AN EXTENDABLE JAVA FRAMEWORK FOR INSTANCE SIMILARITIES IN ONTOLOGIES

Mark Hefke, Valentin Zacharias, Andreas Abecker, Qingli Wang
FZI Research Center for Information Technologies at the University of Karlsruhe, 76131 Karlsruhe, Germany

Ernst Biesalski, Marco Breiter
DaimlerChrysler AG, Plant Wörth, Daimlerstrasse 1, D-76742 Wörth, Germany

Keywords: Ontologies, Similarity, Knowledge Management, Case-Based Reasoning.

Abstract: We present the conceptual basis and a prototypical implementation of a software framework for syntactical and semantical similarities between ontology instances. Our focus comprises both the implementation of specific, ontology-based similarity measures and their flexible, efficient, and extensible combination.

1 INTRODUCTION

The importance of ontologies increased significantly in the recent years with usage scenarios such as the representation of complex knowledge in application domains like manufacturing, transportation, or life-sciences. For instance, the DAML Ontology Library (http://www.daml.org/ontologies/) contains 282 ontologies in different domains. Ontologies often serve as a common model for the semantic integration of heterogeneous information sources. Here, one has to deal with identifying similar entities (concepts, attributes, relations, instances or even instance sets) within and between ontologies, in order to be able to build semantic bridges between distributed ontologies *(ontology mapping),* or to aggregate complementary ontologies to a common one *(ontology merging)*. Due to the complex structure of ontologies (concepts, concept hierarchies, inverse, symmetric or transitive relations between concepts, etc.), traditional syntactical similarity measures like, e.g., string comparison or distance-based similarity for numerical attribute values alone are not able to return reasonable results for the identification of similar entities. Therefore we have developed an extensible framework for the flexible combination and parameterization of syntactical and semantical similarities, in order to compute the similarity between (sets) of ontology instances. The framework is designed such that it can easily be integrated in any ontology-based system.

The paper is organized as follows: In Section 2, we introduce basic notions for defining similarity measures and then construct some ontology-related similarity measures. In Section 3, we show how such approaches can be implemented in a modular and extensible software framework. After discussing related work in Section 4, we conclude in Section 5.

2 ONTOLOGY-BASED SIMILARITY MEASURES

2.1 Ontologies

In Computer Science, ontologies are formal models of a domain which support the communication between human beings and/or machines. On a socio-cultural level, ontologies demand a shared understanding of concepts and relationships. Technically, let us use the following definition – which comprises both schema and instance data in the ontology.

Definition 1 (Ontology).
$O := (C, HC, RC, HR, I, RI, A)$
An ontology O is a tuple consisting of: concepts C of the schema that are arranged in a subsumption hierarchy HC. Between single concepts exist rela-

Hefke M., Zacharias V., Abecker A., Wang Q., Biesalski E. and Breiter M. (2006).
AN EXTENDABLE JAVA FRAMEWORK FOR INSTANCE SIMILARITIES IN ONTOLOGIES.
In *Proceedings of the Eighth International Conference on Enterprise Information Systems - AIDSS*, pages 263-269
DOI: 10.5220/0002464302630269

tions (properties) RC. Relations can also be arranged in a hierarchy HR. Instances I of a specific concept are interconnected by property instances RI. Additionally, one can define axioms A which can be used to infer new knowledge or to formulate integrity constraints.
Common languages to represent ontologies are RDF(S) (http://www.w3.org/RDF/) or OWL (http://www.w3.org/TR/owl-features/).

2.2 Similarity Measures

A similarity measure is a function for quantitatively computing the similarity between things:
Definition 2 (Similarity Measure). A similarity measure is a real-valued function sim(x, y): $S^2 \rightarrow [0, 1]$ on a set S measuring the degree of similarity between x and y.
If we regard two complex objects in the real world which can be formally described as sets of interrelated ontology instances, the similarity of such two complex entities E_a and E_b can be assessed following the *local-global principle* which is based on the assumption that complex objects are built up from smaller units which can be characterized by a number of attributes and their respective values. To compute local – i.e. attribute-specific – similarities, appropriate local similarity measures are used which have to be defined accordingly. For the computation of the global similarity between E_a and E_b, we combine the results of local similarities, i.e.:

Definition 3 (Similarity Measure for Complex Objects). A similarity measure for two complex objects E_a and E_b consisting of smaller units $(e_{a,1},e_{a,2},...,e_{a,n})$ and $(e_{b,1},e_{b,2},...,e_{b,n})$ is a function

$$Sim(E_a,E_b) := A(sim_1(e_{a,1},e_{b,1}),...sim_n(e_{a,n},e_{b,n}))$$

where $sim_x(e_{a,x},e_{b,x})$ is a similarity function defined on smaller units of E_a and E_b. Please note that those subunits may represent simple attributes as well as more complex entities themselves. A is an aggregation function (e.g., a weighted sum).

