# Ontology Mapping – An Integrated Approach

Marc Ehrig and York Sure

Institute AIFB, University of Karlsruhe
{ehrig,sure}@aifb.uni-karlsruhe.de

**Abstract.** Ontology mapping is important when working with more than one ontology. Typically similarity considerations are the basis for this. In this paper an approach to integrate various similarity methods is presented. In brief, we determine similarity through rules which have been encoded by ontology experts. These rules are then combined for one overall result. Several boosting small actions are added. All this is thoroughly evaluated with very promising results.

## 1 Introduction

The Semantic Web community has achieved a good standing within the last years. As more and more people get involved, many individual ontologies are created. Interoperability among different ontologies becomes essential to gain from the power of the Semantic Web. Thus, mapping and merging of ontologies becomes a core question. As one can easily imagine, this can not be done manually beyond a certain complexity, size, or number of ontologies any longer. Automatic or at least semi-automatic techniques have to be developed to reduce the burden of manual creation and maintenance of mappings.

One specific application at Karlsruhe, which requires mapping and merging is derived from the SWAP project (Semantic Web and Peer-to-Peer). The SWAP project[1] wants to enable individuals to keep their own work views and at the same time share knowledge across a peer-to-peer network. For this reason tools are provided for each peer to easily create an own ontology. This ontology represents the view on the local file system, emails, or bookmarks. Through the peer-to-peer network communication between the individual peers becomes possible without relying on a central instance. Formal queries are sent around in this network, and peers which know an answer reply to these queries. (Natural language) Examples for such queries could be: “What is the the email address of York?” or “Which documents on Marc’s computer are about similarity?”. This knowledge can then be integrated into the knowledge repository of the original asking peer. Additionally every peer advertises the topics he has most information on; this expertise is saved by the other peers.
In our scenario mapping becomes necessary for different tasks. Mapping is required every single time a decision is taken on which peer has knowledge about a certain topic, and thus will be addressed with the query. Naturally, a foreign peer can only answer incoming queries, if it can interpret the entities with respect to its own knowledge base.

[1] http://swap.semanticweb.org

Query rewriting is required [5]. Finally, the originally asking peer receives answers. When including this information into the own local knowledge base, the new knowledge has to be linked to already existing knowledge. Equal entities have to be identified.

In this paper we present an approach to combine different similarity measures to find mapping candidates between two or more ontologies. **As our hypothesis H we expect better mapping results from intelligent approaches in combining different similarity identifying measures than today's approaches can provide.**

The next section defines and explains general concepts this work is based on: ontology, similarity, and mapping. In section 3 the similarity methods based on rules derived by human experts are introduced. The section 4 presents our approach for combining and integrating these various methods. In section 5 a thorough evaluation is performed showing the strengths of our approach. Finally related work, the next steps, and a conclusion are given.

## 2 Definitions

In this section our understanding of ontologies is presented. For clarification we also discuss the general meaning of similarity. Additionally follow ideas on how to bring the two worlds together. Our notion of mapping will be presented at the end.

### 2.1 Ontologies

In philosophy an ontology is *a particular theory about the nature of being or the kinds of existents*. The following short definition describes ontologies as used in our scenario. In the understanding of this paper they consist of both schema and instance data.

$$O := < C, H_C, R_C, H_R, I, R_I, A >$$

An ontology $O$ is a tuple consisting of the following. The concepts $C$ of the schema are arranged in a subsumption hierarchy $H_C$. Relations $R_C$ exist between single concepts. Relations (properties)[2] can also be arranged in a hierarchy $H_R$. Instances $I$ of a specific concept are interconnected by property instances $R_I$. Additionally one can define axioms $A$ which can be used to infer knowledge from already existing one. An extended definition can be found in [22]. Common languages to represent ontologies are RDF(S)[3] or OWL[4], though one should note that each language offers different modelling primitives and, thus, a different level of complexity.

### 2.2 Similarity

We start with a short definition of similarity from Merriam Webster's Dictionary: *having characteristics in common: strictly comparable.* From our point of view we want to

[2] In this paper we treat the words *relation* and *property* as synonyms.
[3] http://www.w3.org/RDFS/
[4] http://www.w3.org/OWL/

strictly compare two entities to find identity among them. The definition already gives us a hint on how to check for similarity: two entities need common characteristics to be similar. We also give a formal definition of similarity here derived from [3]:

- $sim(x, y) \in [0..1]$
- $sim(x, y) = 1 \rightarrow x = y$: two objects are identical.
- $sim(x, y) = 0$: two objects are different and have no common characteristics.
- $sim(x, y) = sim(y, x)$: similarity is symmetric.

