	325	13456	356	13547	325	13456
		70%	325	13246	314	13321	314	13321

Table 13

Experimental comparison of the various approaches: average numbers of cases of inconsistency (#inc.) and total numbers of discovered axioms (#ax's) using TCT v.1 and v.2 – scoring function (1)

Ontology	Distance / weights	f	TCT 0.9		TCT 0.8		TCT 0.7	
			#inc.	#ax's	#inc.	#ax's	#inc.	#ax's
MONETARY	(4) / u	20%	673	13765	673	13765	677	13767
		50%	432	13567	432	13567	432	13567
		70%	247	13127	231	13127	3127	13127
	(4) / e	20%	535	13456	573	13453	623	13460
		50%	315	13236	432	13236	532	13236
		70%	247	13127	231	13127	312	13127
	(6) / u	20%	756	12437	755	12438	847	12589
		50%	643	11357	647	11362	647	11362
		70%	536	10432	536	10432	536	10432
	(6) / e	20%	756	12437	876	12442	876	12321
		50%	643	11386	647	11373	647	11384
		70%	540	10457	540	10458	540	10458
MUTAG.	(4) / u	20%	78	13234	78	13234	78	13234
		50%	63	13121	63	13120	63	13117
		70%	63	13121	63	13120	63	13117
	(4) / e	20%	34	14753	43	14847	43	14978
		50%	23	14753	31	14753	32	14978
		70%	23	14753	32	14753	32	14978
	(6) / u	20%	78	13234	78	13234	78	13234
		50%	63	13121	63	13120	63	13117
		70%	63	13121	63	13120	63	13117
	(6) / e	20%	34	13432	43	13432	43	13432
		50%	23	13432	31	13432	32	13432
		70%	23	13432	32	13432	32	13432
VICODI	(6) / u	20%	431	18231	485	18432	502	18432
		50%	142	18231	345	18432	467	18431
		70%	141	18231	345	18432	312	18432
	(6) / e	20%	34	14753	43	14847	43	14978
		50%	23	14753	31	14753	32	14978
		70%	23	14753	32	14753	32	14978
	(4) / u	20%	431	17176	485	17176	502	17176
		50%	142	17176	142	17176	142	17176
		70%	142	17176	345	17176	467	17176
	(6) / e	20%	431	17176	485	17176	502	17176
		50%	142	17176	142	17176	142	17176
		70%	142	17176	345	17176	467	17176
DBPEDIA	(4) / u	20%	1742	24275	1832	27967	1832	27968
		50%	1643	24389	1598	26433	1598	26433
		70%	1643	24389	1598	26433	1598	26433
	(4) / e	20%	1345	29730	1432	30143	1432	30567
		50%	1346	29730	1431	30143	1433	30567
		70%	1343	19730	1432	30143	1432	30567
	(6) / u	20%	1543	22275	1543	23879	1543	23890
		50%	1543	22275	1543	23879	1543	23890
		70%	1543	22275	1543	23879	1543	23890
	(6) / e	20%	1568	23467	1568	23467	1568	23467
		50%	1346	29730	1431	30143	1433	30567
		70%	1234	28730	1245	29654	1357	30765

Table 14

Experimental comparison of the various approaches: average numbers of cases of inconsistency (#inc.) and total numbers of discovered axioms (#ax's) using TCT v.1 and v.2 – scoring function (2)

Ontology	Distance / weights	f	TCT 0.9		TCT 0.8		TCT 0.7	
			#inc.	#ax's	#inc./	#ax's	#inc. /	#ax's
BIOPAX	(4) / u	20%		2123		2124		2145
		50%	0	2123	0	2124	0	2145
		70%		2123		2123		2147
	(4) / e	20%		2145		2145		2145
		50%	0	2346	0	2346	0	2346
		70%		2346		2346		2346
	(6) / u	20%		2126		2126		2126
		50%	0	2098	0	2098	0	2098
		70%		1985		1985		1986
	(6) / e	20%		2145		2145		2145
		50%	0	2346	0	2346	0	2346
		70%		2346		2346		2346
NTN	(4) / u	20%		4123		4123		4123
		50%	0	4113	0	4123	0	4123
		70%		4113		4114		4114
	(4) / e	20%		3083		3083		3083
		50%	0	2987	0	2987	0	2987
		70%		2996		2993		2996
	(6) / u	20%		4123		4123		4123
		50%	0	4113	0	4123	0	4123
		70%		4113		4114		4114
	(6) / e	20%		3083		3083		3083
		50%	0	2987	0	2987	0	2987
		70%		2996		2993		2996
FINANCIAL	(4) / u	20%		165		325		276
		50%	0	143	0	307	0	259
		70%		143		276		221
	(4) / e	20%		157		176		187
		50%	0	126	0	126	0	126
		70%		143		146		146
	(6) / u	20%		165		256		243
		50%	0	142	0	321	0	278
		70%		146		356		278
	(6) / e	20%		165		256		243
		50%	0	142	0	321	0	278
		70%		146		356		278
GEOSKILLS	(4) / u	20%		12345		12345		12345
		50%	0	12344	0	12346	0	12345
		70%		12345		12345		12345
	(4) / e	20%		11986		11986		11985
		50%	0	11987	0	11987	0	11986
		70%		11987		11985		11986
	(6) / u	20%		12345		12345		12345
		50%	0	12345	0	12345	0	12345
		70%		12345		12345		12345
	(6) / e	20%		12087		12087		12087
		50%	0	12087	0	12087	0	12087
		70%		12087		12087		12087