2.3 Semantic Similarity Measures

A semantic similarity measure takes into account not just the parts of complex objects, but the content of an ontology as well; it uses background knowledge to better calculate the similarity of two elements. In the context of this paper, we restrict ourselves to only look at the similarity between two instances.

Definition 4 (Semantic Instance Similarity Measure). A Semantic Instance Similarity Measure is a function calculating the similarity of two entities $E_a, E_b \in I$ with respect to an Ontology O=(C,HC,RC,HR,I,RI,A):

$$Sim(E_a,E_b,O) := A(sim_1(E_a,E_b,O),...sim_n(E_a,E_b,O))$$

where $sim_x(E_a,E_b,O)$ is a similarity function that compares E_a and E_b with regard to a certain aspect.

Examples for aspects of E_x are certain attributes, relations of some type, or the position of E_x in a taxonomy. Let us briefly explain two exemplary kinds of similarity measures that take into account the content of the ontology. The instance aspects that these similarity measures work on are taxonomic relations and general (non-taxonomic) relations, respectively.

2.3.1 Taxonomic Similarity

The taxonomic similarity of two instances is calculated by looking at the relative taxonomic position of the concepts of the regarded instances. One way to formalize the notion of "relative position" is by looking at the "Semantic Cotopy" – the set of all super concepts – of an instance:
Definition 5 (Taxonomic Similarity). Be SC(i) ("Semantic Cotopy") a function that returns the set of all super concepts of an instance, then the taxonomic similarity between instances $i_1, i_2 \in I$ is defined as:

$$sim_{taxonomic}(i_1,i_2) := \frac{|SC(i_1) \cap SC(i_2)|}{|SC(i_1) \cup SC(i_2)|}$$

2.3.2 Set Similarity

The set similarity is not directly used to compare two instances, but other instance similarity measures (like the relational similarity described below) depend on it. Set similarity compares sets of instances; the method described here is just one example how to calculate this. The set similarity we describe here is based on ideas from multi-dimensional scaling. Given a similarity measure that can compare two instances, the set similarity first represents each in-

stance by a vector with the similarity to all instances in one of the compared sets. Then each of the two sets is represented by the average of these vectors, and the similarity between the sets is given by the cosine of the vectors representing the sets.

Definition 6 (Set Similarity).

$$sim_{set} := \cos\left(\frac{\sum_{e\in E}\vec{e}}{|E|}, \frac{\sum_{f\in F}\vec{f}}{|F|}\right)$$

where E is a complex instance $E = \{e_1, e_2 ...\}$, $\vec{e} = (sim(e,e_1), sim(e,e_2), ..., sim(e,f_1), sim(e,f_2), ...)$ F and $\vec{f}$ are defined analogously.

2.3.3 Relational Similarity

Relational similarity looks at the outgoing and incoming relations, thus realizing the idea that instances can be considered more similar if they have relations to similar entities:

Definition 7 (Relational Similarity).

$$sim_{relation}(i_1, i_2) := \frac{\sum_{k=1...n} sim_{set}(J_{k1}, J_{k2})}{n}$$

where J_{k1} is the set of instances linked to i_1 via a relationship, and J_{k2} analogous. Through the index k, all relationships in RC are enumerated.

3 SIMILARITY FRAMEWORK

On top of the KAON ontology management system (http://kaon.semanticweb.org/), we built an extendable Java framework for calculating instance similarities in RDF(S) ontologies. It comes with a set of semantic similarity measures, including those described above. For a concrete application use case, such as calculating the match between job offers and skill profiles described in an ontology (see Section 4, or (Biesalski et al., 2005)), an expert can create a customized similarity measure by writing an XML file that describes how semantic similarity measures are used to compare the different aspects of the complex instances. New semantic similarity measures can be added without changing the framework.