### 2.3 Similarity for Ontologies

What is the meaning of similarity in the context of ontologies? The basic assumption is that knowledge is captured in an arbitrary ontology encoding. Based on the consistent semantics the coherences modelled within the ontology become understandable and interpretable. From this it is possible to derive additional knowledge such as, in our case, similarity of entities in different ontologies. An example shall clarify how to get from encoded semantics to similarity: by understanding that labels describe entities in natural language one can derive that entities having the same labels are similar. A formal definition of similarity for ontologies follow:

- $O_i$: ontology, with ontology index $i \in \mathbb{N}$
- $sim(x, y)$: similarity function
- $e_{ij}$: entities of $O_i$, with $e_{ij} \in \{C_i, R_i, I_i\}$, entity index $j \in \mathbb{N}$
- $sim(e_{i_1 j_1}, e_{i_2 j_2})$: similarity function between two entities $e_{i_1 j_1}$ and $e_{i_2 j_2} (i_1 \neq i_2)$; as shown later this function makes use of the ontologies of the entities compared

The paper focuses on the similarity of pairs of single entities from different ontologies.

### 2.4 Mapping

Due to the wide range of expressions used in this area (merging, alignment, integration etc.), we want to describe our understanding of the term "mapping". We define mapping as cf. [23]: "Given two ontologies A and B, mapping one ontology with another means that for each concept (node) in ontology A, we try to find a corresponding concept (node), which has the same or similar semantics, in ontology B and vice verse." We want to stick to this definition, more specific we will demand the *same* semantic meaning of two *entities*.

Formally an ontology mapping function can be defined the following way:

- $map : O_{i_1} \rightarrow O_{i_2}$
- $map(e_{i_1 j_1}) = e_{i_2 j_2}$, if $sim(e_{i_1 j_1}, e_{i_2 j_2}) > t$ with $t$ being the threshold
  entity $e_{i_1 j_1}$ is mapped onto $e_{i_2 j_2}$; they are semantically identical, each entity $e_{i_1 j_1}$ is mapped to at most one entity $e_{i_2 j_2}$

**The central contribution of this paper is to present an approach for defining this mapping function.** We only consider one-to-one mappings between single entities. Neither do we cover mappings of whole ontologies or sub-trees, nor complex mappings as concatenation of literals (e.g. name corresponds to first name plus last name) or functional transformation of attributes (e.g. currency conversions).

## 3 Similarity Measures

Our mapping approach is based on different similarity measures. In this section we want to describe how the various similarity methods have been created.

### 3.1 Manual Rules

Our implemented approach is based on manually encoded mapping rules. Please note that the mappings itself are not yet encoded through rules (as in [16]). We are using rules to identify possible mappings. This manual effort is necessary because coherences in ontologies are too complex to be directly learned by machines. An expert understanding the encoded knowledge in ontologies formulates machine-interpretable rules out of the information. Each rule shall give a hint on whether two entities are identical, but no rule for itself provides enough support to unambiguously identify a mapping. Naturally, evaluation of these manually created rules has to be a core element of the overall process.

### 3.2 Similarity Stack

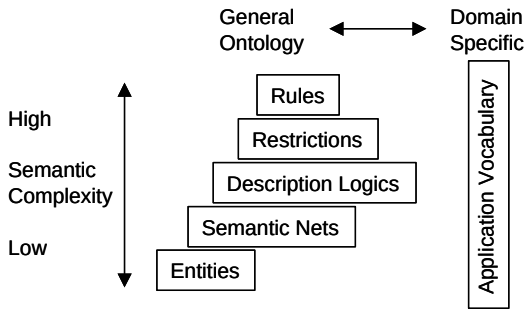


**Fig. 1.** Similarity stack

The presented general idea will now be explicitly used to determine similarity between ontologies. To get a better understanding, the rules are categorized in a similarity stack as shown in graph 1. Ontologies are based on certain vocabularies which are well-defined, well-understood, and with a generally accepted meaning. The left part shows these aspects arranged along their complexity, which is derived from the "layer cake" of [2]. Special shared ontology domains e.g. SWAP-common in the mentioned SWAP project, have their own additional vocabulary. The right part therefore covers domain-specific aspects. As this domain-specific knowledge can be situated at any level of ontological complexity, it is presented as a box across all of them. In the next paragraphs the general semantic meaning of features is described followed by the concrete derived rules, being tagged with a number (**R**$n$) with $n \in (1, ..17)$. As many of the rules are derived from existing literature, we give references where applicable.

