UNIVERSITY
OF TRENTO

**DIPARTIMENTO DI INGEGNERIA E SCIENZA DELL'INFORMAZIONE**

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.disi.unitn.it

# D3.7 SUMMATIVE REPORT ON MATCHING IMPLEMENTATION AND BENCHMARKING RESULTS

Lorenzino Vaccari, Juan Pane, Pavel Shvaiko, Maurizio Marchese,
Fausto Giunchiglia, Paolo Besana and Fiona McNeill

February 2009

Technical Report # DISI-09-017

OpenKnowledge

FP6-027253

# D3.7 Summative report on matching implementation and benchmarking results

Lorenzino Vaccari[1], Juan Pane[1], Pavel Shvaiko [1], Maurizio Marchese[1], Fausto Giunchiglia [1], Paolo Besana[3],Fiona McNeill[3]

[1] University of Trento
{vaccari;pane;pavel;marchese,fausto}@disi.unitn.it
[2] University of Edinburgh
p.besana@sms.ed.ac.uk;f.j.mcneill@ed.ac.uk

Report Version: Final
Report Preparation Date: 15/11/08
Classification: deliverable 3.7
Contract Start Date: 1.1.2006 Duration: 36 months
Project Co-ordinator: University of Edinburgh (David Robertson)
Partners: IIIA(CSIC) Barcelona
Vrije Universiteit Amsterdam
University of Edinburgh
KMI, Open University
University of Southampton
University of Trento

**Abstract.** In this deliverable we report on the work carried out for the evaluation and benchmarking of the matching component of the OpenKnowldge kernel. The main goal of the present work has been a extensive evaluation of the approximate Structure-Preserving Semantic Matching (SPSM) algorithm both on a set of synthesized benchmarks, as well as on a set of real world GIS ESRI ArcWeb services. The results demonstrate the robustness and the performance of the SPSM approach on a large number (ca 700.000) of matching tasks. Moreover, the SPSM approach is capable of reproducing similar results to state-of-the-art matchers when only syntactic variation are considered, while outperforming them when semantic variations are applied
The rest of the deliverable is organized as follows: (1) a brief introduction of the goals and main results of the deliverable and (2) the collection of two papers that describe the approach, the related work and discuss the results so far achieved.

# 1 Introduction

## 1.1 SPSM algorithm and specificity

The matching in the OpenKnowledge project is done by the Structure-Preserving Semantic Matching (SPSM) algorithm presented in detail in the previous deliverables (see in particular D3.4 and D3.6). To briefly summarize it, unlike standard ontology matching (see, for example, which focusses on matching flat terms within a structured ontology, SPSM is concerned with matching structured terms: specifically, first-order terms.

This is essential in the OpenKnowledge kernel because the purpose of the matching is to map between the abilities required in a role, represented by the first-order constraints in the LCC, and the abilities of the service or peer playing that role. The abilities of the service may not naturally be expressed as first-order terms but in most cases it is possible to translate between the service inputs and outputs and a first-order term. We have already created a translation process from WSDL services to first-order terms. These first-order terms then represent the service within its OK Component (OKC), and matching between a service and a role becomes a matter of matching between first-order terms: those of the service abilities in the OKC and those of the role requirements, which are the constraints in the IM.

The SPSM algorithm is a two-step process, consisting of $(i)$ node matching and $(ii)$ tree matching. The node matching step considers the individual words in the terms to be matched. For example, when matching the term $buy(car, price)$ to the term $purchase(price, vehicle, number)$, the node matching step would look for similarities between members of the set $[buy, car, price]$ and the set $[purchase, price, vehicle, number]$.

This step is done using conventional ontology matching methods, particularly through the use of S-Match (see Deliverable D3.4). LCC allows for annotations of terms, meaning that these terms may be annotated with the ontology from which they are drawn, which can then be used in matching. If this is not the case

then methods such as consulting WordNet are used. The tree matching step uses the results of the node matching step to consider the global similarities between the trees by preserving a set of structural properties. The algorithm for this part of the matching is based on a tree-edit distance algorithm, augmented using the theory of abstraction to ensure the matches remain semantically meaningful. For the node matching step, we are able to rely on a large body of existing work but there has been little work done on semantically meaningful tree matching and our approach to this problem is novel.

The SPSM algorithm searches not only for perfect matches, which in an unconstrained system we would expect to encounter only rarely, but also for *good enough* matches, where by good enough we mean that the similarity of the terms is above a certain threshold. The algorithm therefore returns a numerical score in [0 1] indicating the similarity of the terms, which can be compared to the threshold value for the IM. Those exceeding the threshold are considered *good enough*; otherwise the terms are considered not to be similar. The level of the threshold will depend on the type of interaction and the desires of the users.

## 2 Evaluation Goals

Matching in OpenKnowledge has two purposes:

1. To allow peers with abilities that are not identical to the required abilities for a role to understand how they may satisfy the constraints on that role. This is done through building up a map between each element of the peer's ability to each element of a constraint. In the case of non-perfect matching, there may be elements in either the ability or the constraint that remain unmatched, and the matches that do exist may not be between things that are semantically identical. Nevertheless this map enables the peer to use its own abilities to satisfy constraints to the highest degree possible.
2. To allow peers to determine how similar their own abilities are to those required of the role, by considering how close the numerical similarity returned is to a perfect score (1). This judgement can be used:
   (a) to enable a peer to decide whether it wishes to play that particular role, or whether it should look for a similar role in a different IM where the abilities may match better. If a peer opts to play a role for which it is has poor matching, it is likely to either fail completely in that role or else to produce a low-quality performance. This is both a waste of the peer's own time and resources and will lead to a lowering of other peers' trust in it.
   (b) in the case that there are several peers subscribed to play a single role, to assist other peers in judging which of those peers they wish to play with. In the OpenKnowledge kernel, this is done through the combination of the matching score reported by the subscribed peers and the trust score that the choosing peer has in each of them - *good enough answers*. The choosing peer cannot actually verify that the matching score reported is correct, as this can only be calculated through complete observation

of the peer's method of performing a service, which is not public, so it must rely on the veracity of the subscribed peers. However, peers that persistently exaggerate their matching scores will end up with a lower trust score and therefore not be chosen.

The evaluation of the performance of the proposed matching implementation has been the focus of this final deliverable of workpackage 3. To this end, the evaluation of the SPSM algorithm have been carried out in various settings, namely:

1. **Test case 1: real-world ontologies**. We used different versions of the Standard Upper Merged Ontology (SUMO) and the Advance Knowledge Transfer (AKT) ontologies. We extracted all the differences between versions 1.50 and 1.51, and between versions 1.51 and 1.52 of the SUMO ontology and between versions 1 and 2.1, and 2.1 and 2.2 of the AKT-portal and AKT-support ontologies. These are all first-order ontologies (hence, their expressivity is far beyond generalization/specialization hierarchies), so many of these differences matched well to the potential differences between terms that we are investigating. The approach, as well as the tests and the results have been presented in the first of the two attached paper, i.e. "Approximate structure-preserving semantic matching" by Fausto Giunchiglia, Fiona McNeill, Mikalai Yatskevich, Juan Pane, Paolo Besana, Pavel Shvaiko submitetd and accepted at the 7th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE 2008).
2. **Test case 2: systematic benchmarks**. This test case was composed of trees that are alterations of the original trees. Unlike the work on systematic benchmarks in Ontology Alignment Evaluation Initiative-OAEI [5], the original trees here were generated automatically. We have used here both generic trees (see "Approximate structure-preserving semantic matching" paper) and real world GIS ESRI ArcWeb services (see the second attached paper "An evaluation of approximate sontology matching in GIS applications" by Lorenzino Vaccari , Pavel Shvaiko , Paolo Besana , Maurizio Marchese to be submitted to GeoInformatica)
3. **Test case 3: classification**. In this third test, we have investigated the ability of the SPSM algorithm in the unsupervised clustering of a set of meaningful related Web Service operations. In this experiments, the evaluation ssetup corresponded to a manual classication (reference alignment) of a selected set (50) of ARCWeb service operations. We thus compared the constructed manual classication of the selected Web Service operations with the automatic one obtained with the SPSM approach. The theory, the evaluation set-up and the results are presented in the second paper "An evaluation of approximate ontology matching in GIS applications".

A more specific evaluation of the SPSM approach focused not only on the performance of the matching component, but also on the the evaluation of the fulfillment of original purposes envisioned for the matching plug-in in the OpenKnowledge project, have been addressed and detailed in the Trust and GEA deliverable (D4.9) and in the e-Response summative experiment (D6.8)

### 2.1 Summary of results

We report here a brief summary of the main results of our benchmarking experiments, in order to facilitate the reading of the deliverable. The complete details of the evaluation set-up, the evaluation methodology and the complete results are described in the two following papers.

1. **Test case 1: real-world ontologies** The SPSM solution demonstrates high matching quality on the wide range of the internal parameters (cut-off threshold) values. In particular, F-Measure values exceed 70% the extended range $0.2 - 1.0$ of cut-off threshold. A more detailed analysis of the performance of the individual matchers (i.e. string-based, edit-distance, semantic matchers etc..) indicates the quality and range of applicability of each matchers (see also the brief discussion of the results for following test case).
2. **Test case 2: systematic benchmarks**. We have developed evaluation tests to explore the overall behavior and robustness of the proposed SPSM approach towards both typical syntactic alterations and alterations of word meanings in real GIS service operation signatures. All experiments demonstrated the capability of the SPSM approach to self-adapt (i.e. to provide best results) to the empirical threshold used in the experiment to simulate the users' tolerance to errors (i.e. to calculate the set of true positive, false positive and false negative correspondences).
   Moreover, the results show the robustness of the SPSM algorithm over significant ranges of parameters' variation (thresholds and alteration operations' probability); while maintaining high (over 50-60 %) overall matching relevance quality (F-measure).
   Comparison with a baseline matcher (based on edit-distance algorithm) showed how the SPSM approach is always comparable with the baseline when only syntactic alteration are considered, whereas SPSM results were always better (in average more than 20%) when "meaning" alterations were introduced. This is exactly what one would expect, since SPSM approach includes a number of state-of-the-art syntactic matchers (that are first used in the internal matching algorithm) plus a number of semantic matchers that enter into play for the alterations in the meaning of nodes labels. .
3. **Test case 3: classification**. In this experiment, we have investigated how the proposed SPSM approach could be used in determining (in an unsupervised manner) the "class" of a specific GIS operation directly from the information present in its WSDL operation signature. We thus compared a manual classication for the selected Web Service operations with the automatic one obtained with the SPSM approach.
   Classification quality measures despend on the cut-off threshold values and the SPSM solution demonstrated overall good matching quality (i.e. F-measure) on the wide range of these values. In particular, the best F-measure values exceeded 50% for the given GIS operations set. Although the results are encouraging, still 50% of GIS operation were incorrectly classified, due to the limited knowledge present in the signatures only. In this case, the

presence of more informative and semantically structured annotation would improve significantly the automatic classification, though at the expense of a greater effort from the designer/programmer.

Moreover, during all experiments we have been able to access the overall performance of the SPSM algorithm. The evaluations were performed on standard laptop Intel Centrino Core Duo CPU-2Ghz, 2GB RAM, with the Windows Vista (32bit, SP1) operating system, and with no applications running but a single matching system.

Considering all our experiments we executed approximately 700.000 matching tasks using SPSM. The efficiency of SPSM solution is such that the average execution time per matching task in the evaluation under consideration was 43ms (with an average number of the parameters of the WSDL operations around 4). The quantity of main memory used by SPSM during matching did not rise more than 2.3Mb higher than the standby level.

# Approximate structure-preserving semantic matching

Fausto Giunchiglia[1], Fiona McNeill[2], Mikalai Yatskevich[1], Juan Pane[1], Paolo Besana[2], Pavel Shvaiko[3]

[1] University of Trento, Povo, Trento, Italy,
{fausto|yatskevi|pane|pavel|}@dit.unitn.it
[2] University of Edinburgh, Scotland,
f.j.mcneill@ed.ac.uk|p.besana@sms.ed.ac.uk
[3] TasLab, Informatica Trentina, Italy,
Pavel.Shvaiko@infotn.it

**Abstract.** Typical ontology matching applications, such as ontology integration, focus on the computation of correspondences holding between the nodes of two graph-like structures, e.g., between concepts in two ontologies. However, for applications such as web service integration, we need to establish whether full graph structures correspond to one another globally, preserving certain structural properties of the graphs being considered. The goal of this paper is to provide a new matching operation, called *structure-preserving semantic matching*. This operation takes two graph-like structures and produces a set of correspondences, $(i)$ still preserving a set of structural properties of the graphs being matched, $(ii)$ only in the case if the graphs are *globally* similar to one another. Our approach is based on a formal theory of abstraction and on a tree edit distance measure. We have evaluated our solution in various settings. Empirical results show the efficiency and effectiveness of our approach.

## 1 Introduction

Ontology matching is a critical operation in many applications, such as Artificial Intelligence, the Semantic Web and e-commerce. It takes two graph-like structures, for instance, lightweight ontologies [9], and produces an alignment, that is, a set of correspondences, between the nodes of those graphs that correspond semantically to one another [6].

Many varied solutions of matching have been proposed so far; see [6,28,23] for recent surveys[4]. In this paper we introduce a particular type of matching, namely *Structure-preserving semantic matching (SPSM)*. In contrast to conventional ontology matching, which aims to match single words through considering their position in hierarchical ontologies, structure-preserving semantics matching aims to match complex, structured terms. These terms are not structured according to their semantics, as terms are in an ontology, but are structured to express relationships: in the case of our approach, first-order relationships. This structure-preserving matching is therefore a two-step process, the first step of which is to match individual words within the terms through techniques used for conventional ontology matching, and the second - and novel - step of

[4] See, http://www.ontologymatching.org for a complete information on the topic.

which is to match the structure of the terms. For example, consider a first-order relation $buy(car, price)$ and another, $purchase(price, vehicle, number)$, both expressing buying relations between vehicles and cost. If the words used in these terms are from known ontologies, then we can use standard ontology matching techniques to determine, for example, that $buy$ is equivalent to $purchase$ and that $car$ is a sub-type of $vehicle$. If they are not from known ontologies we can still use WordNet to gather this information. Our work is concerned with understanding and using this information about how the words are related to determine how the full structured terms are related. Therefore, SPSM needs to preserve a set of structural properties (e.g., vertical ordering of nodes) to establish whether two graphs are globally similar and, if so, how similar they are and in what way. These characteristics of matching are required in web service integration applications, see, e.g., [20,22,17].

More specifically, most of the previous solutions to web service matching employ a single ontology approach, that is, the web services are assumed to be described by the concepts taken from a shared ontology. This allows for the reduction of the matching problem to the problem of reasoning within the shared ontology [20,26]. In contrast, following the work in [1,25,30], we assume that web services are described using terms from different ontologies and that their behavior is described using complex terms; we consider first-order terms. This allows us to provide detailed descriptions of the web services' input and output behavior. The problem becomes therefore that of matching two web service descriptions, which in turn, can be viewed as first-order terms and represented as tree-like structures. An alignment between these structures is considered as successful only if two trees are *globally* similar, e.g., $tree_1$ is 0.7 similar to $tree_2$, according to some measure in [0 1]. A further requirement is that the alignment must preserve certain structural properties of the trees being considered. In particular, the syntactic types and sorts have to be preserved: $(i)$ a function symbol must be matched to a function symbol and $(ii)$ a variable must be matched to a variable. We are mainly interested in approximate matching, since two web service descriptions may only rarely match perfectly.

The contributions of this paper include: $(i)$ a new approach to approximate web service matching, called *Structure-preserving semantic matching (SPSM)*, and $(ii)$ an implementation and evaluation of the approach in various settings (both with automatically generated tests and real-world first-order ontologies) with encouraging results. SPSM takes two tree-like structures and produces an alignment between those nodes of the trees that correspond semantically to one another, preserving the above mentioned two structural properties of the trees being matched, and only in the case that the trees are globally similar. Technically, the solution is based on the fusion of ideas derived from the theory of abstraction [11,12] and tree edit distance algorithms [3]. To the best of our knowledge, this is the first work taking this view.

The rest of the paper is organized as follows. Section 2 explains how calls to web services can be viewed as first-order trees. It also provides a motivating example. We overview the approximate SPSM approach in Section 3, while its details, such as abstraction operations, their correspondence to tree edit operations as well as computation of global similarity between trees are presented in Section 4 and Section 5, respectively.

Evaluation is discussed in Section 6. Section 7 relates our work to similar approaches. Finally, Section 8 summarizes the major findings.

## 2 Matching Web Services

Our hypothesis is that we can consider web services inputs and outputs as trees and therefore apply SPSM to calls to web services. This kind of structural matching can then allow us to introduce flexibility to calls to web services so that we no longer need to rely on $(i)$ terms used in these calls coming from a global ontology; instead local ontologies adapted to purpose can be used; $(ii)$ the structuring of these calls being fixed.

The structure is important because each argument in a call to a web service is defined according to its position in the input or output. However, expecting this structure to be fixed is just as problematic as expecting a global ontology. Individual web service designers will use different structure just as they will use different vocabulary and changes to web service descriptions over time will be mean that previous calls to web services become inappropriate. In order to remove the need both for a global ontology and a fixed structure for every web service call, we therefore need to employ structured matching techniques for matching between web service calls and returns and web service inputs and outputs.

The first-order terms that we match do not distinguish between inputs and outputs in the same manner as, for example, Web Service Description Language (WSDL). Instead, both inputs and outputs are arguments of the same predicate. In Prolog notation, this is indicated by using a $+$ for an input and a $-$ for an output. Thus the term:

$$purchase(-Price, +Vehicle, +Number)$$

indicates that $Vehicle$ and $Number$ are inputs and $Price$ is an output. During run-time, we can distinguish between inputs and outputs because inputs must be instantiated and outputs must be uninstantiated. In order to use our tree matching techniques for web services, we therefore make use of an automated translation process we have created that will map between a first-order term such as the above and a standard WSDL representation of the same information. This approach can also be used for other kinds of services in addition to web services; all that is required is that a translation process is created to convert between the representation of the service and first-order terms.

We make the assumption that web services written in WSDL will contain some kind of semantic descriptions of what the inputs and outputs are: that arguments are labelled descriptively and not merely as 'input1' and so on. This is after all what WSDL, as a description language, is designed to do. We appreciate that in practice designers of web services adopt a lazy approach and label inputs and outputs with terms that do not describe their semantics, especially when the WSDL files are generated automatically from classes or interfaces written in a programming language. In such cases, our techniques will have a very low success rate. However, such web services are of little value for any automated process and do not make use of the full potential of WSDL. We believe that as they become more widely used, the need for them to be properly descriptive becomes imperative so that they can be located and invoked automatically. In

the meantime, any mark-up that is used to provide semantics for web services outside of the WSDL can also be amenable to our techniques, provided, as is usually the case, that descriptions of inputs and outputs can be expressed as a tree.

