Large Scale Integration of Senses for the Semantic Web

Jorge Gracia
IIS Department
University of Zaragoza
Zaragoza, Spain
jogracia@unizar.es

Mathieu d'Aquin
Knowledge Media Institute
The Open University
United Kingdom
m.daquin@open.ac.uk

Eduardo Mena
IIS Department
University of Zaragoza
Zaragoza, Spain
emena@unizar.es

ABSTRACT

Nowadays, the increasing amount of semantic data available on the Web leads to a new stage in the potential of Semantic Web applications. However, it also introduces new issues due to the heterogeneity of the available semantic resources. One of the most remarkable is redundancy, that is, the excess of different semantic descriptions, coming from different sources, to describe the same intended meaning.

In this paper, we propose a technique to perform a large scale integration of senses (expressed as ontology terms), in order to cluster the most similar ones, when indexing large amounts of online semantic information. It can dramatically reduce the redundancy problem on the current Semantic Web. In order to make this objective feasible, we have studied the adaptability and scalability of our previous work on sense integration, to be translated to the much larger scenario of the Semantic Web. Our evaluation shows a good behaviour of these techniques when used in large scale experiments, then making feasible the proposed approach.

Categories and Subject Descriptors

H.3 [**Information Systems**]: Information Storage and Retrieval; H.4.m [**Information Systems**]: Miscellaneous

General Terms

Algorithms, Performance

Keywords

Semantic Web, scalable sense integration, ontologies

1. INTRODUCTION

An increasing amount of online ontologies and semantic data is available on the Web, enabling a new generation of intelligent applications that exploit this semantic information as source of knowledge [5]. However, the growing amount of semantic content also leads to an increasing semantic heterogeneity.

Heterogeneity is something inherent to the current (and future) Semantic Web. Far from assuming an "ideal" Semantic Web (under the authority of a few high-quality large ontologies, with maximal coverage and expressivity levels), we advocate to learn how to deal with the "real" one instead, where semantics is constructed by different authors, for very different domains, and with very different levels of expressivity and richness.

WWW 2009, April 20–24, 2009, Madrid, Spain.
ACM 978-1-60558-487-4/09/04.

The redundancy problem. As a direct consequence of heterogeneity in the Semantic Web, the problem of *redundancy* arises. It is characterized by the excess of different semantic descriptions, coming from different sources, to describe the same intended meaning.

For example, if one searches ontological terms to describe the keyword "apple" in the Swoogle semantic web search engine[1], 262 different results are obtained[2]. Each of them represents a possible semantic description for a particular meaning of "apple". Obviously, this number is quite above the real polysemy of the word "apple". For example, WordNet corpus[3] identifies only two possible meanings for this word (the fruit and the tree), while Wikipedia[4] identifies around 20. However, it is still far from the number of possible senses given by Swoogle. If we take a look at the first page of results in Swoogle, the redundancy problem becomes evident (e.g., the meaning of "apple" as a kind of "fruit" appears repeated in five, out of ten, ontological terms).

In order to solve the problem of redundancy, we propose *a method to cluster the ontology terms that one can find on the Semantic Web, according to the meaning that they intend to represent.*

Achieving such a clustering is not only essential for human users to better apprehend the results of semantic web search engines, and generally the content of the Semantic Web, but is also crucial for applications that rely on the knowledge from the Semantic Web (see e.g. [5]). Indeed, eliminating redundancy is a way to improve performance for such applications, but more importantly, knowing which ontological entities refer to the same sense, and which refer to different meanings, makes possible the automatic selection of relevant knowledge in a more accurate and precise way.

Proposed solution. Such an ambitious goal cannot be tackled without relying on an index of the ontological elements accessible through the Semantic Web. For this, our work is supported by the Watson system [4]. Watson is a gateway to the Semantic Web that crawls the Web to find

[1] http://swoogle.umbc.edu
[2] Checked on 23 October 2008, using the exact match mode (localname:apple) for the query.
[3] http://wordnet.princeton.edu
[4] http://en.wikipedia.org

and index available semantic resources, providing a single access point to online semantic information. It also provides efficient services to support application developers in exploiting the Semantic Web voluminous distributed and heterogeneous data.

To tackle the problem of redundancy reduction on the Semantic Web, we define a clustering technique that we apply to the base of ontological terms collected by Watson, and that creates groups of ontological terms having similar meanings. The result is a set of concise *senses* for each keyword one can find in Watson.

For example, a search of "apple" will return all the ontology terms that refer to the meaning "fruit", grouped together as a single integrated sense. Different integration levels are possible, to adapt the "granularity" in senses discrimination to the requirements of client applications, and to the different points of view of users. Let us imagine that a user considers that "Apple" as "electronics label" and "Apple" as "computer company" should be treated differently, while for many others they intend the same meaning. Therefore, it is important to make the integration flexible, preserving the original source of information still accessible, just in case the user is not satisfied with the proposed integration.

To achieve this goal of sense integration we do not start from scratch. As a matter of fact, in [21] we presented a technique that integrates various keyword senses, when they are similar enough. It was developed for a system that analyzes a keyword-based user query in order to automatically extract and make explicit, without ambiguities, its semantics.

However, this technique was initially targeted to a small scale context, the one given by the few keywords involved in a user query. Therefore, an additional goal that motivates this paper is *to adapt these small scale integration techniques to be used in a much more ambitious context: the Semantic Web reachable by the Watson gateway.*

The idea of adapting these integration techniques, to be imbricated in the process of indexing the Semantic Web, leads to another more specific task, that is a scalability study, in order to check the feasibility of these techniques when applied to a much larger body of knowledge.

In this work we have tackled both the redundancy and scalability problems. First we propose a large scale integration method that applies our semantic techniques, in order to reduce the *redundancy* problem on the current Semantic Web, and second we contribute with a scalability study to evaluate the feasibility of our proposal.

The rest of the paper is organized as follows. In Section 2 we present some previous work which serves as background for the present study. In Section 3, our proposal for large scale redundancy reduction is presented. Section 4 focuses on how to optimize the integration level and, in Section 5, we detail the scalability study that we have performed in order to check the suitability of our techniques. Some illustrating examples and further discussion can be found in Section 6. Section 7 reviews some related work and, finally, conclusions and future work appear in Section 8.