**Entities** The first level describes entities as is. No ontological considerations are needed for these features. Labels are human identifiers (names) for entities, normally shared by a community of humans speaking a common language. We can therefore infer that *if labels are the same, the entities are probably also the same* (**R1**, see example 1). Several ideas have already been created to compare labels, e.g. the edit distance[12]. Dictionaries (e.g. WordNet) can further be used for comparisons even across languages, although some restrictions apply. Another feature of objects can be an identifier such as URIs, which are unique for every entity. Therefore we know that *if two entities have the same identifier they are identical* (**R2**).

```
<owl:Class rdf:ID=''id1''>
    <rdfs:label>telephone number</label>
</owl:Class>
<owl:Class rdf:ID=''id2''>
    <rdfs:label>phone number</label>
</owl:Class>
```

**Example 1.** Two entities id1 and id2 with similar labels.

**Semantic Nets** The second level is the level of Semantic Nets as e.g. introduced by [18]. A concept is a general class of objects. They are in relation to others through attributes or properties. *If the properties of two concepts are equal, the concepts are also equal* (**R3**). The same is true for properties. *If the domain and range (the original and the result concept) of two properties are equal, the properties are also* (**R4**).

**Description Logics** The third level described here covers ontologies which have the complexity as provided by Description Logics [1]. A taxonomy can be created over concepts, in which a concept inherits all the relations of its super-concepts. Another rule is that if concepts are the same, they will probably have the same super-concepts. We turn the rule around: *if super-concepts are the same, the actual concepts are similar to each other* (**R5**). In practice we calculate the degree of overlap of the two super-concept sets, which provides a number between 0% and 100% [6]. And finally the sub-concepts of two equal classes will also be the same. *If sub-concepts are the same, the compared concepts are similar* (**R6**) [13]. Also, *if concepts have similar siblings (i.e. children of parents), they are also similar* (**R7**). It is also possible to group properties into a taxonomy, with the corresponding rules resulting: *super-properties* (**R8**) and *sub-properties* (**R9**). The next piece of information which can be added are instances. An instance is a specific entity of a general class from which it inherits all the relations. A concept on the other hand can also be defined as a representative for a set of instances. We can therefore infer that *concepts that have the same instances are the same* (**R10**) [10]. Vice versa, *instances that have the same mother concept are similar* (**R11**). It is also interesting to have a look at the possible distribution of instances on concepts. *If concepts have a similar low/high fraction of the instances, the concepts are similar* (**R12**). Like concepts are interconnected via properties, instances are also regarded to be interconnected via properties instances. This means that *if two instances are linked to another instance via the same property, the two original instances are similar* (**R13**). To a certain degree we can also turn this around: *if two properties connect the same two instances, the properties can be similar* (**R14**).

**Restrictions** We continue with ontologies using restrictions. This is covered by e.g. the ontology language OWL. In OWL there are properties such as "sameIndividualAs" or "sameClassAs". *They explicitly state that two entities are the same* (**R15**). A number of further features from OWL could be used, but are discarded at this time, as they do not have any wide distribution yet: property characteristics as symmetry, restrictions of values, equivalence, set operators, enumeration, and disjointness. From all of them new rules to determine similarity can be derived.

**Rules** Higher levels of the ontology "layer cake" [2] can also become interesting for similarity considerations. Especially if similar rules between entities exist, these entities will be regarded as similar. For this one would have to process higher-order relationships. Unfortunately there has not been sufficient research and practical support for the rule layer in the Semantic Web in general, not at all for similarity considerations.

**Application-Specific Vocabulary** One can also exploit clearly defined application-specific vocabulary for similarity considerations. As an example we take the ontology used within the SWAP project, in which every file has a unique hash-code assigned. *If the hash-codes of two files are the same, one can infer that they are the same* (**R16**). Additionally, *files with the same MIME-type are similar*, at least in their format (**R17**).

**Similarity Paths** In a bigger environment one can expect to have to do more complex mapping e.g. of elements of multiple ontologies. In this case we can use the notion of similarity itself to receive information on other mappings. Similarity as defined here has transitive characteristics if A is similar to B, and B is similar to C, A is similar to C. Some relaxation has to be added when the paths become too long.

## 4 Integrated Approach

### 4.1 Combination

According to our hypothesis, a combination of the so far presented rules leads to better mapping results compared to using only one at a time. Clearly not all introduced similarity methods have to be used for each aggregation, especially as some methods have a high correlation. We present both manual and automatic approaches to learn how to combine the methods. Even though quite some methods exist, no research paper focused on the combination and integration of these methods yet.

