Ontology Alignment Through Instance Negotiation: a Machine Learning Approach.

Ignazio Palmisano[†], Luigi Iannone[†], Domenico Redavid[*], Giovanni Semeraro[*]
[*]Dipartimento di Informatica, Università degli Studi di Bari, Via Orabona 4, 70125 Bari, Italy
Email: {redavid, semeraro}@di.uniba.it
[†]Computer Science Department, Liverpool University, Ashton Building, Ashton Street, L69 BX Liverpool, UK
Email: {ignazio, luigi}@csc.liv.ac.uk

***Abstract*— The Semantic Web needs methodologies to accomplish actual commitment on shared ontologies among different actors in play. In this paper, we propose a machine learning approach to solve this issue relying on classified instance exchange and inductive reasoning. This approach is based on the idea that, whenever two (or more) software entities need to align their ontologies (which amounts, from the point of view of each entity, to add one or more new concept definitions to its own ontology), it is possible to learn the new concept definitions starting from shared individuals (i.e. individuals already described in terms of both ontologies, for which the entities have statements about classes and related properties); these individuals, arranged in two sets of positive and negative examples for the target definition, are used to solve a learning problem which as solution gives the definition of the target concept in terms of the ontology used for the learning process. The method has been applied in a preliminary prototype for a small multi-agent scenario (where the two entities cited before are instantiated as two software agents). Following the prototype presentation, we report on the experimental results we obtained and then draw some conclusions.**

I. Motivation

The Semantic Web (SW), being an evolution of the World Wide Web, will inherit Web decentralized architecture. Decentralization, in its turn, was one of the success factors of the Internet, granting its structure with scalability, no single point of failures, and so on. However, in order to implement the SW scenario envisioned in [1], semantic convergence is crucial. This point has been always identified in the employment of ontologies that, even before the rise of SW, were considered as *"shared formalizations of conceptualizations"* [2].

In the SW vision, different actors that want to take advantage from interoperating should be able to converge onto shared ontologies in order to communicate. Such a problem turned out to be crucial and very difficult to work out. Indeed, each party involved has its peculiar view of the domain it shares with other parties. Very often, different applications are interested in different aspects of the same *world state*, e.g. in a typical B2B scenario in which suppliers and final customers usually are interested into different aspects of the goods that are dealt with. Both quantitative and, more likely, qualitative concepts often turn out to be fuzzy depending on those actors interested in assessing them to interoperate. Suppliers may likely consider features like materials, provenance, standard processes, whereas customer satisfaction may depend also on other factors, e.g. warranties, comfort, support, which may turn out to be only marginal for the suppliers, unexpressed or simply difficult to acquire.

In such cases, partially overlapping visions of the same world should be integrated in value chains in order to provide goods. However, The wide range of possible B2B scenarios hinders a centralized approach to ontology sharing for semantic convergence, which contrasts with the architectural paradigm of the SW. Indeed, such a scenario requires flexibility, since it basically builds up open situations where the interacting actors cannot be precisely individuated in advance. Hence, a central ontology should foresee a high number of particular situations, and therefore it would be detailed enough, and therefore useful, only for restricted cases and toy domains. Moreover, single application ontologies are supposed to vary in time, an issue that is not easily manageable in centralized approaches.

We agree on the fact that semantic convergence has to be addressed at the ontology level, but we argue that the approaches for solving such issues should fulfil the following properties to be effectively applied in the mentioned scenarios:

- they have to be automatizable, so that applications could integrate among each other without human intervention (or with as little human intervention as possible);
- they have to be flexible. It should be possible to apply the same convergence schema to any domain on which the actors have to communicate their own (partially overlapping) conceptualizations;
- they have to be on-demand and limited in scope. The aim is not to fully map the actors conceptualization, which is often impossible or even useless, but to reconciliate only those parts that are necessary for the dialogue.

Developing platforms enabling SW systems to communicate with each other, without committing to a *superimposed* ontology, ensures a very loose coupling among them which allows for independent evolution and maintaining of the applications. In our opinion, other SW technologies spanning from Web Service discovery, orchestration, and choreography, up to software agents immersed in the SW, could profit from the solutions that research will find to these issues.

In this paper, we suggest a Machine Learning approach to accomplish semantic integration, taking into account the requirements listed above. The remainder of the paper is or-

ganized as follows: the next section will briefly summarize the state of the art (to our knowledge) on these problems. Sect. III illustrates our approach to Ontology Alignment together with a proposed algorithm. Sect. IV explains in detail the Machine Learning techniques used to implement the algorithm, while Sect. V presents the implemented prototype employed to carry out a preliminary empirical evaluation reported in this section. Finally, in Sect. VI, some conclusions are drawn outlining future work directions.

II. Related Work

In this section, we present a short discussion of some recent relevant research describing different approaches to ontology alignment. Ontology Alignment is a broad umbrella for a variety of subtask that try to solve some of the issues pointed out in the previous section. In order to keep the subject simple and general, we might define alignment as: *any process that enables communication among two or more systems, adopting two or more different ontologies*. Following Noy [3], there are (at least) two types of alignment processes, classified according to their output:

- Ontology Merging: in which the outcome of the alignment of n ontologies is a single new one, embedding the knowledge coming from the n sources
- Ontology Mapping: in which the results of the alignment among n different ontologies consists of the starting ones plus the mapping between similar entities of the input sources.