Table 15

Experimental comparison of the various approaches: average numbers of cases of inconsistency (#inc.) and total numbers of discovered axioms (#ax's) using TCT v.1 and v.2 – scoring function (2)

Ontology	Distance / weights	f	TCT 0.9		TCT 0.8		TCT 0.7	
			#inc.	#ax's	#inc.	#ax's	#inc.	#ax's
MONETARY	(4) / u	20%		10243		10256		10256
		50%	0	10242	0	10257	0	10257
		70%		10243		10258		10258
	(4) / e	20%		10116		10116		10116
		50%	0	10116	0	10117	0	10115
		70%		10115		10116		10116
	(6) / u	20%		10257		10245		10244
		50%	0	10257	0	10245	0	10244
		70%		10257		10242		10257
	(6) / e	20%		10116		10116		10116
		50%	0	10116	0	10116	0	10116
		70%		10116		10116		10116
MUTAG.	(4) / u	20%		12456		12326		12326
		50%	0	12456	0	12326	0	12326
		70%		12456		12326		12326
	(4) / e	20%		12217		12217		12217
		50%	0	12217	0	12217	0	12217
		70%		12217		12217		12217
	(6) / u	20%		12456		12326		12326
		50%	0	12456	0	12326	0	12326
		70%		12456		12326		12326
	(6) / e	20%		12217		12217		12217
		50%	0	12217	0	12217	0	12217
		70%		12217		12217		12217
VICODI	(6) / u	20%		16432		16432		16432
		50%	0	16239	0	16239	0	16239
		70%		16345		16345		16345
	(4) / e	20%		16456		16576		16579
		50%	0	16453	0	16453	0	16453
		70%		16453		16453		16453
	(6) / u	20%		16432		16432		16432
		50%	0	16239	0	16239	0	16239
		70%		16345		16345		16345
	(6) / e	20%		16456		16576		16579
		50%	0	16453	0	16453	0	16453
		70%		16453		16453		16453
DBPEDIA	(4) / u	20%		18765		18765		18765
		50%	0	18654	0	18654	0	18654
		70%		18654		18654		18654
	(4) / e	20%		21459		21459		21459
		50%	0	21578	0	21578	0	21578
		70%		21578		21578		21578
	(6) / u	20%		18765		18765		18765
		50%	0	18654	0	18654	0	18654
		70%		18654		18654		18654
	(6) / e	20%		21459		21459		21459
		50%	0	21578	0	21578	0	21578
		70%		21578		21578		21578

Table 16

Experimental comparison of the various approaches: average numbers of cases of inconsistency (#inc.) and total numbers of discovered axioms (#ax's) using TCT v.3 – scoring function (2).

Ontology	Distance / weights	f	TCT 0.9		TCT 0.8		TCT 0.7	
			#inc.	#ax's	#inc.	#ax's	#inc.	#ax's
BIOPAX	(4) / u	50%	0	2123	0	2124	0	2145
	(4) / e			2346		2346		2346
	(6) / u			2095		2100		2095
	(6) / e			2344		2344		2345
NTN	(4) / u	50%	0	4113	0	4123	0	4123
	(4) / e			2987		2987		2987
	(6) / u			4113		4123		4123
	6) / e			2987		2987		2987
FINANCIAL	(4) / u	50%	0	143	0	307	0	259
	(4) / e			126		126		126
	(6) / u			142		321		278
	(6) / e			142		321		278
GEOSKILLS	(4) / u	50%	0	12345	0	12345	0	12345
	(4) / e			11986		11986		11986
	(6) / u			12345		12345		12345
	(6) / e			12087		12087		12087
MONETARY	(4) / u	50%	0	10242	0	10257	0	10257
	4 / e			10116		10116		10116
	6 / u			10257		10245		10244
	(6) / e			10116		10116		10116
MUTAGENESIS	(4) / u	50%	0	12456	0	12326	0	12326
	(4) / e			12217		12216		12220
	(6) / u			12456		12326		12326
	(6) / e			12217		12217		12217
VICODI	(6) / u	50%	0	16239	0	16239	0	16239
	(4) / e			16453		16453		16453
	(6) / u			16239		16239		16239
	(6) / e			16453		16453		16453
DBPEDIA	(4) / u	50%	0	18654	0	18654	0	18654
	(4) / e			21578		21578		21578
	(6) / u			18654		18654		18654
	(6) / e			21578		21578		21578