Evaluating Ontology-Mapping Tools: Requirements and Experience

Natalya F. Noy and Mark A. Musen

Stanford Medical Informatics, Stanford University
251 Campus Drive, Stanford, CA 94305, USA
{noy, musen}@smi.stanford.edu

Abstract. The appearance of a large number of ontology tools may leave a user looking for an appropriate tool overwhelmed and uncertain on which tool to choose. Thus evaluation and comparison of these tools is important to help users determine which tool is best suited for their tasks. However, there is no "one size fits all" comparison framework for ontology tools: different classes of tools require very different comparison frameworks. For example, ontology-development tools can easily be compared to one another since they all serve the same task: define concepts, instances, and relations in a domain. Tools for ontology merging, mapping, and alignment however are so different from one another that direct comparison may not be possible. They differ in the type of input they require (e.g., instance data or no instance data), the type of output they produce (e.g., one merged ontology, pairs of related terms, articulation rules), modes of interaction and so on. This diversity makes comparing the performance of mapping tools to one another largely meaningless. We present criteria that partition the set of such tools in smaller groups allowing users to choose the set of tools that best fits their tasks. We discuss what resources we as a community need to develop in order to make performance comparisons within each group of merging and mapping tools useful and effective. These resources will most likely come as results of evaluation experiments of stand-alone tools. As an example of such an experiment, we discuss our experiences and results in evaluating PROMPT, an interactive ontology-merging tool. Our experiment produced some of the resources that we can use in more general evaluation. However, it has also shown that comparing the performance of different tools can be difficult since human experts do not agree on how ontologies should be merged, and we do not yet have a good enough metric for comparing ontologies.

1 Ontology-Mapping Tools Versus Ontology-Development Tools

Consider two types of ontology tools: (1) tools for developing ontologies and (2) tools for mapping, aligning, or merging ontologies. By **ontology-development tools** (which we will call *development tools* in the paper) we mean ontology editors that allow users to define new concepts, relations, and instances. These tools usually have capabilities for importing and extending existing ontologies. Development tools may include graphical browsers, search capabilities, and constraint checking. Protégé-2000 [17], OntoEdit [19], OilEd [2], WebODE [1], and Ontolingua [7] are some examples of development tools. **Tools for mapping, aligning, and merging ontologies** (which we will call *mapping tools*) are the tools that help users find similarities and differences between source ontologies. Mapping tools either identify potential correspondences automatically or provide the environment for the users to find and define these correspondences, or both. Mapping tools are often extensions of development tools. Mapping tool and algorithm examples include PROMPT[16], ONION [13], Chimaera [11], FCA-Merge [18], GLUE [5], and OBSERVER [12].

Even though theories on how to evaluate either type of tools are not well articulated at this point, there are already several frameworks for evaluating ontology-development tools. For example, Duineveld and colleagues [6] in their comparison experiment used different development tools to represent the same domain ontology. Members of the Ontology-environments SIG in the OntoWeb initiative[1] designed an extensive set of criteria for evaluating ontology-development tools and applied these criteria to compare a number of projects. Some of the aspects that these frameworks compare include:

- interoperability with other tools and the ability to import and export ontologies in different representation languages;
- expressiveness of the knowledge model;
- scalability and extensibility;
- availability and capabilities of inference services;
- usability of the tools.

Let us turn to the second class of ontology tools: tools for mapping, aligning, or merging ontologies. It is tempting to reuse many of the criteria from evaluation of development tools. For example, expressiveness of the underlying language is important and so is scalability and extensibility. We need to know if a mapping tool can work with ontologies from different languages. However, if we look at the mapping tools more closely, we see that their comparison and evaluation must be very different from the comparison and evaluation of development tools. All the ontology-development tools have very similar *inputs and the desired outputs*: we have a domain, possibly a set of ontologies to reuse, and a set of requirements for the ontology, and we need to use a tool to produce an ontology of that domain satisfying the requirements. Unlike the ontology-development tools, the

[1] http://delicias.dia.fi.upm.es/ontoweb/sig-tools/

ontology-mapping tools *vary with respect to the precise task that they perform, the inputs on which they operate and the outputs that they produce.*

