OpenKnowledge

FP6-027253

Empirical Tests of Ontology Matching

Paolo Besana[1] and Pavel Shvaiko[2]

[1] University of Edinburgh, Edinburgh, UK
p.besana@ed.ac.uk
[2] University of Trento
Department of Information and Communication Technology (DIT),
University of Trento, Povo, Trento, Italy
pavel@dit.unitn.it

Report Version: final
Report Preparation Date: 12.1.2008
Classification: deliverable D7.4
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

This deliverable presents an evaluation of the OpenKnowldge (OK) ontology matching component (OMC). Specifically, we discuss: (i) an evaluation dataset that we have automatically built in order to measure quality and efficiency indicators of OMC, and (ii) the evaluation results of OMC on the dataset constructed.

1 Introduction

The OpenKnowledge system is a peer-to-peer (P2P) network of knowledge or service providers. Each computer in the network is a peer which can offer services to other peers. OK is viewed as an infrastructure, where we only provide some core services which are shared by all the peers, while all kinds of application services are to be plugged on top of it. When a peer needs to perform a task that requires other peers, it asks the discovery service (DS) a list of interaction models (IM) described by a set of keywords: the interaction models, written in Lightweight Coordination Calculus (LCC) [13], define the messages exchanged between the participants. The messages have pre-conditions and post-conditions, that the peer needs to solve: as described in [1], the conditions - or *constraints* - are solved by methods contained in plug-in components in the peer, called OKCs.

However, methods and constraints may have different signatures, that must be compared and matched: the ontology matching component [8, 5, 17] is used for selecting the IM whose constraints best match the peer components and for creating the adaptors between the methods and the constraints. The goal of this deliverable is to conduct empirical tests of the OK ontology matching component.

The rest of the deliverable is organized as follows. Section 2 briefly presents the OK OMC to be evaluated. Then, Section 3 discusses an evaluation dataset that we have automatically built in order to measure quality and efficiency indicators of OMC. Section 4 provides the evaluation results of OMC on the dataset constructed. Finally, Section 5 summarizes the findings of the deliverable and outlines future work.

2 The ontology matching component

The OMC should be able to recognize when a constraint and a method are similar enough, and to identify the correspondences between the arguments when different names and positions are used [17]. In turn, the correspondences are used to generate adaptors [15] that allow the decoupling between plug-ins and protocols: the programmer of a plug-in can use the arguments of a method as defined by the method semantics, without worrying how the method will be called in future interactions. Furthermore, if DS returns many similar IMs for a particular task, the matching scores can be used, together with other measures, such as the popularity of the IMs, to rank and choose IMs when they all match the peer's components with an acceptable score.

Parameters in constraints and methods can be atomic, like strings or integers, or structured objects. Both constraints and methods can be represented as labelled *trees*, with the root being the predicate or the method name and its subtrees the arguments:

```
accept_proposal(
   car(
     brand,
     model,
     engine(cc,hp)),
   cost(
     original_price,
     discount))
```

A branch in the tree corresponds to an element of a parameter. In the example above, the branch /accept_proposal/car/brand identifies a specific value of the first parameter in the constraint.

The OMC takes two tree-like structures, for example, Get_Wine(Region, Country, Color, Price) and Get_Wine(Region(Country, Area), Colour, Cost, Year), and returns a global similarity coefficient in the [0 1] range between these trees (e.g., 0.57) as well as the set of one-to-one correspondences between the semantically related nodes of these trees (e.g., that Color in the first tree corresponds to Colour in the second one). The following two structural properties are preserved: (i) functions are matched to functions and (ii) variables are matched to variables [9, 17].

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 domain specific contextual information of the trees. We use here the S-Match system [8]. Technically, two nodes n_1 and n_2 in trees $T1$ and $T2$ match if: $c@n_1\ R\ c@n_2$ holds based on S-Match. $c@n_1$ and $c@n_2$ are the concepts at nodes n_1 and n_2, and $R \in \{=, \sqsubseteq, \sqsupseteq, \perp\}$. When non of the $R's$ can be explicitly computed a special *"not related"* relation is returned. In particular, in semantic matching [6] as implemented in the S-Match system [8] the key idea is that the relations (e.g., $=, \sqsubseteq$) between nodes are determined by (i) expressing the entities of the tree-like structures as logical formulas, and (ii) reducing the matching problem to a logical validity problem. Specifically, the nodes are translated into logical formulas which explicitly express the concept descriptions as encoded in the tree-like structure and in external resources, such as WordNet [12, 4]. 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 (SAT) solvers [7]. The result of this stage is the set of correspondences holding between the nodes of the trees.