Let us consider an example of approximate SPSM between the following web services: get_wine(Region, Country, Color, Price, Number_of_bottles) and get_wine(Region(Country, Area), Colour, Cost, Year, Quantity), see Figure 1. In this case the first web service description requires the fourth argument of the get_wine function (Color) to be matched to the second argument (Colour) of the get_wine function in the second description. Also, Region in $T2$ is defined as a function with two arguments (Country and Area), while in $T1$, Region is an argument of get_wine. Thus, Region in $T1$ must be passed to $T2$ as the value of the Area argument of the Region function. Moreover, Year in $T2$ has no corresponding term in $T1$. Notice that detecting these correspondences would have not been possible in the case of exact matching by its definition.

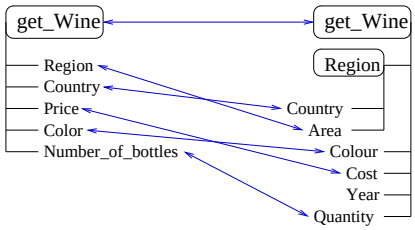


Fig. 1: Two approximately matched web services represented as trees: $T1$: get_wine(Region, Country, Color, Price, Number_of_bottles) and $T2$: get_wine(Region(Country, Area), Colour, Cost, Year, Quantity). Functions are in rectangles with rounded corners; they are connected to their arguments by dashed lines. Node correspondences are indicated by arrows.

In order to guarantee successful web service integration, we are only interested in the correspondences holding among the nodes of the trees underlying the given web services in the case when the web services themselves are similar enough. At the same time the correspondences have to preserve two structural properties of the descriptions being matched: $(i)$ functions have to be matched to functions and $(ii)$ variables to variables. Thus, for example, Region in $T1$ is not linked to Region in $T2$. Finally, let us suppose that the correspondences on the example of Figure 1 are aggregated into a single similarity measure between the trees under consideration, e.g., 0.62. If this global similarity measure is higher than empirically established threshold (e.g., 0.5), the web services under scrutiny are considered to be similar enough, and the set of correspondences showed in Figure 1 is further used for the actual web service integration.

## 3 Overview of the Approach

The matching process is organized in two steps: $(i)$ node matching and $(ii)$ tree matching. Node matching solves the semantic heterogeneity problem by considering only labels at nodes and contextual information of the trees. We use here the S-Match system [14]. Technically, two nodes $n_1 \in T1$ and $n_2 \in T2$ match iff: $c@n_1 \; R \; c@n_2$ holds,

| Matcher name | Execution order | Approximation level | Matcher type | Schema info |
|---|---|---|---|---|
| *WordNet* | 1 | 1 | Sense-based | WordNet senses |
| Prefix | 2 | 2 | String-based | Labels |
| Suffix | 3 | 2 | String-based | Labels |
| Edit distance | 4 | 2 | String-based | Labels |
| Ngram | 5 | 2 | String-based | Labels |

Table 1: Element level matchers. The first column contains the names of the matchers. The second column lists the order in which they are executed. The third column introduces the matcher's approximation level. The relations produced by a matcher with the first approximation level are always correct. Notice that matchers are executed following the order of increasing approximation. The fourth column reports the matcher's type, while the fifth column describes the matcher's input, see [14] for details.

where $c@n_1$ and $c@n_2$ are the concepts at nodes $n_1$ and $n_2$, and $R \in \{=, \sqsubseteq, \sqsupseteq\}$. In semantic matching [10] as implemented in the S-Match system [14] the key idea is that the relations, e.g., equivalence and subsumption, between nodes are determined by $(i)$ expressing the entities of the ontologies as logical formulas and by $(ii)$ reducing the matching problem to a logical validity problem. Specifically, the entities are translated into logical formulas which explicitly express the concept descriptions as encoded in the ontology structure and in external resources, such as WordNet [8]. Besides WordNet, the basic version of S-Match also uses four string-based matchers, see Table 1. This allows for a translation of the matching problem into a logical validity problem, which can then be efficiently resolved using sound and complete state of the art satisfiability solvers [13]. Notice that the result of this stage is the set of one-to-many correspondences holding between the nodes of the trees. For example, initially Region in $T1$ is matched to both Region and Area in $T2$.

Tree matching, in turn, exploits the results of the node matching and the structure of the trees to find if these globally match each other. Specifically, given the correspondences produced by the node matching, the abstraction operations (§4) are used in order to select only those correspondences that preserve the desired properties, namely that functions are matched to functions and variables to variables. Thus, for example, the correspondence that binds Region in $T1$ and Region in $T2$ should be discarded, while the correspondence that binds Region in $T1$ and Area in $T2$ should be preserved. Then, the preserved correspondences are used as allowed operations of a tree edit distance in order to determine global similarity (§5) between trees under consideration. If this global similarity measure is higher than an empirically established threshold, the trees are considered to be similar enough, and not similar otherwise. Technically, two trees $T1$ and $T2$ approximately match iff there is at least one node $n_{1i}$ in $T1$ and a node $n_{2j}$ in $T2$ such that: $(i)$ $n_{1i}$ approximately matches $n_{2j}$, and $(ii)$ all ancestors of $n_{1i}$ are approximately matched to the ancestors of $n_{2j}$, where $i$=1,...,N1; $j$=1,...,N2; N1 and N2 are the number of nodes in $T1$ and $T2$, respectively.

Semantic heterogeneity is therefore reduced to two steps: $(i)$ matching the web services, thereby obtaining an alignment, and $(ii)$ using this alignment for the actual web service integration. This paper focuses only on the matching step.

## 4 Matching Via Abstraction

In this section we first discuss the abstraction operations (§4.1), then discuss how these operations are used in order to drive a tree edit distance computation (§4.2), and, finally, discuss the implementation details (§4.3).

### 4.1 Abstraction Operations

The work in [12] categorizes the various kinds of abstraction operations in a wide-ranging survey. It also introduces a new class of abstractions, called TI-abstractions (where TI means "Theorem Increasing"), which have the fundamental property of maintaining completeness, while loosing correctness. In other words, any fact that is true of the original term is also true of the abstract term, but not vice versa. Similarly, if a ground formula is true, so is the abstract formula, but not vice versa. Dually, by taking the inverse of each abstraction operation, we can define a corresponding refinement operation which preserves correctness while loosing completeness. The second fundamental property of the abstraction operations is that they provide *all and only* the possible ways in which two first-order terms can be made to differ by manipulations of their signature, still preserving completeness. In other words, this set of abstraction/refinement operations defines all and only the possible ways in which correctness and completeness are maintained when operating on first-order terms and atomic formulas. This is the fundamental property which allows us to study and consequently quantify the semantic similarity (distance) between two first-order terms. To this extent it is sufficient to determine which abstraction/refinement operations are necessary to convert one term into the other and to assign to each of them a cost that models the semantic distance associated to the operation.

The work in [12] provides the following major categories of abstraction operations:

**Predicate:** Two or more predicates are merged, typically to the least general generalization in the predicate type hierarchy, e.g., *Bottle(X)* + *Container(X)* $\mapsto$ *Container(X)*. We call *Container(X)* a predicate abstraction of *Bottle(X)* or *Container(X)* $\sqsupseteq_{Pd}$ *Bottle(X)*. Conversely, we call *Bottle(X)* a predicate refinement of *Container(X)* or *Bottle(X)* $\sqsubseteq_{Pd}$ *Container(X)*.

**Domain:** Two or more terms are merged, typically by moving the functions or constants to the least general generalization in the domain type hierarchy, e.g., *Micra* + *Nissan* $\mapsto$ *Nissan*. Similarly to the previous item we call *Nissan* a domain abstraction of *Micra* or *Nissan* $\sqsupseteq_D$ *Micra*. Conversely, we call *Micra* a domain refinement of *Nissan* or *Micra* $\sqsubseteq_D$ *Nissan*.

**Propositional:** One or more arguments are dropped, e.g., *Bottle(A)* $\mapsto$ *Bottle*. We call *Bottle* a propositional abstraction of *Bottle(A)* or *Bottle* $\sqsupseteq_P$ *Bottle(A)*. Conversely, *Bottle(A)* is a propositional refinement of *Bottle* or *Bottle(A)* $\sqsubseteq_P$ *Bottle*.

Let us consider the following pair of first-order terms *(Bottle A)* and *(Container)*. In this case there is no abstraction/refinement operation that makes them equivalent. However, consequent applications of propositional and domain abstraction operations make the two terms equivalent:

$$\textit{(Bottle A)} \mapsto^{\sqsubseteq_P} \textit{(Bottle)} \mapsto^{\sqsubseteq_D} \textit{(Container)}$$

In fact the relation holding among the terms is a composition of two refinement operations, namely *(Bottle A)* $\sqsubseteq_P$ *(Bottle)* and *(Bottle)* $\sqsubseteq_D$ *(Container)*.

The abstraction/refinement operations discussed above allow us to preserve the desired properties: that functions are matched to functions and variables to variables. For example, predicate and domain abstraction/refinement operations do not convert a function into a variable. Therefore, the one-to-many correspondences returned by the node matching should be further filtered based on the allowed abstraction/refinement operations: $\{=, \sqsupseteq, \sqsubseteq\}$, where $=$ stands for equivalence; $\sqsupseteq$ represents an abstraction relation and connects the precondition and the result of a composition of arbitrary number of predicate, domain and propositional abstraction operations; and $\sqsubseteq$ represents a refinement relation and connects the precondition and the result of a composition of arbitrary number of predicate, domain and propositional refinement operations.

Since abstractions and refinements cover every way in which first-order terms can differ (either in the predicate, in the number of arguments or in the types of arguments), we can consider every relation between terms that are in some way related as a combination of these six basic refinements and abstractions. Therefore, every map between first-order trees can be described using these operations. The only situation in which we cannot use these techniques is if there is no semantic relation between the predicates of the two terms, but in this situation, a failed mapping is the appropriate outcome since we do not consider them to be related even though the arguments may agree. Note that we can match non-related arguments using these operations by applying propositional abstraction and then propositional refinement.

### 4.2 Tree Edit Distance Via Abstraction Operations

Now that we have defined the operations that describe the differences between trees, we need some way of composing them so that we can match entire trees to one another. We look for a composition of the abstraction/refinement operations allowed for the given relation $R$ (see §3) that are necessary to convert one tree into another. In order to solve this problem we propose to represent abstraction/refinement operations as tree edit distance operations applied to the term trees.

In its traditional formulation, the tree edit distance problem considers three operations: $(i)$ vertex deletion, $(ii)$ vertex insertion, and $(iii)$ vertex replacement [31]. Often these operations are presented as rewriting rules:

$$(i)\ \upsilon \rightarrow \lambda \qquad (ii)\ \lambda \rightarrow \upsilon \qquad (iii)\ \upsilon \rightarrow \omega$$

where $\upsilon$ and $\omega$ correspond to the labels of nodes in the trees while $\lambda$ stands for the special blank symbol.

Our proposal is to restrict the formulation of the tree edit distance problem in order to reflect the semantics of the first-order terms. In particular, we propose to redefine the tree edit distance operations in a way that will allow them to have one-to-one correspondence to the abstraction/refinement operations. Table 2 illustrates the correspondence between abstraction/refinement and tree edit operations. Let us focus for the moment on the first three columns of Table 2. The first column presents the abstraction/refinement operations. The second column lists corresponding tree edit operations. The third column describes the preconditions of the tree edit operation use.

Table 2: The correspondence between abstraction operations, tree edit operations and costs.

| **Abstraction operations** | **Tree edit operations** | **Preconditions of operations** | $Cost_{T1=T2}$ | $Cost_{T1 \sqsubseteq T2}$ | $Cost_{T1 \sqsupseteq T2}$ |
|---|---|---|---|---|---|
| $t_1 \sqsupseteq_{Pd} t_2$ | $a \rightarrow b$ | $a \sqsupseteq b$; $a$ and $b$ correspond to predicates | 1 | $\infty$ | 1 |
| $t_1 \sqsupseteq_{D} t_2$ | $a \rightarrow b$ | $a \sqsupseteq b$; $a$ and $b$ correspond to functions or constants | 1 | $\infty$ | 1 |
| $t_1 \sqsupseteq_{P} t_2$ | $a \rightarrow \lambda$ | $a$ corresponds to predicates, functions or constants | 1 | $\infty$ | 1 |
| $t_1 \sqsubseteq_{Pd} t_2$ | $a \rightarrow b$ | $a \sqsubseteq b$; $a$ and $b$ correspond to predicates | 1 | 1 | $\infty$ |
| $t_1 \sqsubseteq_{D} t_2$ | $a \rightarrow b$ | $a \sqsubseteq b$; $a$ and $b$ correspond to functions or constants | 1 | 1 | $\infty$ |
| $t_1 \sqsubseteq_{P} t_2$ | $a \rightarrow \lambda$ | $a$ corresponds to predicates, functions or constants | 1 | 1 | $\infty$ |
| $t_1 = t_2$ | $a = b$ | $a = b$; $a$ and $b$ correspond to predicates, functions or constants | 0 | 0 | 0 |

Let us consider, for example, the first line of Table 2. The predicate abstraction operation applied to first-order term $t_1$ results with term $t_2$ ($t_1 \sqsupseteq_{Pd} t_2$). This abstraction operation corresponds to a tree edit replacement operation applied to the term $t_1$ of the first tree that replaces the node $a$ with the node $b$ of the second tree $(a \rightarrow b)$. Moreover, the operation can be applied only in the case that: $(i)$ label $a$ is a generalization of label $b$ and $(ii)$ both nodes with labels $a$ and $b$ in the term trees correspond to predicates in the first-order terms.

### 4.3 Implementation

We have implemented our approximate SPSM solution in Java. Many existing tree edit distance algorithms allow the tracking of the nodes to which a replace operation is applied. According to [31], the minimal cost correspondences are: $(i)$ one-to-one, $(ii)$ horizontal order preserving between sibling nodes, and $(iii)$ vertical order preserving. The alignment depicted in Figure 1 complies with $(i)$, $(iii)$ and violates $(ii)$. In fact, the fourth sibling Color in $T1$ is matched to the second sibling Colour in $T2$ (see below for an explanation).

For the tree edit distance operations depicted in Table 2, we propose to keep track of nodes to which the tree edit operations derived from the replace operation are applied. In particular, we consider the operations that correspond to predicate and domain abstraction/refinement ($t_1 \sqsupseteq_{Pd}$, $t_1 \sqsubseteq_{Pd}$, $t_1 \sqsupseteq_{D}$, $t_1 \sqsubseteq_{D}$). This allows us to obtain an alignment among the nodes of the term trees with the desired properties, i.e., that there are only one-to-one correspondences in it and that functions are matched to functions and variables are matched to variables. This is the case because $(i)$ predicate and domain abstraction/refinement operations do not convert, for example, a function into a variable and $(ii)$ the tree edit distance operations, as from Table 2, have a one-to-one correspondence with abstraction/refinement operations.

At the same time, an alignment used in a tree edit distance computation preserves the horizontal order among the sibling nodes, but this is not a desirable property for the web service integration purposes. In fact, we would want the fourth sibling Colour

in $T1$ to match the second sibling Color in $T2$ of Figure 1. However, as from Table 2, the tree edit operations corresponding to predicate and domain abstraction/refinement ($t_1 \sqsupseteq_{Pd}$, $t_1 \sqsubseteq_{Pd}$, $t_1 \sqsupseteq_D$, $t_1 \sqsubseteq_D$) can be applied only to those nodes of the trees whose labels are either generalizations or specializations of each other, as computed by the S-Match node matching algorithm. Therefore, given the alignment produced by the S-Match node matching algorithm, we identify the cases when the horizontal order between sibling nodes is not preserved and change the ordering of the sibling nodes to make the alignment horizontal order preserving. For example, swapping the nodes Cost and Colour in $T2$ of Figure 1 does not change the meaning of these terms but it allows the correspondence holding between Colour and Color in Figure 1 to be included in the alignment without increasing the cost during the tree edit distance computation. This switching means that the original horizontal order of siblings is not preserved in most cases. If there are arguments with identical names, such cases are resolved with the help of indexing schemes.

## 5 Global Similarity Between Trees

Our goal now is to compute the similarity between two term trees. Since we compute the composition of the abstraction/refinement operations that are necessary to convert one term tree into the other, we are interested in a minimal cost of this composition. Therefore, we have to determine the minimal set of operations which transforms one tree into another, see Eq. 1:

$$\mathit{Cost} = min \sum_{i \in S} k_i * \mathit{Cost}_i \tag{1}$$

where, $S$ stands for the set of the allowed tree edit operations; $k_i$ stands for the number of *i-th* operations necessary to convert one tree into the other and $\mathit{Cost}_i$ defines the cost of the *i-th* operation. Our goal here is to define the $\mathit{Cost}_i$ in a way that models the semantic distance.

A possible uniform proposal is to assign the same unit cost to all tree edit operations that have their abstraction theoretic counterparts. The last three columns of Table 2 illustrate the costs of the abstraction/refinement (tree edit) operations, depending on the relation (equivalence, abstraction or refinement) being computed between trees. Notice that the costs for estimating abstraction ($\sqsupseteq$) and refinement ($\sqsubseteq$) relations have to be adjusted according to their definitions. In particular, the tree edit operations corresponding to abstraction/refinement operations that are not allowed by definition of the given relation have to be prohibited by assigning to them an infinite cost. Notice also that we do not give any preference to a particular type of abstraction/refinement operations. Of course this strategy can be changed to satisfy certain domain specific requirements.

Let us consider, for example, the first line of Table 2. The cost of the tree edit distance operation that corresponds to the predicate abstraction ($t_1 \sqsupseteq_{Pd} t_2$) is equal to 1 when used for the computation of equivalence ($\mathit{Cost}_{T1=T2}$) and abstraction ($\mathit{Cost}_{T1 \sqsupseteq T2}$) relations between trees. It is equal to $\infty$ when used for the computation of refinement ($\mathit{Cost}_{T1 \sqsubseteq T2}$) relation.