2. BACKGROUND

In this section we briefly present some background work that serves as basis for this study.

2.1 Discovering Senses of User Keywords

In a previous work [21] we have developed a system that analyzes a keyword-based user query in order to automatically extract and make explicit its semantics. Firstly, it discovers candidate senses for these keywords among pools of web ontologies (accessed by Watson [4] or Swoogle [8]). Also other local ontologies and lexical resources, as WordNet [15], can be accessed. Secondly, it extracts the ontological context that characterizes each candidate sense, and an alignment and integration step are carried out in order to reduce redundancy. Finally, a disambiguation process is run to pick up the most probable sense for each keyword, according to the context. The result can be eventually used in the construction of a well-defined semantic query (expressed in a formal language) to make explicit the intended meaning of the user [3, 21].

It was not the first technique to match and merge ontological information from different ontologies [7]. Nevertheless, it was quite novel in combining the use of any available pool of online ontologies (without being restricted to a set of predefined ones) as source of knowledge, as well as in the extraction and use of only the relevant portions of semantic descriptions, instead of the whole ontologies (following the recommendation in [1]). Also the absence of pre-processing tasks is a remarkable characteristic of this method.

Among all the different techniques developed in [21], we will focus, in this work, on the alignment and integration services. They give us the tools we need to recognize similarities among ontological terms, and to integrate them when necessary. The latest version of the alignment service receives the name of CIDER (Context and Inference baseD alignER), and it was successfully tested during the Ontology Alignment Evaluation Initiative (OAEI) 2008 [10].

2.2 Watson: A gateway to the Semantic Web

As already mentioned, Watson [4] is a gateway to the Semantic Web. It makes use of a set of dedicated crawlers to collect online semantic content. A range of validation and analysis steps is then performed to extract information from the collected semantic documents, including information about the ontological terms they describe and their relations. This information provides a valuable base of metadata, employed to create indexes of the semantic content currently available on the Web.

Watson provides both a Web interface for human users[5], as well as a set of Web services and APIs for building applications, exploiting the semantic information available on the Web. These services and this interface provide various functionalities for searching and exploring online semantic documents, ontologies, and ontological terms, including keyword search, metadata retrieval, the exploration of ontological term descriptions and formal query facilities.

This combination of mechanisms for searching, retrieving and querying online semantic content provides all the necessary elements enabling applications to select and exploit online semantic resources, without having to download the corresponding ontologies or to identify them at design time. Many applications have already been developed that dynamically exploit online semantic content in an open domain thanks to Watson [5].

[5]http://watson.kmi.open.ac.uk

3. LARGE SCALE INTEGRATION SYSTEM

In this section we summarize our proposed *large scale redundancy reduction technique*. The idea is to carry out an *off-line* process that analyzes the whole set of ontology terms indexed in Watson (classes, properties and individuals), in order to identify clusters that group the terms according to their intended meaning, by exploring how similar they are. These equivalent ontology terms are also integrated, to show a unified semantic description of the meaning they represent.

A second processing step is carried out at run-time, when the system needs to be adapted, in terms of the granularity of the provided clustering, to accommodate the user's view or the application requirements. An overview of the approach is shown in Figure 1, and will be explained in the following subsections.

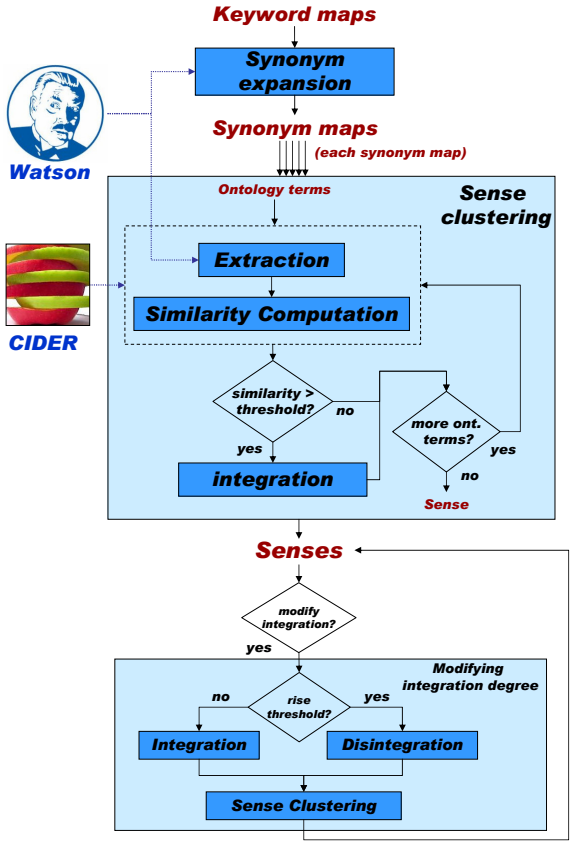

Figure 1: Scheme of the approach

3.1 Preliminary definitions

Let us call C,P,I the sets of all *classes, properties, and individuals*, respectively, indexed by Watson. In the rest of the paper, we will call *ontology term* (and denote *ot*) any element of the union set: $ot \in C \cup P \cup I$.

We will denote OE the set of all *ontological elements* that can be accessed by Watson, considering as elements not only ontology terms, but also any other semantic information that characterizes an ontology: hierarchies, axioms, lexical entries, etc. (we adopt here the characterization of ontology given in [6]).

A *keyword* k will represent, throughout this paper, an element of the set of strings $\mathbb{S}$ (sequences of letters of any length over an alphabet), with any special significance. We will represent the set of all possible keywords as $\mathbb{K}$.

Definition 1. **Watson search**. Let us denote $P(C)$, $P(P)$ and $P(I)$ the σ-algebras formed by the power sets (sets of all subsets) of C, P, I respectively. We define the function *Watson Search* as:

$$wSearch(k) : \mathbb{K} \rightarrow P(C) \cup P(P) \cup P(I)$$

from a keyword k to the set of ontology terms retrieved from Watson for this keyword.

In particular, we employ a dedicated function of the Watson API corresponding to the search of any ontological term that matches the keyword in its label or identifier. The matching that is used is said to be exact, as it would discard any ontological term for which the keyword is only a part of the identifier or label[6].