In the following, we briefly discuss a non-exhaustive list of some remarkable research approaches pursuing such alignment strategies.

A. GLUE.

GLUE [4], [5] exploits Machine Learning techniques to find semantic mappings between concepts stored in distinct and autonomous ontologies. Basically, its strategy falls into the Ontology Merging category as its output is a *mediator* ontology (or database schema in its earlier version). The process relies on a initial manual bootstrapping phase, in which user provides some sample mappings between some entities within the ontologies to be merged. GLUE then, tries to infer a set of rules (or classifiers) that can *synthesize* the mapping strategy followed by the users for the initial mappings. GLUE then, applies what it learnt to find other semantic mappings. One of its strong points is the possibility of using any kind of probabilistic similarity measure (measure neutrality) and, furthermore, also the weighting of the single similarity assessments is learnt by the system. In [4] the authors underline that there are many kinds of mappings that can be found, comparing entities from different ontologies, however GLUE focuses on finding 1-1 mappings between concepts belonging to two taxonomies.

B. PROMPT *Suite.*

The PROMPT Suite by Stanford Medical Informatics [6], [3] provides two tools for alignment: iPROMPT and ANCHOR-PROMPT. iPROMPT is an interactive tool performing ontology merging. It starts from initial mappings among two entities (provided either by users or by a automatic process based on linguistic similarity) and generates suggestions for further mappings between other entities. After the user triggers one among the suggestions proposed (or performs some changes on the resulting ontology), iPROMPT applies the required changes, computes the possible cases of inconsistency trying to suggest fixes for those, and produces new suggestions for further mappings.
ANCHORPROMPT, on the contrary, is an ontology mapping tool. It can be used for supporting iPROMPT in order to individuate new related concepts during the iterations. Unlike iPROMPT, it exploits also non-local contexts for the concept pairs whose similarity has to be assessed. Indeed, it represents ontologies as graphs and compares the paths in which concepts are involved, both within the taxonomy and in slot dependencies and domain/range constraints.

C. APFEL.

APFEL is really a whole framework rather than a single alignment technique. In [7], Ehrig et al. propose a very interesting classification of alignment strategies dividing them into manually predefined automatic methods and learning from instance representations methods. The former are very complex to design, since all the possible peculiarities of the domains they will be applied must be considered; the latter, on the contrary, can be used only in presence of instances (and sometimes this is not the case) though they show more flexibility as the underlying ontology domain changes. The proposed framework (PAM - Parameterizable Alignment Methodology) is fully parameterizable to the extent of being able to reproduce the strategy used in other methodologies by just adjusting some parameters. APFEL tries to learn from pre-validated example mappings the right parameters that would instantiate the fittest PAM for the particular learning problem. Such parameters are:

- *Features*, i.e. the smallest set of features on which to compute similarity among entities
- *Selection criteria* for the next pair of entities to compare
- *Similarity measures*, aggregation, and interpretation strategies
- *Iteration*, that is the extent to which neighbor entity pairs are influenced by the current pair (whose similarity has been evaluated).

III. Ontology Alignment Algorithm

According to Noy's classification cited in the previous section, the approach described in this work falls into the ontology mapping methodology. Indeed, without loss of generality, we assume to work on two ontologies dealing with the same domain. Besides, we also assume that some of the concepts within the input ontologies may partially overlap in their semantics. We intentionally will not define formally what this semantic overlap means, due to the large variety of practical situations in which this may happen in real-world applications. We might have concepts defined with different

names, structures, etc. that share exactly the same meaning. On the other hand, we can have concepts that, though sharing their name, might be different in their actual meaning, and it could be necessary that an application understands what its peer exactly means for such a concept, possibly in terms of its own ontological primitives.

A couple of examples can better explain these situations. The former case can be exemplified as follows. Suppose that there are two applications dealing with two ontologies about cars, differing just for the language: one uses Italian names and the other English names. The concept of, say, Wheel is equivalent to the concept Ruota in the Italian version of the car ontology. The latter case, instead, may happen when you have two ontologies describing a domain from two different perspectives. Suppose that the ontology FoodRestaurant deals with food from the typical point of view of a restaurant and that the ontology FoodNutritionist deals with the same subject from the side of a dietician. The concept of HighQualityFood depends on different aspects of food according to the adopted viewpoint. Indeed, when defining this concept, a dietician, would care of nutritional values of the ingredients, whereas a chef would take into account the provenance of the ingredients, their rarity and so on. In such case, we have two legal HighQualityFood concepts; an agent for dinner arrangements dealing with restaurants and people could be more effective if it knows both senses.