Eq. 1 can now be used for the computation of the tree edit distance score. However, when comparing two web service descriptions we are interested in similarity rather than in distance. We exploit the following equation to convert the distance produced by a tree edit distance into the similarity score:

$$\textit{TreeSim} = 1 - \frac{\textit{Cost}}{\textit{max}(\textit{T1}, \textit{T2})} \tag{2}$$

where $Cost$ is taken from Eq. 1 and is normalized by the size of the biggest tree. Note that for the special case of *Cost* equal to $\infty$, *TreeSim* is estimated as 0. Finally, the highest value of *TreeSim* computed for $\textit{Cost}_{T1=T2}$, $\textit{Cost}_{T1\sqsubseteq T2}$ and $\textit{Cost}_{T1\sqsupseteq T2}$ is selected as the one ultimately returned. For example, in the case of example of Figure 1, when we match $T1$ with $T2$ this would be 0.62 for both $\textit{Cost}_{T1=T2}$ and $\textit{Cost}_{T1\sqsubseteq T2}$.

## 6 Evaluation

On top of the implementation discussed in §4.3 we exploited a modification of simple tree edit distance algorithm from [33]. The evaluation set-up is discussed in §6.1, while the evaluation results are presented in §6.2.

### 6.1 Evaluation Set-up

Ontology and web service engineering practices suggest that often the underlying trees to be matched are derived or inspired from one another. Therefore, it is reasonable to compare a tree with another one derived from the original one. We have evaluated efficiency and quality of the results of our matching solution on two test cases.

**Test case 1: real-world ontologies.** We used different versions of the Standard Upper Merged Ontology (SUMO)[5] and the Advance Knowledge Transfer (AKT)[6] ontologies. We extracted all the differences between versions 1.50 and 1.51, and between versions 1.51 and 1.52 of the SUMO ontology and between versions 1 and 2.1, and 2.1 and 2.2 of the AKT-portal and AKT-support ontologies[7]. These are all first-order ontologies (hence, their expressivity is far beyond generalization/specialization hierarchies), so many of these differences matched well to the potential differences between terms that we are investigating. However, some of them were more complex, such as differences in inference rules, and had no parallel in our work; therefore, these were discarded, and our tests were run on all remaining differences. Specifically, 132 pairs of trees (first-order logic terms) were used. Half of the pairs were composed of the equivalent terms (e.g., journal(periodical-publication) and magazine (periodical-publication)) while the other half was composed from similar but not equivalent terms (e.g., web-reference(publication-reference) and thesis-reference (publication-reference)).

[5] `http://ontology.teknowledge.com/`

[6] `http://www.aktors.org`

[7] See `http://dream.inf.ed.ac.uk/projects/dor/` for full versions of these ontologies and analysis of their differences.

**Test case 2: systematic benchmarks.** Different application programming interfaces (APIs) suggest that the terms within a tree are likely not to be semantically related to each other. Examples from the Java API include: set(index, element) and put(key, value). Thus, trees can be considered as being composed of nodes whose labels are random terms.

This test case was composed of trees that are alterations of the original trees. Unlike the work on systematic benchmarks in Ontology Alignment Evaluation Initiative-OAEI [5], the original trees here were generated automatically. We have generated 100 trees. For each original tree, 30 altered ones were created, see Table 3. Pairs composed of the original tree and one varied tree were fed to our SPSM solution. The experiment described above was repeated 5 times in order to remove noise in the results.

For tree generation, node labels were composed of a random number of words, selected from 9000 words extracted from the Brown Corpus[8]. The average number of nodes per tree was 8; in fact, functions usually have fewer parameters. In turn, the tree alterations were inspired by the approach in [5]. These are summarized in Table 3 and include: $(i)$ syntactic alterations, such as adding or removing characters, and $(ii)$ semantic alterations, word addition in labels by using related words (e.g., synonyms) extracted from the Moby thesaurus[9]. The probabilities used for these two types of alterations represent the fact that in most of the cases (0.8) the modifications made during an evolution process concern the altering in meaning, while syntactic modifications, such as introducing acronyms, usually have less occurrences (0.3).

Table 3: Parameters used for generating and modifying the trees

| **Parameter** | **Syntactic** | **Semantic** | **Combined** |
|---|---|---|---|
| Number of trees | 100 | 100 | 100 |
| Number of modifications per tree | 30 | 30 | 30 |
| Average number of nodes per tree | 8 | 8 | 8 |
| Probability of replacing a word in a node label for a related one | 0.0 | 0.8 | 0.8 |
| Probability of making a syntactic change in a word of a node label | 0.3 | 0.0 | 0.3 |

Since the tree alterations made are known, these provide the ground truth, and hence, the reference results are available for free by construction, see also [5,21]. This allows for the computation of the matching quality measures. In particular, the standard matching quality measures, such as *Recall*, *Precision* and *F-measure* for the similarity between trees have been computed [6]. In computation of these quality measures we considered the correspondences holding among first-order terms rather than the nodes of the term trees. Thus, for instance, journal(periodical-publication$_1$)=magazine(periodical-publication$_2$) was considered as a single correspondence rather than two correspondences, namely journal=magazine and periodical-publication$_1$=periodical-publication$_2$.

The evaluation was performed on a standard laptop Core Duo CPU-2Ghz, 2GB RAM, with the Windows Vista operating system, and with no applications running but a single matching system.

[8] `http://icame.uib.no/brown/bcm.html`

[9] `http://www.mobysaurus.com/`. Since the SPSM node matching uses WordNet 2.1, an alternative thesaurus was used here.

### 6.2 Evaluation Results

The matching quality results for the first test case are shown in Figure 2. Quality measures depend on the cut-off threshold values and the SPSM solution demonstrates high matching quality on the wide range of these values. In particular, F-Measure values exceed 70% for the given range.

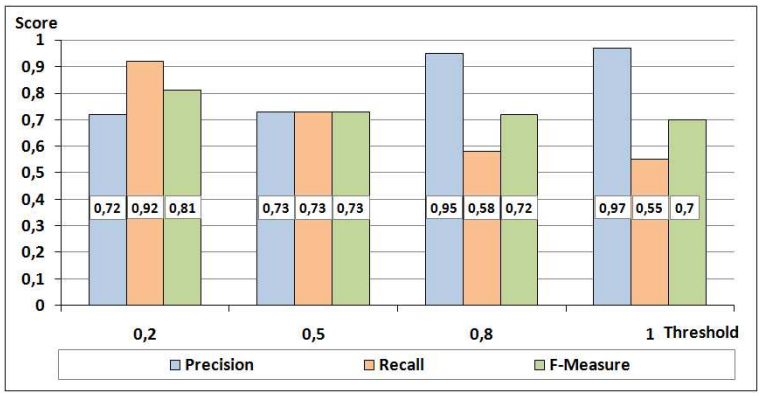


Fig. 2: Test case 1: Evaluation results.

The evaluation results for the second test case are summarized in Figures 3, 4 and 5. In order to obtain those results there have been used: $(i)$ the tree matcher discussed in §4 and §5 and $(ii)$ the various matchers used in isolation (namely, edit distance, NGram, prefix, suffix and WordNet) and all these matchers as combined by S-Match, see Table 1. Figures 3, 4 and 5 are composed of four plots (from top to bottom): $(i)$ standard precision-recall plot, $(ii)$ recall *vs* various cut-off threshold values in [0 1], $(iii)$ precision *vs* various cut-off threshold values in [0 1], and $(iv)$ F-measure *vs* various cut-off threshold values in [0 1].

In particular, Figure 3 shows that for the syntactic alterations, as expected, string-based matchers outperform the WordNet matcher. Also, edit distance performs as well as S-Match. The best performance in terms of F-Measure (which is 0.52) is reached at the threshold of 0.8. In turn, Figure 4 shows that for the semantic alterations, as expected, the WordNet matcher outperforms the string-based matchers. The best performance in terms of F-Measure (which is 0.73) is demonstrated by S-Match and is reached at the threshold of 0.8. Finally, Figure 5 shows that when both types of alterations, namely syntactic and semantics, are applied the best performance in terms of F-Measure (which is 0.47) is demonstrated by S-Match and is reached at the threshold of 0.8.

The efficiency of our solution is such that the average execution time per matching task in the two test cases under consideration was 93ms. The quantity of main memory used by SPSM during matching did not rise more than 3Mb higher than the standby level. Finally, the evaluation results show that conventional ontology matching technology that we previously applied to matching classifications and XML schemas (see [14])

can also provide encouraging results in the web services domain. Of course, additional extensive testing is needed, especially with WSDL services, for example as done in [30].

## 7 Related Work

We believe that this approach to structured matching is unique and therefore it is difficult to perform any comparative analysis. In order to demonstrate that we make use of powerful ontology matching tools for the standard ontology matching step of the process, we can compare S-Match against other ontology matching tools. However, the full structure-preserving semantic matching addresses a previously unsolved problem. In this section, we discuss other methods that address similar problems.

Our work builds on standard work in tree-edit distance measures, for example, as espoused by [27]. The key difference with our work is the integration of the semantics that we gain through the application of the abstraction and refinement rules. This allows us to consider questions such as *what is the effect to the overall* meaning *of the term (tree) if* node a *is relabelled to* node b?, or *how significant is the removal of a node to the overall semantics of the term*? These questions are crucial in determining an intuitive and meaningful similarity score between two terms, and are very context dependent. Altering the scores given in Table 2 enables us to provide different answers to these questions depending on the context, and we are working on giving providing even more subtle variations of answers reflecting different contexts (see Section 8).

Work based on these ideas, such as Mikhaiel and Stroudi's work on HTML differencing [15], tends to focus only on the structure and not on the semantics. This work never considers what the individual nodes in their HTML trees mean and only considers context in the sense that, for example, the cost of deleting a node with a large subtree is higher than the cost of deleting a leaf node; the semantic meanings of these nodes is not considered.

The problem of location of web services on the basis of the capabilities that they provide (often referred as the matchmaking problem) has recently received considerable attention. Most of the approaches to the matchmaking problem so far employed a single ontology approach (i.e., the web services are assumed to be described by the concepts taken from the shared ontology). See [20,22,26] for example. Probably the most similar to ours is the approach taken in METEOR-S [1] and in [25], where the services are assumed to be annotated with the concepts taken from various ontologies. Then the matchmaking problem is solved by the application of the matching algorithm. The algorithm combines the results of atomic matchers that roughly correspond to the element level matchers exploited as part of our algorithm. In contrast to this work, we exploit a more sophisticated matching technique that allows us to utilise the structure provided by the first order term.

Many diverse solutions to the ontology matching problem have been proposed so far. See [28] for a comprehensive survey and [7,24,4,16,2,19,29] for individual solutions. However most efforts has been devoted to computation of the correspondences holding among the classes of description logic ontologies. Recently, several approaches allowed computation of correspondences holding among the object properties (or binary

predicates) [32]. The approach taken in [18] facilitates the finding of correspondences holding among parts of description logic ontologies or subgraphs extracted from the ontology graphs. In contrast to these approaches, we allow the computation of correspondences holding among first order terms.

In summary, much work has been done on structure-preserving matching and much has been done on semantic matching, and our work depends heavily on the work of others in these fields. The novelty of our work is in the combination of these two approaches to produce a structure-preserving semantic matching algorithm, thus allowing us to determine fully how structured terms, such as web service calls, are related to one another.

## 8 Conclusions and Future Work

We have presented an approximate SPSM approach that implements the *SPSM* operation. It is based on a theory of abstraction and a tree edit distance. We have evaluated our solution on test cases composed of hundreds of trees. The evaluation results look promising, especially with reference to the efficiency indicators.

Future work proceeds at least along the following directions: $(i)$ studying a best suitable cost model, $(ii)$ incorporating preferences in order to drive approximation, thus allowing/prohibiting certain kinds of approximation (e.g., not approximating red wine with white wine, although these are both wines), and $(iii)$ conducting extensive and comparative testing in real-world scenarios.

**Acknowledgements.** We appreciate support from the OpenKnowledge European STREP (FP6-027253).

## References

1. R. Aggarwal, K. Verma, J. A. Miller, and W. Milnor. Constraint driven web service composition in METEOR-S. In *Proceedings of IEEE SCC*, 2004.
2. S. Bergamaschi, S. Castano, and M. Vincini. Semantic integration of semistructured and structured data sources. *SIGMOD Record*, 28(1), 1999.
3. W. Chen. New algorithm for ordered tree-to-tree correction problem. *Journal of Algorithms*, 40(2), 2001.
4. M. Ehrig, S. Staab, and Y. Sure. Bootstrapping ontology alignment methods with APFEL. In *Proceedings of ISWC*, 2005.
5. J. Euzenat, A. Isaac, C. Meilicke, P. Shvaiko, H. Stuckenschmidt, O. Šváb, V. S., W. van Hage, and M. Yatskevich. Results of the ontology alignment evaluation initiative 2007. In *Proceedings of the ISWC + ASWC International Workshop on Ontology Matching (OM)*, 2007.
6. J. Euzenat and P. Shvaiko. *Ontology matching*. Springer, 2007.
7. J. Euzenat and P. Valtchev. Similarity-based ontology alignment in OWL-lite. In *Proceedings of ECAI*, 2004.
8. C. Fellbaum. *WordNet: an electronic lexical database*. MIT Press, 1998.
9. F. Giunchiglia, M. Marchese, and I. Zaihrayeu. Encoding classifications into lightweight ontologies. *Journal on Data Semantics*, VIII, 2007.

10. F. Giunchiglia and P. Shvaiko. Semantic matching. *The Knowledge Engineering Review*, 18(3), 2003.
11. F. Giunchiglia and T. Walsh. Abstract theorem proving. In Bassi S., Sridharan N., Bertacco S., and Bonicelli R., editors, *11th international joint conference on artificial intelligence (IJCAI'89)*, volume 1, Detroit, Mich., 20-25 August 1989.
12. F. Giunchiglia and T. Walsh. A theory of abstraction. *Artificial Intelligence*, 57(2-3), 1992.
13. F. Giunchiglia, M. Yatskevich, and E. Giunchiglia. Efficient semantic matching. In *Proceedings of ESWC*, 2005.
14. F. Giunchiglia, M. Yatskevich, and P. Shvaiko. Semantic matching: Algorithms and implementation. *Journal on Data Semantics*, IX, 2007.
15. R. Gligorov, Z. Aleksovski, W. ten Kate, and F. van Harmelen. Accurate and efficient html differencing. In *Proceedings of the 13th IEEE International Workshop on Software Technology and Engineering Practice (STEP)*, pages 163–172. IEEE Press, 2005.
16. R. Gligorov, Z. Aleksovski, W. ten Kate, and F. van Harmelen. Using google distance to weight approximate ontology matches. In *Proceedings of WWW*, 2007.
17. N. Gooneratne and Z. Tari. Matching independent global constraints for composite web services. In *In Proceedings of WWW*, pages 765–774, 2008.
18. W. Hu and Y. Qu. Block matching for ontologies. In *Proceedings of ISWC*, 2006.
19. Y. Kalfoglou and M. Schorlemmer. IF-Map: an ontology mapping method based on information flow theory. *Journal on Data Semantics*, I, 2003.
20. M. Klusch, B. Fries, and K. Sycara. Automated semantic web service discovery with OWLS-MX. In *Proceedings of AAMAS*, 2006.
21. Y. Lee, M. Sayyadian, A. Doan, and A. Rosenthal. eTuner: tuning schema matching software using synthetic scenarios. *VLDB Journal*, 16(1), 2007.
22. L. Li and I. Horrocks. A software framework for matchmaking based on semantic web technology. In *Proceedings of WWW*, 2003.
23. N. Noy, A. Doan, and A. Halevy. Semantic integration. *AI Magazine*, 26(1), 2005.
24. N. Noy and M. Musen. The PROMPT suite: interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies*, 59(6), 2003.
25. S. Oundhakar, K. Verma, K. Sivashanugam, A. Sheth, and J. Miller. Discovery of web services in a multi-ontology and federated registry environment. *Journal of Web Services Research*, 2(3), 2005.
26. M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Semantic matching of web services capabilities. In *Proceedings of ISWC*, 2002.
27. Dennis Shasha and Kaizhong Zhang. Approximate tree pattern matching. In *In Pattern Matching Algorithms*, pages 341–371. Oxford University Press, 1997.
28. P. Shvaiko and J. Euzenat. A survey of schema-based matching approaches. *Journal on Data Semantics*, IV, 2005.
29. U. Straccia and R. Troncy. oMAP: Combining classifiers for aligning automatically OWL ontologies. In *Proceedings of WISE*, 2005.
30. E. Stroulia and Y. Wang. Structural and semantic matching for assessing web-service similarity. *International Journal of Cooperative Information Systems*, 14(4):407–438, 2005.
31. K.-C. Tai. The tree-to-tree correction problem. *Journal of the ACM*, 26(3), 1979.
32. J. Tang, J. Li, B. Liang, X. Huang, Y. Li, and K. Wang. Using Bayesian decision for ontology mapping. *Journal of Web Semantics*, 4(1), 2006.
33. G. Valiente. *Algorithms on Trees and Graphs*. Springer, 2002.