Definition 2. **Extraction**. Given the power set (set of all subsets) of OE, that we denote $P(OE)$, we define *extraction* as a function

$$ext : C \cup P \cup I \rightarrow P(OE)$$

that, for each ontology term, retrieves its neighbouring ontological elements characterizing its meaning.

We are not being more specific in this definition deliberately, leaving open the particular criteria of what is considered as part of the "neighbourhood" of an ontology term. For our purposes, we use the extraction described in [21, 10].

Definition 3. **Ontological context**. Given an *extraction* function *ext*, we define *ontological context* of an ontology term *ot*, and denote OC_{ot}, as the element of $P(OE)$ such that

$$OC_{ot} = ext(ot)$$

Finally, in the following subsections we will introduce the use of a *similarity measure*. Recalling the definition given in [7], *similarity* is a function from a pair of objects to a real number expressing how similar they are. For our purposes, the compared objects will be subsets of OE, even when, for simplicity, we say "similarity between ontology terms" instead of "between their ontological contexts".

3.2 Keyword maps

Due to the huge amount of available semantic information on the current Semantic Web, it is not computationally feasible to establish comparisons among *all* accessible ontology terms, in order to cluster them. We have conceived, instead, a pre-processing step to reduce the space of comparisons, by establishing an initial grouping of all ontology terms with identical labels. In the following, we call them *keyword maps*.

[6]However, this matching is not case sensitive and relies on simple tokenization techniques to properly match compound terms independently of the employed separators (e.g. "cup-of-tea" matches "CupOfTea").

Definition 4. **Keyword map**. Given a keyword $k \in \mathbb{K}$, and a set of ontology terms $OT \in P(C) \cup P(P) \cup P(I)$, we define the *keyword map* of k as the tuple $\langle k, OT \rangle$, such that $OT = wSearch(k)$

Here, and in the rest of the paper, we will use the term *map* to mean any abstract structure that contains, among other information, an *associative array* (a collection of unique keys and their associated values) of ontology terms, where their URIs act as unique keys.

3.3 Synonym maps

We subsequently enrich these *keyword maps* by including all the possible synonym labels, obtained by the Watson synonymy service[7]. We call these sets of ontology terms *direct synonym maps* (or simply *synonym maps*). More formally:

Definition 5. **Direct synonymy**. We define *direct synonymy* between two different keywords $k_1, k_2 \in \mathbb{K}$, and denote it as $k_1 \equiv k_2$, as a relationship that satisfies:

$$wSearch(k_1) \cap wSearch(k_2) \neq \emptyset$$

That is, different keywords are considered synonyms if they return the same ontology term in a Watson search.

Definition 6. **Direct synonym map**. We define *direct synonym map* as the tuple $\langle SYN, OT \rangle$, where $SYN \subseteq \mathbb{K}$ and $OT \in P(C) \cup P(P) \cup P(I)$, such that

$$\forall k_i \in SYN \ \exists k_j \in SYN \mid k_i \equiv k_j$$

$$\forall ot \in OT \ \exists k \in SYN \mid ot \in wSearch(k)$$

Up to this point, we have grouped all ontology terms corresponding to an equivalent syntactical expression. It is symbolized in Figure 2, where a synonym map is represented, not only grouping the ontological terms that correspond to "apple" but also to any of its direct synonyms. Note that if multilingual ontologies have been accessed, terms in different languages could be provided as synonyms, as for example "manzana"[8]:

$$wSearch(\text{“}manzana\text{”}) \cap wSearch(\text{“}apple\text{”}) \neq \emptyset$$

3.4 Sense clustering

Each execution of this process receives as input a synonym map, containing all the ontology terms that are candidates to describe the meaning of certain keyword. An iterative algorithm takes each ontology term, *extracts* the ontological context that describes the term in the ontology, and computes its *similarity* degree with respect to each of the other terms (and corresponding ontological contexts) in the synonym map, by using the measure described in [10, 21].

The extracted ontological context depends on the type of term, and it includes synonyms, comments, hypernyms, hyponyms, domains, ranges, etc. A reasoner enhances this extraction step, by adding some inferred facts that are not

[7]http://watson.kmi.open.ac.uk/API/term/synonyms

[8]"manzana" is the translation of "apple" in Spanish.

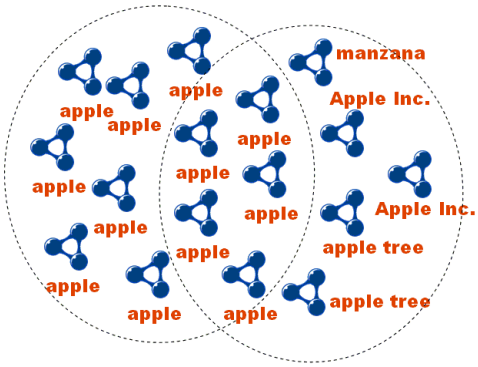

Figure 2: Synonym map: expanding the keyword map with synonyms.

present in the asserted ontology. Although different inference levels can be applied, we use simple inferences based on the transitivity of the subclass relation, due to the fact that it has shown a good balance between quality of results and processing time according to our experiments. Also the access methods in the Watson API provide ways to access classes and subclasses inferred through transitivity (thus saving us much time in local processing and reasoning).

The similarity computation explores the ontological context of the compared terms, providing a measure of how similar they are. When the obtained value is under a given threshold, we consider them as different senses, and the algorithm continues comparing other terms. If, on the contrary, the similarity is high enough, the terms are considered *equivalent* and they are *integrated* into a single sense. The comparison process is then reinitiated among the new sense and the rest of terms in the synonym map. If we find a new equivalence between the new sense and another ontology term, a further integration is done. We can consider it as an *agglomerative clustering* technique [14].

In the following definitions we clarify what we mean by *integration* of ontology terms and *equivalence* between two terms.

