Generating and Integrating Evidence for Ontology Mappings

Ludger van Elst and Malte Kiesel

German Research Center for Artificial Intelligence
– Knowledge Management Department –
{elst,kiesel}@dfki.uni-kl.de

Abstract. For more than a decade, ontologies have been proposed as a means to enable sharing and reuse of knowledge. While originally relatively narrow information landscapes have been in mind (e.g., knowledge sharing between a few expert systems) the application areas proposed nowadays (e.g., organizational knowledge management or the Semantic Web) are rather broad and open.

From abstract considerations about the distributed nature of knowledge as well as from observation of actual (human) ontology negotiation processes it seems clear that *globally* agreed-upon conceptualizations are probably not obtainable. Therefore, ontology matching and mapping procedures play an essential role in more open information landscapes.

In this paper, we present a framework that collects and integrates heuristic *evidence* for ontology mappings, allows a knowledge engineer to browse a space of (assessed) mapping candidates in order to select adequate candidates and then leverage them to a level of formal statements for ontology merging. A simple example session shows the intended handling of the prototype and demonstrates strengths and weaknesses of particular sources of matching evidence.

1 Motivation

Within the *sharing and reuse effort*, ontologies have been widely proposed as a means to alleviate model mismatches at the knowledge level [17]. The scenarios envisioned more than a decade ago were relatively narrow: Knowledge sharing between a few expert systems on the one hand and reuse of knowledge models by a couple of system engineers on the other. Nowadays, ontologies are proposed for much broader and more open information landscapes, e.g., as backbone technology for organizational knowledge management (KM) systems [1] or even as silver bullet in e-commerce applications and for the semantic web [10]. While already much research has been carried out in the areas of ontology representation, acquisition and inferencing, the broadening of the scope of ontology technology leads to (at least) two additional challenges:

1. *Involvement of end users* in ontology–related processes: While with the "expert system scenario" mainly knowledge engineers were the contributors and costumers of ontologies, in the broader application areas often also end users have to understand them (at least at a certain level of abstraction, e.g., in the case of web portals) or even are a valuable source for their maintenance (e.g., KM in product development).

2. *Scaling–up* the idea of ontologies: From the viewpoint of the topology of communication, the "shared conceptualization approach" tries to reduce the exponentially growing number of one-to-one mappings between models by introducing the ontology as a "hub" or mediator, resulting in a star–shaped topology. However, in more open worlds a centralized "knowledge topology" is hardly reachable. While having OWL as a W3C recommendation[1] is a significant achievement with respect to a *representation* ontology for the semantic web, for conceptual as well as for pragmatic reasons it is unlikely that we will see a similarly high agreement on specific domain ontologies[2]. Consequently, the question of mediating between several domain ontologies will become more and more important.

Although we will concentrate on the latter topic we believe that for a viable approach to ontologies in more open information landscapes, in the long term both issues have to be tackled in an integrated manner[3].

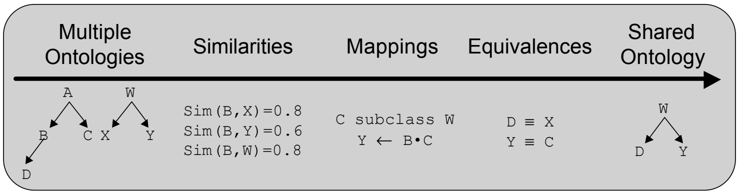


Fig. 1. Examplary Operation Points in the "Ontology Matching Continuum"

Ontology matchings can be seen as statements about relationships between elements (e.g., concepts) of two or more ontologies. Figure 1 exemplifies that these statements can occur on various levels of formalization: *Similarities* are relatively informal as the semantics of this relation is typically ill–defined or application specific (see, e.g., the discussion about similarity and utility in case-based reasoning [3]). *Mappings* can take various forms, from more heuristics-like to formally grounded ones (e.g., "class C is a subclass of class W", or "concept Y is a combination of the concepts B and C"). *Equivalences* state that (parts of) the ontologies indeed intend to express the same conceptualizations. The ultimate result of ontology matching may be a *shared ontology* that comprises the conceptualizations captured by the input ontologies.

Ontology matching procedures perform transitions along this continuum, from heterogeneous ontologies to shared conceptualizations. More logics-oriented approaches, for example, try to formally infer mappings from the input ontologies. It is well known that such a semantic unification is a complex, highly knowledge–intensive task that is in general not solvable (see [23] for recent results in this area). While these approaches have the attractiveness of being logically sound, they are quite heavy–weight and—even more important—rely on some common vocabulary in the definitions of the input ontologies. This pre–requisite might be satisfied relatively easily on closed–world scenarios

[1] see http://www.w3.org/News/2004#item14

[2] For an extended discussion on the distributed nature of knowledge, see, e.g., [8].

[3] With the notion of *ontological societies* we have made a first step into that direction [6, 21].

(e.g., schema matching in federated databases), but it is a serious problem in the case of more open information landscapes (like the semantic web and organizational knowledge management) with their continuously evolving domain ontologies.