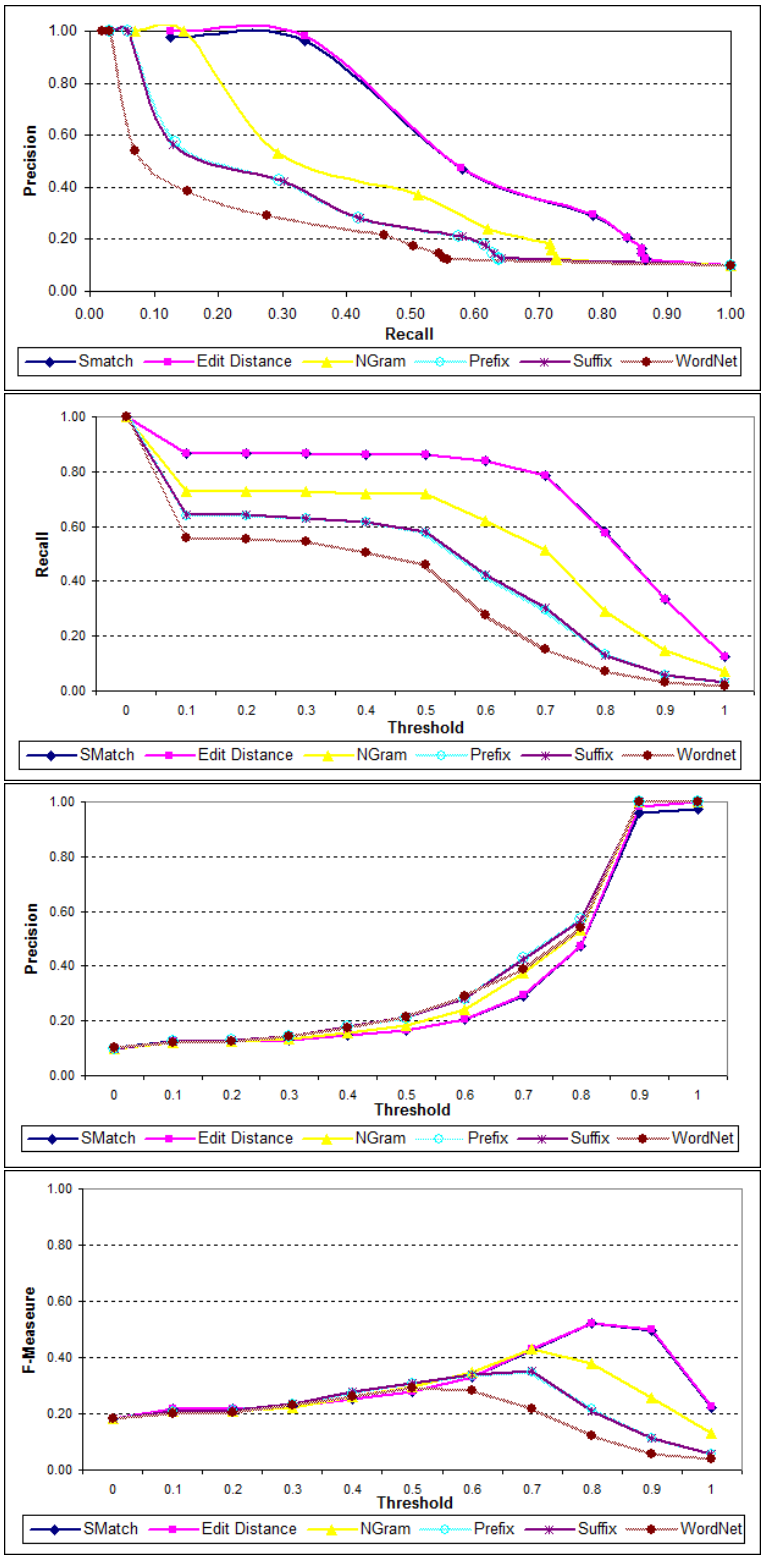


Fig. 3: Test case 2: Evaluation results for syntactic changes.

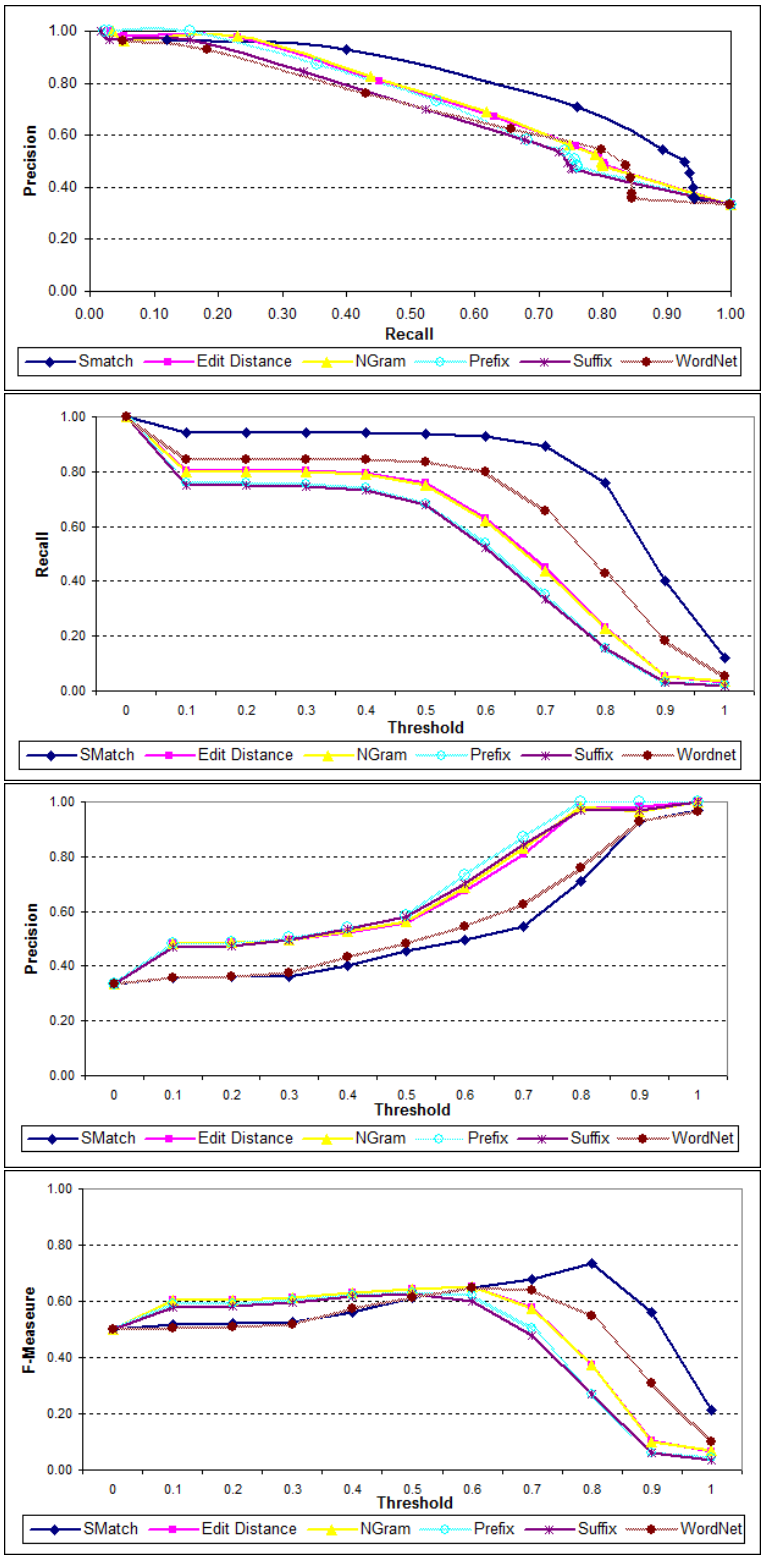


Fig. 4: Test case 2: Evaluation results for semantic changes.

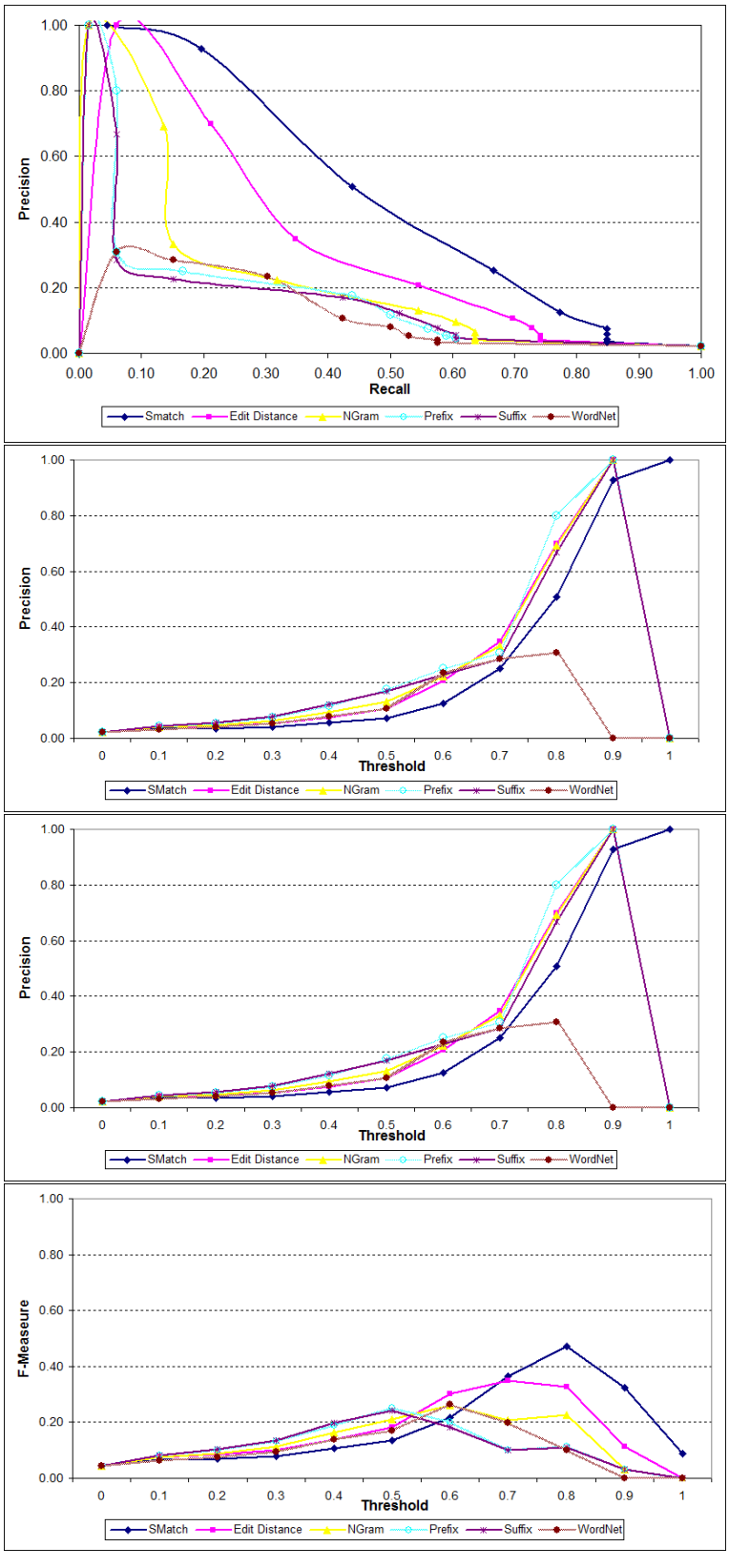


Fig. 5: Test case 2: Evaluation results for combined changes.

# An evaluation of approximate ontology matching in GIS applications

Lorenzino Vaccari[1], Pavel Shvaiko[2], Paolo Besana[3],
Maurizio Marchese[1], Juan Pane[1]

[1] DISI, University of Trento, Italy
{vaccari|pane|marchese@disi.unitn.it}
[2] TasLab, Informatica Trentina Spa, Italy
{pavel.shvaiko@infotn.it}
[3] University of Edinburgh, UK
{p.besana@sms.ed.ac.uk}

**Abstract.** Matching between concepts describing the meaning of representing for heterogeneous information sources is a key operation in many application domains, including web service coordination, data integration, peer-to-peer information sharing, query answering, and so on. In this paper we present an evaluation of an ontology matching approach, specifically of Structure-Preserving Semantic Matching (SPSM) solution which we applied within an emergency response scenario for GIS web service coordination. First we provide a detailed description and formalization of scenario under consideration. Then, we discuss the SPSM approach used to resolve the semantic heterogeneity problem among web services. We evaluate the SPSM solution both on a set of synthesized benchmarks, as well as on real world GIS ESRI ArcWeb services. The results demonstrate the robustness and the performance of the SPSM approach on a large number (ca. 700.000) of matching tasks.

**Keywords**. Ontology matching, ontology matching evaluation, semantic heterogeneity, GIS web services, SDI integration, similarity measurement.

## 1 Introduction

Geospatial semantics and ontologies are keep gaining an increasing attention within and outside geospatial information communities. The main reason is that there is an overwhelming necessity to share this information between different stakeholders, such as departments in public administration, professionals and citizens; thus, diverse information systems are tackling with interoperability issues, in order to obtain a coherent and contextual use of geographic information.

This necessity forms the basis for a number of international and national initiatives, to set up global, international, national and regional infrastructures for the collection and dissemination of geographical data. The key initiatives

include: the INfrastructure for SPatial InfoRmation in Europe (INSPIRE) directive[1], the Shared Environmental Information System (SEIS) initiative[2], the Water Information System for Europe (WISE) system[3], the Global Monitoring for Environmental and Security (GMES - Kopernikus) initiative[4] and the Global Earth Observation System of Systems (GEOSS)[5]. All these initiatives are based on the integration of geographic information locally maintained by Geographic Information Systems (GIS) and globally shared using Spatial Data Infrastructures (SDI) [3,17,27,24,31].

Based on these recent developments pushed by the above mentioned initiatives, access to geographic data has radically changed in the past decade. Previously, it was a specific task, for which complex desktop GIS were built and geographic data were maintained locally, managed by a restricted number of technicians. Thus, each organization maintained its local domain vocabulary of terms related to geographic features and relations among them. These systems were often based on semantics, sometimes even with an explicitly encoded, though not shared locally, defined semantics.

With the advent of the (Semantic) Web an increasing and different number of Geo Web services became available from different sources. Nowadays, the challenge is to discover and chain these services in order to obtain a user defined goal. Of course, each service provider publishes its services based on its background knowledge and discovering and chaining services requires a semantic interoperability level between different providers. As described in [19] , *in today's GIS service architectures, the interfaces between agents, computational and human, are those of web services...*and*...the interface of a service is formally captured by its signature.* If we consider signatures (name, inputs and outputs) of web services to be tree-like structures, we can use some of-the-shelf ontology matching systems to facilitate the resolution of the semantic heterogeneity problem.

In fact, ontology matching is often proposed as a solution to this heterogeneity problem in many applications, including integration of web services, and specifically GIS services. Ontology matching takes two graph-like structures such as, for instance, lightweight ontologies [9] and produces an alignment (set of correspondences) between the nodes of those graphs that correspond semantically to one another. Many solution to the ontology matching problem has been proposed so far (see [8,29,38] for recent surveys). In this paper we focus on a particular type of ontology matching, namely *Structure-Preserving Semantic Matching* (SPSM) and evaluate a concrete solution proposed in [10], on a set of ESRI ArcWeb services SOAP methods[6]. We base the evaluation, both on a set of synthesized benchmarks, as well as between services from real world GIS ESRI ArcWeb services.

[1] http://www.ec-gis.org/inspire/
[2] http://ec.europa.eu/environment/seis/index.htm
[3] http://ec.europa.eu/environment/water/index_en.htm
[4] http://www.gmes.info/index.php?id=home
[5] http://www.earthobservations.org/geoss.shtml
[6] http://www.arcwebservices.com/v2006/help/index.htm

The contributions of this paper include: *(i)* a description of an emergency response scenario focused on GIS web services coordination and *(ii)* an extensive evaluation of the SPSM approach using ESRI ArcWeb services in various settings. This paper is an expanded and updated version of an earlier conference paper [23], where the first contribution mentioned above was originally claimed and substantiated. The most important extension of this paper over the previous work in [23] includes an extensive evaluation of the SPSM matching approach on a set of real-world ESRI ArcWeb services.

The structure of the rest of the paper is as follows. We discuss the related work in Section 2. We introduce our application scenario and its formalization in Section 3. In Section 4 we outline a Structure-Preserving Semantic Matching approach used to reduce the semantic heterogeneity problem as required by the scenario. We provide the evaluation of the solution discussed in Sections 5 (dataset description), (evaluation method) and (evaluation results). Finally, we present the major findings of the paper as well as future work in Section 8.

## 2 Related Work

In this section we first review advances in the GIS semantic heterogeneity management for service integration (2.1). Then, we overview relevant ontology matching solutions in artificial intelligence, semantic web and database communities (2.2). Finally, we discuss evaluation efforts made so far in ontology matching evaluation (2.3), including evaluation of web service.

### 2.1 GIS web service management

Heterogeneity of GIS ontologies has been addressed in many works during the last decades, see, e.g., [30] and [42]. The importance of semantics in the GIS domain has been described in [21] where the authors presented an approach to ontology based GI retrieval. In the case of GIS Web services, in [22][7] the authors presented a rule-based description framework (a simple top-level ontology as well as a domain ontology) and an associated discovery and composition method that helps service developers to create such service chains from existing services. In [20] the authors developed a methodology that combines service discovery, abstract composition, concrete composition, and execution. In this approach, the authors presented the use of domain ontologies for the different steps in geographic service chaining.

For geo-services specific case a comparison between Business Process Execution Language (BPEL)[8] and Web Services Modeling Ontology (WSMO)[9] approaches has been made in [16]. In this work the authors proposed semantic web service composition using WSMO as an improvement of BPEL limitations.

[7] ORCHESTRA project, http://www.eu-orchestra.org/

[8] http://www.oasis-open.org/apps/group_public/download.php/23974/wsbpel-v2.0-primer.pdf

[9] http://www.wsmo.org/

The work in [43] proposed a toolset to compose geo-web services using BPEL. In turn, the work in [40] combined WSMO and IRS-III for semantically composing geo-spatial web services. The composition of geo-web services is analyzed also in in [6], where the authors used OWL [2], OWL-S[10] and BPEL to give meaning to diverse data sources and geo-processing services. To chain and discover geo-web services the authors applied an OWL reasoner as the inference engine for the knowledge-base.

Most of the previous solutions employs a single ontology approach, that is, the web services are assumed to be described by the concepts taken from a shared ontology [20,16,40]. This allows for the reduction of the matching problem to the problem of reasoning within the shared ontology [18,33]. Moreover, the adoption of a common ontology for the geographic information communities is not practical, because the development of a common ontology has proven to be difficult and expensive [**?**]. In contrast, following the works in [1,32,39], we assume that web services are described using terms from different ontologies and that their behavior is described using complex terms. The problem becomes therefore that of matching two web service descriptions, which in turn, can be represented as tree-like structures [10].

### 2.2 Ontology matching

A substantial amount of work that tackles the problem of semantic heterogeneity has been done in the semantic web, artificial intelligence and database domains, where ontology matching is viewed as a plausible solution, see, e.g., [28], [38] for recent surveys, while examples of individual approaches addressing the matching problem can be found on http://www.OntologyMatching.org. These solutions take advantage of the various properties of ontologies[11] (e.g., labels, structures) and use techniques from different fields (e.g., statistics and data analysis, machine learning, linguistics). These solutions share some techniques and attack similar problems, but differ in the way they combine and exploit their results. A detailed analysis of the different techniques in ontology matching has been given in [8].

The most similar to the solution that we used in our scenario are the approaches taken in [1,32,39], where the services are assumed to be annotated with the concepts taken from various ontologies. The matching algorithms of those works combine the results of atomic matchers that roughly correspond to the element level matchers[12] exploited as part of the work in [15] and [10] which we applied in our scenario.

[10] http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/

[11] The term ontology is used here in a wide sense, and, hence, encompasses sets of terms, classifications, database schemas, thesauri, etc.

[12] Element level matching techniques compute correspondences by analyzing concepts in isolation, ignoring their relationships with other concepts. In turn, structure level matching techniques compute correspondences by exploiting the results of element level matchers and by analyzing relationships between concepts.