**Summarizing** A general formula for this integration task can be given by summarizing over the $n$ weighted similarity methods.

$$sim(e_{i_1 j_1}, e_{i_2 j_2}) = \sum_{k=1}^{n} w_k sim_k(e_{i_1 j_1}, e_{i_2 j_2})$$

with $w_k$ being the weight for a specific method $sim_k$ and $n \in \mathbb{N}$

Please note our assumption that similarities can be aggregated and are increasing strictly. The weights could be assigned manually or learned e.g. through maximization of the f-measure (see section 5) of a training set. In our approach we are basically looking for similarity values supporting the thesis that two entities are equal. If a measure doesn't support the thesis, it still doesn't mean that it's opposing it. These considerations are directly derived from the open world assumption which we respect in this paper.

**Sigmoid Function** A more sophisticated approach doesn't only weight the similarity methods but performs a functional computation on each of them. In the given case the most promising function would be the sigmoid function, which has to be shifted to fit our input range of $[0 \dots 1]$ (see figure 2).

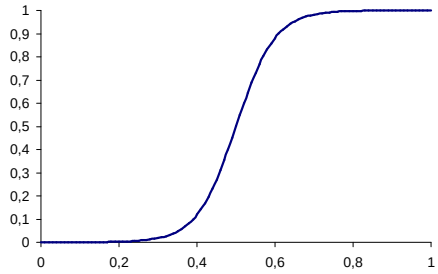


**Fig. 2.** Sigmoid function

$$sim(e_{i_1 j_1}, e_{i_2 j_2}) = \sum_{k=1}^{n} w_k \times sig_k(sim_k(e_{i_1 j_1}, e_{i_2 j_2}) - 0.5)$$
with $sig(x) = \frac{1}{1+e^{-ax}}$ and $a$ being a parameter for the slope

The idea behind using a sigmoid function is quite simple: a high similarity value should be weighted over-proportionally whereas a low value practically can be abandoned. An example shall make this clear. When comparing two labels the chance of having the same entity if only one or two letters are different is very high. On the other hand if only three or four letters match there is no information in this similarity at all. The parameters of the sigmoid function can be regarded as an extension of the similarity methods, as they have to be adjusted according to the method they are applied to.

**Machine Learning with Neural Networks** A very convenient way of determining how to combine the methods is to use a machine learning approach. As we have continuous inputs, only some machine learning approaches make sense. In our work we focus on neural networks [11], as they represent an appropriate way to learn non-linear functions. Specifically we choose a three layer fully connected neural network consisting of a linear input layer, a hidden layer with a tanh-function, and a sigmoid output function. A lot of literature discusses how to choose the number of layers, nodes, and edges. We

will stick to a simple approach, as we focus on similarity considerations rather than efficient machine learning. Support vector machines are another alternative. Unfortunately one needs a large number of examples for training, which is currently difficult to obtain.

### 4.2 Cut-off

After the just described steps we have a list which consists of the most similar entities of two ontologies plus the corresponding similarity value. Now remains the question which level of similarity is appropriate to indicate equality for the mapping and which strongly indicates inequality? It is important to make the decision where to put the cut-off. We use the thresholds as cf. [9]. Every similarity value above the cut-off indicates a match; everything below the cut-off is dismissed.

**Constant Similarity Value** For this method a fixed constant $c$ is taken as cut-off.

$$b = c, \text{ with } b \text{ being the cut-off}$$

The difficulty is to determine this value. Possible approaches are to take an average which maximizes the f-measure in several test runs. Alternatively it might make sense to let experts determine the value, which only works if the similarity value can be interpreted completely (e.g. with the sigmoid summarization).

**Delta Method** For this method the cut-off value for similarity is defined by taking the highest similarity value of all and subtracting a fixed value $c$ from it.

$$b = max(sim(e_{i_1 j_1}, e_{i_2 j_2}) | \forall e_{i_1 j_1} \in O_{i_1}, e_{i_2 j_2} \in O_{i_2}) - c$$

**N Percent** This method is closely related to the former one. Here we take the highest similarity value and subtract a fixed percentage $p$ from it.

$$b = max(sim(e_{i_1 j_1}, e_{i_2 j_2}) | \forall e_{i_1 j_1} \in O_{i_1}, e_{i_2 j_2} \in O_{i_2})(1 - p)$$

The latter two approaches are motivated from the idea that similarity is also dependent on the domain. The calculated maximum similarity can be an indicator for this and is fed back into the algorithm.

Our approach focuses on classifying the found mappings into two groups: equal or not equal. As a potential extension in future we foresee a *three layer semi-automatic* approach having: correct mappings, mappings to be confirmed manually, and dismissed mappings.

### 4.3 Additional Actions

Using small additional actions can lead to significantly better results.