First, the *tasks* for which the mapping tools are designed, differ greatly. On the one hand, all the tools are designed to find similarities and differences between source ontologies in one way or another. In fact, researchers have suggested a uniform framework for describing and analyzing this information regardless of what the final task is [3, 10]. On the other hand, from the user's point of view the tools differ greatly in what tasks this analysis of similarities and differences supports. For example, Chimaera and PROMPT allow users to merge source ontologies into a new ontology that includes concepts from both sources. The output of ONION is a set of articulation rules between two ontologies; these rules define what the similarities and differences are. The articulation rules can later be used for querying and other tasks. The task of GLUE, AnchorPROMPT [14] and FCA-Merge is to provide a set of pairs of related concepts with some certainty factor associated with each pair.

Second, different mapping tools rely on *different inputs*: Some tools deal only with class hierarchies of the sources and are agnostic in their merging algorithms about slots or instances (e.g., Chimaera). Other tools use not only classes but also slots and value restrictions in their analysis (e.g., PROMPT). Other tools rely in their algorithms on the existence of instances in each of the source ontologies (e.g., GLUE). Yet another set of tools require not only that instances are present, but also that source ontologies share a set of instances (e.g., FCA-Merge). Some tools work independently and produce suggestions to the user at the end, allowing the user to analyze the suggestions (e.g., GLUE, FCA-Merge). Some tools expect that the source ontologies follow a specific knowledge-representation paradigm (e.g., Description Logic for OBSERVER). Other tools rely heavily on interaction with the user and base their analysis not only on the source ontologies themselves but also on the merging or alignment steps that the user performs (e.g., PROMPT, Chimaera).

Third, since the tasks that the mapping tools support differ greatly, the *interaction between a user and a tool* is very different from one tool to another. Some tools provide a graphical interface which allows users to compare the source ontologies visually, and accept or reject the results of the tool analysis (e.g., PROMPT, Chimaera, ONION), the goal of other tools is to run the algorithms which find correlations between the source ontologies and output the results to the user in a text file or on the terminal–the users must then use the results outside the tool itself.

The goal of this paper is to start a discussion on a framework for evaluating ontology-mapping tools that would account for this great variety in underlying assumptions and requirements. We argue that many of the tools cannot be compared directly with one another because they are so different in the tasks that they support. We identify the criteria for determining the groups of tools that *can* be compared directly, define what resources we need to develop to make such comparison possible and discuss our experiences in evaluating our merging tool, PROMPT, as well as the results of this evaluation.

2 Requirements for Evaluating Mapping Tools

Before we discuss the evaluation requirements for mapping tools, we must answer the following question which will certainly affect the requirements: what is the goal of such potential evaluation? It is tempting to say "find the best tool." However, as we have just discussed, given the diversity in the tasks that the tools support, their modes of interaction, the input data they rely on, it is impossible to compare the tools to one another and to find one or even several measures to identify the "best" tool.

Therefore, we suggest that the questions driving such evaluation must be user-oriented. A user may ask either *what is the best tool for his task* or *whether a particular tool is good enough for his task*. Depending on what the user's source ontologies are, how much manual work he is willing to put in, how important the precision of the results is, one or another tool will be more appropriate. Therefore, the first set of evaluation criteria are **pragmatic criteria**. These criteria include but are not limited to the following:[2]

Input requirements What elements from the source ontologies does the tool use? Which of these elements does the tool require? This information may include: concept names, class hierarchy, slot definitions, facet values, slot values, instances. Does the tool require that source ontologies use a particular knowledge-representation paradigm?

Level of user interaction Does the tool perform the comparison in a "batch mode," presenting the results at the end, or is it an interactive tool where intermediate results are analyzed by the user, and the tool uses the feedback for further analysis?

Type of output What is the result of working with the tool? Is it a set of articulation rules? Is it a merged ontology? Is it an (instantiated) ontology representing the mappings? Is it a list of pairs of related concepts (possibly with a certainty factor associated with them)?

Content of output Which elements of the source ontologies are correlated in the output? These elements can include relations between classes, slots, values, or instances.