The intuition behind our approach is that a useful mapping is the one in which at least one of the two parties can *understand* what the other means in terms of its own ontology. Indeed, if an application is to use information coming from another system, it should be able to frame this information in its own knowledge model, hence, in our opinion, it has to grasp some meaning using primitive concepts and properties from its own ontology. Though this may seem straightforward, many approaches limit themselves to find a correspondence between equivalent (or very similar) entities, provided that such an equivalence exists. Yet such approaches disregard the importance of maintaining different definitions for the same entity (the viewpoints discussed in the previous example) and of having one's own definitions, in order to communicate with different parties.

Therefore, a way is needed for allowing a system to build a representation of a concept belonging to another ontology using its own terminology. We argue that the inductive reasoning techniques of Machine Learning are a suitable way to accomplish this objective. In particular, we employ the *Concept Learning* paradigm [8] for treating this problem. Indeed, with Concept Learning aims at inducing an intensional descriptions of concepts starting from its examples and (possibly) counterexamples.

In particular, we intend to implement the following scenario. Suppose there are two applications A and B with their respective ontologies O_A and O_B and suppose that application A wants to communicate about some concept, say $a : C$, with B (we suppose to indicate concepts with the usual URIs, i.e. namespace:conceptLocalName) regarding a concept that does not appear in O_B. Then, in our scenario B could ask A to provide some instances (examples) of $a : C$ and, possibly, also some counterexamples (i.e.: instances of the complement class of $a : C$). If such examples occur as instances within O_B, then B, by means of a Concept Learning algorithm, can infer a definition of $a : C$ in terms of the descriptions of those instances contained in O_B. In this way B obtains a definition of $a : C$ using its own terminology, that is, in other words, B *understands* $a : C$ according to what it knows (its ontology). In the following section (Sect. IV) the particular Concept Learning algorithm employed is described; however, for the purposes of this section the particular algorithm is irrelevant since the only requirement is that it solves a generic Concept Learning problem, described formally in the following definition:

Concept Learning problem

Given a knowledge base $\mathcal{K}$ and a set of instances divided into examples and counterexamples $E = E^+ \cup E^-$ and described in $\mathcal{K}$

Learn the definition of a concept C such that all elements of E^+ none of E^- are instances of C.

The result of the learning problem can be further revised in the following way. B identifies a suitable subset of instances (different from those provided by A as a training set), classifies them as belonging or not to the learnt concept and passes them to A asking the peer to do the same. If A and B disagree on the classification of some individuals, it will be the case that B refines properly the learnt definition in order to fix the discrepancy with the classification provided by A.

After learning a definition of $a : C$, let us call it $b : C$ supposing there are no clashes with pre-existing names, B has to compare it with the concepts in its O_B, evaluating the similarity of $b : C$ to pre-existing concepts. This can accomplished using suitable similarity measures such as the one proposed in [9] that score the semantic affinity/diversity of a pair of concepts within an ontology. If the similarity between $b : C$ and another concept in O_B exceeds a given threshold or, better, if the equivalence can be proved by a reasoner, then an equivalence axiom can be added, otherwise the definition of $b : C$ is left unchanged within O_B.

Careful readers should have already found out that there is an assumption on which the whole process relies on. The assumption is that there exists a subset of individual URIs in common between O_A and O_B. This corresponds to the initial knowledge that is provided in other systems during the bootstrapping phase. To cite only the related work, GLUE needs a complete example of mapping to start the learning phase, whereas iPROMPT relies on initial suggestions on linguistic similarity, assuming that concept names are expressed in their natural language form (or very close to it). ANCHORPROMPT, on the contrary, exploits the graph structure assuming that, for ontologies partially overlapping on the same domain, their graph structure should be similar. Our assumption seems at least as reasonable as these, also because it can be relaxed thinking that, instead of having some identical individual URIs across ontologies, we have mechanisms to map some URIs

into some others (either manually or automatically).

Consider, for instance, the ontologies about paper indexing systems, e.g. DBLP and the ACM Digital Library, and suppose there exist their OWL (or RDF) [1] versions. Each system has automatic strategies for generating identifiers and there is a large subset of research works that are indexed by both of them. It could be easy then to provide URIs translator for the two systems. Actually, ACM and DBLP bibliographic information are used in one of the test cases we devised for our system; more details in Sect. V-C.

IV. Machine Learning in Description Logic

In this section, we describe in more details the learning algorithm we developed for solving a Concept Learning Problem in Description Logic. In particular, we focus on concept descriptions in the $\mathcal{ALC}$[10] Description Logic as it is the least expressive subset of OWL-DL allowing for disjunction and ensuring a reasonable expressiveness without features (like role hierarchies) that would make learning intractable.