3.1 Similarity Measures

As stated already, complex similarity measures can be defined by an XML file that describes how to combine the atomic measures. A simple example:

```
<similarity name='Example'
   concept='fzi.de/kmir#Profile'>
   <instanceRelationSimilarity weight='3'

relationType='fzi.de/kmir#hasProblem'>

      <similarity>
         <syntacticSimilarity
language='de'
            attributeURI='labelValue'/>
      </similarity>

   </instanceRelationSimilarity>
</similarity>
```

The first `<similarity>` tag states that this measure has the name `"Example"` and that it is used exclusively to calculate the similarity between instances of the concept `fzi.de#Profile`. This outermost tag creates a new complex similarity measure; its content describes how it is calculated. The measure `"Example"` only considers one aspect of instances when comparing them: their relation `fzi.de/kmir#hasProblem`.

The similarity of two profile instances only depends on the similarity of the problems they are associated with. The `<instanceRelationSimilarity>` tag gives information about this part of the similarity measure. The attribute `relationType` states which relation we are talking about; the `weight` attribute defines the relative weight of this aspect compared to other aspects of a complex similarity measure (here we only have one and could have omitted this attribute).

We have now defined that the similarity between `"#Profile"` instances only depends on the similarity of the instances they have `#hasProblem` relations to; what we haven't said yet, is how these problem instances are compared. As a last resort the system has a `"defaultSimilarity.xml"` that defines a similarity method that would be used if no other can be found, but this is a very rudimentary. Another possibility is to include a similarity measure for `#Problem` instances in the same XML file; this similarity measure would be preferred to the default one. The third possibility has been used in the example: the description of the similarity measure used for the related instances is nested inside the `<instanceRelationSimilarity>` tag. In the example, the similarity measure used to compare the problem instances again only considers one aspect: the syntactic similarity of the (German) labels – the similarity of two problem instances depends on the Levenshtein distance of their labels.

3.2 Filters

Two different kinds of filters allow to specify which instances are considered in the calculation and to precisely define what results are returned. They can be individually combined from atomic filters.

Pre-filters allow a precise definition of the instances that should be considered for similarity computation. These filters are important because similarity calculations can be computationally very expensive.

```
<preFilter>
   <inclusiveConceptFilter
      concept="fzi.de#Human"/>
   <inclusiveConceptFilter
      concept="fzi.de#chimp"/>
</preFilter>
```

The example defines a filter that consists of two atomic filters. Both state that instances of a certain concept should be included – here instances of the concepts `#Human` and `#Chimp`.

It is possible to use filters based on KAON Queries (cf. "Developers Guide for KAON") in the XML-based definition of filters in order to further reduce the number of considered instances. The KAON conceptual query language allows easy and efficient locating of elements in KAON OI-models. For instance, a filter with the query

```
[#Profile] AND
SOME(<#hasOrganisationType>,!#SME!)
```

restricts the instances that are considered to those of type `#Profile` and with an organisation type `#SME` (small and medium enterprises)

Post filters determine how many instances are returned after similarity computation. There are two filter types, `minSimilarityFilter` and `maxCountofInstancesFilter`. The first defines the minimum similarity to the query instance for an instance to be included in the result. The second defines a cut-off for the number of results returned.

```
<postFilter>
   <minSimilarityFilter minimum="0.3"/>
   <maxCountOfInstancesFilter
maximum="10"/>
</postFilter>
```

Here, a filter is defined that restricts the number of results to 10 and returns only instances with a similarity of at least 0.3 to the query instance.

3.3 Adding Similarity Measures

For use cases where the atomic similarities included in the system are not sufficient, it is possible to add new similarity measures.

A new similarity measure must implement the interface *Similarity* or *InstanceRelationSimilarity*: The second one is for similarity measures that will need information about the similarity of instances related to the currently compared ones (like the relational similarity described earlier). Once the new similarity measure has been implemented, the framework is alerted about its existence through a change in the tag library `"similarity.tld"`:

```
<tag>
   <name>taxonomicSimilarity</name>
   <tagclass>
      de.fzi.wim.similarity.TaxSim
   </tagclass>
</tag>
```

This example adds a tag `<taxonomicSimilarity>` and defines which class implements it. This tag can then be used alongside the other similarity measures in XML-files.

4 USE CASES

The strength of our framework is that is easily adaptable to different domains and use cases. We shortly sketch two successful use cases.