The approach presented in this paper therefore abandons the path of formally sound reasoning for matching ontologies and instead establishes a framework that collects and integrates heuristic *evidence* for ontology mappings. As the sources of evidence are various forms of similarities in the input ontologies, the framework comprises the whole range depicted in Figure 1. The prototypical implementation computes these evidences and allows a knowledge engineer to browse a space of (assessed) mapping candidates in order to select adequate candidates and then leverage them to a level of formal statements for ontology merging.

The remainder of this paper is organized as follows: Section 2 gives an overview of the framework and its algorithmic basics. In section 3, we describe a prototypical implementation of the framework on top of the Protégé ontology environment. A simple example session shows the intended handling of the tool and demonstrates strengths and weaknesses of particular sources of matching evidence. Finally, we summarize the basic features of the approach and give a brief outlook on future work.

2 An Evidence-Based Framework for Ontology Mapping

The main task in merging ontologies is to identify relationships between elements of the input ontologies, most basically between the ontologies' classes. These relationships are necessary to determine which actions to perform in order to create a merged ontology (see Figure 2). As already stated above, in this paper we will not *formally infer* class relationships, but *gather evidence* for such relationships. In the literature, ontology merging operations are mainly based on two sources of evidence:

- *Term–based evidence* considers similarities in the the textual description (i.e., the "name") of concepts in the source ontologies. Examples are the Chimaera ontology environment [15, 14] and Protégé's PROMPT tab [18].
- *Topology–based evidence* considers the structure of the source ontologies, e.g., by determining similarities of the graphs representing concepts and their relationships, as done by the Similarity Flooding algorithm [16].

The charm of term–based evidence is that a variety of well–understood algorithms for the determination of string similarities makes this approach easy to implement. However, all precision and recall problems known from string–based information retrieval (e.g., due to synonyms or homonyms) directly apply here. Therefore, thesauri or lexica are sometimes incorporated as additional background knowledge to alleviate these problems.

Most methods that focus on topology–based evidence are, strictly speaking, hybrid: Approaches using formal logic apply matching and unification procedures that rely on some common vocabulary (i.e., they rely basically on term identity); the similarity flooding algorithm presented in [16] is hybrid as it derives its initial activation values from term similarities.

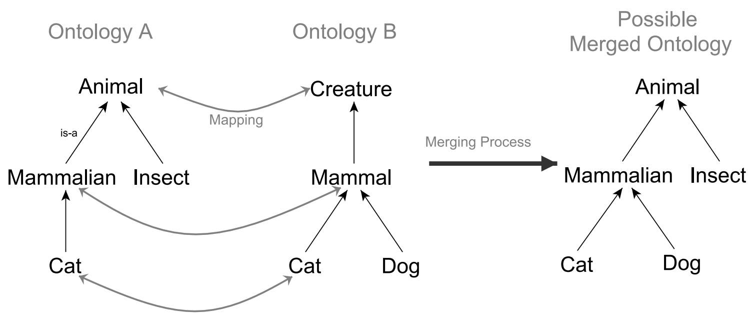


Fig. 2. Merging Ontologies

A third source of evidence for detecting ontology mappings are similarities in the *instances*. Two basic forms of underlying heuristics for instance–based mapping are

– "IF the instances of class A in ontology O_1 are very similar to the instances of class B in O_2 THEN suggest $A \approx B$", and
– "IF the instances of class A in ontology O_1 are classified as class B instances with respect to O_2 THEN suggest $A \subseteq B$".

The keys to these heuristics are obviously the definition of *instance similarity* and the *classifier*, respectively, without presupposing a shared terminology. How can we determine the similarity of instance a, formulated in terms of ontology O_1, and instance b, formulated in terms of ontology O_2? How can we classify instance a, formulated with the O_1 vocabulary, with respect to O_2? In general, we here face the same mapping and matching problems on the instance level that we actually wanted to solve on the ontology level. A first idea would be to step another level down and determine local similarities on the basic datatypes (e.g., number or string similarity) of properties. However, it is very unlikely that these basic property similarities really contain enough knowledge to reflect semantical similarity of the instances. Is an object o_1 that has a slot foo_1 with a value of 50 really similar to an object o_2 with a foo_2 slot value of 51? Without any additional knowledge this similarity would look rather random.

Fortunately, we are in the position that in many application areas envisioned in the semantic web and in organizational knowledge management, we actually can rely on better similarity and classifier functions. In these scenarios, domain ontologies are often not primarily used to manage "real" instances with the proper is-a semantics ("Allan is-a researcher"), but to *annotate* (text) documents or parts of them for better retrieval ("This document is about ontology mapping, semantic web, and knowledge management"). On these text documents we can, from experiences in information retrieval (e.g., [2]), rightly expect to obtain the required classification and similarity functions that also capture some semantics of the "instances". So, the above sketched heuristics for instance–based mappings can be re-formulated as

– "IF the documents annotated with concept A (of ontology O_1) are very similar to the documents annotated with concept B (of O_2) THEN suggest $A \approx B$".
  This similarity can for example be defined as vector similarity (typically the cosine measure) in a vector space model as is often used in document retrieval [20, 2].

– "IF the documents annotated with concept A (of ontology O_1) are classified as class B documents in O_2 THEN suggest $A \subseteq B$".

  Such text classifiers can be automatically learned from the example documents that are annoted with B and than easily be applied to previously unknown documents, e.g., those documents that are annotated with A.