First we cast the generic Concept Learning Problem reported in Def. III to the Description Logic setting.

learning/tuning problem In a search space of concept definitions $(\mathcal{S}, \sqsubseteq)$ ordered by subsumption
Given a knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ and a set of positive and negative assertions $\mathcal{A}_C = \mathcal{A}_C^+ \cup \mathcal{A}_C^-$ regarding the membership (or non-membership) of some individuals to a concept C such that: $\mathcal{A} \not\models_{\mathcal{T}} \mathcal{A}_C$
Find a new T-box[2] $\mathcal{T}' = (\mathcal{T} \setminus \{C \equiv D\}) \cup \{C \equiv D'\}$ such that: $\mathcal{A} \models_{\mathcal{T}'} \mathcal{A}_C$

A Concept *Tuning* (or Revision) Problem is similar but for the fact that learner already knows an incorrect definition for C which has to be revised. This means that there are some assertions within $\mathcal{A}_C$ that are not logical consequences of the knowledge base and the current definition of C. Such incorrect definition can be *incomplete*, i.e.: there are some individuals that are said to belong to C in $\mathcal{A}_C$ but this cannot be entailed, and/or it can be *incorrect*, meaning that there are individuals that are deduced to belong to C from the definition of C, while they actually don't belong to it.

In general, Concept Learning relies on the intuition that the solution of any problem can be found traversing a proper search space.

A learning algorithm, then, is cast as a search that moves across such a space with two kinds of possible movements, called *refinements*:

- Specific towards general (*generalization*)
- General towards specific (*specialization*)

We provide here the formal (though generic) definitions for both kinds of refinement.

downward refinement operator - ρ Let $(\mathcal{S}, \preceq)$ be a search space of concept descriptions C ordered by subsumption $\sqsubseteq$. We define the function $\rho : \mathcal{S} \to 2^{\mathcal{S}}$ such that $\rho(C) \subseteq \{E \in 2^{\mathcal{S}} \mid E \sqsubseteq C\}$.

upward refinement operator - δ Let $(\mathcal{S}, \preceq)$ be a search space of concept descriptions C ordered by subsumption relation $\sqsubseteq$ we define the function $\delta : \mathcal{S} \to 2^{\mathcal{S}}$ such that $\delta(C) \subseteq \{E \mid E \in 2^{\mathcal{S}} \wedge C \sqsubseteq E\}$.

Notice that it does not hold in general that:

$$E \sqsubseteq C \to E \in \rho(C) \text{ or } C \sqsubseteq E \to E \in \delta(C)$$

Refinement has been thoroughly studied in literature (see [11], [12]) also specifically for Description Logic knowledge representations [13]. For the sake of brevity we will not expose here all the theoretical framework for refinement. It suffices to saying that, unlike many other approaches, we try to use examples and counterexamples of the concept to be learnt to bias the search and, hence, the refinement strategies, rather than using them just for testing intermediate candidate refinements. Indeed, during learning, as the algorithm searches for a solution of the problem, it may find various intermediate hypotheses that satisfy the required constraints[3] but not all of them. Hence refinement operators must be applied to fulfil such constraints. The problem is that at each refinement step there are many alternative directions that can be chosen. Examples come into play in this choice, guiding the refinement process towards selecting the most promising candidates, thus pruning the search space and avoiding the learner to backtrack for the sake of efficiency.

Our solution to the learning problem is the algorithm implemented in the learning system named YinYang that is an evolution of the system described in [14]. In YinYang examples are not processed at the individual description level but, for each of them, a concept level representative is computed. This is very frequent in Machine Learning when the example and hypothesis languages do not coincide, and is called *Single Representation Trick*. Concept level representatives should be as much adherent as possible to the examples they stand for. In Description Logic there exist a non standard inference called Most Specific Concept (denoted msc) that fulfills such a requirement but, unfortunately, it needs not exist for many Description Logics (such as $\mathcal{ALC}$). Hence it will be approximated provided that it has to be possible to distinguish among concept level representative of positive and negative examples. In other words it cannot exist a unique concept level representative for a positive and a negative example.

The algorithm starts choosing a seed that is a concept level representative of an example. Then, it keeps generalizing it until there are residual positive to be covered or until the generalization covers some negative example (overgeneralization). In the former case, the algorithm exits returning a complete and correct generalization. In the latter, specialization is necessary in order to correct the overly general hypothesis (concept definition).

[1] `http://www.w3.org/2004/OWL/`, `http://www.w3.org/RDF/`

[2] T-box (terminological box), is the portion of the knowledge base containing the theory ($\mathcal{T}$), while A-box (assertional box) is the portion containing the ground statements

[3] It should covers a subset of examples and does not cover a part of counterexamples, but there would remain some uncovered examples and/or some covered counterexamples.

Specialization can take place in two different ways. The first one is based on the notion of *counterfactuals* [15]. The idea behind is that if we have a concept that covers some negative examples one can single out the parts of such definitions that are *responsible* for the coverage and rule them out by means of negation. There is a well known *non-standard* inference in DLs called *concept difference* or *residual* [16] used for the aforementioned identifications of part of definitions which are responsible for the incorrect coverage (also known as *blame assignment*). For eliminating in one single negation all the different parts that are responsible for the coverage of each negative, a generalization of the blamed parts is needed in order to obtain a single description to be negated (i.e. a counterfactual). Thus, again generalization is needed, for the negative residuals with respect to the overly general hypothesis. Such negative residuals, will now become positive examples in a new nested learning problem and the solution of such problem will be our counterfactual. Since its negation has to be conjoined to the starting incorrect hypothesis in order to specialize, one do not want to incur in the dual error, i.e. overspecialization. Indeed, the refined resulting concept definition must keep covering the positive examples it covered before specialization. Hence the counterfactual should not cover the residual of covered positive example representatives w.r.t. the covering hypothesis to specialize. This means that they represent negative examples for the new nested learning problem solved by the recursive call.