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. As from [9], two trees $T1$ and $T2$ approximately match if and only if there is at least one node $n1_i$ in $T1$ and node $n2_j$ in $T2$, such that: (i) $n1_i$ approximately matches $n2_j$, (ii) all ancestors of $n1_i$ are approximately matched to the ancestors of $n2_j$.

3 Evaluation set-up

The ontology matching evaluation theme has been given a chapter account in [3]. Its more recent advances have been described in [14] and several summaries have been reported in [16, 19]. A peculiarity of OpenKnowledge is that constraints or methods are simple labelled trees and not fully-fledged ontologies. Moreover, these trees are usually not too deep and large, and thus, the OMC will have to deal with many small trees. Also, we make the following two assumptions in our evaluation:

- Having looked at various application programming interfaces (APIs), we do an assumption that the terms are likely not to be semantically related between each other within a single constraint or method. Examples from the Java API include: set(index, element) and put(key, value). Thus, a tree representing a method or a constraint can be considered as being composed of nodes whose labels are random terms.

- Often methods and constraints to be matched are derived or inspired one from the other: a developer sees an IM that (s)he wants to use, and writes the corresponding OKC; then the IM is altered in order to include a new case that was not considered; a new, similar IM is written; the OKC is refactored; and so on. Therefore, it is reasonable to compare a tree with another one derived from the original one. This also allows to verify what types of alterations of the tree are beyond the capabilities of OMC, and thus, help revealing its weaknesses.

Since evaluation datasets for OMC from the OK testbeds, such as emergency response and bioinformatics are still under way [19], we followed here an alternative approach. In particular, bearing the above mentioned assumptions in mind and similarly to the work on systematic benchmarks of the Ontology Alignment Evaluation Initiative (OAEI) [2], our dataset was composed of trees that are alterations of the actual trees. Unlike the work on systematic benchmarks in OAEI, the actual trees here were generated automatically. The process used for tree generation and alteration is described in Table 1. Notice that using the same external resource or oracle as employed by OMC when replacing a term with another related term can bias the results by overestimating them: the OK matching component uses WordNet. Therefore, as mentioned in Table 1, the term is extracted both from WordNet 3.0 (notice that OMC uses version 2.1 of WordNet) and from the Moby Thesaurus[1]. However, while in WordNet the relations between words are always defined, Moby is more generic, and simply returns a list of related words, not only synonyms. A brief analysis of the returned terms has shown that some of the related words can have very little relation with the original term.

We have generated 100 trees. For each original tree, 30 altered ones are created. Notice that the transformations applied to a tree generate a new tree that can be more or less distant from the original one: changing the root element with an unrelated

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

- **Tree creation:**
  - Trees are currently one level deep.
  - Labels are composed of a random number of words (minimum 1), selected from 9000 words extracted from part of the Brown Corpus [10].
  - The number of children (corresponding to arguments in methods and constraints) is random. We used Gaussian distribution with average 2 and deviation 2. These settings for the Gaussian distribution were used in order to emulate functions, that are rarely longer than 4 parameters.
- **Tree alteration:**
  - Label replacement, with probability 0.2 (obtaining nodes that are not related to the original one).
  - Label syntactic alterations, with probability 0.2 (letters dropped, added, changed).
  - Words addition or removal in labels, with probability 0.15.
  - Words replacement in labels, with probability 0.4, using:
    - synonyms, hyponyms, hypernyms (extracted from WordNet 3.0, using all the possible parts of speech of the word);
    - related words (extracted from the Moby thesaurus), in order to reduce the bias due to WordNet 2.1 use by OMC.
  - Nodes deletion (the number of nodes to remove is computed using a Gaussian distribution with average 0 and standard deviation 0.9).
  - New nodes addition, with probability 0.25.
  - Nodes shuffling, with probability 0.4.

  The probabilities of the alteration operations have been chosen by trial and error in order to obtain reasonably altered trees, without having completely unrelated trees.

Table 1: Tree creation and alteration process.

label puts the generated tree at a maximum distance, but dropping or adding nodes, adding errors, using synonyms put the trees at variable distances. Pairs composed of the original tree and one varied tree are fed to the OMC that returns the similarity score between trees and the correspondences between their nodes. The experiment described above was repeated 5 times in order to remove noise in the results.