Examples for instance–based ontology mapping in document–centered applications can be found in [5, 13, 19]. For an overview of some merging tools and algorithms along with the kind of information they exploit, see table 1.

Table 1. Comparison of Ontology Merging Approaches

Reference	Term	Topology	Instance
COMA [4]	x	x[4]	
Chimaera [14]	x		
CAIMAN [13]		x[5]	x
Similarity Flooding [16]	x[6]	x	
PROMPT [18]	x		

The three basic sources of evidence, *term–based*, *topology–based*, and *instance–based* are of course not completely independent; certainty coming from one of the sources may reinforce or eliminate other evidence, and approval of instance– or term–based mapping evidence, for example, has direct impact on the topology. Which of the sources is to most adequate seems highly dependant on the input ontologies (and instances). This fact and the awareness that, by now, we just gather evidence, but not construct mappings, leads to the following two additional elements of the framework:

– *Evidence Integration*: The goal of this step is to generate a comprehensive view on the various interacting and potentially conflicting evidences. Possible integration techniques range from relatively light–weight voting procedures, like they are often used in classification tasks (e.g., Borda Count or Highest–Rank) [11], to more heavy–weight approaches like logic–based formalisms for evidences. An example for the latter is the Dempster–Shafer theory of evidence that also explicitly handles the absence of evidence (for an overview see [12]). A middle course would be to use global similarity measures known from case–based reasoning. [9] presented this approach for alignment of OWL-lite ontologies. COMA [4] is a system that also allows for combining different matchers for database schemas.
– *User Interaction*: It is unlikely that there is only one "true" solution to the problem of mapping ontologies. Every ontology is founded on design decisions. Therefore, incorporating a knowledge engineer's feedback is reasonable to catch an adequate representation bias (or: the right ontological commitment) in the merged ontology. Conceiving ontology merging as transitions in a continuum as described in section 1 suggests to have this interaction right at the interfaces of these transitions. This means the user decides if and when an integrated evidence for a relation between elements of the input ontologies is really leveraged to a mapping (which then may influence other evidences again) and how the mappings are used for an actual merger.

Figure 3 gives an overview of the framework for evidence–based ontology merging described in this section. One of its main features is that the conceptually quite different sources of evidence are also separated in the framework. The only point where combination of the approaches takes place is when combining the results of evidence generation algorithms. Thereby, one can on the one hand estimate the usefulness of the separate algorithms without any interference caused by the early combination of the approaches, and, on the other hand, more easily tune the overall system to the characteristics of particular input ontologies. Moreover, our approach emphasizes the need for user feedback, incorporating and enhancing its match proposals with every match the user confirms or rejects. If, for example, the user confirms that two classes match, the algorithm considers it more likely that subclasses of these two classes match, too.

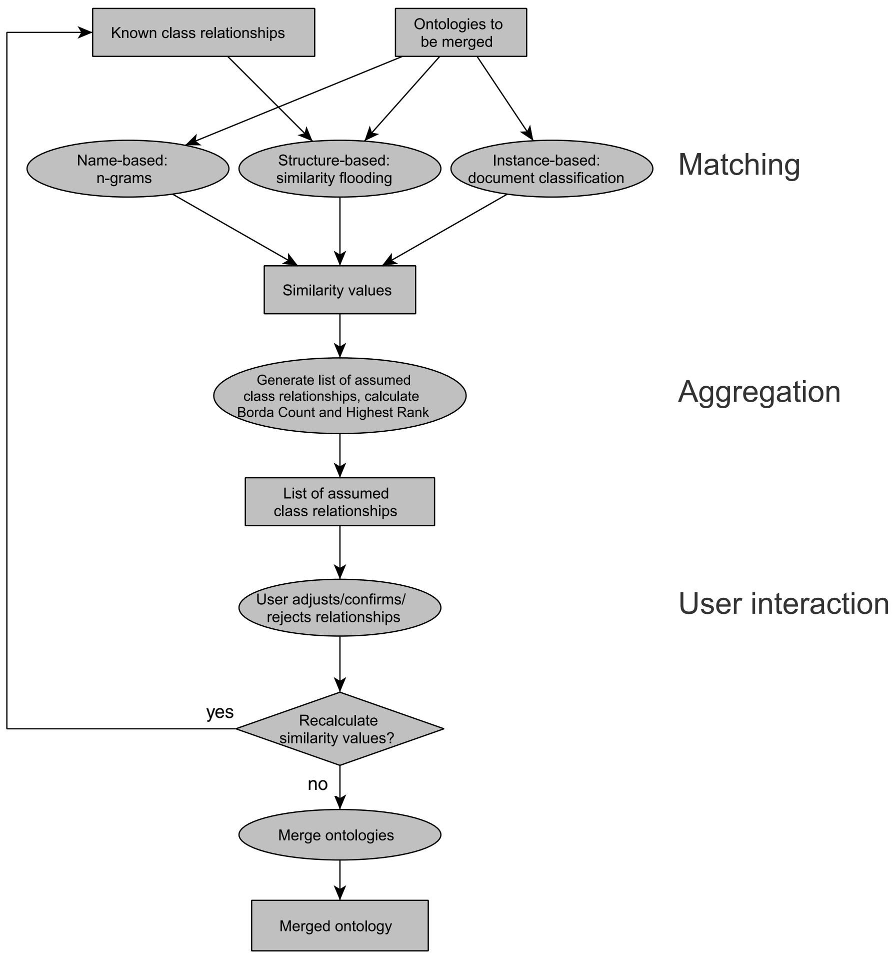

Known class relationships
Ontologies to be merged
Name-based: n-grams
Structure-based: similarity flooding
Instance-based: document classification
Matching
Similarity values
Generate list of assumed class relationships, calculate Borda Count and Highest Rank
Aggregation
List of assumed class relationships
User adjusts/confirms/ rejects relationships
User interaction
yes
Recalculate similarity values?
no
Merge ontologies
Merged ontology