4.1 Case-Based Reasoning

The KMIR (Knowledge Management Implementation and Recommendation) Framework (Hefke, 2004) supports organizations in the implementation of Knowledge Management (KM) by providing recommendations wrt. technological, organizational, and human aspects. These recommendations are based on best practice cases (BPCs) for successful KM introduction. BPCs are structured and stored in an ontology. A typical KM introduction scenario is described as follows: An organization intends to introduce KM. With a web-based self-description component, it describes its profile in consideration of organizational structure (business size, sector, legal form, processes, etc.) and infrastructure (used tools and technologies), financial ratios (e.g., turnover, profit) as well as economic aspects for KM introduction (e.g., planned implementation time, amortization time and target costs). Moreover, the organization can define knowledge problems and requirements, as well as normative, strategic and operative (knowledge) goals wrt. KM. Finally, the organization assigns weights to all described aspects in order to attach more or less importance to them. After that, the "profile" is stored as a set of instances/ relations in the ontology representing in this scenario the case base. In a next step, a matching component identifies the most similar case(s) in the case base which con-

sider the same "problem situation" concerning the introduction of KM by matching the organization profile against already existing BPCs.
In order to retrieve BPCs that are most similar to a newly created profile achieved from the self-description process, a matching component matches the profile against already existing BPCs from the case base. This is done by combining syntactical similarity measures (***distance-based similarity, syntactical similarity*** and ***equality***) with semantical similarity measures (***relation similarity, taxonomic similarity*** and ***set similarity***). Finally, the most similar BPC(s) from the case base is/are presented to the requesting organization including solutions and methods for solving the similar problem situation.
Distance-based Similarity is used to compare values of numeric data types (e.g. turnover, profit, number of KM workers, etc.) from the organization profile with those of existing BPCs. ***Syntactical Similarity*** and ***Equality*** are used for string comparisons in order to compare problem or goal descriptions, the name of specific tools or technologies. ***Relation similarity*** is used for, e.g., comparing instantiations of the concept "problem" that are linked to further instantiations of the concept "Core process" using the relation "(problem) addresses core process". ***Set similarity*** compares each instance or set of instances in an organisation profile with those of the BPC(s). ***Taxonomic Similarity*** identifies similar profile instances based on their position in the taxonomy. For instance, an organization is searching for an extension of its existing groupware system G_1 in order to achieve better search results. The matching component identifies a similar groupware system G_2 in the case base (which has been extended with a semantic search functionality) by regarding all instances of the corresponding concept "groupware" resp. of more general/specific ones and recommends the assigned solution to the requesting organization.
Finally, a weighted average determines the global similarity of all computed local similarities between the organisation profile, and each of the BPCs, and presents a ranked list of the best matching results.

4.2 Case-Based Reasoning

A further application domain of the similarity framework has been introduced at DaimlerChrysler AG, Wörth. Efficient skill management is a key factor wrt. human resources. Here, matching skill profiles with position requirements is an essential, yet complex task that is performed in order to staff positions or project teams, to provide strategically optimized training recommendations, or to perform succession planning. Current approaches often lack comprehensive means to compare skill profiles considering interrelations of skills, synonyms, varying skill metrics of different data sources and application domains. Our approach (Biesalski et al., 2005) allows to overcome these challenges with an integrated, ontology-based skill catalogue for storing and managing individual skills as well as profiles.
On the basis of the presented similarity framework, we were able to provide decision makers with flexible similarity measures based on different ***compound similarity measures***:

- **Direct skill comparison:** we require an *exact match* of as-is and to-be. So we can specify *K.O. criteria* for the central requirements, especially in strategically important jobs.
- **Proportional similarity:** we identify also *partially fulfilled requirements*. This is also important if we can plan for additional teaching and qualification, or for "training on the project".
- **Compensatory similarity:** we identify not only partially fulfilled requirements, but also *overqualifications*; so, additional expertise on one hand may compensate deficiencies on the other hand. If several employees fulfil the K.O. criteria, this can be used to find the most suited one.
- **Taxonomic similarity:** the taxonomic structure of the skill ontology is taken into account to find "close matches" in the case that no employee has exactly the required qualifications. Also usable for deciding between several candidates, and for refining profile specifications.