There is no single "best" set of answers to these questions. If the user's ontologies include instance data, the tools that use this data in their analysis will provide more precise suggestions. However, if the instance data is not available, these tools are inappropriate. Similarly, if the user needs only approximate mappings between the sources, the tools that provide less precise mappings but require less interaction may be what the user is looking for. In other words, if we create a comparison matrix of tools and their features (as most current comparisons of the tools do), the best tool is not the tool that gets the largest number of checkmarks in the matrix. Rather, the best tool is the tool whose set of checkmarks best matches the user's conditions. From a practitioner's point of view,

[2] This list is a summary of many of the parameters used in the OntoWeb initiative for comparing mapping tools.

existing comparison frameworks have perhaps over-emphasized the importance of getting as many features in a single tool as possible (e.g., [9]).

These pragmatic criteria will help a user identify a group of tools that will be useful to him. Within a group of tools that use the same type of input data and produce similar types of outputs with similar level of interaction, which one should the user choose? At this point the quality of comparison algorithms comes into play. All other things being equal, it is the tool that produces "better" suggestions that will be most beneficial to us. Therefore, we need to define what "better" is in this context and how to find which tool is indeed better according to this **performance criterion**. Defining what "better" is, is fairly easy: the tool with better recall and precision wins. We can define **recall** as the fraction of correct matches that the algorithm identifies. We can define **precision** as the fraction of correct matches among the matches that the tool identifies. These notions of recall and precision are similar to recall and precision used in information retrieval: recall measures how much of the useful information the tools finds; precision measures how much of the information that the tool finds is useful. Therefore, we can perform experiments comparing the tools in the same group directly to one another to determine the quality of the comparison algorithms. Ontology-mapping tools employ a variety of techniques to compare source ontologies. Some tools use machine learning (e.g., GLUE and FCA-Merge), others analyze graph structure (e.g., AnchorPROMPT, ONION), yet other tools use heuristic-based analyzers (e.g., ONION, PROMPT).

To compare the tools within one group, we as a community need to develop the following resources:

Source ontologies We need (preferably several sets of) pairs of ontologies covering similar domains. We would like to have sets of ontologies of different sizes, with different levels of overlap, some of them complicated and some of them close to simple hierarchies.

Benchmark results For each pair of the source ontologies, we need human-generated correspondences between them. Again, we would like to have these correspondences at different levels and in different forms: pairs of related concepts, logic rules showing more complex transformations, ontologies representing the mappings.

Metrics for comparing the performance of the tools For the tools that produce a list of correspondences between concepts in the source ontologies, we can use the measures of recall and precision that we defined earlier. For the tools that result in new merged ontologies, we must compare the resulting ontologies with the benchmark ones. Therefore, we need some precise measure of the "distance" between ontologies. There are several proposals on measuring distance between individual concepts. However, we need distance between ontologies as a whole. As an alternative to measuring distance between ontologies, we can consider evaluating the ontologies resulting from experiments based on analysis of taxonomic relationships proposed by Guarino and Welty [8]. We can use the information on essence, rigidity, and other

properties defined in the benchmark results to determine whether these properties hold in the ontologies resulting from the experiments.

Last but not least, when we perform the comparison experiments, we must be careful to maintain a set of **experiment controls**: level of the users' expertise with a particular tool and with the mapping process in general, the amount of training the users get, the documentation that is available, and so on. The more uniform these controls are, the more valid the experiments will be.

Ideally, researchers that do not have a vested interest in any of the tools should create the resources that we have listed. Otherwise, the selection of resources and the benchmarks will inevitably be skewed towards one or the other tool. However, in practice, this approach may not be possible, unless there is specific funding to create the resources. Therefore, realistically, these resources are most likely to come initially as results of stand-alone evaluation experiments of specific tools. These experiments evaluate whether a particular tool is "good enough" for the user's task. To perform these individual experiments, researchers will need to find source ontologies covering the same domain. They will need to create manually a gold-standard ontology to serve as a benchmark. Alternatively, the merged ontologies that the experiment participants will produce could also serve as benchmarks. In the experiment, the researchers will likely need to compare the resulting ontologies, thus developing some metric of the distance. Therefore, many of the resources for future comparative evaluation of different tools can come as results of such stand-alone experiment. We performed such an experiment evaluating PROMPT—an ontology-merging tool developed in our laboratory [16]. We describe the experiment in the rest of this paper.

