S-Match: an algorithm and an implementation of semantic matching[1]

Fausto Giunchiglia, Pavel Shvaiko, Mikalai Yatskevich

Dept. of Information and Communication Technology
University of Trento,
38050 Povo, Trento, Italy
{fausto, pavel, yatskevi}@dit.unitn.it

Abstract. We think of *Match* as an operator which takes two graph-like structures and produces a mapping between those nodes of the two graphs that correspond semantically to each other. Semantic matching is a novel approach where semantic correspondences are discovered by computing and returning as a result, the semantic information implicitly or explicitly codified in the labels of nodes and arcs. In this paper we present an algorithm implementing semantic matching, and we discuss its implementation within the *S-Match* system. We also test *S-Match* against three state of the art matching systems. The results, though preliminary, look promising, in particular for what concerns precision and recall.

1. Semantic matching

Preliminary to the definition of semantic matching is the definition of *concept of/at a node*, which, in turn, is based on the notion of *concept of/at a label*. Let us analyze these two notions in turn, starting from the second. The trivial but key observation is that labels in classification hierarchies are used to define the set of documents one would like to classify under the node holding the label. In other words, a label has an intended meaning, which is what this label means in the world. However, when using labels for classification purposes, we use them to denote the set of documents which talk about their intended meaning. This consideration allows us to define the "*concept of/at a label*" as "*the set of documents that are about what the label means in the world*". Trees add structure which allows us to perform the classification of documents more effectively. Thus, we have that "*the concept of/at a node*" is "*the set of documents that we would classify under this node*", given it has a certain label and it is positioned in a certain place in the tree.

We can now proceed to the definition of semantic matching. Let a *mapping element* be a 4-tuple $< ID_{ij}, n1_i, n2_j, R >$, $i=1,...,N1$; $j=1,...,N2$; where ID_{ij} is a unique identifier of the given mapping element; $n1_i$ is the *i-th* node of the first graph, $N1$ is the number of nodes in the first graph; $n2_j$ is the *j-th* node of the second graph, $N2$ is

[1] Long version in Proceedings of the European Semantic Web Symposium, LNCS 3053, pp. 61-75, 2004. Full version and the description of the schemas tested can be found at: http://www.dit.unitn.it/~accord/

Dagstuhl Seminar Proceedings 04391
Semantic Interoperability and Integration
http://drops.dagstuhl.de/opus/volltexte/2005/37

the number of nodes in the second graph; and *R* specifies a *semantic relation* which holds between the concepts of nodes $n2_j$ and $n1_i$. Possible semantic relations are: *equivalence* (=), *more general* (⊒), *less general* (⊑), *mismatch* (⊥), *overlapping* (⊓). Thus, for instance, the concepts of two nodes are equivalent if they have the same extension, they mismatch if their extensions are disjoint, and so on for the other relations. *Semantic matching* can then be defined as the following problem: given two graphs *G1*, *G2* compute the *N1 ´ N2* mapping elements $<ID_{ij}, n1_i, n2_j, R¢>$, with $n1_i \in G1$, $i=1,\ldots,N1$, $n2_j \in G2$, $j=1,\ldots,N2$ and *R¢* the strongest semantic relation holding between the concepts of nodes $n1_i$, $n2_j$.

2. Semantic matching algorithm

The S-Match algorithm is organized in the following four macro steps:

- *Step 1*: for all labels in the two trees, compute concepts at labels
- *Step 2*: for all nodes in the two trees, compute concepts at nodes
- *Step 3*: for all pairs of labels in the two trees, compute semantic relations among concepts denoted by labels
- *Step 4*: for all pairs of nodes in the two trees, compute semantic relations among concepts at nodes

Let us consider for instance the two trees depicted in Figure 1a.

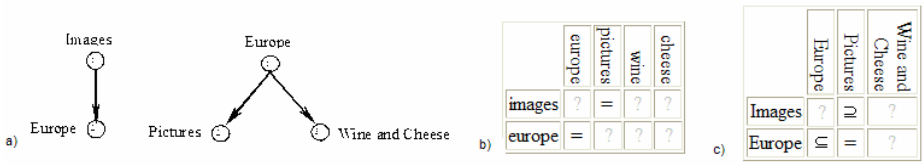

Fig. 1. Two trees (a). The matrix of semantic relations between concepts at labels in the trees (b). The matrix of semantic relations between the concepts at nodes in the trees (matching result) (c).

During Step 1 we try to capture the meaning of the labels in the trees. In order to perform this we first tokenize the complex labels. For instance "*Wine and Cheese*" from Figure 1 becomes <Wine, and, Cheese>. Then we lemmatize tokens; and "*Images*" becomes "*image*". Then, an Oracle (at the moment we use WordNet 2.0 as an Oracle) is queried in order to obtain the senses of the lemmatized tokens. Afterwards, these senses are attached to the atomic concepts. Finally, the complex concepts are built from the atomic ones. Thus, the concept of label *Wine and Cheese*, $C_{Wine\ and\ Cheese}$ is computed as $C_{Wine\ and\ Cheese}$ = <*wine*, $\{senses_{WN}\#4\}$>&<*cheese*, $\{senses_{WN}\#4\}$>, where <*cheese*, $\{senses_{WN}\#4\}$> is taken to be the union of the four WordNet senses.