Yet since decision makers need to be able to put a different emphasis on individual skills when staffing, specification of different weights for certain skill requirements or definition of subsets as mandatory elements had to be allowed. In order to support skill matching with these additional constraints, we had to introduce customized ***set similarity, instance relation*** and ***compound similarity measures,*** which dynamically compute weights that are specified in the ontology rather than statically given in the similarity configuration file. This was accomplished by definition of dedicated weight properties of instances and provision of references to this weight setting in the customized configuration format:

```
<instanceRelationSimilarity weight="#has-
weight" relationType="#has-skill"
depth="3">
</instanceRelationSimilarity>
```

Another challenge in skill management is to adjust and optimize efficiency and effectiveness considering the average number of skills within a profile, the size of the skill catalogue, or the granularity of

weight metrics. Our approach simplifies the definition and ongoing adjustment of the similarity measures within the life cycle of the skill management solution. This was accomplished by provision of adaptable configuration files which can be customized according to the demands of the ever-changing environment.
Our approach allows us to successfully provide executives with context-dependent similarity measures of skill profiles and position requirements.

5 RELATED WORK

Related work can be coarsely separated into two categories: (i) the ontology mapping / alignment approaches, and (ii) the object similarity approaches.
Ontology alignment and mapping approaches are concerned with finding corresponding concepts and instances in different ontologies. They are mostly motivated by information integration problems where different ontologies for the same domain need to be joined or aligned. Approaches comprise the FOAM Framework for Ontology Alignment and Mapping (Ehrig & Sure, 2005), AnchorPROMPT (Noy & Musen, 2003), or GLUE (Doan et al., 2002). These systems often exploit anchor-based approaches to conclude semantic similarity from similar connection structures in the graph constituted by the relationships between ontology instances.
Object similarity approaches are more similar to the work presented here and compute the similarity of objects (instances or concepts) within the same ontology.
Bernstein and colleagues created the SimPack framework (Bernstein et al., 2004) that uses a set of similarity measures to calculate the similarity of different concepts. The major difference to our similarity framework is that they are trying to build a generic similarity measure that works for all domains, although they do acknowledge the need for what they call "personalised similarity measures".
SemMF (Oldakowski & Bizer, 2005) is a simple customizable framework for instance similarities in ontologies; it finds the most similar objects for a query instance in a set of resource objects. The properties of the query instance need not exactly correspond to properties of the resource objects (a mapping between them can be specified in an RDF file), but the concept taxonomy needs to be the same. Compared to our approach, SemMF lacks support for the recursive and set like characteristics of relational similarity – SemMF cannot compute the similarity of objects based on their relations to other objects; relations can only be treated like attributes.
Culmone and colleagues (Culmone et al., 2002) describe a simple mechanism to calculate the similarity between concepts based on the number of relations to identical concepts.
(Diaz-Agudo & Gonzalez-Calero, 2001) describe the architecture of a CBR framework that encompasses similarity of cases described in an ontology. jColibri (Bello-Tomás et al., 2004) implements a CBR framework for the entire CBR lifecycle (retrieve, reuse, revise, remember). Similarity of cases with respect to background knowledge is not a part of its core, but this functionality can be added through Problem-Solving Methods (PSMs).

6 CONCLUSION

We have presented a flexible and extensible similarity framework for instance-similarity computation in ontologies. In order to later guarantee the usefulness of the framework in different application domains, we started with a collection, analysis and selection of commonly used traditional similarity measures, as well as of ontology-related similarity measures. Here, we rely among others on the theoretical results of (Ehrig et al., 2005) which provide "*a comprehensive framework for measuring similarity within and between ontologies as a basis for the interoperability across various application fields"*. Further, we applied our framework in two different application domains (Knowledge and Skill Management).
The framework is implemented in Java 1.4 on top of KAON and uses Xerxes for parsing XML configuration files (http://xml.apache.org/xerces2-j/).
For the future, we plan to technically enhance the framework, which currently supports RDF(s) and the more expressive, proprietary KAON language. Therefore, we will also support the web ontology language OWL (http://www.w3.org/TR/owl-ref/). OWL, for instance, provides different data types, further special property characteristics, and property restrictions, which are not supported in KAON, but promise added value for similarity computation. We also plan to consider OWL DL's complex classes (e.g., by regarding the set operators "intersectionOf", "unionOf" and "complementOf"), which enables new ways of filtering similarity sets by directly using the ontology.