3 PROMPT Evaluation

PROMPT [16] is a tool for interactive ontology merging. It is a plugin for Protégé-2000.[3] PROMPT leads the user through the ontology-merging process, identifying possible points of integration, and making suggestions for operations that should be done next, what conflicts need to be resolved, and how to resolve them. The tool compares names of concepts, relations among them, constraints on slot values, and instances of concepts to make its suggestions.

We evaluated the quality of the suggestions that the tool provides by asking several users to merge two source ontologies using PROMPT. We recorded their steps, which suggestions they followed, which suggestions they did not follow, and what the resulting ontology looked like.

3.1 Source ontologies

In order to evaluate the performance of the PROMPT merging tool, we chose two ontologies that were developed independently by two teams in the DAML project.[4] We imported two ontologies from the DAML ontology library [4]:

[3] http://protege.stanford.edu
[4] http://www.daml.org

1. An ontology for describing individuals, computer-science academic departments, universities, and activities that occur at them developed at the University of Maryland (UMD), and
2. An ontology for describing employees in an academic institutions, publications, and relationships among research groups and projects developed at Carnegie Mellon University (CMU).

These two ontologies constituted a good target for the merging experiment because on the one hand, they covered similar subject domains (research organizations and projects, publications, etc.) and on the other hand, their developers worked completely independent of one another and therefore there was no intensional correlation among terms in the ontologies. In addition, the domain is easy to understand for everyone.

Figure 1 presents snapshots of the two hierarchies. Note that many of the concepts in the two ontologies are similar, but they are represented by different terms: *Industrial_org* versus *CommercialOrganization*, *Governmental_org* versus *GovernmentOrganization*, *Student* versus *Students*, *Organisation* versus *Organization*. The structure of the hierarchy is also different: In the CMU hierarchy, for example, *Students*, *Faculty*, *Management* are subclasses of the class *Employment_Categories*, whereas in the UMD hierarchy these types of classes are subclasses of *Person*. Even though the CMU hierarchy has a class *Person*, its only subclass is *Employee*.

Given these differences in the sources, the merged ontologies produced by different users will inevitably be different: There are many design decisions that could go either way. For example, will the *Organization* class in the merged ontology be at the top level, as it is in the CMU hierarchy, or will it be a subclass of *SocialGroup*, as it is in the UMD hierarchy? Are the classes *Employment_Categories* from the CMU ontology and *Employee* from the UMD ontology essentially the same classes? The easiest way to answer these questions is to have the designers of the two original ontologies get together and merge them. However, in practice this scenario is unrealistic. Therefore, if our task requires that we merge the two ontologies producing one uniform ontology, the user performing the merge will have to make these decisions and different users may make different decisions.

3.2 Experiment setup

We asked users to use the PROMPT tool to merge the two ontologies described in the previous section. All the users were previously familiar with Protégé, but have not tried to use PROMPT before. None of the users has addressed the problem of merging ontologies prior to the experiment. There were four participants in the experiment—all of them students at the Stanford Medical Informatics who answered our call for participation. Each participant received a package containing:

– the PROMPT software

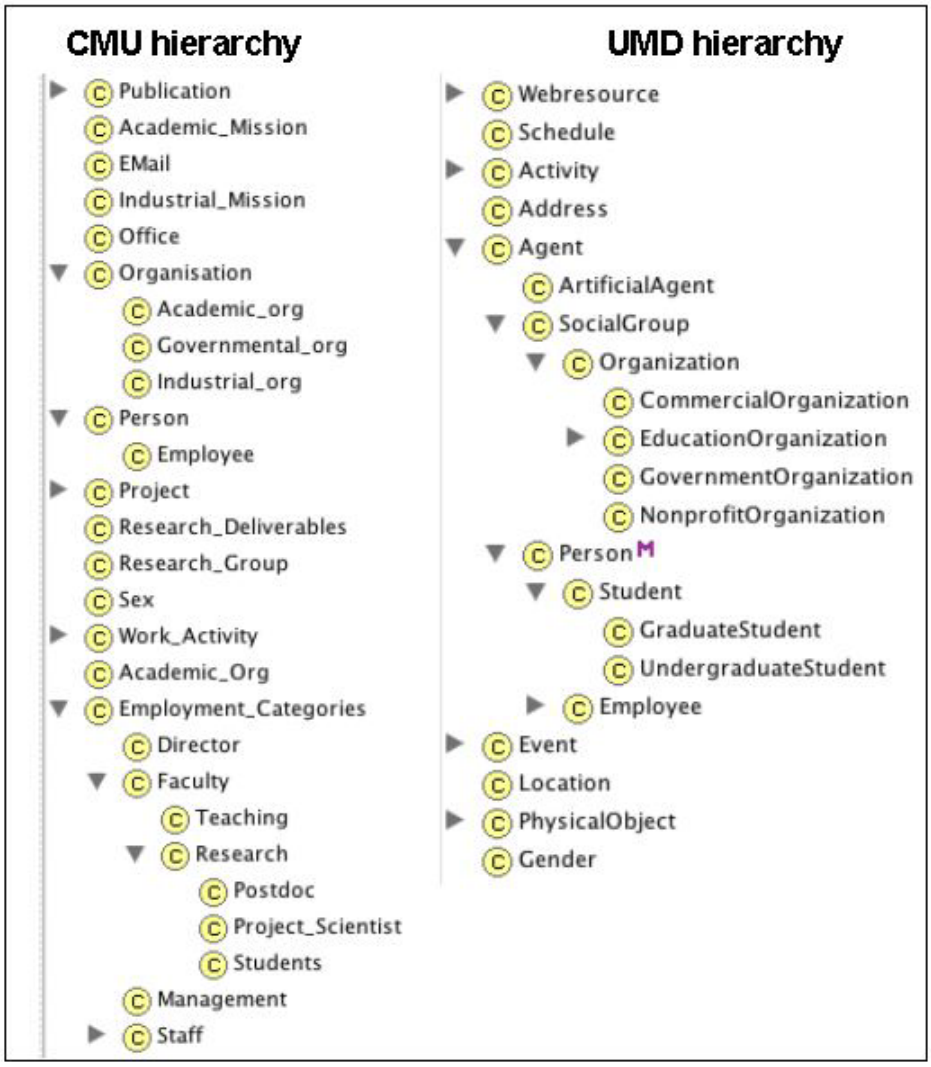

Fig. 1. Snapshots of the class hierarchies in the two source ontologies for the experiment.

– documentation of the tool
– a detailed tutorial
– a tutorial example
– materials for the evaluation

The users performed the evaluation at their convenience on their own computers. We asked each participant to install the tool, to read the tutorial and to follow the examples in the tutorial using the tutorial ontologies. We suggested that they may then look through the documentation to get additional insights. After that the participants were to perform the evaluation itself by merging the two source ontologies. After they were done, they sent back the resulting ontology and the log files for our analysis.

In order to minimize differences in the results, we arbitrarily set up the CMU ontology to be the preferred one. That is, if the users were merging two classes with different names, the name from the CMU ontology was used.

3.3 Using PROMPT: Experiment Results

The primary goal of our experiment was to evaluate the *quality of PROMPT's suggestions*. In addition, to experiment with metrics for finding a distance between ontologies, we compared the ontologies that the participants have produced.

Quality of PROMPT's suggestions To evaluate the quality of PROMPT's suggestions, we evaluated the precision and recall measures that we have described in Section 2: Precision is the fraction of the tool's suggestions that the users decided to follow. Recall is the fraction of the operations that the users performed that were suggested by the tool. In our experiments, the average precision was 96.9% and the average recall was 88.6%. The precision was in fact a lot higher than we expected. There are several possible explanations for this remarkable result. First, the users were not experts in ontology merging and did not have any particular task in mind when merging the ontologies. As a result, they found it easier simply to follow the tools suggestions, as long as they seemed reasonable rather than explore the ontologies deeper and come up with their own operations. Second, there was a significant overlap in the content and structure of the source ontologies, which made automatic generation of correct suggestions easier.