Definition 7. **Integration**. Let us call T either C, P or I. Given a certain *extraction* function, we define *integration* of $n \in \mathbb{N}$ ontological terms, and denote $int(ot_1, ..., ot_n)$, as the function

$$int : T^n \rightarrow P(OE)$$

such that, $int(ot_1, ..., ot_n) = OC_{ot_1} \cup \ ... \ \cup \ OC_{ot_n}$

Definition 8. **Equivalence**. Let us call T either C, P or I. Given a similarity measure sim between two sets of ontological elements, and a certain *threshold*, we define *equivalence* between two different ontology terms $ot_1, ot_2 \in T$ with respect to sim, as a relationship, denoted $ot_1 \equiv ot_2$, such that $\exists t \in T^k$ $(k \in \mathbb{N})$ that satisfies one of the following:

$$sim(OC_{ot_1}, int(ot_2, t)) \geq threshold$$

$$sim(int(ot_1, t), OC_{ot_2}) \geq threshold$$

Note that a consequence of the previous definition is that,

$$sim(OC_{ot_1}, OC_{ot_2}) \geq threshold \Rightarrow ot_1 \equiv ot_2$$

(in case $t = \emptyset$). That is, two ontology terms are equivalent if their contexts are similar enough, or if one of them belongs to an *integration* which is similar enough to the other. In general, we use $sim(ot_1, ot_2)$ as an equivalent notation for $sim(OC_{ot_1}, OC_{ot_2})$.

The output of this process is a set of integrated senses. Each sense groups the ontology terms that correspond to the same intended meaning, as exemplified in Figure 3. We will store the obtained integrated senses in the output structures that we define in the following:

Definition 9. **Sense map**. We define *sense map* as the tuple $\langle SYN, OT \rangle$, where $SYN \subseteq \mathbb{K}$, and $OT \in P(C) \cup P(P) \cup P(I)$ such that,

$$\forall ot_i, ot_j \in OT \Rightarrow ot_i \equiv ot_j$$

$$\forall k \in SYN \ \exists ot \in OT \mid ot \in wSearch(k)$$

Sense maps group together ontological terms that, according to their similarity, are expected to represent the same meaning. As we have to deal with their integrated ontological information, they are part of the more complete definition of *sense*. The following definition accommodates (and slightly extends) the idea of *sense* presented in [21]:

Definition 10. **Sense**. We define a *sense* as the tuple $\langle SYN, OT, int(OT), descr, pop, syndgr \rangle$, where $\langle SYN, OT \rangle$ is a sense map, $int(OT)$ is the integration of the involved ontology terms, including also the graph that describes topologically the integration, *descr* is a textual description, *pop* (popularity) is the number of integrated ontology terms: $pop = |OT|$, and *syndgr* is a value corresponding the resultant similarity degree of the integration.

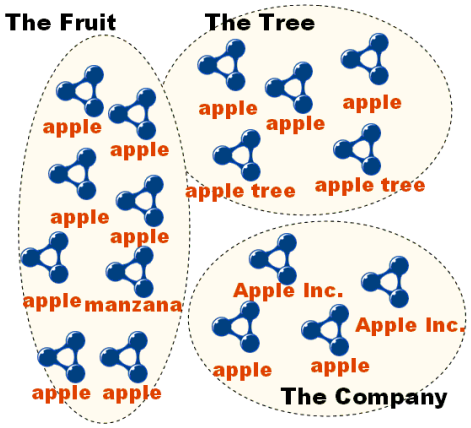

Figure 3: Sense maps obtained for "apple"

Note that, by definition, a sense map only contains ontology terms of the same type. Therefore classes, properties and individuals are treated separately, thus never trying to integrate ontology terms of different nature.

Our process never "destroys" semantic information because, even when the final *sense* includes an *integration*, containing only parts of the involved ontologies, it also preserves all original ontological terms in a *sense map*. It makes possible that, when needed, a particular application traverses the sense map to reach further information from the original ontologies.

This clustering process is repeated with the rest of the synonym maps, to create eventually *a pool of integrated senses which covers all ontology terms in Watson indexes*.

3.5 Modifying integration degree

As commented before, a threshold is given as input of our system, to decide whether to integrate or not two senses. Later, in Section 4, we explore possible ways to decide an "optimal" threshold.

However, a fixed threshold leads to a particular integration scheme. That is, the obtained clusters represent what our application interprets as optimal integration, but may differ from other opinions or necessities. Therefore, we consider that different integration levels should be possible, to adapt the "granularity" in senses discrimination to the requirements of client applications, or to the different points of view of users.

We propose a subsequent run-time process that further integrates or disintegrates the initial clusters, as represented in Figure 4. Without entering into details, the idea is that, given a new threshold, and given the set of *senses* associated to a certain keyword, the system have to explore how to modify the integration, according to the new threshold.

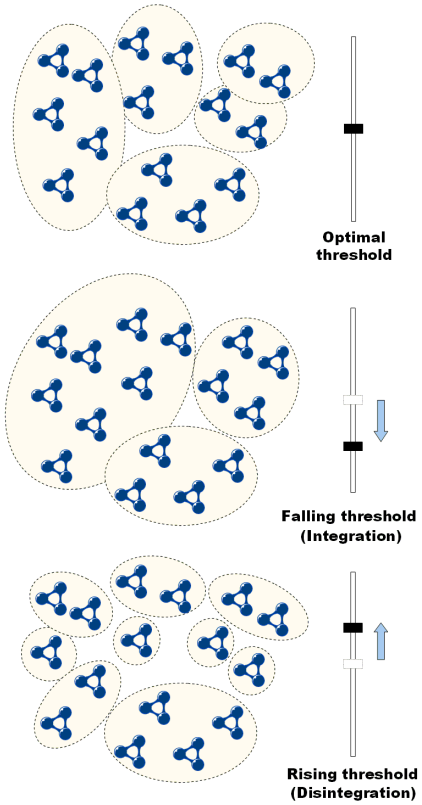

Figure 4: Integration/disintegration example.

A rising threshold ($newthreshold > threshold$) means a lower integration level. Each considered sense s is disintegrated, and the sense clustering algorithm is applied again to its ontology terms $s|_{OT}$. More than one new sense can be created during the process, covering the meaning that the initial s represented.

A falling threshold ($newthreshold < threshold$) means a higher integration level. In this case all senses are candidate for further integration, so the clustering algorithm is applied again on all the senses (not necessarily on their elemental ontology terms, as in the disintegration case). A number of senses equal or lower than the initial set is created.

