Instance-Based Ontology Matching Using Regular Expressions

Katrin Zaiß, Tim Schlüter, and Stefan Conrad

Institute of Computer Science
Heinrich-Heine-Universität Düsseldorf
D-40225 Düsseldorf, Germany
{zaiss, schlueter, conrad}@cs.uni-duesseldorf.de

Many algorithms dealing with the matching task have been proposed in the past, most of them not considering instances. Only a few existing systems like [EM07] or [DMDH04] use the information provided by instances in their matching algorithms. Due to the fact that ontologies offer the possibility to model instances within the ontology, these should definitely be used to increase the accuracy of the matching results. Additionally, the set of instances probably provides more information about the meaning of a concept than its label. Thus, we propose a new instance-based ontology matcher which should extend existing libraries to enhance the matching quality.

The basic idea behind our approach is to consider the content of an ontology and not mere the outer form (e.g. the name of a concept or the data type of an attribute). Our algorithm scans a sample of the instances in order to describe the contents of every concept through a set of regular expressions. These sets can easily be converted into single vectors, each of them representing a concept. These concept vectors establish a basis for calculating a similarity value (e.g. the cosine similarity) to discover similar concepts and thus to match ontologies. The flow of our matching process is shown in figure 1.

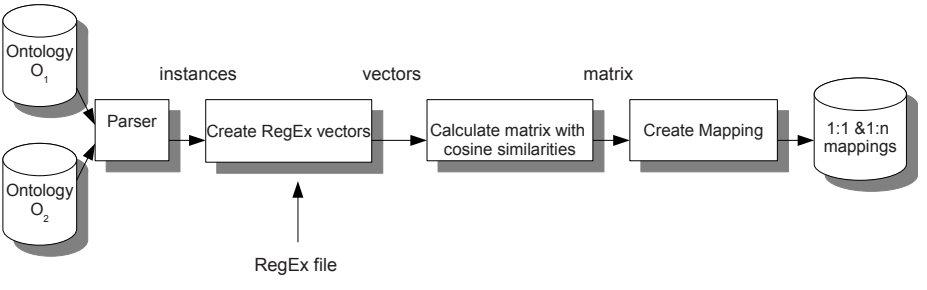

Fig. 1. Matching Process

Building the RegEx Vectors

A domain expert has to create a list of regular expressions that fits to (the instances of) the ontologies for a special domain. The instances are compared to this RegEx list, and the first fitting regular expression is assigned to the instance, whereas the regular expression that is assigned to the majority of the instances belonging to a certain attribute is assigned to this attribute.

R. Meersman, Z. Tari, and P. Herrero (Eds.): OTM 2008 Workshops, LNCS 5333, pp. 40–41, 2008.

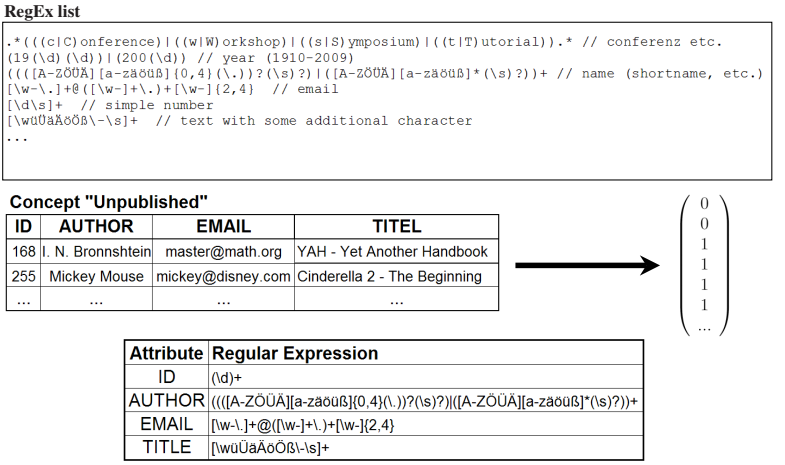

Fig. 2. Creating the RegEx vector

In order to avoid a false assignment in this process, the regular expressions have to be ordered from very specialized at the beginning of the list to more universal at the end.

The RegEx vector for a concept c is a vector $v \in \mathbb{N}^d$ with d being the number of different regular expressions. In every component v_i it contains the number of assignments of the i-th regular expression in RegEx list l to an attribute in c. An illustration of the whole process is given in figure 2.

Creation of Candidate Mapping

For determining a mapping we need the cosine similarities of all possible concepts pairs and a certain predefined threshold. $1:1$ mappings are created by gradually finding the concepts pairs with the highest similarity assuming that every concept has at most one best-fitting matching partner. The whole process is repeated until there are no more similarity values above the threshold, i.e. there are probably no more $1:1$ mappings.

To find $1:n$ mappings we regard the remaining concepts and compute the cosine similarity between one concept vector and the sum of n other vectors. A resulting similarity value above the threshold is a hint for a $1:n$ mapping.

References

[DMDH04] Doan, A., Madhavan, J., Domingos, P., Halevy, A.Y.: Ontology Matching: A Machine Learning Approach. In: Handbook on Ontologies, pp. 385–404. Springer, Heidelberg (2004)

[EM07] Engmann, D., Maßmann, S.: Instance Matching with COMA++. In: Datenbanksysteme in Business, Technologie und Web (BTW 2007), Workshop Proceedings, Aachen, Germany, März 5-6 (2007)