**Multiple Rounds** For calculating the similarity of one entity pair many of the described methods rely on the similarity input of other entity pairs. The first round always has to be a general method like the comparison based on labels, which does not rely on any other pairs. By doing the calculation in several rounds one can then access the already calculated pairs and receive a better similarity. Several possibilities when to stop the calculations have been described in the literature: a fixed number of rounds, no changes in the mappings, changes below a certain threshold, or dynamically depending on how much time and calculation power can be provided.

**Best Mappings Only** When having more than one round of calculation the question arises if the results of each round should be converted/adjusted before they are fed back for the next round. One approach is to reuse only the similarity of the best mappings found. A possible way could be to give the best match a weight of 1, the second best of $\frac{1}{2}$, and the third of $\frac{1}{3}$. Potentially correct mappings are kept with a high probability but leave a path for second best mappings to replace them. The danger of having the system being diverted by low similarity values is minimized.

**Deletion of Doubles** The goal of the current approach is to gain a single mapping between two entities from the best similarity values. As there can be only one *best* match, every other match is a potential mistake, which should be dropped. Practically we do cleansing in the mapping table by removing entries with already mapped entities.

### 4.4 Process

All the ideas presented so far describe how two entities can be compared to one another and determine a mapping measure between them. We use the following methodology (see figure 3):

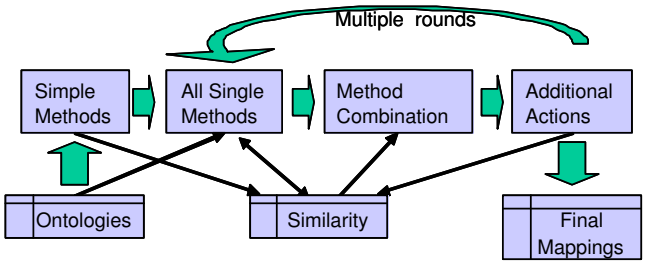


**Fig. 3.** Mapping Process

1. Starting point are two ontologies which have to be mapped. We will therefore calculate the similarities between any valid pair of entities.
2. In a first round basic similarities are set via measures which are independent of other similarities. In our case we rely on the label similarity, equal URIs, or the sameAs relation (**R1**, **R2**, and **R15**). The complete similarity matrix is calculated from this.
3. In a second step the overall similarities between the entities are calculated based on all the introduced similarity measures (**R1** through **R17**), always using the now existing previous similarities of other entities if required.
4. Following the presented additional actions steps two and three are repeated for multiple rounds (either a fixed number of times, or until the number of changes per round drops below a threshold value). In a last step doubles are deleted and similarities which are too little (i.e. below the cut-off value and therefore not worth to mention) are removed and only the best similarities are displayed.

5. These will then be used as the final mapping table. They will be evaluated as explained in the next section.

## 5 Evaluation

The problem of mapping between ontologies already produced some interesting approaches. A thorough evaluation of our new approach is presented here.

### 5.1 Evaluation Scenario

Our evaluation is based on the introductory example given in section 1. We have presented an application scenario for mappings at the beginning of this paper.

We basically take two ontologies and create mappings between the entities based on a given strategy. These mappings are validated against the correct mappings which had been created in beforehand. Our goal was to reach the best number of mappings, which is quantified in the f-measure (see next section). As the absolute quality of mappings is highly dependent of the complexity of the ontologies themselves, we focus on the relative performance of different mapping strategies.

The implementation itself was done in Java using the KAON-framework[5] for ontology access and maintenance. All the tests were run on a standard notebook.

### 5.2 Metrics

To allow for comparability not only between our own test series, but also with existent literature we will focus on using standard information retrieval metrics. The definitions of precision and recall is adapted by us to fit the given evaluation scenario [8].

**Recall** $r = \frac{\#correct\_found\_mappings}{\#possible\_existing\_mappings}$

**Precision** $p = \frac{\#correct\_found\_mappings}{\#all\_found\_mappings}$

**F-Measure** combines the two mentioned measures.
$f = \frac{(b^2+1)pr}{b^2p+r}$ with $b = 1$ being a factor to weight precision and recall.

**11-Point Measure** is the 11-point interpolated average precision at the TREC[6]. We adjusted it to the similarity scenario: eleven points are equally distributed between the best match and the least match. This way we can gain an average precision, recall, or f-measure.

**Measures at Cut-Off** takes into account how well the algorithm can determine which mappings are still valid and which should be dismissed.

### 5.3 Data Sets

Four data sets each consisting of at least two ontologies were used for evaluation purposes. From the differences of them we expect a representative evaluation.