Fig. 3. Framework for Evidence-based Ontology Merging

In the next section, we describe a prototype implementation of the proposed framework. This implementation is realized as a plug-in for the Protégé ontology engineering environment.

3 PHASE: A Prototypical Implementation of the Merging Framework

The prototype presented in this section implements the general framework described in section 3. It is realized on three building blocks:

1. The *Protégé*[7] *ontology environment* is used for the representation of the source and result ontologies as well as for representing the mappings.
2. The *PROMPT tab [18]* is a tool for ontology merging and alignment with Protégé. As it already supports the main methodology and interaction cycle of framework, we used it as a backbone for our implementation.
3. The classification and document similarity procedures for the instance–based matching were realized on top of the *NextBot engine* that is the core of ProFiler, a commercial tool for information organization and retrieval by brainbot technolgies AG[8].

3.1 Basic Algorithms

In the following, we briefly sketch how the elements of the framework were instantiated. As we were mainly interested in the integration of the various sources of evidence we did not concentrate much on the optimization of the basic evidence generators.

Term-Based Matching. From the many well–known algorithms for determining string similarities we chose the n–gram approach to replace to original substring matcher in PROMPT. n-grams compute the similarity of two strings by comparing all substrings of length n. The degree of fitness of a particular match candidate is determined by the number of n-grams it matches. A threshold for the minimal n-gram similarity leads to *match pairs*, tuples of matching classes and a respective confidence value.

Structure-Based Matching. A structure-based algorithm that is already used in the area of databases for schema mapping is Similarity Flooding [16]. Similarity Flooding is a generic approach for determining matching nodes in graphs, taking two graphs as input and returning a set of match pairs with corresponding similarity values. The main part of Similarity Flooding is execution of a fixpoint calculation. In the original approach, Similarity Flooding uses a set of similarity values obtained by string comparison of the classes' names as an initial value of the similarity vector. However, as we do not want to mix name-based and structure-based approaches, we use a canonical vector for initialization. Also, the initialization vector is used to inject knowledge of user-confirmed matches. Match pairs which have been confirmed by the user get a higher

[7] http://protege.stanford.edu
[8] http://brainbot.com

initial similarity value than pairs which have not been confirmed. Accordingly, potential match pairs which the user rejected get an initial similarity value of zero.

Instance-Based Matching. Instance-based matching exploits similarities of the *instances* associated with the ontologies' classes. For example, instances of the class "car" can be named "VW Beetle", "Porsche 911", and so on, and it is likely that in another ontology that features the class "automobile" similar instances can be found. In our prototype, we do not compare "real" instances of the ontologies, but use text or HTML documents that are annotated with the concepts of the ontologies. Thereby, the documents define a kind of "semantic context" of the concepts. The commercial NextBot engine that was used for document classification and document similarities is based on the *vector space model* [20] and standard information retrieval methods [2] (mainly the TF–IDF weighting schema and the cosine similarity measure). This engine can automatically learn text classifiers with respect to a concept taxonomy. Input for the learning algorithm are *examples* that explicitly relate documents to one ore more concepts. Additionally, we use a simple algorithm for determining class similarities shown in figure 4. Again, *examples* of a class are documents that are a priori used for learning the classifier for a class in the vector space model. *Soft matches* are a document's (previously unknown) matches to other classes. These are automatically determined by applying the learned document classifier. So, the algorithm takes as input the examples, the (automatically learned) soft matches and their resemblance values (i.e., the strength of a soft match) and delivers pairwise similarities of classes that belong to different ontologies[9]. Basically, the algorithm aggregates all evidence that the example documents e_i for a class c of ontology O_1 also belong to class d of ontology O_2 and delivers this aggregate as $similarity(c, d)$.

```
for every class $c$ in ontology $O_n$
    $i = 0, s = \{\}$
    for every example $e$ of class $c$
        $i = i + 1$
        for every soft match $m$ of example $e$
          if $m.class$ is not in $O_n$ /* no mappings within one ontology */
            $inc(similarity(c, m.class), m.resemblanceValue)$
            $s = s \cup m.class$
    for every class $d$ in $s$
        $similarity(c, d) = similarity(c, d)/i$
```

Fig. 4. Algorithm for Instance–based Evidence Generation

Detecting Different Kinds of Relationships. In many ontology merging approaches, detection and handling of relationships is limited to one-to-one matchings with "is-equal" semantics. This means that merging algorithms only search for direct semantic correspondence of classes of the ontologies to merge. However, due to the semantics of the subclass relation that is typically defined as set inclusion, subclasses and superclasses can be discovered by the instance-based matcher. If many instances of class a

[9] For convenience reasons, *one* NextBot engine is used to store all input ontologies. This accounts for the test into which ontology a soft match points (see Figure 4).

in ontology O_1 match class b in ontology O_2, but only few instances of class b match class a, it is likely class b represents a superclass of class a. Also, name-based matchers can be used for discovering evidence for subclass/superclass relationships as the names of subclasses often include the name of their respective superclass. However, as the classes reside in separate ontologies that probably use varying diction, discovery of subclass/superclass relationships using name-based approaches needs sophisticated synonym handling. Therefore, name-based discovery of subclass and superclass relationships is not used in our approach.