4. OPTIMIZATION STUDY OF THE SIMILARITY THRESHOLD

As mentioned before, a threshold is used to decide whether two terms are similar enough or not. However, one question arises: What is the best threshold value to identify a correct synonymy between ontology terms? We have different ways to decide it:

1. Experimenting with ontology matching benchmarks. That is, running our similarity measure with known alignment test cases, and comparing results with the provided reference alignments.

2. Contrasting with human opinion, in experiments where direct or indirectly some human evaluators assess the correctness of our measure.

3. Optimizing response time. As different thresholds lead to different integration degrees, we expect different execution times as well. We can try to discover a threshold that minimizes response time.

In the rest of the section, we explore all these possibilities and provide a method to decide the optimal threshold, which relies on the latter point (that is, it prioritizes time response).

4.1 Finding threshold from OAEI benchmarks

Regarding a comparison to benchmarks, we submitted CIDER system (our alignment service) to the OAEI'08 competition[9], in order to evaluate the capabilities of our semantic similarity measure for ontology matching tasks [10]. We participated in benchmark and directory tracks, obtaining good results in both of them. As benchmark track was open (organizers provided the reference alignment), we were able to deduce the threshold for the similarity measure (around 0,13 in an interval [0,1]) that lead to the best results.

Nevertheless, we do not trust this value very much, as the nature of the comparisons in the benchmark data sets is quite restricted[10], not representing the great variety of real semantic descriptions one can find on the Semantic Web. In general, we expect higher thresholds when integrate senses because, with benchmark test cases, it is highly probable that minimal similar information, as for example same labels, assures that two terms are equivalent (as ontologies are very similar). However, equal labels are not enough, in general, to assess equivalent meanings when more heterogeneous semantic data are involved. Therefore, we accept the obtained threshold, but only as a lower bound for an optimal one, to be used in a more general scenario.

[9]http://oaei.ontologymatching.org/2008

[10]Most comparisons are between an ontology and itself, with some modification rules applied, and always in the bibliographic domain.

4.2 Finding threshold from human assessment

Another possibility, as it was above mentioned, is to devise an experiment where human evaluation is present to assess the quality of the matching that our measure provides. We already did such an experiment in [9], where a method to improve a background-based ontology matching technique was proposed. Without entering into further details (see [9]), our measure was used to add a confidence level to 354 mappings initially provided for the background-based technique, between two ontologies of the agricultural domain. We applied different thresholds to this confidence level, to decide the validity of the mappings, and we compared this to an assessment made by human observers.

A particular optimum threshold did not arise from this experiment, depending the particular choice on the balance between precision and recall one needs. However, a range of reasonable good values was found between 0.2 and 0.3.

4.3 Finding threshold from performance optimization

We have measured the variation of the response time of the system, during alignment and integration, when the threshold is modified. Our goal is to discover in which ranges our integration works better, in terms of performance. In fact, we expect different response time depending on the threshold, because the number (and size) of clusters differs with the integration level, therefore consuming different time when comparing and merging them during our iterative integration algorithm.

For this purpose we have set up an small experiment with 505 randomly selected keywords, with a number of associated ontology terms ranging from 2 to 511, which were used as input of our alignment and integration system. We have experimented with different levels of integration (that is, corresponding to different thresholds). We evaluated the *integration level* as

$$integration\ level = 1 - \frac{\#\ final\ integrated\ senses}{\#\ initial\ ontology\ terms} \quad (1)$$

The computer used for this test had the following characteristics: 4 x Intel® Core™2 Quad CPU Q6600 at 2.40GHz, 8GB RAM (with Ubuntu 8.04 Linux).

Figure 5 shows the averaged integration levels obtained. As expected, the higher the threshold, the lower the integration level, which is consistent with the discussion in Section 3.5. In Figure 6, the average time that our similarity computation and integration processes took in the experiment, for different thresholds, is shown.

Analyzing the results, we can see two local maximum values in the graphic, corresponding to the highest and the lowest integration levels respectively. In fact, each iteration, in our integration algorithm, needs to compare a new ontology term to the others already examined. If there are no (or few) clusters, this number of comparisons could be high, thus consuming more time. On the contrary, if there is a high integration level, the time is spent merging ontological data, as well as comparing each new integration with the rest

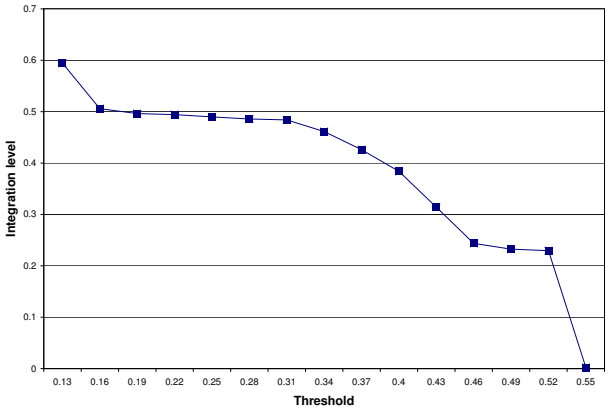

Figure 5: Integration level with different threshold.

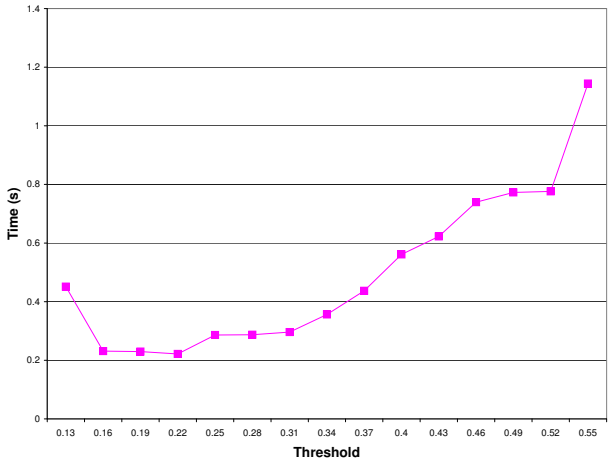

Figure 6: Response time with respect to threshold.

of the ontology terms again. The results show that the best performance corresponds to an integration level around 0.5. In particular, we found the best performance for a threshold value of 0.22 in this experiment.

4.4 Proposed optimization method

As conclusion of our experience, described in previous paragraphs, we advocate for an optimal threshold established to maximise performance. The main reasons are:

1. It will reduce the response time of the overall system. Figure 6 shows how an optimal threshold selection can easily double or triple the speed with respect to other thresholds.