[5] http://kaon.semanticweb.org/

[6] http://trec.nist.gov/

**Russia 1** In this first set we have two ontologies describing Russia. The ontologies were created by students with the task to represent the content of two independent travel websites about Russia. These ontologies have approximately 400 entities, including concepts, relations, and instances. The total number of theoretical mappings is at 280, which have been assigned manually. This scenario is an easy scenario, with which many individual methods can be tested.

**Russia 2** The second set again covers Russia. This time the two ontologies have been additionally altered by deleting entities and changing the structure as well as the labels at random. Each ontology has about 300 entities with 215 possible mappings, which were captured during the generation. Many of these mappings can not even be identified by humans any longer.

**Tourism** Two ontologies which were created separately by different groups of people describe the tourism domain. Both ontologies consist each of about 500 concepts and relations, but no instances though. 300 manual mappings were created.

**SWRC** The SWRC (Semantic Web Research Community) ontology describes the domain of universities and research. Its size is about 300 entities, with very little instances. For this setting three more very small ontologies (about 20 entities each) were created. In total we have 20 possible mappings (manually determined) against the SWRC ontology. This scenario was derived from the SWAP case where small queries (plus additional context) are sent to an existing ontology for validation.

### 5.4 Strategies

For the tests we chose to use five similarity strategies:

**Label (S1)** For this strategy only the labels of entities were regarded (**R1**). This strategy can be regarded as the baseline against which we need to evaluate the other strategies with more advanced measures.

**All (S2)** As a next step all described similarity methods (**R1** through **R15**) are integrated through simple addition.

**Weighted (S3)** All similarity methods are integrated including different weights for each method. The weights were calculated by maximizing the overall f-measure in the four test data sets. Additionally five rounds of similarity calculation are done and doubles are removed.

**Sigmoid (S4)** Again all methods (**R1** to **R15**) are taken, but they are weighted with the sigmoid function. The parameters of the sigmoid functions were assigned manually with respect to the underlying similarity method. In the five rounds only the best results were fed back into the next round. Finally doubles were removed. A constant was used to determine the cut-off.

**Neural Nets (S5)** The results of the methods are fed into a neural network. A fraction (20%) of the evaluation examples was taken for training purposes. The rest was then used for evaluation. A constant value for cut-off was determined from the same training set manually.

### 5.5 Results

For space purposes we will only present an excerpt of the concrete results as to be published in an upcoming technical report[7]. We will focus on the averaged table for the discussion, which already covers the complete results.

In figure 4 we present the results of the first data set with the two strategies S1 Labels and S4 Sigmoid. All mappings are arranged by their similarity value - with the left side of the graph showing the highest value and the right side showing the lowest value. With each new mapping we recalculate the other measures. The graphs show the respective precision, recall, and f-measure values. The marked points show the cut-off border. This is the point we have to measure if we are looking for exactly one value. Alternatively we also measured the 11-point average to gain a view of the whole curve.

We will now compare the two strategies with respect to the defined evaluation measures. The highest mappings are all correct, what one can see from the precision value of 1 for both strategies. But what one can also see is that S4 Sigmoid keeps the precision value high for many more mappings than S1 Label (97% vs. 80% at cut-off). Recall only reaches a medium level for S1 Label; the final level is much higher for S4 Sigmoid: 0.75 vs. 0.5. A consequence of these two measures is that the f-measure is also higher for the advanced approach in comparison to the naive approach.

A word about the other strategies: all lie in between the two presented approaches. However, determining the cut-off point was much more difficult in those strategies. They often missed the highest f-measure value considerably. A general comparison graph is plotted in figure 4. This comparison graph shows the average results over all four data sets, each with the different strategies. Precision, recall, and f-measure reach their highest values with S4 Sigmoid. In general one can say that there is an increase with the rise of strategy complexity. For S1 to S3 we plotted the two results from different cut-offs. We will now discuss these results in more detail.

### 5.6 Discussion

Our original hypothesis H is widely fulfilled:

**Semantics** can help to determine better mappings (S1 vs. S4).

**Precision** is considerably higher for the more advanced combination methods. Especially interesting is the fact that precision generally is higher for these methods, no matter where the cut-off is placed in the mapping table. This is important when thinking of full-automatic solutions, where we want to keep the wrong mappings as low as possible.

**Recall** also rises along the richness of methods.

**F-measure** as our core evaluation measure reaches the highest value for the S4 Sigmoid strategy for every data set.

**Average Increase of 20%** in precision, recall and f-measure.

**Naive Combinations** of mapping methods often do not make the results better, but worse (S2 All, and even S3 Weighted) . The effort for advanced carefully determined methods is therefore very important.

[7] We refer to http://www.aifb.uni-karlsruhe.de/WBS/meh/publications/ESWS for the complete graphs as well as the used ontologies.