Handling Confirmed Matches. As knowledge of confirmed or rejected matches shall be incorporated in subsequent matching steps, we need to find a way to feed this knowledge into the matching process. The structure–based matching algorithm, Similarity Flooding, offers a way to achieve this by adjusting the algorithm's initial similarity vector according to the confirmed or rejected matches. However, while handling of confirmed "is-equal" matches is straightforward this way, handling of confirmed subclass or superclass matches is more difficult. In order to represent these matchings, we create *virtual classes* in the ontologies (see figure 5). Now, we can set the initial similarity value of the virtual class and the respective subclass to a higher value than normal, causing the Similarity Flooding matching algorithm to use the subclass relationship for improving its matching results by a certain degree.

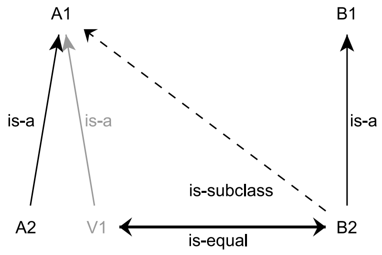


Fig. 5. Handling confirmed subclasses

Superclass relationships are handled accordingly. Note, that confirmed "is-equal" matches take precedence over subclass and superclass relationships of the same classes when building the initial similarity vector of Similarity Flooding.

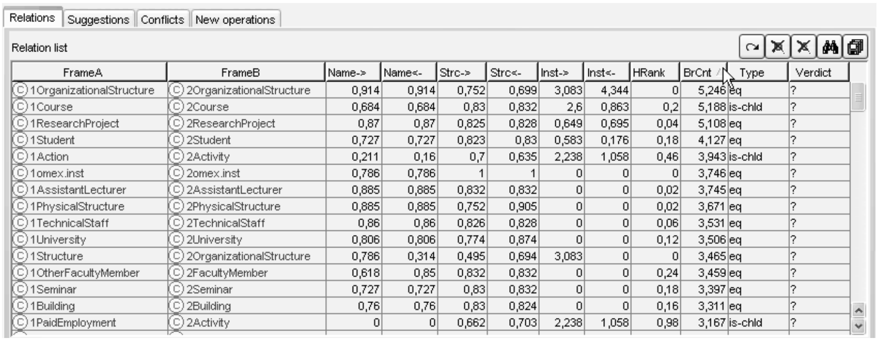
Aggregation of Match Results. Typically, the output of the matching algorithms is ambiguous. Term–based matchers for example may conclude that class "University" from ontology A matches class "University" from ontology B while a structure-based matcher does not assign a high similarity value to this pair, perhaps because when taking the ontologies' structures into consideration it becomes clear that class "University" of ontology A means a set of buildings while in the context of ontology B, the class "University" denotes an organizational structure.

We apply a modified Borda Count method for combining matcher results using *rankings*. In the following, we will briefly explain the approach (for an overview see

[11]): A *ranking* is a list of match pairs, ordered by similarity value. Aggregation based on rankings is not affected by problems with different scales of the similarity values. Rankings are common scales for the matchers, regardless of the algorithms the matchers use or the scale of the similarity values the matchers return. The Borda Count method uses the rankings of match pairs computed by the matchers as input and outputs an real number (score) associated with every match pair as output. The real number can be used to construct a combined ranking of match pairs. Basically, the Borda Count for a match pair is the sum of the number of pairs ranked below it by each matcher. In order to account for ties in the input rankings, we group all pairs with the same similarity values on one rank when constructing the input rankings. In order to compensate for the fact that now each matcher's ranking probably has a different number of ranks, we assign a ranking score to each rank. The ranking score is $1 - (rank - 1)/(ranks - 1)$. For each pair, we sum up its ranking scores as each pair gets a separate score for each matcher. The pairs, ordered by summed-up ranking scores, represent the resulting aggregated ranking.

3.2 The PHASE Tab

The PROMPT tab [18] is an extension to Protégé and allows to manage multiple ontologies. Specifically, it provides means to compare and merge ontologies, extracts parts of an ontology and to move frames between including projects. PROMPT's merging part tries to identify semantically corresponding classes solely by using a simple term–based matcher. The PHASE tab depicted in figure 6 builds on PROMPT by replacing the original term—based matching algorithm by the evidence generation algorithms described above.