Specialization by means of counterfactuals can be proved not to be complete in the sense that it sometimes fails in finding a correct counterfactual for properly specializing overly general hypotheses. That is why another specialization strategy was introduced, named Add Conjunct, which is applied whenever the Counterfactual routine fails. Such a strategy aims at finding a conjunct per positive example that does not appear yet in the hypothesis to be specialized nor in none of the negatives that are currently covered. In order to minimize the number of conjuncts, each positive provides such a conjunct only if there are no conjuncts (already provided by some other positives) that cover it. After having computed such conjuncts, they are generalized into their least common subsumer (lcs) [10] and such lcs is conjoined to the overly general hypothesis. Since the requirements for a conjunct to be chosen for each positive are very tight, it may sometimes happen that such specialization strategy fails as well as counterfactual before it. In such cases, the overly general hypothesis is simply replaced by the lcs of its covered positive example representatives, such lcs is added as a disjunct to the main generalization and the algorithm chooses another seed starting again the inner loop.

V. Prototype Description and Preliminary Evaluation

We developed a preliminary prototype that implements the methodology in Sect. III. We thought that one of the most suitable settings could be a Multi-Agent System. Indeed, it seems natural to think of communicating actors as autonomous agents that have their own knowledge bases and wish to dialogue with each other. In such environments, it is very likely that there is no common ontology to commit to; this can offer plenty of possible scenarios in which ontology alignment can be evaluated.

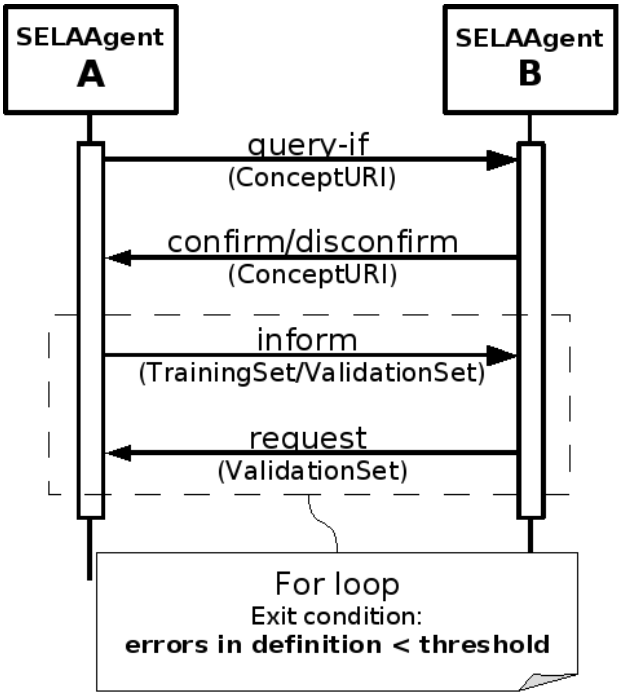

Fig. 1. SELA Alignment Protocol

Here we present the prototype in terms of its architecture and we propose an evaluation on some simple alignment problems that are characteristic (in our opinion) of typical situations that can happen in such settings. We conducted such simple tests in order to single out the direction towards which a mature system should move as well as the practical issues that arise. Therefore, these tests are not to be considered a thorough empirical evaluation, which will follow once robust solutions for the issues described in the following have been devised.

Our prototype is called SELA (Self Explaining and Learning Agents) and consists of a software agent (namely SELAAgent) that has two main tasks: it can either explain concepts, providing examples and validating classifications, or it can learn concepts from examples acquired by some other SELAAgent and ask for validations of what it learned. The agent has been developed within the framework provided by the Java Agent Development Environment (JADE `http://jade.tilab.com`) and the alignment methodology has been designed as a protocol as sketched in Fig. 1.

We can describe it briefly as follows. A SELAAgent A (the teacher) initiates a conversation with another SELAAgent B (the learner) asking whether B knows the concept $a : C$ which

is what A wants to speak about. B can confirm it knows $a : C$ or disconfirm, stating it does not. In case of disconfirmation A provides some individual names that are instances of $a : C$ and some names of counterexamples of $a : C$ (individuals in the extension of the complement of $a : C$). After receiving the names of the instances (both examples and counterexamples) B takes into account only the subset of individuals it owns in its ontology. Then B starts learning as described in the previous Section. Notice that instance descriptions are not to be provided by A, but they are built on the assertions that B has about such individuals. When learning terminates, B has a definition of $a : C$ in terms of its ontology vocabulary. As we shall see in the following, this needs not to be exactly corresponding to the definition of $a : C$ that is actually stored into the ontology of A. B then, has somehow to be sure that its *idea* of $a : C$ corresponds to the one of A. This can be accomplished to some extent by means of the cyclic remainder of the protocol. B chooses some individuals and classifies them employing the definition of $a : C$ it has learned. Then, it sends the classification results to A that validates them. A answers with validation results and then B can fine-tune (see Def. IV) its definition $a : C$. The loop stops when the number of discordances between B classifications and A validations is below a given threshold.