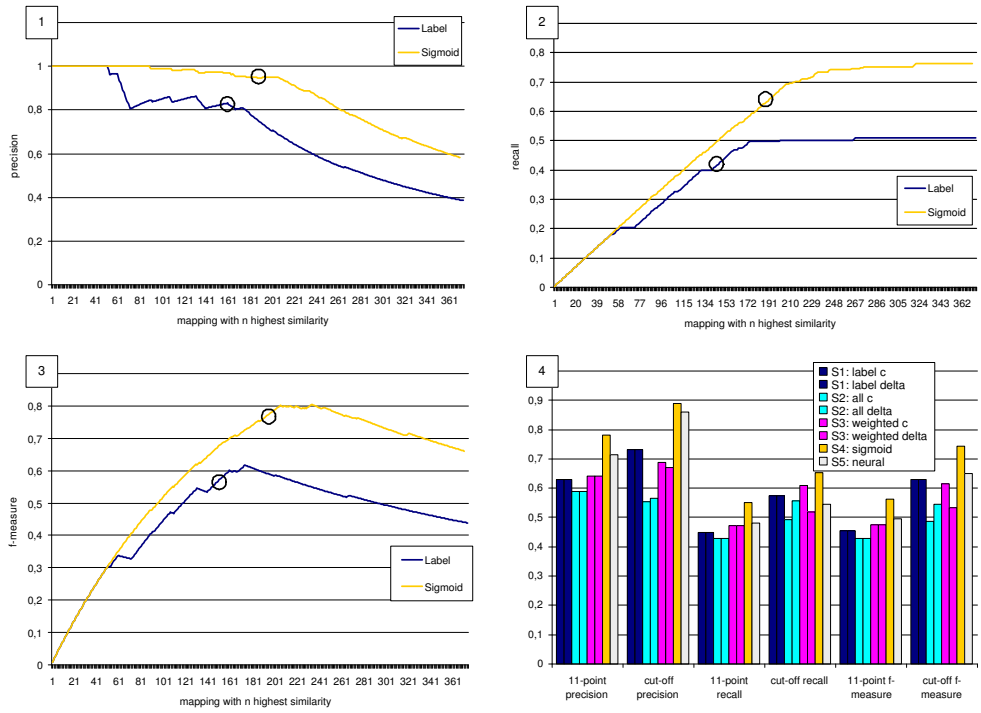


**Fig. 4.** Results of strategies on dataset Russia1 (1. precision, 2. recall, 3. f-measure); Average results of all strategies over all test scenarios (4.)

**Machine Learning** might help (S5). The problems we encountered are general problems of machine learning such as over-fitting. We also faced the problem that the additional actions could not be completely integrated into the machine learning approach, which lead to lower results.

# 6 Outlook

## 6.1 Related Work

Most of the ideas for measuring similarity are derived from common sense and can be easily understood. To our knowledge existing approaches focus on specific methods to determine similarity rather than using an overall integrating approach.

Some authors have tried to find a general description of similarity with several of them being based on knowledge networks. [20] give a general overview of similarity. As the basic ontology mapping problem has been around for some years first tools have already been developed to address this. The tools PROMPT and AnchorPROMPT [17] use labels and to a certain extent the structure of ontologies. Their focus lies on ontology merging i.e. how to create one ontology out of two. [10] already used a general

approach of relaxation labelling in their tool GLUE. Most of their work is based on the similarity of instances only. [15] created a tool for mapping called Chimaera. Potential matches are presented to the user in all mentioned tools for confirmation. In their tool ONION [16] the authors take up the idea of using rules and inferencing for mapping, but the inferencing is based on manually assigned mappings or simple heuristics (as e.g. label comparisons). Besides equality first steps are taken in the direction of complex matches. These could also include concatenation of two fields such as "first name" and "last name" to "name"[9]. [4] further present an approach for semantic mappings based on SAT. Despite the large number of related work, there are very little approaches on how to combine the many methods as we do. The other mentioned tools do not raise the issue, they presumably use only naive summarization approaches.

[19] express their insights from a database view. Many ideas from the database community, especially concerning efficiency [14], should also be regarded. Another community involved in similarity and mapping are object-oriented representations in which little work seems to has been done, [21] for UML being an exception. Agent communication greatly benefits from mapping as shown in [24].