2. The optimal threshold we can find this way (Section 4.3), is compatible with the range of good ones found by comparing the results with human evaluations (Section 4.2).

3. It is not always feasible to have a large enough number of humans, or reference alignments, in order to assess the behaviour of a similarity measure. On the contrary the performance-based method is completely automatic, not needing human supervision.

4. Finally, as explained in Section 3.5, the ideal integration level is dependent on the user's view and on the requirements of particular applications. Therefore, it is reasonable to chose an initial integration that can be established quickly, letting the user, or client application, chose later how it should be modified to fit his particular needs.

Therefore, we propose to automatically train the system with a large number of test cases, analyzing the response time with respect to the threshold. Then, we deduce the threshold that minimize the time value, that will be proposed as the optimal threshold of our system.

Finally, the information of subsequent "manual" tuning of the threshold can be stored, and utilized to improve the initial threshold by using machine learning schemes.

5. SCALABILITY STUDY

In this section we explore the scalability of the techniques we presented in [21], when used to align and integrate ontological information extracted from Watson indexes. This way, we can check whether the scenario described in Section 3 is feasible or not.

5.1 Empirical analysis of costs

The method developed in [21] to integrate senses of user keywords, was conceived initially for a context of use where only a reduced set of keywords (corresponding to a user query) were involved. For this case, the set of tests we run showed a good behaviour of the method, in terms of performance. However, the new goal we describe in this paper is much more ambitious, as the context of application grows enormously in scale. Therefore, a scalability study is needed in order to figure out whether our integration algorithm is applicable to a large scale context (the whole set of online ontologies indexed by Watson) or not.

Generally speaking, we say that a system is scalable when the cost per unit of output remains relatively constant with proportional changes in the number of units (or size) of the inputs. There are two ways of analyzing the cost in time of an algorithm: theoretically or empirically. We have chosen the second way, because we already dispose of an advanced implementation of the system. To empirically study the response time of the system, we need to run it with a statistically significant sample of input data. Then, we will infer the performance of the system from the obtained results.

Experimental setup. For these experiments, we used the set of synonym maps that Watson indexes hosted on June 2008, from which we randomly selected a 10%, taking a representative keyword from each map to be used as input for our experiment, resulting in an initial number of 28800. After removing the keywords with only one ontology term associated (it makes no sense to apply our integration process on them)[11], we counted 9156 different keywords, associated to 73169 different ontology terms to be clustered in this experiment. We used a threshold = 0.27 during our sense clustering step.

[11]A small number of keywords that could not been processed, due to different technical reasons, was also removed (e.g., not available in Watson indexes at the time the test was run).

The computer utilized for this tests had these characteristics: 2 x Intel® Xeon® CPU X5355 at 2.66GHz, 6GB RAM (with Red Hat Enterprise Linux 5).

5.2 Studying the size of keyword maps

The target of this experiment is to study the response in time of our system, with respect to the number of ontology terms per keyword that have to be processed during the clustering step. We expect that the greater the number of ontology terms associated to a keyword, the longer it takes to be integrated. Our objective is to figure out whether this time is always limited and controllable, even for a high amount of terms, or not. It is something hard to predict in advance, due to the great variability of input data, in terms of the quantity and quality of semantic information.

In Figure 7 we see how final results distribute. It shows the total response time per number of ontology terms in each *keyword map*. Each point corresponds to a test case (a total of 9156 cases were run, one per different input keyword). We have adjusted the given results to different curves, finding that the best fit is a *linear regression* with equation $y = 367x$, and correlation coefficient $R = 0.97$. If we consider the distribution of time responses, the *mean* value of all cases is 2.5 seconds, and the *median* value is 0.5 seconds

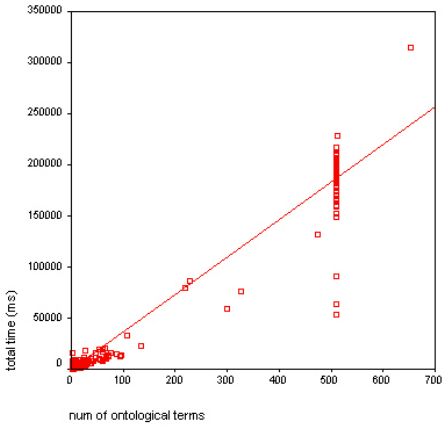

Figure 7: Time vs. number of initial ontology terms per keyword.

From the given results we observe:

1. 99% of cases corresponds to keywords with less than 100 associated ontology terms. In all these cases the system's response takes no more than a few seconds, thus confirming its capacity of being used for real-time purposes (as it was first conceived in [21]).

2. The expected result time is always controlled and can be approximately predicted for any value of the input, as the ratio *processing time/num. ontological terms* remains constant.

As our alignment and integration algorithms show a controllable behaviour, we can focus more on technical optimization issues to improve performance (parallelization, cache schemes, etc.) when the real application scenario requires it (e.g., it is expected that Watson indexes continue growing with time).

5.3 Studying the size of ontologies

As an additional result of our scalability study, we were interested in confirming that, due to the use of the Watson inverted indexes, the size of the accessed ontologies has not effect on the response time of our techniques. Our results confirmed it, as it is shown by the cloud shaped dispersion in Figure 8 (zoomed for convenience), where the time result according to the size of the involved ontologies appears. After having statistically analyzed the results, we have not found any dependence between ontology size and response time of our system.

This independence between the size of ontologies and the response time of our system, even if expected, shows a remarkable advantage of accessing the ontological information by using Watson, instead of other sources that requiere a preliminary load step before using the ontologies. In that cases, the time taken for downloading (and querying) the ontologies depends on their size.

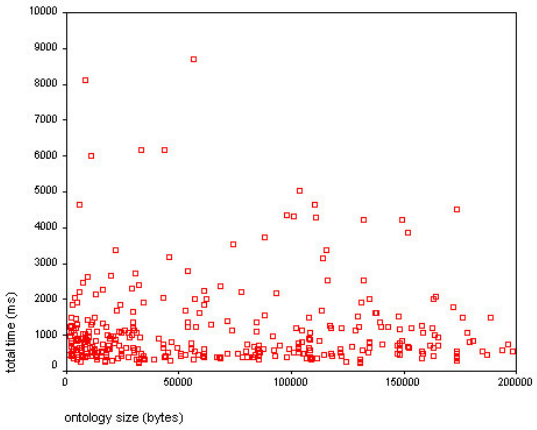

Figure 8: Time vs. size of accessed ontologies.