### 2.3 Ontology matching evaluation

The ontology matching evaluation theme has been given a chapter account in [8]. There are several individual approaches to the evaluation of matching approaches in general, see, e.g., [13], as well as with web services in particular, see, e.g., [39,25,32]. Beside, there are annual Ontology Alignment Evaluation Initiative (OAEI) campaigns [4,7]. OAEI is a coordinated international initiative that organizes the evaluation of the increasing number of ontology matching systems. The main goal of OAEI is to support the comparison of the systems and algorithms on the same basis and to allow anyone to draw conclusions about the best matching strategies. Unfortunately, at present, matching of web services has not been addressed by OAEI. In turn, there is the Semantic Web Service (SWS) challenge initiative which aims at evaluation of various web service mediation approaches [35]. However, as also noticed in [37], the key problem with the current evaluations of web service matching approaches is the lack of real world web service data sets as well as their size, for example as from [35], the participants of SWS were operating with 20 web services.

The goal of this paper is the evaluation of the SPSM approach on real world GIS web services. Specifically, we conduct two kinds of experiments: *(i)* an evolution experiment with ca. individual 1.600 matching tasks (that we repeat for varying thresholds parameters and strength of the alteration operations) where we deal with meaning and syntactic robustness tests, and *(ii)* a classification robustness experiment with 50 matching tasks, where we compare the results of an automatic and a manual classification of a selected set of GIS ESRI ArcWeb service operations.

## 3 Application scenario

Emergency management activities, that in the following we will reference as emergency response (eResponse) activities, are developed and implemented by means of the essential analysis of information and the coordination of the involved actors, including emergency personnel, army, volunteers, etc. Disaster data and events can be acquired, modeled, fused and displayed in state-of-the-art SDIs using distributed data sources, the majority of which are spatial. SDIs are pervaded by interoperability problems, including: *(i)* (geo) data interoperability, specifically, geographical datasets have particular properties (e.g., maps as implicit interfaces) to be handled, which are different from other types of data (see [34]) and *(ii)* web-service interoperability issues, such as discovery, composition and coordination [21].

We have analyzed the organizational model of the distributed GIS Agency infrastructure of Trentino, Italy. The framework is represented by a number of specialized GIS agencies: civilian protection, urban planning, forestry, roads, agriculture, cadastral, environment, and geologic survey. Each GIS agency is responsible for providing a subset of the geographic information for the local region. To support interoperability among different GIS agencies the regional

information infrastructure is shifting from a traditional GIS system to a distributed SDI.

Within the general Trentino SDI management scenario, we focus here - for descriptive purposes - on the most commonly used specific use case, i.e., Map Request Service. Let us discuss one particular, but the most typical, request that can be made by a service requestor peer of a GIS agency: a digital map request. The service requestor, both in an emergency or normal situation, needs to visualize a map of a region with geo-referenced information. Usually, the searched map is a composition of different geographic layers offered by a GIS service provider. A simplified interaction for the Map Request Service is illustrated in Figure 1.

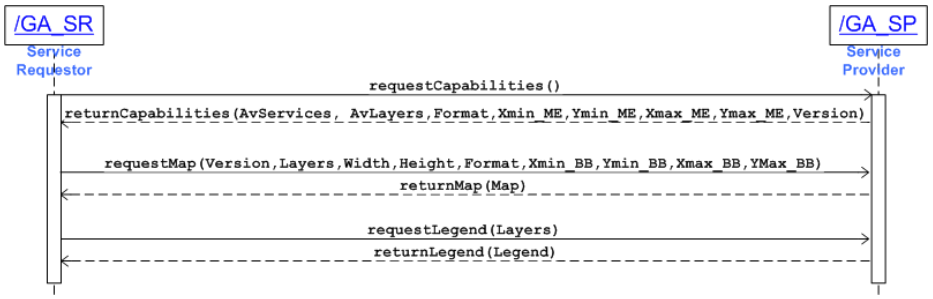


**Fig. 1.** Map request service.

Interactions between a map service requestor and a map service provider are briefly described as follows:

- The requestor (GIS Agency Service Requestor, *GA_SR*) asks the provider (GIS Agency Service Provider, *GA_SP*) for the characteristics of the provided services (*requestCapabilities()*).
- *GA_SP* returns its characteristics (*returnCapabilities*), in particular: the list of available services (*AvServices*), the list of geographic datasets managed by the server (*AvLayers*), the file format of the returned services (*Format*), the geographic bounds of the available services (*XMin_ME, YMin_ME, XMax_ME, YMax_ME*) and the software version (*Version*) of the adopted service.
- Then, *GA_SR* asks for the map service (*requestMap*) using the information received from the previous step . This message contains the software version of the adopted service (*Version*), the requested geographic layers (*Layers*, a subset of the *AvailLayers*), the image dimension of the map (*Width, Heigth*), the image format of the map (*Format*) and the spatial coverage of the map (*XMin_BB, YMin_BB, Xmax_BB, YMax_BB*).
- *GA_SP* provides the map (*return(Map)*) requested by the requestor.

– Finally, *GA_SR* asks for the graphic legend that describes the previous map (*requestLegend(Layers)*) and *GA_SP* sends the legend (*returnLegend(Legend)*) to *GA_SR*.

### 3.1 Use case formalization

In this subsection, we first describe the Lightweight Coordination Calculus (LCC) [36], the choreography language employed to specify the interactions among participants of our scenario. Then, we focus on the selected coordination problem depicted in the previous subsection and provide its formalization with LCC.

**LCC basics.** LCC is a protocol language used to describe interactions among distributed processes, e.g., agents and web services. LCC was designed specifically for expressing P2P style interactions within multi-agent systems, i.e., without any central control; therefore, it is well suited for modeling coordination (choreography) of software components running in an open environment. The applicability of P2P style protocol modeling, such as LCC, in the eResponse domain is driven by its potential to deal with a distributed-knowledge and dynamic environment.

Interactions in LCC are expressed as message passing behaviors associated with roles. The most basic behaviors are to send or receive messages, where sending a message may be conditional on satisfying a constraint (pre-condition) and receiving a message may imply constraints (post-condition) on the peer accepting it.

$$
\begin{array}{l}
a(r1, A1) :: \\
\quad ask(X) \Rightarrow a(r2, A2) \leftarrow need(X)\ then \\
\quad update(X) \leftarrow return(X) \Leftarrow a(r2, A2) \\
\\
a(r2, A2) :: \\
\quad ask(X) \Leftarrow a(r1, A1)\ then \\
\quad return(X) \Rightarrow a(r1, A1) \leftarrow get(X)
\end{array}
$$

**Fig. 2.** LCC example: double arrows ($\Rightarrow, \Leftarrow$) indicate message passing, single arrow ($\leftarrow$) indicates constraint satisfaction.

A basic LCC interaction is shown in Figure 2. The peer identified by the value of the variable *A1* playing the role *r1* verifies if it needs the info *X* (pre-condition *need(X)*); if it does, *A1* asks the peer identified by the value of the variable *A2* for *X* by sending the message *ask(X)*. *A2* receives the message, *ask(X)* from *A1* and then obtains the info *X* (pre-condition *get(X)*) before sending back a reply to *A1* through the message *return(X)*. After having received the message *return(X)*, *A1* updates its knowledge (post-condition *update(X)*). The constraints embedded into the protocol express its semantics and could be written as first-order logic predicates (e.g., in Prolog) as well as methods in an object-oriented language (e.g., in Java). The characteristic of modularity allows separating the protocol

from the agent engineering. While performing the protocol, peers can therefore exchange messages, satisfy constraints before/after messages are sent/received and jump from one role to another so that a flexible interaction mechanism is enabled still following a structured policy, which is absolutely necessary for team-execution of coordinated tasks.

**Use case formalization with LCC.** Figure 3 and Figure 4 show the LCC code for the interaction model depicted in Figure 1. The interaction model contains the interactions from the viewpoint of a GIS agency (map) service Requestor (*ga_sr*, Figure 3) and of the GIS agency (map) service provider (*ga_sp*, Figure 4).

– Specifically, in Figure 3, the GIS agency service requestor (*ga_sr*) asks the service provider (*ga_sr*) its capabilities.

$$
\begin{array}{l}
a(ga\_sr, R) :: \\
\quad requestCapabilities() \Rightarrow a(ga\_sr, R)\ then \\
\quad returnCapabilities(AvailableServices, AvailableLayers, Format, \\
\qquad XMin\_ME, YMin\_ME, XMax\_ME, YMax\_ME, Version) \\
\qquad \Leftarrow a(ga\_sp, P)\ then \\
\quad requestMap(Version, Layers, Width, Height, Format \\
\qquad XMin\_BB, YMin\_BB, XMax\_BB, YMax\_BB) \Rightarrow a(ga\_sp, P) \\
\qquad\quad \leftarrow selectLayers(AvailableLayers, Layers) \wedge needMap(Width, Height) \wedge \\
\qquad\quad selectBoundingBox(XMin\_ME, YMin\_ME, XMax\_ME, YMax\_ME, \\
\qquad\qquad XMin\_BB, YMin\_BB, XMax\_BB, YMax\_BB)\ then \\
\quad returnMap(Map) \Rightarrow a(ga\_sp, P)\ then \\
\quad requestLegend(Layers) \Rightarrow a(ga\_sp, P)\ then \\
\quad returnLegend(Legend) \Leftarrow a(ga\_sp, P)
\end{array}
$$

**Fig. 3.** LCC fragment for the GIS agency service requestor role.

After that, the service requestor waits until the service provider returns the list of the available services (*AvailableServices*), the list of the available layers (*AvailableLayers*), the format of the returned file (*Format*), and the geographic coverage of the available services (map extent, *XMin_ME, YMin_ME, Xmax_ME, YMax_ME*). Then the map requestor asks the service provider for a map. It selects some of the available geographic layers (*selectLayers(AvailableLayers, Layers)*), defines the map dimension (*needMap(Width, Height)*) and selects an area from the available geographic extension (*selectBoundingBox(XMin_ME, YMin_ME, XMax_ME, YMax_ME, XMin_BB, YMin_BB, XMax_BB, YMax_BB)*). Finally it requests the map legend of the selected layers (*requestLegend(Layers)*).

– In Figure 4 the GIS agency service provider (*ga_sp*) waits for one of the following requests: *requestCapabilities*, *requestMap* and *requestLegend*. After receiving one of them, it performs, respectively, the following actions:
  - it builds its capabilities (*getCapabilities(MapFile, Version, AvailableServices, AvailableLayers, Format, Xmin_ME, YMin_ME, Xmax_ME, YMax_ME)* constraint) and passes them to the requestor;
  - it builds a digital map (*getMap* constraint) and sends it to the service requestor;

$$
\begin{array}{l}
a(ga\_sp, P) :: \\
\quad \left( \begin{array}{l}
requestCapabilities() \Leftarrow a(ga\_sp, P)\ then \\
returnCapabilities(AvailableServices, AvailableLayers, Format, \\
\quad XMin\_ME, YMin\_ME, XMax\_ME, YMax\_ME, Version) \\
\quad \Rightarrow a(ga\_sr, R) \\
\quad \leftarrow getCapabilities(Version,\ AvailableServices,\ AvailableLayers, \\
\quad\quad Format,\ XMin\_ME,\ YMin\_ME,\ XMax\_ME,\ YMax\_ME)
\end{array} \right) \ or \\
\quad \left( \begin{array}{l}
requestMap(Version, Layers, Width, Height, Format, \\
\quad XMin\_BB, YMin\_BB, XMax\_BB, YMax\_BB) \Leftarrow a(ga\_sr, R)\ then \\
returnMap(Map) \Rightarrow a(ga\_sr, R) \\
\quad \leftarrow \underline{getMap(Version,\ Layers,\ Width,\ Height,\ Format,} \\
\quad\quad \underline{XMin\_BB,\ YMin\_BB,\ XMax\_BB,\ YMax\_BB)}
\end{array} \right) \ or \\
\quad \left( \begin{array}{l}
requestLegend(Layers) \Leftarrow a(ga\_sr, R)\ then \\
returnLegend(Legend) \Rightarrow a(ga\_sr, R) \\
\quad \leftarrow getLegend(Layers, Legend)
\end{array} \right)
\end{array}
$$

**Fig. 4.** LCC fragment for the GIS agency service provider role.

- it builds a legend of the requested layers (*getLegend(Layers, Legend)* constraint) and returns it to the service requestor.

In the following section we will use *getMap* constraint (underlined in Figure 4) as part of the motivating example to the matching approach employed.

**The emergency GUI.** In order to complete the scenario, let us to describe, as an example, how the detailed procedures and their formalization discussed previously are actually used by final users. To this end, we developed an initial prototype in which the coordination of the web services between the network peers can be executed, visualised and analysed. In this prototype, the ongoing simulation of an emergency situation and the resulting map acquired by the interaction proposed in the previous subsection, together with movements of the emergency peers are visualized through a GUI as shown in Figure 5.

The GUI represents a control panel used by the emergency coordinators. Through the emergency GUI the users can visualize the map and the events of the emergency situation. The map shows static geographic datasets (topographic map, probable flooding areas, escape roads, meeting points, refuge centers, sensor network) as well as dynamic datasets (e.g. Figure 5 shows the position of the firefighters involved in the simulation). Moreover, through the GUI, emergency coordinators can perform actions (enact the emergency plan, recall digital services, change the map legend, search for other GIS datasets, send statements to the emergency peers, etc) as well as ask information about the emergency situation (i.e., evacuated people, list of the emergency peers, blocked roads, situation of the meeting points and of the refuge centers, etc).

## 4 Structure-Preserving Semantic Matching

In our scenario peers are selected at run time and they can change every time. Let us suppose that we want to match a constraint on a role, such as: *getMap(Version, Layers, Width, Height, Format, XMin_BB, YMin_BB, XMax_BB, YMax_BB)* (*T1* in Figure 6, see also highlighted in Figure 4) with the capabilities of a peer

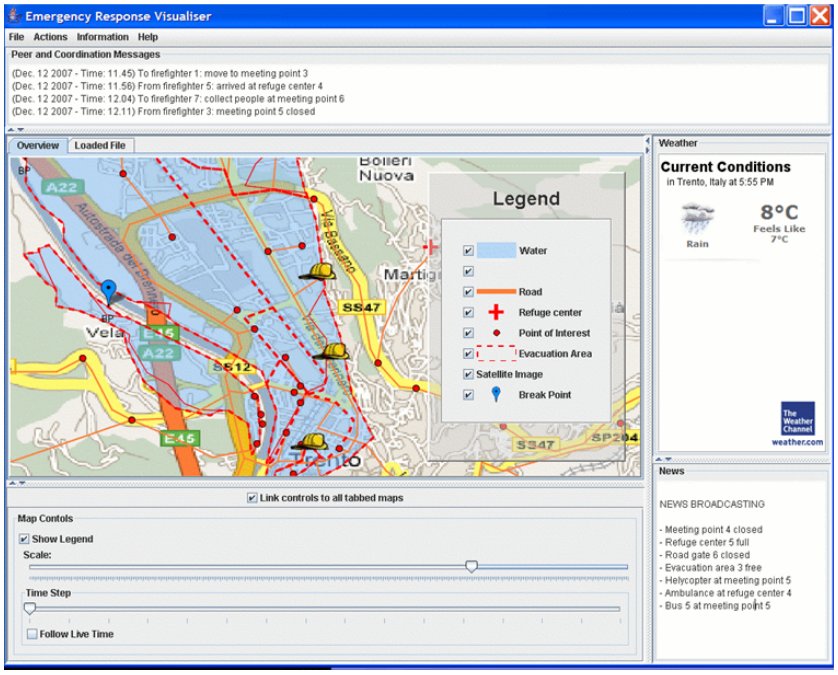


**Fig. 5.** Emergency response visualiser.

willing to play that role, such as: *getMap(Dimension(Width, Height), Edition, Layers, DataFormat, Request, Xmin, Ymin, Xmax, Ymax)* (*T2* in Figure 6). These can be also viewed as a web service operation descriptions, which in turn, can be represented as tree-like structures.

As shown in Figure 6, the first description requires the second argument of *getMap* function (*Layers*) to be matched to the fourth one (*Layers*) of *getMap* function in the second description. The value of *Version* in the first description must be passed to the second web service operation as the *Edition* argument. Moreover, *Request* (this parameter indicates which web service operation (map service, download service, etc) is being invoked) in *T2* has no corresponding term in *T1*.

The purpose of Structure-Preserving Semantic Matching is to reduce semantic heterogeneity in peer role descriptions. Specifically, a semantic similarity measure is used to estimate similarity between peer role descriptions under consideration. This scenario poses additional constraints on conventional ontology matching. In particular, we need to compute the correspondences holding among the full graph structures and preserve certain structural properties of the graphs under consideration. Thus, the goal here is to have a Structure-Preserving Semantic Matching operation. This operation takes two graph-like structures and produces a set of correspondences between those nodes of the graphs that correspond semantically to one another, *(i)* still preserving a set of structural proper-

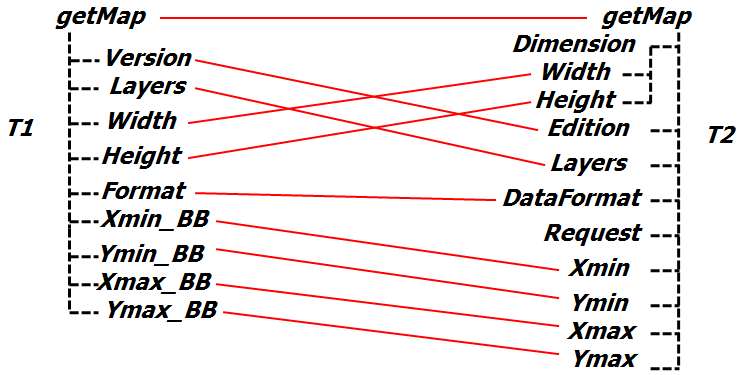


**Fig. 6.** Two web service descriptions (trees) and correspondences (lines) between them.

ties of the graphs being matched, namely that functions are matched to functions and variables to variables; and *(ii)* only in the case if the graphs are globally similar to one another, e.g., $graph_1$ is 0.65 similar to $graph_2$ according to some measure.

### 4.1 The approach

We briefly report here the approach (see [10] for the in-depth discussion) and present it with the help of examples from the GIS domain. We focus on tree-like-structures, see Figure 6. The SPSM matching process is organized in two steps: *(i)* node matching and *(ii)* tree matching.