### 6.2 Problems and Future Steps

Even though the shown approach retrieves good results, the results do not reach 100% correctness. Unfortunately, if full-automatic mapping is done, and inferencing builds on top of it, wrong results can bring down the value of the whole mapping process. Implications of fuzzy inferencing[7] will have to be understood well when using it. Semi-automatic processing is a common approach to circumvent this problem. Another problem is a general problem when doing comparisons. Especially with big ontologies complexity of similarity calculations can grow dramatically. In our approach one can expect a complexity of $O(log^2(n) \times n^2)$. It is derived from: $O(log(n))$ for entity access, $O(log(n))$ for the method complexity, and $O(n^2)$ for the full comparison of all possible pairs. Approaches to reduce complexity from other domains (e.g. databases) might be a good start. As data in ontologies expresses certain semantics the calculations might be channelled using these semantics e.g. starting with comparisons of top-level elements in the hierarchy. Both problem areas have potential for future work.

### 6.3 Conclusion

The mapping problem arises in many scenarios. We have shown a methodology for identifying mappings between two ontologies based on the intelligent combination of manually encoded rules. Evaluation proved our initial hypothesis, i.e. the combination of our presented similarity measures leaded to considerably better results than the usage of one at a time. One can summarize that precision, recall, and f-measure increase by 20% compared to label-based approaches. Semantics helps bridging the mapping gap.

*Acknowledgements.* Research reported in this paper has been partially financed by EU in the IST projects SWAP (IST-2001-34103) and SEKT (IST-2003-506826). Many thanks to our colleagues for the fruitful discussions.

## References


1. F. Baader, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider. *The Description Logic Handbook*. 2003.
2. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001.
3. G. Bisson. Why and how to define a similarity measure for object based representation systems. *Towards Very Large Knowledge Bases*, pages 236–246, 1995.
4. P. Bouquet, B. Magnini, L. Serafini, and S. Zanobini. A SAT-based algorithm for context matching. Technical report, University of Trento, Trento, Italy, 2003.
5. D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. What to ask to a peer: Ontology-based query reformulation. In *Proc. of the 9th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2004)*, 2004.
6. S. V. Castano, M. G. D. Antonellis, B. Fugini, and C. Pernici. Schema analysis: Techniques and applications. *ACM Trans. Systems*, 23(3):286–333, 1998.
7. Z. Ding and Y. Peng. A probabilistic extension to ontology language owl. In *Proceedings of the 37th Hawaii International Conference On System Sciences (HICSS-37)*, Big Island, Hawaii, January 2004.
8. H. Do, S. Melnik, and E. Rahm. Comparison of schema matching evaluations. In *Proceedings of the 2nd Int. Workshop on Web Databases (German Informatics Society)*, 2002.
9. H. Do and E. Rahm. Coma - a system for flexible combination of schema matching approaches. In *Proc. of the 28th VLDB Conference*, Hong Kong, China, 2002.
10. A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *Proc. to the WWW-11*, Honolulu, USA, 2002.
11. J. Heaton. *Programming Neural Networks in Java*. 2002.
12. I. V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory*, 1966.
13. A. Maedche, B. Motik, N. Silva, and R. Volz. Mafra - a mapping framework for distributed ontologies. In *Proc. of the EKAW 2002*, 2002.
14. A. McCallum, K. Nigam, and L. H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Knowledge Discovery & Data Mining*, 2000.
15. D. L. McGuinness. Conceptual modeling for distributed ontology environments. In *International Conference on Conceptual Structures*, pages 100–112, 2000.
16. P. Mitra, G. Wiederhold, and M. Kersten. A graph-oriented model for articulation of ontology interdependencies. *Lecture Notes in Computer Science*, 1777:86+, 2000.
17. N. F. Noy and M. A. Musen. Anchor-PROMPT: Using Non-Local Context for Semantic Matching. In *WS Ontologies & Information Sharing at IJCAI-2001*, Seattle, USA, 2001.
18. M. R. Quillan. Word concepts: A theory and simulation of some basic capabilities. *Behavioral Science*, 12:410–430, 1967.
19. J. Roddick, K. Hornsby, and D. de Vries. A unifying semantic distance model for determining the similarity of attribute values. In *Proc. of ACSC2003*, Adelaide, Australia, 2003.
20. M. Rodríguez and M. Egenhofer. Determining semantic similarity among entity classes from different ontologies. *IEEE Trans. on Knowledge and Data Eng.*, 15(2):442–456, 2003.
21. R. Rufai. Similarity metric for UML models. Master's thesis, King Fahd University of Petroleum and Minerals, 2003.
22. G. Stumme et al. The Karlsruhe View on Ontologies. Technical report, University of Karlsruhe, Institute AIFB, 2003.
23. X. Su. A text categorization perspective for ontology mapping. Technical report, Norwegian University of Science and Technology, Norway, 2002.
24. P. C. Weinstein and W. P. Birmingham. Agent communication with differentiated ontologies. Technical Report CSE-TR-383-99, 7, 1999.