Step 2 takes into account the structural schema properties. The logical formula for a concept at node is constructed, most often as the conjunction of the formulas in the concept path to the root. Element level semantic matchers are applied during Step 3. They determine the semantic relations holding between the atomic concepts at labels. For example, from WordNet we can derive the information that *image* and *picture* are synonyms. Therefore, the equivalence relation between concepts at labels (C_{Images} =

$C_{Pictures}$) can be inferred. The relations between the atomic concepts at labels for the trees depicted on Figure 1a are depicted on Figure 1b. Element level semantic matchers provide the input to the structure level matcher, which is applied on Step 4 and produces the set of semantic relations between concepts at nodes as the matching result (see Figure 1c). On this step the tree matching problem is reformulated into the set of node matching tasks. Further, each of node matching tasks is reduced into a set of propositional validity problems.

3. A platform implementing semantic matching

Our approach is to develop a *platform* for semantic matching, namely a highly modular system where single components can be plugged, unplugged or suitably customized. The logical architecture of the system we have developed, called *S-Match,* is depicted on Figure 2.

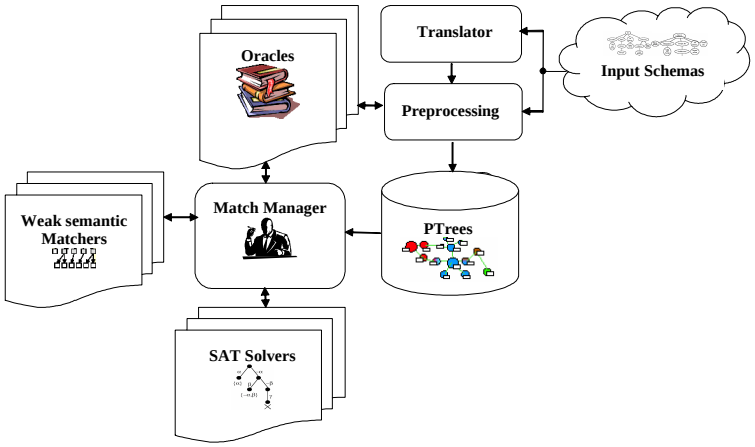

Fig. 2. Architecture of the S-match platform

Let us discuss it from a data flow perspective. The module taking input schemas does the *preprocessing*. It takes in input trees codified into a standard internal XML format. This internal format can be loaded from a file manually edited or can be produced from a input format dependent *translator*. This module implements the preprocessing phase and produces, as output, enriched trees which contain concepts of labels and concepts of nodes. These enriched trees are stored in an internal database (the database labeled *PTrees* on Figure 2) where they can be browsed, edited and manipulated. The preprocessing module has access to the set of *oracles* which provide the necessary *a priori* lexical and domain knowledge. In the current version WordNet is the only oracle we have. The *Matching Manager* coordinates the execution of steps 3 and 4 using the *oracles* library (used here as element level *strong semantics matchers*), the library of element level *weak semantic matchers*, and the library of *SAT solvers*.

4. A comparative evaluation

We have done some preliminary comparison between *S-Match* and three state of the art matching systems, namely Cupid [3], COMA [1], and SF [5] as implemented within the Rondo system [4]. In our evaluation we have used three examples: the simple catalog matching problem, presented in the paper and two small examples from the academy and business domains. The business example describes two company profiles: a standard one (mini) and Yahoo Finance (mini). The academy example describes courses taught at Cornell University (mini) and at the University of Washington (mini). As match quality measures we have used the following indicators: *precision*, *recall*, *overall*, *F-measure*, *overall* and *time*. All the tests have been performed on a P4-1700, with 256 MB of RAM, with the Windows XP operating system, and with no applications running but a single matcher. The evaluation results are shown on Figure 3. From the point of view of the quality of the matching results *S-Match* clearly outperforms the other systems.

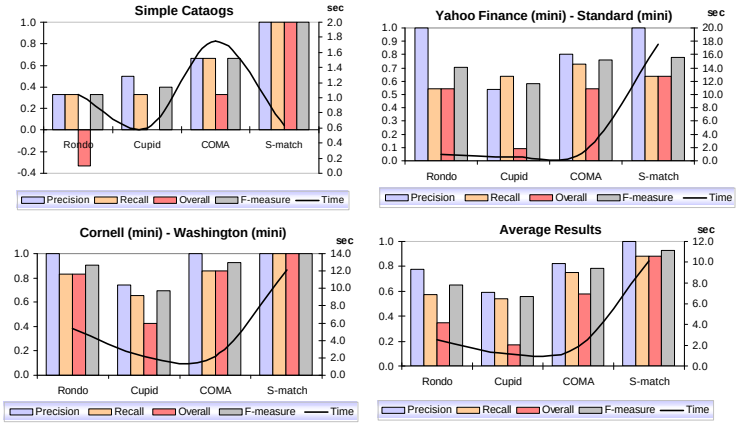

Fig. 3. Experimental results

References

1. Do H.H., Rahm E.: COMA – A System for Flexible Combination of Schema Matching Approach. Proceedings of VLDB'02, (2002) 610-621.
2. Giunchiglia F., Shvaiko P.: Semantic Matching. To appear in "The Knowledge Engineering Review" journal 18(3).
3. Madhavan J., Bernstein P., Rahm E.: Generic schema matching with Cupid. Proceedings of VLDB'01, (2001) 49-58.
4. Melnik S., Rahm E., Bernstein P.: Rondo: A programming platform for generic model management. Proceedings of SIGMOD'03, (2003) 193-204.
5. Melnik, S.,Garcia-Molina H., Rahm E.: Similarity Flooding: A Versatile Graph Matching Algorithm. Proceedings of ICDE, (2002) 117-128.