Some of the lower recall figures (it was 49% for *merge* operations in one of the experiments) result from questionable choices that the users made. For example, one user merged *Publication* and *DocumentRepresentation* classes, *EMail* and *ElectronicDocument*, *ProjectScientist* and *Research_Assistant*; slots *publisher* and *publishDate*. PROMPT also did not identify such pairs of related classes as *Academic_org* and *EducationOrganization* or *Industrial_org* and *CommercialOrganization*.

Even though we did not formally evaluate usability, the fact that all the users were able to use the tool and completely merge the source ontologies after a brief

handout tutorial, indicates that the tool is fairly easy to use. Even though we told the participants that while they were still going through the tutorial, they could ask questions about the tool and about the process, only one of them ended up asking a question.

Resulting ontologies In addition to comparing the recall and precision of PROMPT's suggestions, we looked at the resulting ontologies. Since we did not have a benchmark ontology (we pulled both ontologies from the Internet from an ontology library), we could compare the ontologies resulting from the experiment only to one another.

We have noted in Section 2 that we need a distance measure between ontologies. We can treat the ontologies resulting from the experiment as versions of the same ontology—after all, they all result from merging the same ontologies. Therefore, we can use the notion of diff between versions to find a distance between two ontologies. In our earlier work [15], we defined the notion of a **structural diff** between two versions of the same ontology.

Definition 1 (Structural diff). *Given two versions of an ontology O, V_1 and V_2, a* **structural diff** *between V_1 and V_2, $D(V_1, V_2)$, is a set of frame pairs $\langle F_1, F_2\rangle$ where:*

- *$F_1 \in V_1$ or $F_1 = null$; $F_2 \in V_2$ or $F_2 = null$*
- *F_2 is an* **image** *of F_1 (***matches** *F_1), that is, F_1 became F_2. If F_1 or F_2 is null, then we say that F_2 or F_1 respectively does not have a match.*
- *Each frame from V_1 and V_2 appears in at least one pair.*
- *For any frame F_1, if there is at least one pair containing F_1, where $F_2 \neq null$, then there is no pair containing F_1 where $F_2 = null$ (if we found at least one match for F_1, we do not have a pair that says that F_1 is unmatched). The same is true for F_2.*

Note that the definition implies that for any pair of frames F_1 and F_2, there is at most one entry $\langle F_1, F_2\rangle$.

The structural diff describes which frames have changed from one version to another. However, for a diff to be more useful to the user, it should include not only *what* has changed but also some information on *how* the frames have changed. A PromptDiff table provides this more detailed information [15].

Definition 2 (PromptDiff table). *Given two versions of an ontology O, V_1 and V_2, the* PromptDiff **table** *is a set of tuples $\langle F_1, F_2, rename\_value, operation\_value, mapping\_level\rangle$ where:*

- *There is a tuple $\langle F_1, F_2, rename\_value, operation\_value, mapping\_level\rangle$ in the table iff there is a pair $\langle F_1, F_2\rangle$ in the structural diff $D(V_1, V_2)$.*
- *$rename\_value$ is true if frame names for F_1 and F_2 are the same; $rename\_value$ is $false$ otherwise.*
- *$operation\_value \in OpS$, where $OpS = \{add, delete, split, merge, map\}$*
- *$mapping\_level \in MapS$, where $MapS = \{unchanged, isomorphic, changed\}$.*

	u1-u2	u1-u3	u1-u4	u2-u3	u2-u4	u3-u4
Frames in ontology 1	251	251	251	253	253	216
Frames in ontology 2	253	216	232	216	232	232
Unmatched entries from ontology 1:	3	37	19	39	22	11
Unmatched entries from ontology 2:	5	2	0	2	1	27
Changed rows in the table:	30	50	46	48	54	45
Difference (in number of frames)	38	89	65	89	77	83
Difference (in %)	14.8%	35.2%	25.9%	34.9%	30.3%	34.2%

Table 1. The difference between pairs of ontologies in the experiment. There were four users, u1, u2, u3, and u4. Each column represents a comparison of ontologies that each pair of users produced.