A. Artificial Ontologies

We prepared two sample pairs of ontologies. The first pair is made of two ontologies that are structurally identical but for concept and property names. In fact, we prepared an ontology in the academic domain with an English nomenclature and then its translation in Italian (see Fig. 2 for its layout in Protégé).

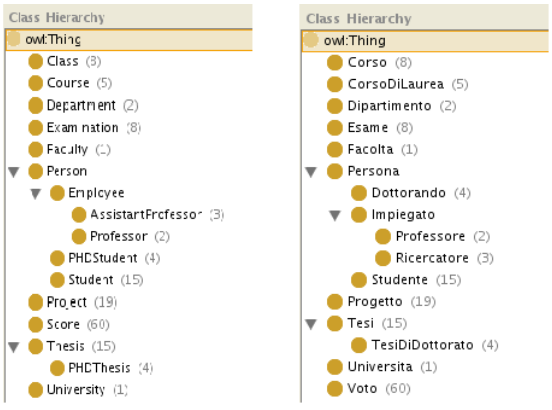

Fig. 2. English and Italian Academy ontologies

The second pair presents a different situation. We started from a common ontology in a fancy restaurants domain that deals with food, meals, and courses. Actually, it is a simplified version of the famous food ontology sample on the W3C website.[4] Then we derived two ontologies containing both the concept called HighQuality defined, however, in two different namespaces (in order to represent two different points of views for it). One deals with HighQuality from the perspective of a particular restaurants defining it as:

$$\mathsf{HighQuality} \equiv \mathsf{Meal} \sqcap ((\exists \mathsf{hasCourse.Starter} \sqcap \exists \mathsf{hasCourse.MainCourse} \sqcap \exists \mathsf{hasCourse.Dessert}) \sqcup \exists \mathsf{hasCourse.}(\exists \mathsf{hasDrink.Alcoholic}))$$

The other one defines its notion of HighQuality from the hypothetical perspective of a dietician as:

$$\mathsf{HighQuality} \equiv \mathsf{Meal} \sqcap (\forall \mathsf{hasCourse.}(\forall \mathsf{hasDrink.}(\neg \mathsf{Alcoholic}) \sqcap \exists \mathsf{hasFood}(\mathsf{LowCaloric} \sqcup \mathsf{NegligibleCaloric} \sqcup \mathsf{ModerateCaloric})))$$

It is obvious that there are two different alignment purposes and situation. Indeed, in the former, the process should detect that there is a complete correspondence among the entities of the two ontologies. In the latter, there is not a 1-1 mapping between the two conceptualizations of HighQuality but it could be interesting if each party (restaurant owner and dietician) could build up, in their respective knowledge base, the point of view of their counterpart during the dialogue.

As concerns the mapping between the two academic ontologies, we report briefly that for each concept the learning SELAAgent was able to build up a definition that was equivalent to the corresponding translated concept in its own ontology, even when a few instances where exchanged. Such a good result depends obviously on the structural similarity of the ontologies: the individuals in common between the two ontologies have the same structure but for the names of the relations; moreover, the A-box contains all the relevant information for the individuals, which in general may not be the case (see Sect. V-C for further details).

In the restaurant domain the results were not as satisfactory as the previous case. Consider the more likely scenario in which the agent with the restaurant owner version of HighQuality tries to learn the dietician's concept of HighQuality. It learns a definition that is more specific than the desired one though it never required tuning in all the iterations we ran through. This is likely due to the limitedness of the number and especially of the variety of examples. It is indeed well known in inductive learning that too few examples not so different among each other may yield a phenomenon called *overfitting*. Overfitting occurs when the learned definition is too adherent to the training set used for building it up and hence suffers for poor predictive accuracy on future unseen examples classification. Future experiments will aim to devise also suitable strategies for choosing examples and, above all, counterexamples. As a matter of fact, a careful selection of counterexamples could improve the effectiveness of learning. Winston [17] claimed the usefulness of *near-misses* counterexamples that are instances that very close to the concept to be learnt but not belonging to its extension. In our case, we could exploit the taxonomy for individuating the most likely near-misses: e.g. the instances of the superconcepts of the concept to be learned (or those of its sibling concepts) that do not

[4] http://www.w3.org/TR/2002/WD-owl-guide-20021104/food.owl

belong to it.