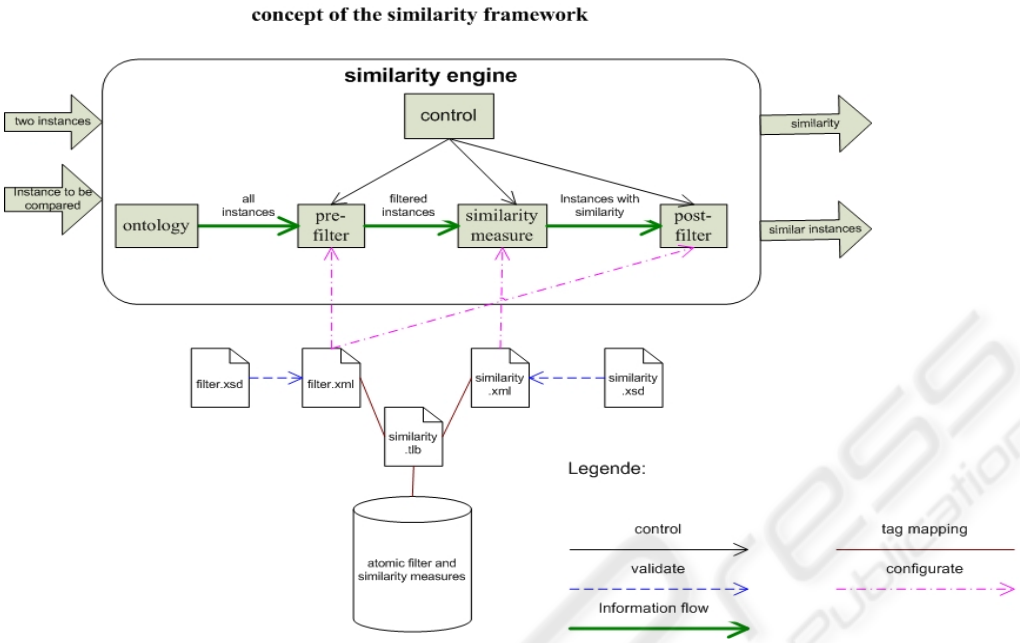

Figure 1: Major Components of the Similarity Framework.

ACKNOWLEDGMENTS

The work presented has been supported by the European Commission under grant OntoGov (IST-507237) and DIP (IST-507483), and by the German National Ministry for Education and Research bmb+f under grant Modale (FKZ 01ISC28H). Most of the work presented was implemented in the Diploma theses of Mrs. Qingli Wang and Mr. Marco Breiter.

REFERENCES

Bello-Tomás, J.J., González-Calero, P.A., Díaz-Agudo, B., 2004. JColibri: an Object-Oriented Framework for Building CBR Systems. In Funk, P., González-Calero, P.A., (eds.): Advances in Case-Based Reasoning, Proc. ECCBR-2004. LNAI 3155, Springer

Bernstein, A., Kaufmann, E., Bürki, C., Klein, M., 2004. Object Similarity in Ontologies: A Foundation for Business Intelligence Systems and High-Performance Retrieval. In: 25th Int. Conf. on Information Systems

Biesalski, E., Breiter, M., Abecker, A., 2005. Towards Integrated, Intelligent Human Resource Management. 1st workshop "FOMI 2005", Formal Ontologies Meet Industry

Culmone, R., Rossi, G. and Merelli, E., 2002. An Ontology Similarity Algorithm for BioAgent (Poster), NETTAB02 Agents in Bioinformatics, Bologna

Díaz-Agudo, B., González-Calero, P. A., 2001. A Declarative Similarity Framework for Knowledge Intensive CBR. In Aha, D., Watson, I., (eds.): Case-Based Reasoning Research and Development, ICCBR-2001, LNAI 2080, Springer

Doan, A., Madhavan, J., Domingos, P. and Halevy, A., 2002. Learning to Map Between Ontologies on the Semantic Web. In Proc. of the Eleventh Int. World Wide Web Conference, Honolulu, Hawaii, USA

Ehrig, M., Sure, Y., 2005. Adaptive Semantic Integration – Proc. ODBIS workshop at the VLDB-2005. Trondheim, Norway

Ehrig, M., Haase, P., Stojanović, N., Hefke, M., 2005. Similarity for Ontologies - A Comprehensive Framework. In 13th European Conf. on Information Systems

Hefke, M., 2004. A Framework for the Successful Introduction of KM Using CBR and Semantic Web Technologies. J. Universal Computer Science 10(6):731-739, Springer

Noy, N.F., Musen., M.A., 2003. The PROMPT Suite: Interactive Tools for Ontology Merging and Mapping. Int. J. of Human-Computer Studies, 59(6):983-1024

Oldakowski, R., Bizar, C., 2005. SemMF: A Framework for Calculating Semantic Similarity of Objects Represented as RDF Graphs, Poster at the 4th Int. Semantic Web Conference