Similar to the work in [2, 11], since the tree alterations made are known, these provide the ground truth, and hence, the reference results are available for free by construction. This allows for the computation of the matching quality measures [19]. In particular, the standard matching quality measures (*recall*, *precision* and *F-measure* being a harmonic mean of recall and precision [18, 19]) for the similarity between trees have been computed.

4 Evaluation results

We are interested in the following three aspects of the evaluation results:

- The proportion of trees correctly evaluated as similar (§4.1). The similarity is used to select the IMs that the peer is able to perform given its OKCs.
- The proportions of branches correctly matched (§4.2). The correspondences form the adaptors used within the OKC methods to access the parameters.
- Efficiency of OMC on the test cases of the dataset (§4.3).

4.1 Similarity between trees

Let us look at how close are the scores returned by the OMC to those computed during the tree transformations. As shown in Figure 1, the majority (63%) of the results computed by OMC are within a distance of 0.1 from the expected value.

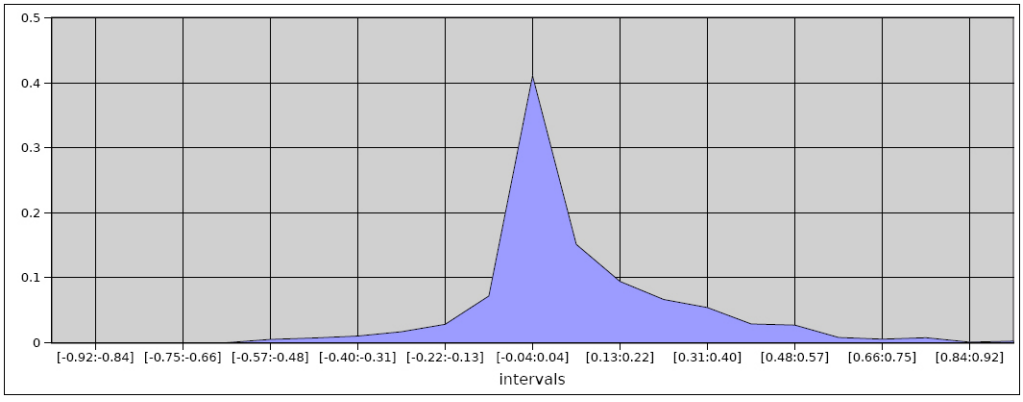

Figure 1: Error distribution: a negative error means that the expected value was lower than the one computed by OMC, while a positive error value means that the expected value was higher than the computed one.

This error directly influences the ranking of the IMs: it may give a wrong position to an IM in a list provided by the DS, possibly prompting the peer to choose an IM that is not the best given its OKCs.

Another issue is whether the error influences the selection of good enough IMs, independently of the rankings. It may well be that the error does move the score, but keeping it above the good-enough threshold: if the expected score is 0.9 and the computed score is 0.7, the error is 0.2, but with a threshold of 0.7 it does not influence the selection. Given a threshold, it is possible to compute the standard matching quality measures. Different thresholds result in different values of matching quality measures, see Figure 2. For example, setting the threshold at 0.5 yields precision of 0.92 and recall of 0.78, while setting it at 0.7 results in precision of 0.93 and recall of 0.67.

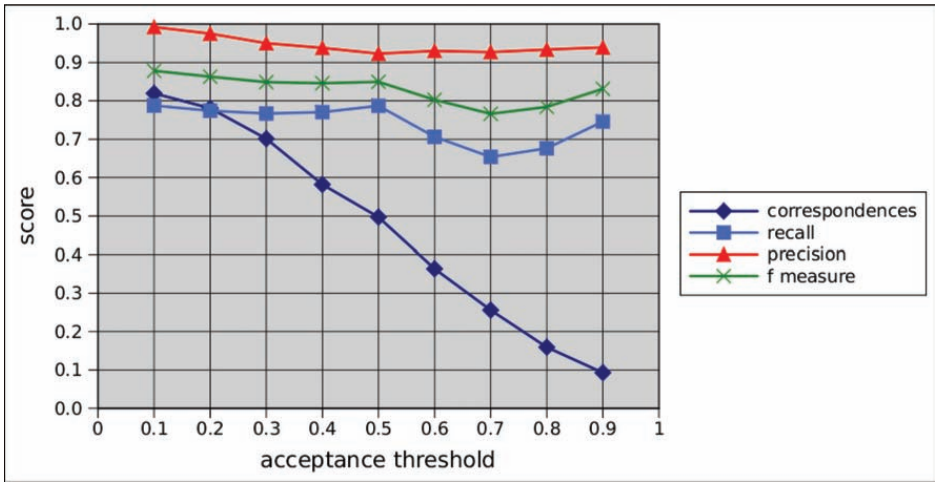

Figure 2: Evaluation results. Different precision, recall and F-Measure given different good-enough thresholds. The correspondences are expressed as ratio between the expected correspondences and all the possible trees to be matched.