Node matching tackles the semantic heterogeneity problem by considering only labels at nodes and domain specific contextual information of the trees. SPSM uses the *S-Match* system as proposed in [15]. Technically, two nodes *n1* and *n2* in trees *T1* and *T2* match if and only if: *c@n1* $R$ *c@n2* holds based on S-Match. *c@n1* and *c@n2* are the concepts, that represent entities of the local ontologies, at nodes *n1* and *n2* and $R \in \{=, \sqsubseteq, \sqsupseteq,$ *"not related"*$\}$. In particular, in semantic matching [11] as implemented in the *S-Match* system the key idea is that the relations (e.g., $=, \sqsubseteq$) between nodes are determined by *(i)* expressing the entities, that is the concepts, of the ontologies as logical formulas and *(ii)* reducing the matching problem to a logical validity problem.

Specifically, the concepts are translated into logical formulas which explicitly express the concept descriptions as encoded in the ontology structure and in external resources, such as WordNet [26]. This allows for a translation of the matching problem into a logical validity problem, which can then be efficiently resolved using sound and complete state-of-the-art satisfiability solvers [14]. Notice that the result of this stage is the set of correspondences holding between the nodes of the trees. For example, that *getMap* and *Version* in *T1* correspond to *getMap* and *Edition* in *T2*, respectively.

Tree matching, in turn, exploits the results of the node matching and the structure of the trees to find if these globally match each other. Technically, two trees *T1* and *T2* approximately match if and only if there is at least one node $n2_i$ in *T1* and node $n2_j$ in *T2* such that: *(i)* $n1_i$ matches $n2_j$, *(ii)* all ancestors of $n1_i$ are matched to the ancestors of $n2_j$, where $i = 1 \dots N1; j = 1 \dots N2; N1$ and $N2$ are the number of nodes in *T1* and *T2*, respectively.

### 4.2 The implementation

The implementation of SPSM is based on *(i)* a formal theory of abstraction and *(ii)* a tree edit-distance algorithm [41]. Let us present, in the following, a brief summary of them.

**Abstraction operations** . The work in [12] categorizes the various kinds of abstraction operations, including:

**Predicate (Pd)** : two or more predicates are merged, typically to the least general generalization in the predicate type hierarchy, e.g., $Height(X) + Dimension(X) \rightarrow Dimension(X)$. We call $Dimension(X)$ a predicate abstraction of $Height(X)$, namely $Dimension(X) \sqsupseteq_{Pd} Height(X)$. Conversely, we call $Height(X)$ a predicate refinement of $Dimension(X)$, namely $Height(X) \sqsubseteq_{Pd} Dimension(X)$.

**Domain (D)** : two or more terms are merged, typically by moving constants to the least general generalization in the domain type hierarchy, e.g., $Xmin\_BB + Xmin \rightarrow Xmin$. We call $Xmin$ a domain abstraction of $Xmin\_BB$, namely $Xmin \sqsupseteq_D Xmin\_BB$. Conversely, we call $Xmin\_BB$ a domain refinement of $Xmin$, namely $Xmin\_BB \sqsubseteq_D Xmin$.

**Propositional (P)** : one or more arguments are dropped, e.g., $Layers(L1) \rightarrow Layers$. We call $Layers$ a propositional abstraction of $Layers(L1)$, namely $Layers \sqsubseteq_P Layers(L1)$. Conversely, $Layers(L1)$ is a propositional refinement of $Layers$, namely $Layers(L1) \sqsubseteq_P Layers$.

Let us consider the following example: $(Height(H))$ and $(Dimension)$. In this case there is no abstraction/refinement operation that makes those first-order terms equivalent. However, consequent applications of propositional and domain abstraction operations make the two terms equivalent: $Height(X) \sqsubseteq_P Height \sqsubseteq_D Dimension$.

The abstraction/refinement operations discussed above preserve the desired properties: that functions are matched to functions and variables to variables. For example, predicate and domain abstraction/refinement operations do not convert a function into a variable. Thus, for instance, the correspondences between $Height$ (variable) and $Width$ (variable) in $T1$ and $Dimension$ (function) in $T2$, although returned by the node matching, should be further discarded, and therefore, are not shown in Figure 6.

**Global similarity measurement** . The key idea is to use abstractions / refinements as allowed tree edit-distance operations in order to estimate the similarity of two tree structures. Tree edit-distance is the minimum number of tree edit operations, namely node *insertion*, *deletion*, *replacement*, required to transform one tree to another. The goal is to: *(i)* minimize the editing cost, i.e., computation of the minimal cost composition of abstractions/refinements, *(ii)* allow only those tree edit operations that have their abstraction theoretic counterparts in order to reflect semantics of the first-order terms.

A uniform proposal here is to assign an empirical unit cost (see Table 1) to all operations that have their abstraction theoretic counterparts, while operations not allowed by definition of abstractions/refinements are assigned an infinite cost. The following three relations between trees are considered: $T1 = T2$, $T1 \sqsubseteq T2$, and $T1 \sqsupseteq T2$. A global similarity (*TreeSim*) between two trees $T1$ and $T2$ ranges in $[0 \dots 1]$ and is computed as follows:

$$TreeSim(T1, T2) = 1 - \frac{min \sum_{i \in S} n_i \cdot Cost_i}{max(N1, N2)} \quad (1)$$

where, $S$ is the set of allowed tree edit operations, $n_i$ is the number of *i-th* operation necessary to convert one tree into the other, $Cost_i$ is the cost of the *i-th* operation. The minimal edit-distance is normalized by the size of the biggest tree. Finally, a normalized distance (denoting dissimilarity) is converted into a similarity score. When $Cost_i$ is infinite (see Table 1), $TreeSim$ is estimated as zero.

The highest value of $TreeSim$ among $T1 = T2$, $T1 \sqsubseteq T2$, and $T1 \sqsupseteq T2$ is returned as the final similarity score. For the example of Figure 6, 10 node-to-node correspondences, namely 6 equivalence and 4 abstraction/refinement relations, were identified by the node matching algorithm. The biggest tree is $T2$ with 12 nodes. Then, these are used to compute $TreeSim$ between $T1$ and $T2$ by exploiting the above mentioned formula. In our example $TreeSim$ is 0.54 for both $T1 = T2$ and $T1 \sqsubseteq T2$ (while it is 0 for $T1 \sqsupseteq T2$). The tree similarity value could be useful to select trees whose similarity value is greater than a cut-off threshold. In our example $Treesim$ is higher than the cut-off threshold of 0.5, and, therefore, the two trees globally match as expected and the correspondences connecting the nodes of the term trees are further used for data translation purposes.

## 5 The GIS web service evaluation dataset

The SPSM solution allows that the web services are described, in the corresponding WSDL files and eventually in other formats, such as OWL-S and WSMO. However, until actual services with such semantic specifications are not published, we limit our evaluation to the names of the WSDL SOAP methods (operations) and of their parameters as carriers of meaningful information about the behavior and the semantics of the services. The SPSM approach thus assumes that

**Table 1.** The correspondence between abstraction operations, tree edit operations and costs.

| **Abstractions** | **Operation** | **Preconditions** | $Cost_{T1=T2}$ | $Cost_{T1 \sqsubset T2}$ | $Cost_{T1 \sqsupset T2}$ |
|---|---|---|---|---|---|
| $t_1 \sqsupseteq_{Pd} t_2$ | $replace(a, b)$ | $a \sqsupseteq b$; $a$ and $b$ correspond to predicates | 1 | $\infty$ | 1 |
| $t_1 \sqsupseteq_{D} t_2$ | $replace(a, b)$ | $a \sqsupseteq b$; $a$ and $b$ correspond to functions or constants | 1 | $\infty$ | 1 |
| $t_1 \sqsupseteq_{P} t_2$ | $insert(a)$ | $a$ corresponds to predicates, functions or constants | 1 | $\infty$ | 1 |
| $t_1 \sqsubseteq_{Pd} t_2$ | $replace(a, b)$ | $a \sqsubseteq b$; $a$ and $b$ correspond to predicates | 1 | 1 | $\infty$ |
| $t_1 \sqsubseteq_{D} t_2$ | $replace(a, b)$ | $a \sqsubseteq b$; $a$ and $b$ correspond to functions or constants | 1 | 1 | $\infty$ |
| $t_1 \sqsubseteq_{P} t_2$ | $delete(a)$ | $a$ corresponds to predicates, functions or constants | 1 | 1 | $\infty$ |
| $t_1 = t_2$ | $a = b$ | $a = b$; $a$ and $b$ correspond to predicates, functions or constants | 0 | 0 | 0 |

the web services described in WSDL will be annotated with some kind of meaningful descriptions of: *(i)* what the operations are (e.g. *find_Address_By_Point*); *(ii)* what the inputs and outputs are: i.e. that arguments are labeled descriptively and not merely as "*input1*", "*var1*" and so on. Any additional mark-up that is used to provide semantics for web services outside of the WSDL files can also be amenable to our techniques, provided, as is usually the case, that descriptions of inputs and outputs can be captured in a tree structure.

In our scenario, we base our test cases on ESRI ArcWeb WSDL operations and we compare labeled trees that correspond to their signatures.

We conducted two different kinds of tests. The first one has been inspired by the work on systematic benchmarks of the Ontology Alignment Evaluation Initiative (OAEI) [7][13]. In this experiment we matched original labeled trees to synthetically alterated trees. Moreover, we compared the performance of the SPSM algorithm against the performance of a baseline algorithm, such as edit-distance[14]. In the second experiment we compared a manual classification of our GIS ArcWeb services dataset, the so-called "reference alignment", to the unsupervised one discovered by SPSM.

Finally, we evaluated efficiency and quality of the results of SPSM matching solution on these test cases. The evaluation was performed on a standard laptop Intel Centrino Core Duo CPU-2Ghz, 2GB RAM, with the Windows Vista (32bit,

[13] http://oaei.ontologymatching.org/2006

[14] The edit-distance between two strings is given by the minimum number of operations needed to transform one string into the other, where an operation is an insertion, deletion, or substitution of a single character

SP1) operating system, and with no applications running but a single matching system.

### 5.1 Test case 1: evolution scenario

Ontology and web service engineering practices suggest that often the underlying trees to be matched are derived or inspired from one another using different kind of operations to change the meaning and the syntax of the original tree [10]. Therefore, it is reasonable to compare a tree with another one (derived from the original tree). We evaluated SPSM following this scenario in which we performed syntactical and semantical alterations to the nodes in trees, with a random probability ranging in $[0.1 \ldots 0.9]$.

The evaluation dataset was composed of trees that are alterations of several original trees. Initially, 80 "original" trees were extracted from the ESRI ArcWeb services collection. Some examples include:

- *find_Address_By_Point( point, address_Finder_Options, part)*,
- *get_Distance(location1, location2, num_Points, return_Geometry, token, units)* and
- *convert_Map_Coords_To_Pixel_Coords( map_Area, map_Size, map_Coords, token).*

Then, 20 altered tree were automatically generated for each original tree. Pairs composed of the original tree and one varied tree, thereby resulting in 1600 matching tasks, were then used as input by the SPSM solution. The alteration operations were applied to node names (node names being composed of labels, as discussed in Section 4), and correspond to the following four alteration categories (the underscored labels indicate modifications):

1. **Replace a node name with an unrelated node name**: a node name was replaced with an unrelated node name randomly selected from a generic dictionary. In our test we used the Brown corpus[15], a standard corpus of present-day American English. Some examples include:
   - Original tree:
     *find_Address_By_Point(point, address_Finder_Options, part)*
   - Modified tree:
     *find_Address_By_Point( atom_firmer, discussion, part)*
2. **Add or remove a label in a node name**: the label of a node name was either dropped or added. A label was dropped only if the node name contained more than one label. Label addition in node names was performed by using words extracted from the Brown corpus. Some examples include:
   - Original tree:
     *find_Address_By_Point(point,address_Finder_Options,part)*
   - Modified tree:
     *find_By_Point(toast_point, address_Milledgeville_Finder_Options, part)*

[15] http://icame.uib.no/brown/bcm.html

3. **Alter syntactically a label**: this test aimed at mimicking potential misspellings of the node labels. First we decided randomly whether or not to modify a node name. Then, we randomly generated the number of labels to be modified and, for each word, we randomly decided how to modify it by using three types of alterations: character dropped, added, or changed. Some examples include:
   - Original tree:
     *find_Address_By_Point(point,address_Finder_Options,part)*
   - Modified tree:
     *finm_Address_By_Poioat(einqt,ddress_Finder_Optxions,vparc)*
4. **Replace a label in a node name with a related (e.g. synonyms, hyponyms, hypernyms) one**: this test aimed at simulating the selection of an operation which meaning was similar (equivalent, more general or less general) to the original one. In the implementation of these type of alterations we used a number of generic sources like WordNet 3.0 and Moby[16]. Some examples include:
   - Original tree:
     *find_Address_By_Point(point,address_Finder_Options,part)*
   - Modified tree:
     *locate_Address_By_Point(sail, address_Finder_Options, part)*

We implemented evaluation tests to explore the robustness of the SPSM approach towards both typical syntactic alterations (i.e. replacements of node names, modification of node names and misspellings) and typical meaning alterations (i.e. usage of related synonyms, hyponyms, hypernyms) of node names.

### 5.2 Test case 2: classification scenario

In this test case, we aimed at investigating the capability of the SPSM algorithm in the unsupervised clustering of a set of meaningfully related web service operations. The evaluation setup corresponds to a manual classification (reference alignment) of a selected set (50) of ArcWeb service operations. The rest of the previously discussed 80 operations were dropped during the second step of the construction procedure. The construction of the reference alignment included the following steps.

1. Classify manually the set of operation conforming to the WSDL file description of the operations.
2. Eliminate some general (valid for all the group) operations, e.g., *get_Info (data_Sources, token)*; these operations did not contribute to operation-specific information to the classification process.
3. Refine the classification by logically regrouping some operations, e.g., *find_Place(place_Name, place_Finder_Options, token)* was grouped with the *"address finder"* set of operations.

[16] http://www.mobysaurus.com

We thus compared the constructed manual classification of the selected Web Service operations with the automatically obtained by SPSM approach. Specifically, we:

- compared each operation signature with all other one signatures;
- computed a similarity measure between each signature and all the other signatures;
- classified the operations depending on a given similarity threshold.

## 6 Evaluation method

We used standards measures as precision, recall and F-measure to evaluate quality of the SPSM matching results[8]. Specifically, for both the experiments, we based calculation of these measures on the comparison of the correspondences produced by a matching system ($R$) with the reference correspondences considered to be correct ($C$). We also define the sets of true positives ($TP$), false positives ($FP$) and false negatives ($FN$), as, respectively, the set of the correct correspondences which have been found, the set of the wrong correspondences which have been found and the set of the correct correspondences which have not been found. Thus:

$$R = TP \cup FP \tag{2}$$

$$C = TP \cup FN \tag{3}$$

Precision, recall and F-measure are defined as follows:

- *Precision:* varies in the $[0 \dots 1]$ range; the higher the value, the smaller the set of false positives which have been computed. Precision is a measure of correctness and it is computed as follows:

$$Precision = \frac{\mid TP \mid}{\mid R \mid} \tag{4}$$

- *Recall:* varies in the $[0 \dots 1]$ range; the higher the value, the smaller the set of true positives which have not been computed. Recall is a measure of completeness and it is computed as follows:

$$Recall = \frac{\mid TP \mid}{\mid C \mid} \tag{5}$$

- *F-measure:* varies in the $[0 \dots 1]$ range; it is global measure of the matching quality, which increases if the matching quality increases. The version presented here was computed as a harmonic mean of precision and recall:

$$F - measure = \frac{2 * Recall * Precision}{Recall + Precision} \tag{6}$$

### 6.1 Test case 1 (evolution scenario) evaluation set-up.

Since the generated tree alterations were known, these provided the ground truth, and hence, the reference results were available by construction (see also [7]). This allowed for the computation of the matching quality measures. In particular, we computed the standard matching quality measures, such as precision, recall, and F-measure for the similarity between trees. In the computation of the quality measures, the proposed SPSM considered discovered correspondences between first-order terms rather than the nodes of the term trees. Thus, for instance:

- *get_Standard_Geography_Report(standard_Geographies) = pickup(standard_Geographiejournal)*

was considered as a single correspondence rather than two correspondences, namely:

- *get_Standard_Geography_Report = pickup*

and

- *standard_Geographies = standard_Geographiejournal*

Moreover, we assigned to each node a label that described the type of the relation with the original one. Initially we set the value of similarity to 1 and the value of the relation to "equivalent". Then each alteration operation, applied sequentially to each node, reduced the similarity value and changed the relation value. We changed the rate of the reduction and the value of the relation according to the following empirical rules.

1. **Replace a label with an unrelated label**: when applied, we classified the two nodes as "not related" and we set the node score to 0.
2. **Add or remove a label in a node name**: when applied, we multiplied the current node score by 0.5. If the parent node was still related, we considered the initial node either "more general" (when the label was added) or "less general" (when the label was removed) than the modified node . Some examples include:
   - "more general":
     - Original node:
       *find_Address_By_Point(part)*
     - Modified node (label added):
       *find_disturbed_Address_By_Point(part)*
   - "less general":
     - Original node:
       *find_Address_By_Point(address_Finder_Options)*
     - Modified node (label removed):
       *find_Address_By_Point(address_Finder)*

3. **Alter syntactically a label**: when applied, for each letter dropped, added or changed, we empirically decreased the similarity value by $(0.5/(total\ number\ of\ letters\ of\ the\ node\ label))$. We did not change the relation value between the original node and the modified one.
4. **Replace a label in a node name with a related one**: when applied, if the two nodes were related, we did not change the score if the new label was a synonym. If the new label was a hypernym or a hyponym of the original node, we changed the relation value to, respectively, "less general" and "more general" and therefore we applied to the similarity value a reduction of 0.5.

When all the alteration operations were applied, the expected score (*ExpScore*) between two trees $T1$ (the original one) and $T2$ (the modified one) ranged between $[0 \dots 1]$ and was computed as follows:

$$ExpScore(T1, T2) = \frac{\sum_{i \in N} Score_i}{N} \tag{7}$$

where $N$ is the node number of $T1$, $T2$ and $Score_i$ is the resulting similarity value assigned to each node of $T2$. The expected score is normalized by the size of the trees.

The reference correspondences, used to compute true positive and false positive correspondences, were the altered trees whose expected score is higher than a fixed threshold ($corrThresh$). This threshold separates the trees that a human user would, on average, consider as still similar to the original from those that are too different.