FrameA	FrameB	Name->	Name<-	Strc->	Strc<-	Inst->	Inst<-	HRank	BrCnt	Type	Verdict
1OrganizationalStructure	2OrganizationalStructure	0,914	0,914	0,752	0,699	3,083	4,344	0	5,246	eq	?
1Course	2Course	0,684	0,684	0,83	0,832	2,6	0,863	0,2	5,188	is-chld	?
1ResearchProject	2ResearchProject	0,87	0,87	0,825	0,828	0,649	0,695	0,04	5,108	eq	?
1Student	2Student	0,727	0,727	0,823	0,83	0,583	0,176	0,18	4,127	eq	?
1Action	2Activity	0,211	0,16	0,7	0,635	2,238	1,058	0,46	3,943	is-chld	?
1omex.inst	2omex.inst	0,786	0,786	1	1	0	0	0	3,746	eq	?
1AssistantLecturer	2AssistantLecturer	0,885	0,885	0,832	0,832	0	0	0,02	3,745	eq	?
1PhysicalStructure	2PhysicalStructure	0,885	0,885	0,752	0,905	0	0	0,02	3,671	eq	?
1TechnicalStaff	2TechnicalStaff	0,86	0,86	0,826	0,828	0	0	0,06	3,531	eq	?
1University	2University	0,806	0,806	0,774	0,874	0	0	0,12	3,506	eq	?
1Structure	2OrganizationalStructure	0,786	0,314	0,495	0,694	3,083	0	0	3,465	eq	?
1OtherFacultyMember	2FacultyMember	0,618	0,85	0,832	0,832	0	0	0,24	3,459	eq	?
1Seminar	2Seminar	0,727	0,727	0,83	0,832	0	0	0,18	3,397	eq	?
1Building	2Building	0,76	0,76	0,83	0,824	0	0	0,16	3,311	eq	?
1PaidEmployment	2Activity	0	0	0,662	0,703	2,238	1,058	0,98	3,167	is-chld	?

Fig. 6. The PHASE Tab

As we want to differentiate between matching evidence the matching algorithms gather and the actions that may result from them, a new subtab is present in the PHASE tab that does not exist in PROMPT (see figure 6). The tab is named "relations". Its content is a list of (probably) matching classes. Along with the two classes' names, the similarity values for the three sources of evidence are shown. Also, the (modified) Borda Count value is shown. There is a column that shows the assumed relationship type (equal/subclass/superclass) between the classes which the user may change in case the estimate is wrong. Finally, there is a “verdict” column in which the user can confirm or reject the respective class pair.

3.3 An Example Session

Having described the software environment, we will go through an example merging session, outlining the typical steps necessary for merging ontologies and following the general framework depicted in Figure 3. We try to merge two ontologies that model data from the domain of science and teaching. These ontologies have been constructed in order to meet certain requirements that enable us to show strengths and weaknesses of the algorithm. Part of the ontologies we try to merge are depicted in figure 7. The ontologies each consist of about 30 classes.

Note that while both ontologies are roughly similar, there are also small differences that render most approaches that are solely based on one aspect (be it the classes names or the ontology structure) ineffective. As a start, many class names do not match directly. Also, the ontologies’ structure is not identical. Some classes cannot be matched at all. Instances are not shown in the figure. The instances are HTML documents taken from the “ 4 Universities Data Set”[10].

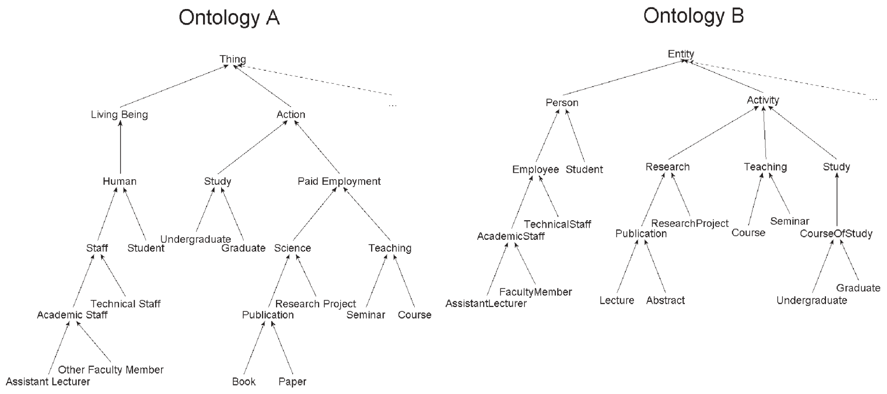


Fig. 7. Example Ontologies

In figure 6, the initial list of proposed matches can be seen, sorted using the modified Borda Count. The user should now confirm or reject matches. When assigning verdicts, it is important to verify that the proposed relationship types are correct or, if this is not the case, to select the proper relationship type. This is especially true if we confirm a match.

After having assigned verdicts to a number of matches, the user can decide to let the structural similarity values get recalculated using the new information.

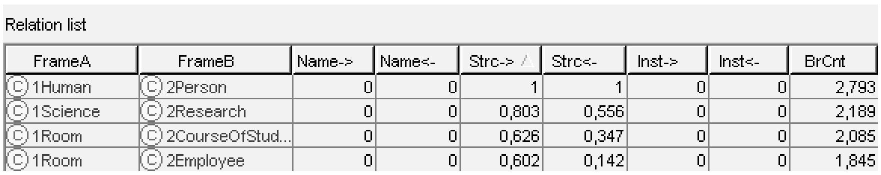
[10] http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/theo-20/www/data/

Generally, discovering matching classes is done using the following steps:

– Browse/sort the relation list in order to find matches. At the beginning, using the Borda Count for sorting is most promising. Later, sorting by single matcher's values should be considered, too.
– Confirm or reject a number of match pairs. Adjust relationship types if necessary.
– Let the PHASE tab recalculate the structure-based similarity.
– Repeat until no more matches remain.