B. Ontology Alignment Evaluation Initiative Ontologies

Our second test is based on two ontologies taken from the Ontology Alignment Evaluation Initiative test set for 2005, namely onto101 and onto206[5]; onto101 is the reference ontology for the contest, which consists on a set of ontologies that are modifications of onto101 (e.g., hierarchy flattening, deletion of concepts, translation of names, URIs, comments, or deletion of comments). In particular, onto206 is the French translation of onto101, where all local names of the defined classes and properties have been translated from English to French. These ontologies consist of 39 named classes, 24 properties and over 70 individuals, of which about 50 are identified by an URI; these individuals have the same URI in both onto101 and onto206, and this enables us to apply our algorithm to them; in particular we assigned onto101 to one of our agents (the teacher) and onto206 to the other agent, then we made the teacher agent try to teach the definition of the `http://oaei.inrialpes.fr/2005/benchmarks/101/onto.rdf\#Institution`[6] concept; however, the resulting definition:

$$\exists \text{adresse}.\text{Adresse} \sqcap \text{Éditeur}$$

is poor w.r.t. the original definition (Fig. 3) and moreover it is overfitting (being Éditeur is not necessary to be Institution); while the second issue only depends on the available individuals (only two individuals of class Éditeur were available in the ontology), the first issue depends on the targeted DL, since the expressivity of the ontologies we use here is greater than that of the language our learner uses (specifically, cardinality restrictions are not handled). This example clearly shows that at least cardinality restrictions should be added to the representation language of the algorithm in order for it to be useful in real world scenarios.

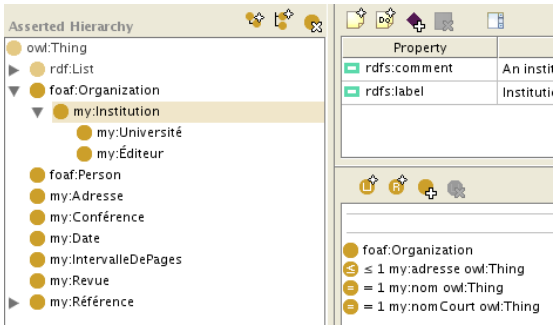

Fig. 3. inria206:Institution Definition

[5]Full test set at `http://oaei.ontologymatching.org/2005/`; as a side note, it is necessary to operate some corrections on the ontologies, since there are some small errors: the literals for year, month and day should be typed with the corresponding XSD types, but they are plain literals in both ontologies, and the DIG reasoner we use (Pellet, `http://www.mindswap.org/2003/pellet/`) finds the ontology inconsistent

[6]Namespaces will be shortened in the following

C. ACM and DBLP Test Case

Our last test case has been built from ACM SIGMOD dataset[7] and from DBLP RDF dump[8]; the ACM dataset is an XML file with associated DTD; in order to translate it into OWL, we designed a very small ontology reflecting the DTD, and then produced the OWL ontology we needed.

In order to find common individuals, we analyzed the available data and found that a good way to identify matching individuals for this setting is to look at the paper titles; this enabled us to choose a subset of the DBLP individuals (amounting to roughly 700 individuals) that were described both in terms of the DBLP ontology and of our homemade ACM SIGMOD ontology (both sketched in Fig. 4[9]; the learning problem here consisted in finding the definition for the concept Article in the ACM SIGMOD ontology; the definition we obtain for Article in the original ACM SIGMOD ontology is:

$$\text{sigmod:Article} \sqcap \exists \text{sigmod:author}.\top$$

while the definition w.r.t. the DBLP ontology is:

$$\text{dblp:Article} \sqcap \exists \text{dblp:url}.\top \sqcap \exists \text{dblp:ee}.\top$$

The definitions appear to be very short; in fact, this depends on the two chosen ontologies being very small, and having mainly datatype properties, which are not useful in the learning algorithm.

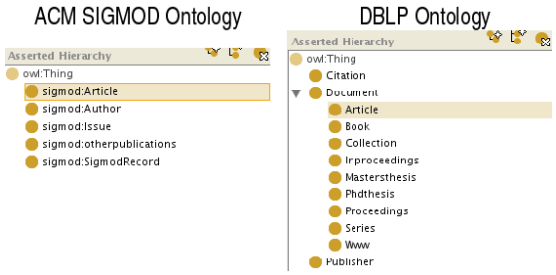

Fig. 4. ACM SIGMOD and DBLP Ontologies

At the time of writing, no quantitative tests have been run on this dataset; the reason is that, being all the individuals in DBLP and ACM SIGMOD almost equal from the structural point of view, a very small amount of examples is enough to converge on the presented definitions. These definitions score 100% precision on the presented datasets, but the reason for this high performance lies in the regularity of the dataset (the actual number of examples used in the tests is 10 positive examples and 5 negative examples out of 700 available individuals). Also, the fact that so many properties in real ontologies

[7]available at `http://www.cs.washington.edu/research/xmldatasets/www/repository.html`

[8]available at `http://www.semanticweb.org/library/`

[9]it is worth noting that the DBLP RDF dump needed a little data cleaning, since it was not completely conformant RDF; in particular, many URIs in the data contained spaces, that needed handling before being submitted to the reasoner

are defined as datatype, even when they in fact represent entities (e.g. dblp:author is a datatype property, while usually an author of a paper is a person, and so is typically modeled as an individual) raises the idea that building more structured datasets is necessary before attempting a complete empirical evaluation of the system.