We computed recall, precision and F-measure values as shown by the equations 4, 5 and 6. We calculated the correspondences produced by SPSM solution ($R$) and the reference correspondences considered to be correct ($C$) as follows:

$$R = \{T2 \in Res \mid TreeSim(T1, T2) \geq thresh\} \tag{8}$$

$$C = \{T2 \in Res \mid ExpScore(T1, T2) \geq corrThresh\} \tag{9}$$

where $ExpScore$ was computed for each tree modification ($T2$), $TreeSim$ was the similarity value returned by the SPSM solution, $thresh$ was the $Treesim$ cut-off threshold and $Res$ was, for each original tree $T1$, the set of the modified trees.

The set of true positive, false positive and false negative correspondences were computed as follows:

$$TP = \{T2 \mid T2 \in R \wedge T2 \in C\} \tag{10}$$

$$FP = \{T2 \mid T2 \in R \wedge T2 \notin C\} \tag{11}$$

$$FN = \{T2 \mid T2 \in C \wedge T2 \notin R\} \tag{12}$$

Examples in Table 2 show results for the alteration operation "*Add or remove a label in a node name*".

**Table 2.** Example of quality measures results.

| Number of matching tasks | Cut-off threshold | C | R | TP | FP | FN | Recall | Precision | F-measure |
|---|---|---|---|---|---|---|---|---|---|
| 1600 | 0.1 | 593 | 1598 | 593 | 1005 | 0 | 1.000 | 0.371 | 0.541 |
| 1600 | 0.2 | 593 | 1585 | 593 | 992 | 0 | 1.000 | 0.374 | 0.545 |
| 1600 | 0.3 | 593 | 1568 | 593 | 975 | 0 | 1.000 | 0.378 | 0.549 |
| 1600 | 0.4 | 593 | 1496 | 593 | 903 | 0 | 1.000 | 0.396 | 0.568 |
| 1600 | 0.5 | 593 | 1391 | 593 | 798 | 0 | 1.000 | 0.426 | 0.598 |
| 1600 | 0.6 | 593 | 758 | 588 | 170 | 5 | 0.992 | 0.776 | 0.871 |
| 1600 | 0.7 | 593 | 642 | 513 | 129 | 80 | 0.865 | 0.799 | 0.831 |
| 1600 | 0.8 | 593 | 397 | 315 | 82 | 278 | 0.531 | 0.794 | 0.636 |
| 1600 | 0.9 | 593 | 143 | 112 | 31 | 481 | 0.189 | 0.783 | 0.304 |

In addition, for a fixed threshold, we compared SPSM recall, precision and F-measure values with the ones obtained from a baseline matcher, such as edit-distance.

Also, we evaluated recall and precision using combined results obtained by varying the "*add or remove a label in a node name with a related one*" (meaning) alteration operation combined with the "*alter syntactically a label*" (syntactic) alteration operation. We computed recall, precision and F-measure for each alteration precision. Also in this case, we compared our results with the ones calculated by the edit-distance baseline matcher.

We repeated all experiments described above 10 times and the presented results correspond to the average values. The maximum value of standard deviation for the plotted average values was 0.013.

### 6.2 Test case 2: (classification scenario) evaluation set-up.

In this second test, a selected set of ArcWeb operations was first manually classified. As described in the evaluation set-up, this classification followed mainly the WSDL description of the operations. We built a $n$ x $n$ matrix (where $n$ was the number of the WSDL operations) that contained our "ground truth", i.e, the manual classification of each pair of operations, which we considered to be correct. Then, we used the SPSM algorithm to independently classify the same operations in an automatic way. For each pair of operations, the SPSM algorithm returned a similarity measure that was compared with a cut-off threshold in the range $[0 \ldots 0.9]$.

We computed recall, precision and F-measure comparing the set of the relevant (manual) classifications and the set of the retrieved (automatic) correspondences. The set of true positives ($TP$) contained the pairs of operations which were manually classified in the same group and which similarity calculated by SPSM ($TreeSim$) was greater than the cut-off threshold. The set of false positives ($FP$) contained the pairs of operations that were not manually classified in the same group and which ($TreeSim$) similarity was greater than the cut-off threshold. The set of false negatives ($FN$) contained the pairs of the operations that were manually classified in the same group but which ($TreeSim$) similarity was lower than the cut-off threshold.

For example, we manually classified the pair of operations:

– *find_Address_By_Point(point, address_Finder_Options)*

and

– *find_Location_By_Phone_Number(phone_Number, address_Finder_Options)*

in the same group. SPSM returned a $TreeSim$ similarity value of 0.67. The returned SPSM correspondence is a true positive or a false negative depending on the value of the set cut-off threshold(i.e. for a cut-off threshold of 0.60 is a true positive, for a cut-off threshold of 0.70 is a false negative).

## 7 Evaluation results

### 7.1 Test case 1 (evolution scenario) results.

For each alteration operation, quality measures are functions of the $TreeSim$ cut-off threshold values and are given in the following as recall, precision and F-measure 3D diagrams. In all 3D graphs, we represent the variation of the probability of the alteration operation on Y axis, the used $TreeSim$ cut-off threshold on X axis and the resulting measures of recall, precision and F-measure on Z axis. Moreover, in all reported graphs, we used an empirically established threshold $corrThresh = 0.6$ . We also investigated the variation of this threshold and we discuss the results in the following subsections.

1. **Replace a node name with an unrelated node name:** this alteration operation replaced an entire node name with an unrelated one, randomly selected from a thesaurus. Graphs in Figures 7 and 8, show the relationship between the variation of the probability of the alteration operation, the variation of the used $TreeSim$ cut-off threshold and the resulting measures of recall, precision and F-measure.
   Figures 7 and 8 indicate that for all alterations' probability the value of recall is very high up to a $TreeSim$ cut-off threshold (around 0.6), after which it drops rapidly. Thus, we can say that, in our experiments, the SPSM approach retrieves all the expected (relevant) correspondences until the empirically fixed threshold ($corrThresh = 0.6$), that mimics the user's tolerance to errors, is reached. We made similar experiments and we observed that, when we varied $corrThresh$, all our diagrams shifted to the new value of the empirical threshold.
   The behavior of the precision is complementary: precision improves rapidly as the $TreeSim$ threshold exceeds the empirically fixed threshold. On the other hand, precision decreases steadily as a function of the alterations' probability while the $TreeSim$ threshold is below the empirically fixed threshold. We observed that this behavior, when we increased the probability of the alteration operation, depended on the decreasing number of true positives, while the number of false positives remained stable.

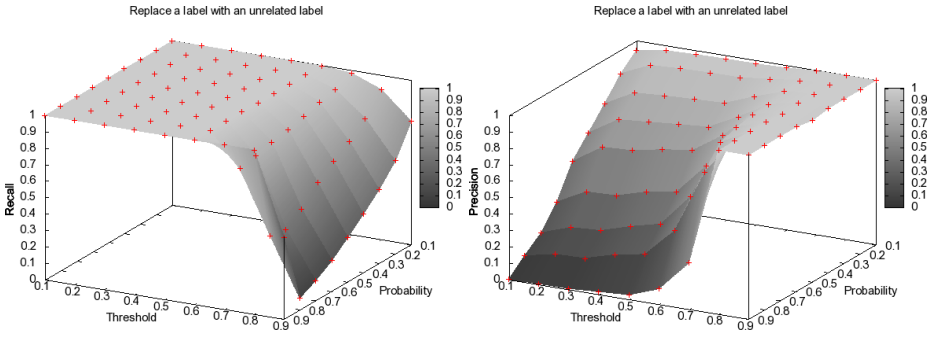


**Fig. 7.** Recall (on the left) and precision (on the right) figures of "*Replace a node name with an unrelated one*" alteration operation.

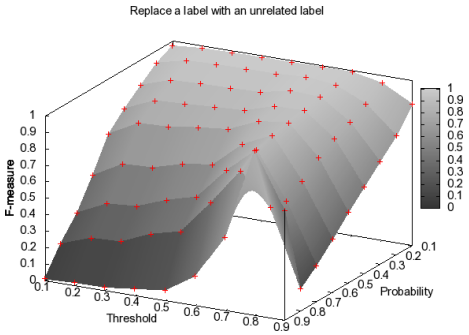


**Fig. 8.** F-measure figure of "*Replace a node name with an unrelated one*" alteration operation.

F-measure diagram 8 summarizes clearly the overall quality performance for the SPSM algorithm: the best global measures of matching quality are obtained around a threshold of $TreeSim = 0.6$, i.e. around the empirically fixed threshold ($corrThresh$) used to calculate the set of true positives, false positives and false negatives correspondences (see equations 10, 11 and 12). Analyzing the data, we observe that this is in fact the threshold where we can find a good balance between the number of the true positives correspondences and the number of the false positives correspondences. Even when the probability of the alteration is very high the balance between correctness and completeness is high. For instance, at the optimal $TreeSim$ cut-off threshold (0.6), for an alteration probability of 80%, F-measure is more than 74%. These data prove the robustness of the SPSM approach up to significant syntactic modifications in the node names.

2. **Add or remove a label in a node name:** this alteration operation added or removed a label in a node name. Figures 9 and 10 show the relation-

ship between the variation of the probability of the alteration operation, the applied *TreeSim* cut-off threshold and the resulting measures for recall, precision and F-measure.

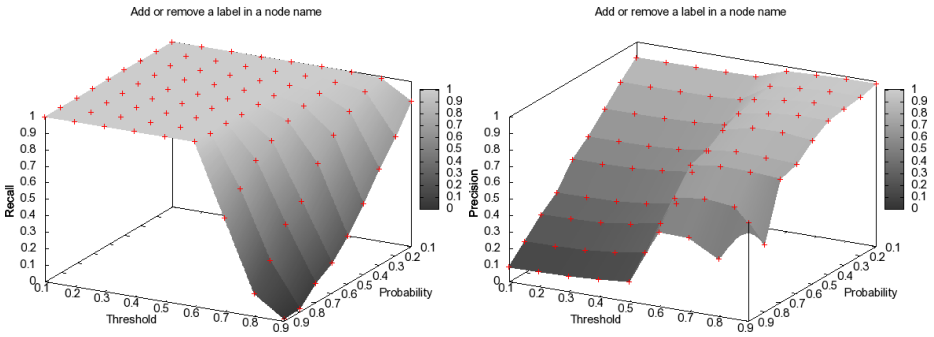


**Fig. 9.** Recall (on the left) and precision (on the right) figures of "*Add or remove a label in a node name*" alteration operation.

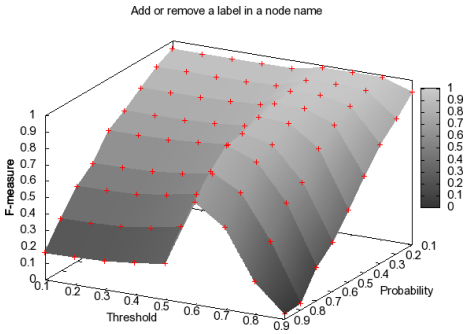


**Fig. 10.** F-measure figure of "*Add or remove a label in a node name*" alteration operation.

The behavior is similar to the ones of the previous test. Thus, the previous arguments hold also here and we can conclude equally in this case that the SPSM approach is robust up to significant syntactic alteration (probability$\sim$ 80%) of node names.

3. **Alter syntactically a label in a node name:** this alteration operation altered syntactically a label in a node name, by modifying (drop, add, delete) its characters. See Figures 11 and 12 for the evaluation results.

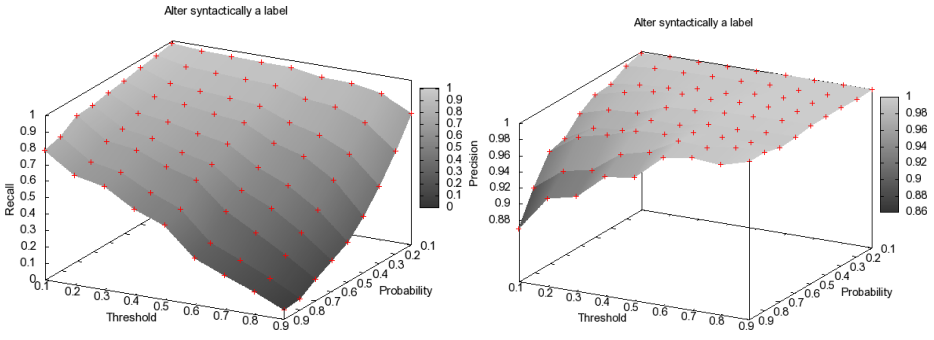


**Fig. 11.** Recall (on the left) and precision (on the right) figures of "*Alter syntactically a label*" alteration operation.

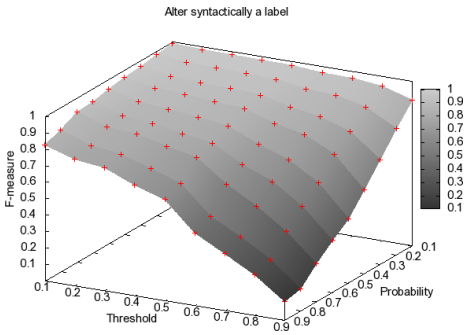


**Fig. 12.** F-measure figure of "*Alter syntactically a label*" alteration operation.

This test evaluated the robustness of the SPSM approach simulating errors and alterations that a programmer could make while writing the service operations signatures. In this test, recall decreases steadily as a function of increasing both probability of the alteration and $TreeSim$ cut-off threshold. Precision is always high, in the range $[0.87 \ldots 1.0]$. This is due to a high number of true positive correspondences and to a simultaneous low number of false positive correspondences. This means that such "misspelling" alterations were always recognized as potential correspondences by the SPSM algorithm.

Therefore, F-measure diagram (Figure 12) essentially reproduces the recall diagram (Figure 11). F-measures values of $\sim 70\%$ were obtained for alterations' probability up to 70% and $TreeSim$ cut-off thresholds up to 0.6.

4. **Replace a label in a node name with a related one:** this alteration operation replaced a label in a node name with a related one, by using synonyms, hyponyms, hypernyms from a number of generic thesauri. Diagrams in Figure 13, report the resulting measures of recall and F-measure. Preci-

sion results are not shown as the values were always close to 1. In fact we "always" used related (i.e. synonyms, hyponyms, hypernym) terms in the alteration operations. Therefore, almost all the semantic correspondences between the labels were found by SPSM (by construction of the altered set). Thus, a very small number of false positives correspondences were found and precision was always close to 1.

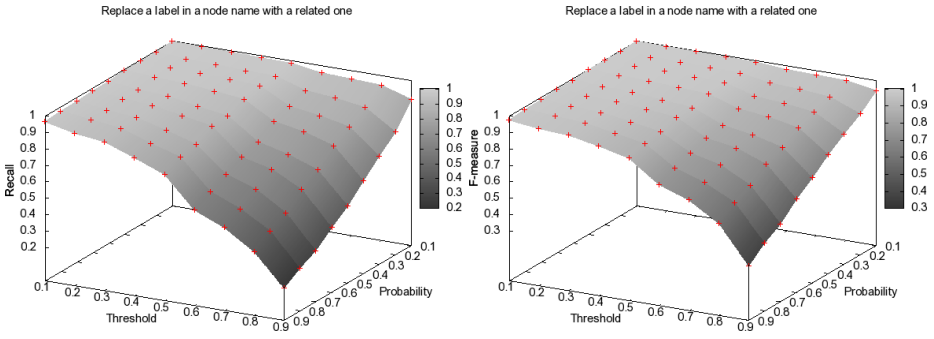


**Fig. 13.** Recall (on the left) and F-measure (on the right) figures of "*Replace a label in a node name with a related one*" alteration operation.

In this experiment, we evaluated the robustness of the SPSM approach to "semantic" alterations of the nodes: we did not change the core concept of the node name, but we used either an "equivalent" or a "more general" or a "less general" label in a node name. In this case recall decreases slowly when both the alteration operation probability and the *Treesim* cut-off threshold increase.

**Test case 1: comparison of SPSM and baseline.** The goal of this experiment was to compare SPSM results with a baseline matcher. In order to appropriately compare the two series of results, we used the same evaluation method of the previous experiment. Thus: *(i)* we used the same alteration operations, described in the previous section, to modify the original trees, and *(ii)* we used the results of the previous experiments to identify the best alteration probability to make the comparison between the best results. The baseline matcher was built using a simple edit-distance algorithm. We made the comparison using all the alteration operations: "*Replace a node name with an unrelated node name*", "*Add or remove a label in a node name*", "*Alter syntactically a label*" and "*Replace a label in a node name with a related one*".

Results for the syntactic modification are, as expected, very similar. Therefore, we focused our analysis to node's names "meaning" alterations. Figure 14 shows the results when "*Replace a node name with an unrelated node name*" is applied and Figure 15 shows the results when "*Replace a label in a node*

*name with a related one*" is applied. We plot the results for the most interesting alteration operation probability ($\sim 0.6$) for both the syntactic and semantic alterations.

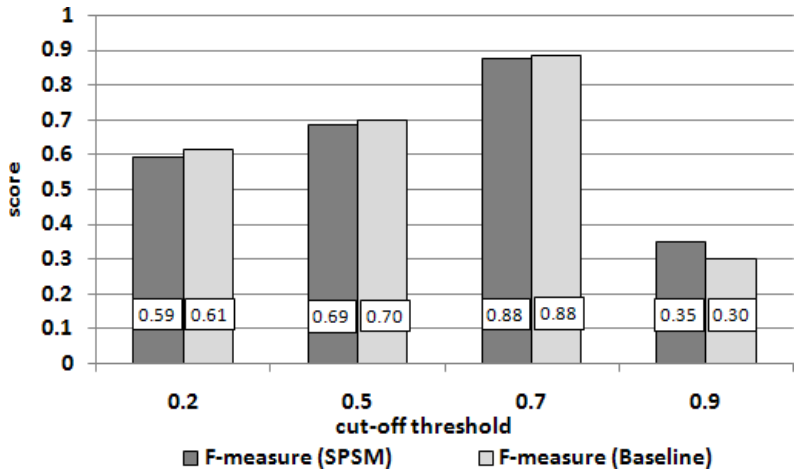


**Fig. 14.** SPSM vs. baseline on "*Replace a label with an unrelated one*" alteration operation.

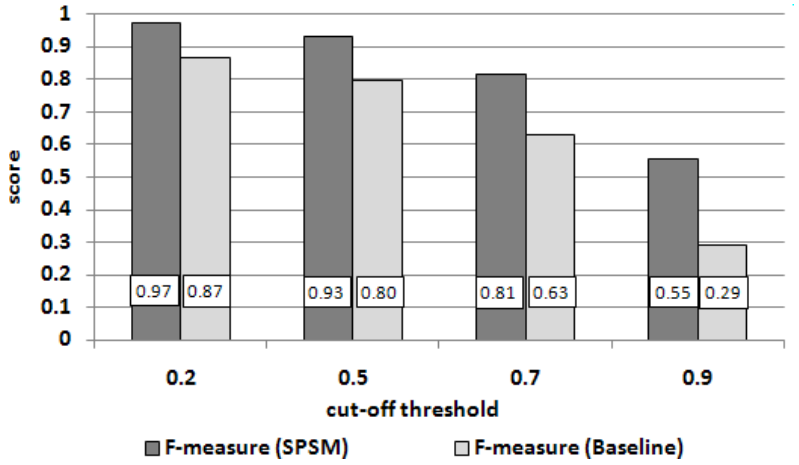


**Fig. 15.** SPSM vs. baseline on "*Replace a label in a node name with a related one*" alteration operation.

As the figures 14 and 15 show, SPSM approach is always comparable with the baseline matcher when we made syntactic alterations (Figure 14). SPSM approach results are significantly better than the baseline matcher (more than 20%, Figure 15) when we made meaning alterations.