Then, the list of confirmed matches can be used for building a list of merging actions to perform. From then on, everything works as known from PROMPT.

As seen in figure 8, *structure–based matching* smoothly works where the other matchers cease to work. Neither the name-based matcher nor the instance–based matcher (due to lack of instances for the respective classes) can match "Human" and "Person" or "Science" and "Research". However, as can be seen in figure 7, the ontologies' topology in the vicinity of these classes is very similar. The structure-based matcher discovers this. Note that at the time the structure-based matcher computed the values shown in the figure, the user already confirmed some other matches, contributing to the structure–based matcher's accuracy. Also note that there is no matching class for "Room" in ontology B.


Relation list

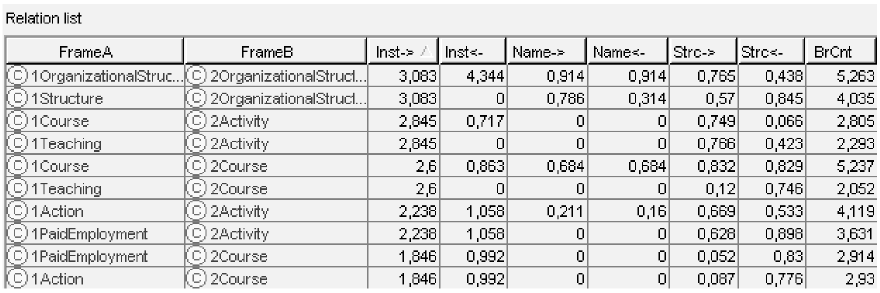
FrameA	FrameB	Name->	Name<-	Strc->	Strc<-	Inst->	Inst<-	BrCnt
1Human	2Person	0	0	1	1	0	0	2,793
1Science	2Research	0	0	0,803	0,556	0	0	2,189
1Room	2CourseOfStud...	0	0	0,626	0,347	0	0	2,085
1Room	2Employee	0	0	0,602	0,142	0	0	1,845

Fig. 8. Structure-based Matcher

Not surprisingly, every match pair proposed by *instance–based matching*(see figure 9) represents a semantic match or at least denotes some semantic relationship. This probably looks even better when using corpora including more documents, as the zero values shown in the right instance–based similarity column are due to missing example documents for the respective classes.

4 Summary and Outlook

In this paper, we presented a framework that collects and integrates heuristic *evidence* for ontology mappings, allows a knowledge engineer to browse a space of (assessed) mapping candidates in order to select adequate candidates and then leverage them to a level of formal statements for ontology merging. The framework employs three basic sources of evidence for ontology mappings, namely *term–based*, *topology–based*, and *instance–based* evidence. In the prototypical implementation that is based on Protégé


Relation list

FrameA	FrameB	Inst->	Inst<-	Name->	Name<-	Strc->	Strc<-	BrCnt
Ⓒ 1OrganizationalStruc...	Ⓒ 2OrganizationalStruct...	3,083	4,344	0,914	0,914	0,765	0,438	5,263
Ⓒ 1Structure	Ⓒ 2OrganizationalStruct...	3,083	0	0,786	0,314	0,57	0,845	4,035
Ⓒ 1Course	Ⓒ 2Activity	2,845	0,717	0	0	0,749	0,066	2,805
Ⓒ 1Teaching	Ⓒ 2Activity	2,845	0	0	0	0,766	0,423	2,293
Ⓒ 1Course	Ⓒ 2Course	2,6	0,863	0,684	0,684	0,832	0,829	5,237
Ⓒ 1Teaching	Ⓒ 2Course	2,6	0	0	0	0,12	0,746	2,052
Ⓒ 1Action	Ⓒ 2Activity	2,238	1,058	0,211	0,16	0,669	0,533	4,119
Ⓒ 1PaidEmployment	Ⓒ 2Activity	2,238	1,058	0	0	0,628	0,898	3,631
Ⓒ 1PaidEmployment	Ⓒ 2Course	1,846	0,992	0	0	0,052	0,83	2,914
Ⓒ 1Action	Ⓒ 2Course	1,846	0,992	0	0	0,087	0,776	2,93

Fig. 9. Instance-based Matcher

and the PROMPT tab, we use a rather simple voting schema for the aggregation of evidence, the so–called Borda Count.

Further work will comprise

- *more flexible aggregation of evidence*, e.g., by using logic–based evidence integration and by adaptive aggregations functions whose parameters may be learned,
- *support for other types of relationships*, e.g., by additional heuristics for the instance–based evidence generation, and
- *richer mapping languages* that also allow for mapping composed concepts.

In addition to a more systematical evaluation of the ontology evidence framework, we also aim at an integration with the concept of ontology societies [5], leading to a comprehensive view on *ontology negotiation* which integrates both, the knowledge formalization aspect and the social aspect that are contained in the definition of ontologies as "explicit, shared conceptualizations". We think that such a comprehensive view on ontologies really makes them adequate tools for handling knowledge in today's open information landscapes.