6. ILLUSTRATING EXAMPLES

The main target of our test series has been to study the scalability and performance of our integration techniques. In this section we want to give also an idea of the quality of the results, by inspecting some selected clustering examples. A larger human-based evaluation is pending as future work. Meanwhile, we have inspected the results given with these polysemic words: "turkey", "plant", and our motivating example "apple". Due to space limitations, we only detail the first example, summarizing the results of the other two.

Case 1: *Turkey*. After synonymy expansion, the synonym map $\langle SYN, OT \rangle$ contained the keywords $SYN = \{$*Turkey, Türkei, Türkiye*$\}$, and a set of 58 ontology terms: $|OT| = 58$. Our sense clustering step was carried out with *threshold* $= 0.27$. The system grouped the initial 58 ontology terms into 9 different senses (*integration level* $= 0.84$, according to Equation 1). Table 1 summarizes some aspects of the integrated senses[12].

[12]Ontologies mentioned in Table 1 can be found, respectively, at http://islab.hanyang.ac.kr/damls/Country.daml, http://139.91.183.30:9090/RDF/VRP/Examples/tap.rdf, http://reliant.teknowledge.com/DAML/Economy.daml,

#	type	pop	syndgr	comments
1	class	1	0	It appears as "a country" in Country.daml ontology.
2	class	1	0	In tap.rdf ontology (however its ontological context does not clarify its meaning).
3	class	2	0.48	Integration of "turkey" as "a livestock" from the Economy.daml and Economy.owl ontologies.
4	class	2	0.28	Integration of "turkey" as "a bird" from the ontosem.owl and mesh.owl ontologies.
5	individual	12	0.61	It integrates many instances of "turkey" as "light meal".
6	individual	34	0.42	It integrates many instances of "turkey" from annotated conversations in social webs, most of them referring to meanings similar to "place", "country", "area".
7	individual	3	0.62	Integration of "turkey" as an instance of "a place", from different versions of epitaph.rdf ontology.
8	individual	2	0.41	It integrates "turkey" as an instance of "document" in annotated blogs.
9	individual	1	0	It comes from a test ontology, being an instance of "Foo".

Table 1: Summary of obtained senses for "turkey".

In order to experiment with different integration levels, we decreased the threshold to 0.17, resulting in an *integration level* = 0.88. In particular senses 6 and 7 were integrated into a single one (with *pop* = 37, *syndgr* = 0.38), as well as senses 3 and 4 (*pop* = 4, *syndgr* = 0.38). The others remained the same. On the contrary, if we increase the threshold to 0.37, the number of obtained senses rises to 12 (*integration level* decreases to 0.79). Particularly, sense 4 is split in two, and sense 5 is split in three new senses.

Case 2: *Plant*. The obtained synonymy expansion was {*plant, plants*}. Watson retrieved 52 ontology terms for these keywords. After running our clustering step, with *threshold* = 0.17, 16 final senses were obtained: 14 classes and two individuals (*integration level* = 0.73). The most remarkable one was a class that integrated 32 ontology terms, all with the meaning of "plant" as a kind of "living organism".

Case 3: *Apple*. Watson retrieved initially 38 terms for this keyword. The synonym expansion did not add significant synonyms. Our sense clustering step, with *threshold* = 0.27, grouped the initial ontology terms into nine different senses (*integration level* = 0.76): five individuals and four classes. Among the possible senses, we find "apple" as "a fruit", as an "ingredient", as an annotation tag in Flickr[13], and as an instance of "document" in many annotated blogs (most of them refer to the meaning of "the electronics company").

Discussion. Our examples show the ability of our system to reduce the number of repeated senses one can discover on the Semantic Web, to represent the meaning of a keyword. It is specially useful to cluster terms form different versions of the same ontology. Also references in social tags, where semantics is quite poor, are easily grouped. We have also found that very different meanings are not wrongly mixed easily (e.g., "turkey" as "country" with "turkey" as "food").

On the other hand, it is hard to obtain a total integration of *all* terms of the same type referring to the same meaning. It is due to the very different semantic descriptions sometimes used to model the same concepts. We have also found that some expected meanings do not appear among our results (e.g., "apple" as "a tree"). It is a consequence of the still limited coverage of the current Semantic Web.

http://reliant.teknowledge.com/DAML/Economy.owl, http://morpheus.cs.umbc.edu/aks1/ontosem.owl, http://www.berkeleybop.org/ontologies/obo-all/mesh/mesh.owl, and http://139.91.183.30:9090/RDF/VRP/Examples/epitaph.rdf

[13]http://flickr.com/

7. RELATED WORK

Our proposal represents the first large-scale effort, so far, to provide a pool of clusters of cross-ontology terms free of redundancy. Therefore, direct comparison with other alternatives is quite difficult. Clusty[14], for example, is a metasearch engine that groups search results in clusters that group pages about a similar topic. The general idea could be equivalent, but our search domain is the Semantic Web instead of the Web, and we are interested in clustering senses instead of web pages. Regarding other systems that index online semantic content [17, 8, 11], neither of them have incorporated redundancy reduction techniques yet.

Technologically speaking, even though we reuse ontology matching and integration techniques, our work is neither solely about ontology matching [7, 6], nor about ontology integration or merging [18, 13, 12], in the sense of *obtaining a new ontology from matched ontologies* [7]. It is closer, however, to the idea of ontology construction from online ontologies given in [1]. In his work, Alani proposes a general strategy to dynamically segment, map and merge online ontologies in order to support ontology construction. Our target is not the same, but we share some commonalities, and we think that our proposal could indeed support and benefit such kind of systems.