Increasing the threshold reduces the correspondences considered good-enough: the steepest drop takes place between 0.5 and 0.7 where the correspondences drop plunge from more than 50% to less than 30%: this drop corresponds to a drop in recall. The drop in recall, never fully recovered, is partially caused by the change in proportion of true positives and true negatives: as the number of correspondences decreases, errors have stronger influence on recall: if 80 are wrongly excluded at threshold 0.3, the influence on recall is 0.12 (80/650), while if 80 are excluded at threshold 0.7, the influence is above 0.25 (80/330). The threshold interval between 0.5 and 0.7 is likely to be critical because the alterations applied to the trees are possibly on the edge of the OMC capabilities. For lower thresholds errors in evaluating similarity are less important, as most of the matches have scores higher than low thresholds. Higher thresholds, on the other hand, involve trees only slightly altered, that are more easily recognized by OMC, diminishing the errors and improving the recall.

4.2 Branch matching

The errors we have observed in §4.1 are in turn caused by errors in identifying the correspondences between branches in different trees: if two similar branches are not recognized by OMC, then the overall score of tree similarity decreases. Conversely, two unrelated branches incorrectly considered similar by OMC increases the similarity score, possibly causing an interaction to fail, because of a constraint cannot be satisfied at run time, or causing unpredictable results if a method accesses the wrong parameter.

Quality results for branch matching are as follows: recall is 0.71, precision is 0.99, and F-measure is 0.83. The expected correspondences, whose number is overestimated by about 2-3% as we have observed in §4.1, constitute 64% of the total comparisons (3000 for each experiment).

Analyzing the statistics about the relations found between nodes, the *not related* relation is normally - and correctly - found between nodes whose label has been replaced with an unrelated one or whose parent node was not related. However, it is possible to spot a number of errors, in particular when hypernyms or hyponyms were used to replace terms in the label. Terms replaced with related words are often difficult to match, as the replacing word may be not be directly linked in WordNet to the old one. The OMC is predictably and quite consistently unable to match two nodes when a label in a node belonging to the original node has been replaced in the altered tree with a synonym/hypernym/hyponym and then some syntactic changes were applied as well.

4.3 Performance evaluation

The trees matched by OMC were small, being composed on average of 4.4 nodes. The experiments were run on an Intel Dual Core (1.86 Ghz) with 1Gb of RAM. The execution time of OMC between the original tree and the altered version takes on average 18ms, that is about 4ms per single node comparison. The quantity of main memory used by OMC during matching did not rise more than 3Mb higher than the standby level.

5 Conclusions and future work

In this deliverable we have presented an automatically created dataset for evaluating quality and efficiency indicators of the OK ontology matching component. The evaluation results look promising, especially for what concerns the efficiency indicators. Also, precision is consistently high: this is particularly important, as the consequences of matching constraints to the wrong methods can be much worse than the consequences of failing to find an interaction. However, the evaluation approach followed in this deliverable, based on random transformation of randomly generated trees, may suffer from some limitations. In particular, due to the limited availability of good

thesauri for the generation of related terms. This limitation biases the performances both positively (terms extracted from WordNet are certainly known by OMC) and negatively (poor thesauri do not distinguish the related terms, and often include terms which are very weakly related).

Future works proceeds at least along the following directions: (i) extensive testing using deeper trees and wider sets of generated and altered trees in order to evaluate the behavior of OMC in more complex scenarios (this requires an additional tree alteration operation, that allows branches to be moved into different positions, etc.); (ii) devising a baseline matching solution and conducting a comparative evaluation that involves OMC, a baseline solution, etc.

References

[1] Paolo Besana, David Dupplaw, and Adrian De Pinnick. Openknowledge deliverable 1.3: Plug-in components. `http://www.cisa.informatics.ed.ac.uk/OK/Deliverables/D1.3.pdf`, 2007.