From the computational complexity point of view, it is well known that OWL DL reasoning algorithms have exponential worst-case complexity[10]; however, most real world ontologies do not exploit all DL constructors, and therefore reasoning with these ontologies can be done in reasonable time. We conducted a very preliminary test on the DBLP dataset presented, where we sliced the available examples in 10 disjoint sets and ran the learning algorithm separately; each learning problem was made up of 50 positive examples (randomly chosen) and 5 negative examples, and the elapsed time for building the definition is around one minute for each one of the ten trials. The last trial was made using all individuals at once, so that there were 500 positive examples and 50 negative examples; in this case, the required time is 80 seconds. So far, then, the number of examples seems not to hamper the practical use of the algorithm.[11]

VI. Conclusions and Future Work

Giving solutions to the ontology alignment problem is one of the most compelling issues in the roadmap to realize the Semantic Web as a real-world infrastructure. We recalled the motivations for the investigation on this subject and underlined our own viewpoint on alignment. In such a vision, we prefer to slightly stray from the most widespread idea of a rigid centralized (top level) ontology and to adopt on-demand partial mappings among ontologies owned by the parties in play. We have proposed a concept learning algorithm to accomplish this that works under assumptions justified throughout the paper. We have illustrated the prototype implemented these ideas together with some preliminary results using limited datasets.

We plan to improve our work along the following directions. First, we should evaluate suitable concept similarity measures for assessing the closeness of learned concept definitions to the pre-existing conceptualizations in the ontologies to be aligned. We will also investigate on methodologies for aligning properties and not only concepts, using techniques that are very similar to those employed in tools like GLUE (see Sect. II). Last, but not least, we will evaluate the impact of different strategies for example selection in terms of the quality (effectiveness and/or efficiency) of the induced definitions and, therefore, of the computed alignment themselves.

References

[1] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, May 2001.

[2] T. R. Gruber, "A translation approach to portable ontology specifications," 1993.

[3] N. F. Noy, "Tools for mapping and merging ontologies." in *Handbook on Ontologies*, ser. International Handbooks on Information Systems, S. Staab and R. Studer, Eds. Springer, 2004, pp. 365–384.

[4] A. Doan, J. Madhavan, P. Domingos, and A. Y. Halevy, "Learning to map between ontologies on the semantic web." in *WWW*, 2002, pp. 662–673.

[5] A. Doan, P. Domingos, and A. Y. Halevy, "Learning to match the schemas of data sources: A multistrategy approach." *Machine Learning*, vol. 50, no. 3, pp. 279–301, 2003.

[6] M. A. M. Natalya F. Noy, "The prompt suite: interactive tools for ontology merging and mapping," *International Journal of Human-Computer Studies*, vol. 59, pp. 983–1024, 2003.

[7] M. Ehrig, S. Staab, and Y. Sure, "Bootstrapping ontology alignment methods with apfel." in *International Semantic Web Conference*, ser. Lecture Notes in Computer Science, Y. Gil, E. Motta, V. R. Benjamins, and M. A. Musen, Eds., vol. 3729. Springer, 2005, pp. 186–200.

[8] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997, ch. Concept Learning and General to Specific Ordering, pp. 20–51.

[9] C. dAmato, N. Fanizzi, and F. Esposito, "A semantic similarity measure for expressive description logics," in *Proceedings of CILC 2005*, 2005.

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

[11] P. R. J. van der Laag and S.-H. Nienhuys-Cheng, "Existence and nonexistence of complete refinement operators." in *ECML*, ser. Lecture Notes in Computer Science, F. Bergadano and L. D. Raedt, Eds., vol. 784. Springer, 1994, pp. 307–322.

[12] S. Nienhuys-Cheng and R. de Wolf, *Foundations of Inductive Logic Programming*, ser. LNAI. Springer, 1997, vol. 1228.

[13] L. Badea and S.-H. Nienhuys-Cheng, "A refinement operator for description logics," in *Proceedings of the 10th International Conference on Inductive Logic Programming*, ser. LNAI, J. Cussens and A. Frisch, Eds., vol. 1866. Springer, 2000, pp. 40–59.

[14] F. Esposito, N. Fanizzi, L. Iannone, I. Palmisano, and G. Semeraro, "Knowledge-intensive induction of terminologies from metadata," in *Proceedings of the 3rd International Semantic Web Conference, ISWC2004*, ser. LNCS, S. A. McIlraith, D. Plexousakis, and F. van Harmelen, Eds., vol. 3298. Springer, 2004, pp. 411–426.

[15] S. Vere, "Multilevel counterfactuals for generalizations of relational concepts and productions," *Artificial Intelligence*, vol. 14, pp. 139–164, 1980.

[16] G. Teege, "A subtraction operation for description logics," in *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning*, P. Torasso, J. Doyle, and E. Sandewall, Eds. Morgan Kaufmann, 1994, pp. 540–550.

[17] P. Winston, *Learning Structural Descriptions from Examples*. MIT, 1970, ph.D. dissertation.

[10] http://www.cs.man.ac.uk/ ezolin/logic/complexity.html

[11] Note that this last test was designed as a ten-fold cross validation experiment, however the results of the test seem too influenced by the specific dataset to be taken into account to measure the performance of the algorithm in terms of precision and recall, and that's the reason they're not presented here in detail.