Regarding our particular techniques, notice that our definitions of *extraction*, *ontological context* and *equivalence*, given in Section 3, are general enough to accommodate other similarity measures, always that they come with a clear notion of *neighbourhood*, as for example in [19] (where they propose the idea of virtual documents, containing neighbouring information, to describe the intended meaning of the ontology terms). Also, other segmentation techniques [20, 16, 2] could substitute the extraction we use. However, the main target of these techniques is usually to reduce the size of large ontologies, to make them treatable and scalable, instead of characterizing a single sense. Furthermore, the good results given by the CIDER alignment service, when compared to others in the OAEI contest [10], confirm the

[14]http://www.clusty.com

validity of its extraction and similarity computation, when applied to ontology matching tasks.

8. CONCLUSIONS AND FUTURE WORK

We have proposed a method to perform a large scale integration of ontology terms, to cluster the most similar ones into integrated senses, when indexing huge amounts of online semantic information. The method is flexible enough to dynamically modify the integration level according to the user point of view, by tuning the similarity threshold. We have also explored possible methods to find an optimal integration level, finally proposing one that is based on performance optimization.

The work presented here is quite innovative in some aspects. For example, it is the first large-scale effort, so far, to provide a pool of clusters of cross-ontology terms free of redundancy (to a certain extent).

A scalability study has been performed in order to investigate the feasibility of our proposal. The results show that our techniques lead to processing times that are controllable and predictable, thus making scalability possible.

As future work, we plan to complete the implementation of the system, only partially covered by our current prototype. It will be provided as a service in Watson eventually. In this way, many applications, that exploit Watson to dynamically access online semantic content, will take advantage of our method.

We plan also to perform a large scale test with human opinion, to judge the quality of the obtained senses.

Acknowledgments. We thank Jordi Bernad for his valuable comments on our mathematical formalization. This work is supported by the CICYT project TIN2007-68091-C02-02, and the Government of Aragon-CAI grant IT17/08. Mathieu d'Aquin is partly funded by the NeOn project sponsored by the European Commission as part of the Information Society Technologies (IST) programme.

9. REFERENCES

[1] H. Alani. Position paper: Ontology construction from online ontologies. In *Proc. of 15th International World Wide Web Conference (WWW2006), Edinburgh, UK*, May 2006.

[2] M. Bhatt, C. Wouters, A. Flahive, W. Rahayu, and D. Taniar. Semantic completeness in sub-ontology extraction using distributed methods. In *In Proc. Int. Conf. on Computational Science and its Applications (ICCSA), Assisi, Italy*, May 2004.

[3] C. Bobed, R. Trillo, E. Mena, and J. Bernad. Semantic discovery of the user intended query in a selectable target query language. In *Proc. of 7th International Conference on Web Intelligence (WI 2008), Sydney, Australia*, December 2008.

[4] M. d'Aquin, C. Baldassarre, L. Gridinoc, S. Angeletou, M. Sabou, and E. Motta. Characterizing knowledge on the semantic web with Watson. In *Proc. of 5th International EON Workshop, at ISWC'07, Busan, Korea*, 2007.

[5] M. d'Aquin, E. Motta, M. Sabou, S. Angeletou, L. Gridinoc, V. Lopez, and D. Guidi. Toward a new generation of semantic web applications. *IEEE Intelligent Systems*, 23(3):20–28, 2008.

[6] M. Ehrig. *Ontology Alignment: Bridging the Semantic Gap (Semantic Web and Beyond)*. Springer-Verlag New York, Inc., 2006.

[7] J. Euzenat and P. Shvaiko. *Ontology matching*. Springer-Verlag, 2007.

[8] T. Finin, L. Ding, R. Pan, A. Joshi, P. Kolari, A. Java, and Y. Peng. Swoogle: Searching for knowledge on the Semantic Web. In *Proc. of AAAI'05 (intelligent systems demo)*, July 2005.

[9] J. Gracia, V. Lopez, M. d'Aquin, M. Sabou, E. Motta, and E. Mena. Solving semantic ambiguity to improve semantic web based ontology matching. In *Proc. of 2nd Ontology Matching Workshop (OM'07), at ISWC'07, Busan, Korea*, November 2007.

[10] J. Gracia and E. Mena. Ontology matching with CIDER: Evaluation report for the OAEI 2008. In *Proc. of 3rd Ontology Matching Workshop (OM'08), at ISWC'08, Karlsruhe, Germany*, October 2008.

[11] A. Harth, J. Umbrich, and S. Decker. Multicrawler: A pipelined architecture for crawling and indexing semantic web data. In *Proc. of 5th International Semantic Web Conference (ISWC'06), Athens, GA, USA*. Springer, 2006.

[12] P. Hitzler, M. Krötzsch, M. Ehrig, and Y. Sure. What is ontology merging? - a categorytheoretic perspective using pushouts. In *Proc. of First International Workshop on Contexts and Ontologies: Theory, Practice and Applications, at AAAI'05, Pittsburgh, Pennsylvania, USA*, July 2005.

[13] C. M. Keet. Aspects of ontology integration, 2004. Technical Report, Napier University, Edinburgh, Scotland.

[14] B. Liu. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data*. Springer-Verlag, 2007.

[15] G. Miller. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[16] N. F. Noy and M. A. Musen. Specifying ontology views by traversal. In *Proceedings of International Semantic Web Conference (ISWC'04), Hiroshima, Japan*, November 2004.

[17] E. Oren, R. Delbru, M. Catasta, R. Cyganiak, H. Stenzhorn, and G. Tummarello. Sindice.com: A document-oriented lookup index for open linked data. *International Journal of Metadata, Semantics and Ontologies*, 3(1):37–52, 2008.

[18] H. S. Pinto, A. Gómez-Pérez, and J. P. Martins. Some Issues on Ontology Integration. In *Proc. of the IJCAI Workshop on Ontologies and Problem-Solving Methods, Stockholm, Sweden*, August 1999.

[19] Y. Qu, W. Hu, and G. Cheng. Constructing virtual documents for ontology matching. In *Proc. of 15th International World Wide Web Conference (WWW'06), Edinburgh, UK*, May 2006.

[20] J. Seidenberg and A. Rector. Web ontology segmentation: analysis, classification and use. In *Proc. of the 15th international conference on World Wide Web (WWW'06), Edinburgh, UK*, May 2006.

[21] R. Trillo, J. Gracia, M. Espinoza, and E. Mena. Discovering the semantics of user keywords. *Journal on Universal Computer Science (JUCS). Special Issue: Ontologies and their Applications*, 13(12):1908–1935, December 2007.