[2] Jérôme Euzenat, Antoine Isaac, Christian Meilicke, Pavel Shvaiko, Heiner Stuckenschmidt, Ondřej Šváb, Vojtěch Svátek, Willem Robert van Hage, and Mikalai Yatskevich. Results of the ontology alignment evaluation initiative 2007. In *Proceedings of the ISWC + ASWC International Workshop on Ontology Matching (OM)*, pages 96–132, 2007.

[3] Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer, Heidelberg (DE), 2007.

[4] Christiane Fellbaum. *WordNet: an electronic lexical database*. The MIT Press, Cambridge (MA US), 1998.

[5] Fausto Giunchiglia, Fiona McNeill, Mikalai Yatskevich, Zharko Alekovski, Alan Bundy, Frank van Harmelen, Spyros Kotoulas, Vanessa Lopez, Marta Sabou, Ronny Siebes, and Annette ten Tejie. Openknowledge deliverable 4.1: Approximate semantic tree matching in openknowledge. `http://www.cisa.informatics.ed.ac.uk/OK/Deliverables/D4.1.pdf`, 2006.

[6] Fausto Giunchiglia and Pavel Shvaiko. Semantic matching. *The Knowledge Engineering Review*, 18(3):265–280, 2003.

[7] Fausto Giunchiglia, Mikalai Yatskevich, and Enrico Giunchiglia. Efficient semantic matching. In *Proceedings of the European Semantic Web Conference (ESWC)*, pages 272–289, 2005.

[8] Fausto Giunchiglia, Mikalai Yatskevich, and Fiona McNeill. Structure preserving semantic matching. In *Proceedings of the ISWC+ASWC International workshop on Ontology Matching (OM)*, pages 13–24, Busan (KR), 2007.

[9] Fausto Giunchiglia, Mikalai Yatskevich, and Pavel Shvaiko. Semantic matching: Algorithms and implementation. *Journal on Data Semantics*, IX:1–38, 2007.

[10] Henry Kucera and W. Nelson Francis. Brown Corpus Manual. `http://icame.uib.no/brown/bcm.html`, 1979.

[11] Yoonkyong Lee, Mayssam Sayyadian, AnHai Doan, and Arnon Rosenthal. eTuner: tuning schema matching software using synthetic scenarios. *VLDB Journal*, 16(1):97–122, 2007.

[12] George A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[13] David Robertson. A lightweight coordination calculus for agent systems. In *Declarative Agent Languages and Technologies*, pages 183–197, 2004.

[14] Pavel Shvaiko, Jérôme Euzenat, Heiner Stuckenschmidt, Malgorzata Mochol, Fausto Giunchiglia, Mikalai Yatskevich, Paolo Avesani, Willem Robert van Hage, Ondřej Šváb, and Vojtěch Svátek. *KnowledgeWeb Deliverable 2.2.9: Description of alignment evaluation and benchmarking results*. `http://exmo.inrialpes.fr/cooperation/kweb/heterogeneity/deli/kweb-229.pdf`, 2007.

[15] Pavel Shvaiko, Fausto Giunchiglia, Juan Pane, and Paolo Besana. *OpenKnowledge Deliverable 4.3: Plug-in component supporting query answering*. `http://www.cisa.informatics.ed.ac.uk/OK/Deliverables/D4.3.pdf`, 2007.

[16] Pavel Shvaiko, Fausto Giunchiglia, Marco Schorlemmer, Fiona McNeill, Alan Bundy, Maurizio Marchese, Mikalai Yatskevich, Ilya Zaihrayeu, Bo Ho, Vanessa Lopez, Marta Sabou, Joaqín Abian, Ronny Siebes, and Spyros Kotoulas. *OpenKnowledge Deliverable 3.1: Dynamic Ontology Matching: a Survey*. `http://www.cisa.informatics.ed.ac.uk/OK/Deliverables/D3.1.pdf`, 2006.

[17] Pavel Shvaiko, Fausto Giunchiglia, Mikalai Yatskevitch, Juan Pane, and Paolo Besana. Openknowledge deliverable 3.6: Implementation of the ontology matching component. `http://www.cisa.informatics.ed.ac.uk/OK/Deliverables/D3.6.pdf`, 2007.

[18] Cornelis Joost (Keith) van Rijsbergen. *Information retrieval*. Butterworths, London (UK), 2nd edition, 1979.

[19] Mikalai Yatskevich, Fausto Giunchiglia, Fiona McNeill, and Pavel Shvaiko. Openknowledge deliverable 3.3: A methodology for ontology matching quality evaluation. `http://www.cisa.informatics.ed.ac.uk/OK/Deliverables/D3.3.pdf`, 2007.