Acknowledgement. The work described here has been supported by the German Ministry for Research under grant 01 IW 901 FRODO and 01 IW C01 EPOS. We thank brainbot technology AG for providing us with the NextBot engine and Björn Endres for implementing the JAVA API for the engine. We also thank the anonymous reviewers for their valuable feedback.

References

1. A. Abecker and L. van Elst. Ontologies for knowledge management. In *[22]*, pages 435–454. Springer, 2004.
2. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* Addison Wesley, New York, 1999.
3. R. Bergmann, M. M. Richter, S. Schmitt, A. Stahl, and I. Vollrath. Utility-oriented matching: A new research direction for case-based reasoning. In S. Schmitt, I. Vollrath, and U. Reimer, editors, *Proceedings of the 9th German Workshop on Case-Based Reasoning*, 2001.

4. H. H. Do and E. Rahm. COMA — A system for flexible combination of schema matching approaches. In Philip A. Bernstein et al., editors, *VLDP 2002: proceedings of the Twenty-Eighth International Conference on Very Large Data Bases, Hong Kong SAR, China, 20–23 August 2002*, pages 610–621, Los Altos, CA 94022, USA, 2002. Morgan Kaufmann Publishers.
5. L. van Elst and A. Abecker. Negotiating domain ontologies in distributed organizational memories. In Paolo Bouquet, editor, *Meaning Negotiation (MeaN-02). Technical Report WS-02-09*, pages 32–35. AAAI Press, 2002.
6. L. van Elst and A. Abecker. Ontologies for information management: Balancing formality, stability, and sharing scope. *Expert Systems with Applications*, 23(4):357–366, November 2002.
7. L. van Elst, V. Dignum, and A. Abecker, editors. *Agent Mediated Knowledge Management, International Symposium AMKM 2003, Stanford, CA, USA, March 24-26, 2003, Revised and Invited Papers*, volume 2926 of *LNAI*. Springer, 2004.
8. L. van Elst, V. Dignum, and A. Abecker. Towards agent-mediated knowledge management. In *[7]*, pages 1–31, 2004.
9. J. Euzenat and P. Valtchev. An integrative proximity measure for ontology alignment. In A. Doan, A. Halevy, and N. Noy, editors, *Proceedings of the 1st Intl. Workshop on Semantic Integration*, volume 82 of *CEUR*, 2003.
10. D. Fensel. *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Springer, 2001.
11. T.K. Ho. *A Theory of Multiple Classifier Systems And Its Application to Visual Word Recognition*. PhD thesis, State University of New York at Buffalo, 1992.
12. J. Kohlhas and P.A. Monney. References theory of evidence – a survey of its mathematical foundations, applications and foundational aspects. *ZOR – Mathematical Methods of Operations Research*, 39:35–68, 1994.
13. M.S. Lacher and G. Groh. Faciliating the exchange of explicit knowledge through ontology mappings. In I. Russell and J. Kolen, editors, *Fourteenth International FLAIRS conference*, pages 305–309, 2001.
14. D.L. McGuinness, R. Fikes, J. Rice, and S. Wilder. The chimaera ontology environment. In *Proc. of the 17th National Conf. on Artificial Intelligence (AAAI 2000)*, Austin, Texas, 2000.
15. D.L. McGuinness, R. Fikes, J. Rice, and S. Wilder. An environment for merging and testing large ontologies. In *Proc. of the Seventh Int. Conf. on Principles of Knowledge Representation and Reasoning (KR2000)*, Breckenridge, Colorado, USA, 2000.
16. S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity Flooding: A Versatile Graph Matching Algorithm and ist Application to Schema Matching. In *Proceedings 18th ICDE*, San Jose, CA, February 2002.
17. R. Neches, R. Fikes, T. Finin, Th. Gruber, R. Patil, T. Senator, and W.R. Swartout. Enabling technology for knowledge sharing. *AI Magazine*, 12(3):36–56, 1991.
18. N. Fridman Noy and M. Musen. PROMPT: Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the 7th Conference on Artificial Intelligence (AAAI-00) and of the 12th Conference on Innovative Applications of Artificial Intelligence (IAAI-00)*, pages 450–455, Menlo Park, CA, July 30– 3 2000. AAAI Press.
19. S. Prasad, Y. Peng, and T. Finin. A tool for mapping between two ontologies using explicit information. In Paolo Bouquet, editor, *Meaning Negotiation – Papers from the AAAI Workshop*, volume WS–02–09 of *Technical Report*. AAAI, 2002.
20. G. Salton. *Automatic information organization and retrieval.* McGraw Hill, New York, 1968.

21. M. Schaaf and L. van Elst. An approach to cooperating organizational memories based on semantic negotiation and unification. In *AAAI-2002 Workshop on Meaning Negotiation, Technical Report WS-02-09*, pages 13–16. AAAI Press, 2002.
22. S. Staab and R. Studer. *Handbook on Ontologies*. International Handbooks on Information Systems. Springer Verlag, Heidelberg, 2004.
23. H. Wache. *Semantische Mediation für heterogene Informationsquellen*. Akademische Verlagsgesellschaft Aka GmbH, Berlin, 2003.