Figure 16 shows the comparison between our approach and the baseline when "*Replace a label in a node name with a related one*" and "A*lter syntactically a label*" alterations were combined together. The diagrams show the scores of F-measure (we selected a cut-off threshold of 0.7) for both SPSM and edit-distance (baseline) matchers.

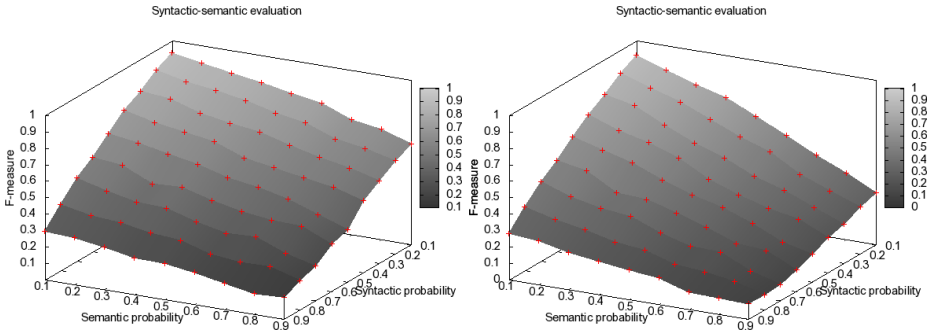


**Fig. 16.** F-measure values for SPSM (on the left) and edit-distance (baseline, on the right) matchers.

Again the diagrams suggest the same conclusion: the SPSM approach behavior is similar to the one of the baseline matcher when syntactic alterations were made, while its performance is constantly better than the baseline when the "meaning" of the label was modified.

### 7.2 Test case 2: classification scenario.

In this experiment, we wanted to investigate whether the proposed SPSM approach can be used in determining (in an unsupervised way) the "class" of a specific GIS operation from the its signature. To this end we first classified a selected set of ArcWeb, in order to obtain the "truth" classification set for our evaluation. This manual classification followed mainly the WSDL description of the operations. Subsequently, we used the SPSM matcher to independently classify the same operations in an automatic way. For each operation the SPSM matcher returned similarity measures in respect to all others operations. These were compared with a selected threshold and similar signatures were grouped together. Finally, we compared the manual classification with the automatic one performed by SPSM. Figure 17 shows, for each $TreeSim$ threshold, recall, precision and F-measure scores. Classification quality measures depend on the cut-off threshold values and the SPSM solution demonstrates good overall matching

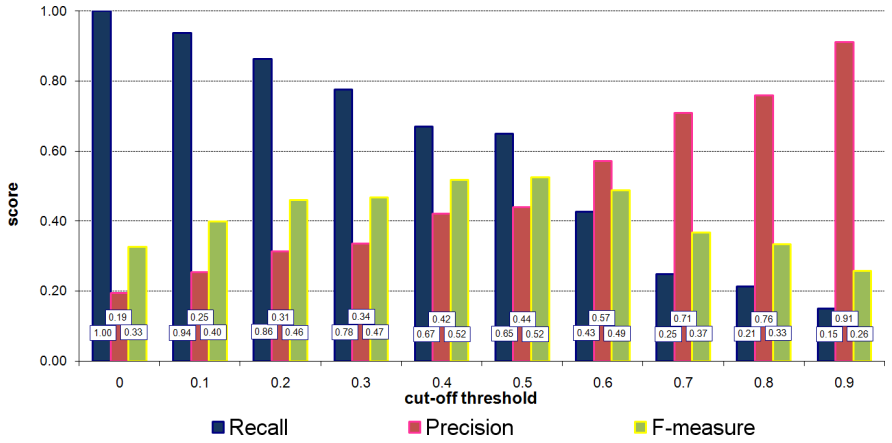


**Fig. 17.** Recall, precision and F-measure values for classification robustness test.

quality (i.e. F-measure) on the wide range of these values. In particular, the best F-measure values exceed 50% for the given GIS operations set. For example, for a *TreeSim* threshold of 0.5, precision is 0.44, recall is 0.65, and F-measure is 0.52.

### 7.3 Evaluation Summary

We can summarize the extensive evaluation of the SPSM matching approach on the selected set of real-world GIS web services as follows:

**SPSM behavior and robustness.** We developed evaluation tests to explore the overall behavior and robustness of the proposed SPSM approach towards both typical syntactic alterations and meaning alterations of the GIS service operation signatures. All experiments demonstrated the capability of the SPSM approach to self-adapt (i.e. to provide best results) to the empirical threshold (*ExpScore*) used in the experiment to simulate the users' tolerance to errors (i.e. to calculate the set of true positive, false positive and false negative correspondences). Moreover, the results showed the robustness of the SPSM algorithm over significant ranges of parameters' variation (thresholds and alteration operations' probability); while maintaining high (over 50-60%) overall matching relevance quality (F-measure).
Comparison with a baseline matcher (based on edit-distance algorithm) showed how the SPSM approach is always comparable with the baseline when only syntactic alteration are considered, whereas SPSM results were always better (in average more than 20%) when "meaning" alterations were introduced. This is what we expected, since the SPSM approach includes a number of state-of-the-art syntactic matchers (that are first used in the internal matching algorithm) plus a number of semantic matchers that enter into play for the alterations in the meaning of nodes labels [10].

**SPSM unsupervised clustering capabilities.** In this experiment, we investigated how the proposed SPSM approach could be used in determining (in an unsupervised manner) the "class" of a specific GIS operation directly from the information present in its WSDL operation signature. Classification quality measures depended on the cut-off threshold values and the SPSM solution demonstrated overall good matching quality (i.e. F-measure) on the wide range of these values. In particular, the best F-measure values exceeded 50% for the given GIS operations set. Although, the results are encouraging, still 50% of GIS operation were incorrectly classified, due to the limited knowledge presented in the signatures only. In this case, the presence of more informative and semantically structured annotation would improve significantly the automatic classification at the expense obviously of a greater effort from the designer/programmer.

**SPSM performance.** With all our experiments we executed approximately 700.000 matching tasks using SPSM. The efficiency of the SPSM solution is such that the average execution time per matching task in the evaluation under consideration was 43ms (the average number of the parameters of the WSDL operations was 4). The quantity of main memory used by SPSM during matching did not rise more than 2.3Mb higher than the standby level. These number are very good for the application domain we want to address, i.e the run-time replaning of web services composition chains.

## 8 Conclusions and future work

In this paper, we tackled the semantic heterogeneity problem in typical spatial data infrastructures, which include geo-data sharing through geo-services provided by GIS agencies. We formalized specific GIS service requests in a e-Response scenario. The scenario was based on the current organizational model for the distributed GIS Agency infrastructure of Trentino, Italy.

We discussed an application of the approximate Structure Preserving Semantic Matching approach in the field of coordination of Web services. We reported theoretically foundation of the approach and, finally, we conducted an extensive set of empirically tests to evaluate quality and efficiency indicators of the SPSM approach on a set of 80 real ArcWeb WSDL operations. After an extensive investigation (ca. 700.000 matching tasks) we analysed in details both the SPSM robustness and its clustering capabilities.

We recall here, that matching in our OpenKnowledge framework, and in particular in our e-Response scenario, has two main purposes:

1. To support unsupervised or semi supervised service chaining between services operations that are not identical to the one required in a formally described interaction model. This is done through building up a map between each element of the service signature to each element of a interaction constraint. In the case of non-perfect matching, there may be elements in either the ability or the constraint that remain unmatched, and the matches that do exist may not be between things that are semantically identical. Nevertheless this map

enables the service' users to use its own abilities to satisfy constraints to the highest degree possible.
2. To allow service's users to determine how similar their own abilities are to those required of the role, by considering how close the numerical similarity returned is to a perfect score.

Currently, the matching solution is elementary in the sense that it provides means to match web services available in the LCC interaction models. However, to run an interaction model, a peer should know which interaction model it wants to execute and with which peers it will be interacting. The ultimate goal is to provide a unifying framework based on interaction models that are mobile among peers, being a mechanism for web service composition, capable of capturing (local) peers semantics emerging from their interactions and enabling ad hoc peer coalition formation as required by hastily formed networks [5].

Moreover, the evaluation approach followed for the first test case in this paper, based on random alteration of the signatures, may suffer from some limitations. In particular, the generation of the reference score ("ExpScore") has to be further investigate in order to statistically compare the computed score to the one that a human user would make.

In turn, future work on the approximate SPSM proceeds at least in the following directions: *(i)* conducting extensive and comparative evaluation, including other kind of GIS web services like the ones available from OGC specifications and GRASS package, *(ii)* extending the matching approach for dealing with fully-fledged GIS ontologies like the ones provided by INSPIRE directive, *(iii)* incorporating domain specific preferences in order to drive approximation, thus allowing/prohibiting certain kinds of approximation (e.g not approximating vector maps with raster maps, although these are both maps), and *(iv)* use different kind of thesaurus like the multilingual GEMET or AGROVOC thesauri or to support multilingual matching.

**Acknowledgments.** This work has been supported by the FP6 OpenKnowledge European STREP (FP6-027253)[17]. We thank Fausto Giunchiglia and Fiona McNeill for many fruitful discussions on the Structure-Preserving Semantic Matching and on the evaluation set-up. We thank David Dupplaw for the development of the emergency GUI.

## References

1. Rohit Aggarwal, Kunal Verma, John Miller, and William Milnor. Constraint driven web service composition in METEOR-S. In *Proceedings of the 1st IEEE International Conference of Services Computing (SCC)*, pages 23–30, 2004.
2. Grigoris Antoniou and Frank van Harmelen. *Web Ontology Language: OWL.* Springer-Verlag, 2003.

[17] www.openk.org

3. Lars Bernard, Max Craglia, Michael Gould, and Werner Kuhn. Towards an SDI research agenda. In *Proceedings of the 11th European Commission-Geographic Information (EC-GI) and GIS Workshop*, pages 147–151, 2005.
4. Caterina Caracciolo, Jérôme Euzenat, Laura Hollink, Ryutaro Ichise, Antoine Isaac, Vŕonique Malais¸ Christian Meilicke, Juan Pane, Pavel Shvaiko, Heiner Stuckenschmidt, Ondřej Šváb Zamazal, and Vojtěch Svátek. Results of the ontology alignment evaluation initiative 2008. In *Proceedings of the International Workshop on Ontology Matching (OM) at the 7th International Semantic Web Conference (ISWC)*, 2008.
5. Peter J. Denning. Hastily formed networks. *Communication of the ACM*, 49(4):15–20, 2006.
6. Liping Di, Peisheng Zhao, Wenli Yang, and Peng Yue. Ontology-driven automatic geospatial-processing modeling based on web-service chaining. In *Proceedings of the 6th Earth Science Technology Conference (ESTC) - CDROM*, 2006.
7. Jérôme Euzenat, Antoine Isaac, Christian Meilicke, Pavel Shvaiko, Heiner Stuckenschmidt, O. Šváb, Vojtech Svátek, Willem Robert van Hage, and Mikalai Yatskevich. Results of the ontology alignment evaluation initiative OAEI. In *Proceedings of the Workshop on Ontology Matching (OM) at the 6th International Semantic Web Conference (ISWC) + the 2nd Asian Semantic Web Conference (ASWC)*, pages 96–132, 2007.
8. Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer, 2007.
9. Fausto Giunchiglia, Maurizio Marchese, and Ilya Zaihrayeu. Encoding classifications into lightweight ontologies. *Journal of Data Semantics*, VIII:57–81, 2007.
10. Fausto Giunchiglia, Fiona McNeill, Mikalai Yatskevich, Juan Pane, Paolo Besana, and Pavel Shvaiko. Approximate structure-preserving semantic matching, 2008.
11. Fausto Giunchiglia and Pavel Shvaiko. Semantic matching. *The Knowledge Engineering Review*, 18(3):265–280, 2003.
12. Fausto Giunchiglia and Toby Walsh. A theory of abstraction. *Artificial Intelligence*, 57(2-3):323–389, 1992.
13. Fausto Giunchiglia, Mikalai Yatskevich, Paolo Avesani, and Pavel Shvaiko. A large scale dataset for the evaluation of ontology matching systems. *The Knowledge Engineering Review Journal*, 24(2), 2009, to appear.
14. Fausto Giunchiglia, Mikalai Yatskevich, and Enrico Giunchiglia. Efficient semantic matching. In *Proceedings of the 2nd European Semantic Web Conference (ESWC)*, pages 272–289, 2005.
15. Fausto Giunchiglia, Mikalai Yatskevich, and Pavel Shvaiko. Semantic matching: Algorithms and implementation. *Journal on Data Semantics*, IX:1–38, 2007.
16. Moses Gone and Sven Shade. Towards semantic composition of geospatial web services using WSMO in comparison to BPEL. *submitted to International Journal of Spatial Data Infrastructures Research (IJSDIR)*, 3(2008), 2008.
17. Richard Groot and John McLaughlin. *Geospatial Data Infrastructure: Concepts, Cases and Good Practice*. Oxford University Press, 2000.
18. Matthias Klusch, Benedikt Fries, and Katia Sycara. Automated semantic web service discovery with OWLS-MX. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2006.
19. Werner Kuhn. Geospatial semantics: Why, of what, and how? *Journal on Data Semantics, special issue on Semantic-based Geographical Information Systems*, 3534:1–24, 2005.
20. Rob Lemmens, Andreas Wytzisk A., Rolf de By, Carlos Granell, Michael Gould M., and Peter van Oosterom. Integrating semantic and syntactic descriptions to chain geographic services. *IEEE Internet Computing*, 10(5):42–52, 2006.

21. Michael Lutz and Eva Klien. Ontology-based retrieval of geographic information. *International Journal of Geographic Information Science*, 20(3):233–260, 2006.
22. Michael Lutz, Roberto Lucchi, Anders Friis-christensen, and Nicole Ostländer. A rule-based description framework for the composition of geographic information services. *Geospatial Semantics*, 4853:114–127, 2007.
23. Maurizio Marchese, Lorenzo Vaccari, Pavel Shvaiko, and Juan Pane. An application of approximate ontology matching in eresponse. In *Proceedings of the 5th International Conference on Information Systems for Crisis Response and Management (ISCRAM)*, pages 294–304, 2008.
24. Ian Masser. *Creating Spatial Data Infrastructures*. ESRI Press - RedLands - California, 2005.
25. Rimon Mikhaiel and Eleni Stroulia. Examining usage protocols for service discovery. In *Proceedings of the 4th International Conference on Service Oriented Computing (ICSOC)*, pages 496–502, 2006.
26. George A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.
27. Douglas Nebert. *Developing Spatial Data Infrastructures. The SDI CookBook*. Global Spatial Data Infrastructure (GSDI), 2004.
28. Natalia F. Noy. Semantic integration: A survey of ontology-based approaches. *SIGMOD Record*, 33(4):65–70, 2004.
29. Natalya F. Noy, AnHai Doan, and Alon Y. Halevy. Semantic integration. *AI Magazine*, 26(1):7–9, 2007.
30. Timothy L. Nyerges. Schema integration analysis for the development of GIS databases. *International Journal of Geographical Information Systems*, 3(2):153–183, 1989.
31. H. Onsrud. *Research and Theory in Advancing Creating Spatial Data Infrastructure Concepts*. ESRI Press - RedLands - California, 2007.
32. Swapna Oundhakar, Kunal Verma, Kaarthik Sivashanmugam, Amit Sheth, and John Miller. Discovery of web services in a multi-ontology and federated registry environment. *International Journal of Web Services Research*, 2(3):1–32, 2005.
33. Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia Sycara. Semantic matching of web services capabilities. In *Proceedings of the 1st International Semantic Web Conference (ISWC)*, pages 333–347, 2002.
34. Christine Parent, Stefano Spaccapietra, and Esteban Zimanyi. *Conceptual modeling for traditional and spatio-temporal applications. The MADS approach*. Springer, 2006.
35. Charles Petrie, Tiziana Margaria, Ulrich Kuster, Holger Lausen, and Michal Zaremba. Sws challenge: Status, perspectives, and lessons learned so far. In *Proceedings of the 9th International Conference on Enterprise Information Systems (ICEIS)*, pages 447–452.
36. David Robertson. A lightweight coordination calculus for agent systems. *Declarative Agent Languages and Technologies*, pages 183–197, 2004.
37. Stefan Schulte, Julian Eckert, Nicolas Repp, and Ralf Steinmetz. An approach to evaluate and enhance the retrieval of semantic web services. In *Proceedings of the 5th International Conference on Service Systems and Service Management (ICSSSM)*, pages 237–243, 2008.
38. Pavel Shvaiko and Jérôme Euzenat. A survey of schema-based matching approaches. *Journal on Data Semantics*, IV:146–171, 2005.
39. Eleni Stroulia and Yiqiao Wang. Structural and semantic matching for assessing web-service similarity. *International Journal of Cooperative Information System*, 14(4):407–438, 2005.

40. Vlad Tanasescu, Alessio Gugliotta, John Domingue, Rob Davies, Leticia Gutiérrez-Villarías, Mary Rowlatt, Marc Richardson, and Sandra Stinčić. A semantic web services gis based emergency management application. pages 959–966, 2006.
41. Gabriel Valiente. *Algorithms on Trees and Graphs*. Springer, 2002.
42. Michael F. Worboys and S. Misbah Deen. Semantic heterogeneity in distributed geographic databases. *SIGMOD Record*, 20(4):30–34, 1991.
43. Peisheng Zhao and Liping Di. Semantic web service based geospatial knowledge discovery. pages 3490–3493.