The operations in the operation set *OpS* indicate to the user how a frame has changed from one version to the other: whether it was added or deleted, whether it was split in two frames, or whether two frames were merged. We assign a *map* operation to a pair of frames if none of the other operations applies. The *mapping_level* indicates how different the two frames are from each other. If the *mapping_level* is *unchanged*, then the user can safely ignore the frames—nothing has changed in their definitions. If two frames are *isomorphic*, then their corresponding slots and facet values are images of each other, but not necessarily identical images. The *mapping_level* is *changed* if the frames have slots or facet values that are not images of each other.

Therefore, we can measure the distance between two ontologies by considering the number of frames in each ontology and the number of rows in the PROMPTDIFF table that have *add*, *delete*, or *changed* in their operation or mapping-level column. In other words, the "difference" between two ontologies is comprised by the frames that either do not have matches or have changed significantly: classes have different superclasses, metaclasses, or slots; slots are attached to different classes or have different facet values. Table 1 presents the results of this comparison. For the four users in the experiment, there are six pairs of ontologies. It is interesting that even with the tool that may have been "steering" the users in a certain direction, the resulting ontologies differed by about 30%. This result indicates that even when human experts are merging ontologies, there is very little agreement and users make very different design decisions. This observation has serious implication for the possibility of even having a benchmark ontology (something that we said is needed to compare merging tools fairly). Consider the *Employment_Categories* and *Employee* classes from Figure 1. Some users decided to merge the class *Employment_Categories* with the *Employee* classes in both ontologies, creating one class out of three. Others kept the distinction that was present in the CMU hierarchy. In one ontology, *Proceedings* is a subclass of *Book* and in the other it is a subclass of *Publication*. Even though most users merged the two *Publication* classes, the two *Proceedings* classes, and the two *Book* classes, they made different decisions on where to place the *Proceedings* class in the merged ontology.

Our approach to measuring the distance may also be inflating the actual distance. For example, if two users both merged the classes *GraduateStudent*

and *UndergraduateStudent* but did not merge their superclasses, then both *GraduateStudent* and *UndergraduateStudent* will appear as "changed" when we compare the merged ontologies: their superclasses are different. Therefore, a better measure may be some sort of weighted measure reflecting how much the concepts have changed.

4 Concluding Remarks

Our evaluation experiment was not ideal. We were limited by available resources and, in some cases, by circumstances. We had four participants in the experiment. The number of users was still too small and the variability in user's expertise with Protégé too large to get meaningful estimates on whether the tool really saves time. If we had more users performing the experiment, the time data would have been one of the interesting points to compare.

Such an experiment however would still have answered only one of the possible user's questions that we discussed in Section 2: is the PROMPT tool good enough. And even that answer assumes that recall and precision figures taken in isolation from other tools, are meaningful. What we really need is a larger-scale experiment that compares tools with similar sets of pragmatic criteria. For example, we would then compare PROMPT with other tools that use classes, slots, facets, and instances in their analysis and that produce suggestions about merging all these knowledge-base elements. In general, it is pointless to compare performance of PROMPT and FCA-Merge for example: FCA-Merge requires that source ontologies not only have instance data but also share the instances. PROMPT does not have such a requirement. Therefore, if a user's ontology does not have instance data, FCA-Merge will be unusable.

In order to help users sort through existing tools and find the right ones for his task, we as a community need to develop the resources that would allow us to perform meaningful experiments comparing different tools:

- Create a library of ontologies covering similar domains. Many ontologies in the DAML ontology library can serve this purpose.
- Manually define mapping between concepts in these ontologies to create consensus benchmark ontologies. Ideally, get the authors of the original ontologies involved in the process of creating mappings.
- Define formal metrics for comparing the distance between ontologies, allowing experimenters to compare the ontologies produced by participants to one another and to the benchmark ontology.
- Define experimental protocols with fixed controls that will make the results of different evaluations comparable.

Our evaluation of PROMPT produced one pair of ontologies that could be used in the library of ontologies for other experiments. Also, since PROMPT is an interactive tool and human experts validate the merged ontologies, we have a set of benchmark ontologies that result from merging the two source ontologies. However, these ontologies differ significantly, which means that there may not be

a single "correct" merged ontology. Our evaluation has also produced an initial metric for comparing ontologies resulting from different experiments. All these resources will be useful in a more general evaluation comparing the performance of different ontology-mapping tools.

In addition to developing these resources, we need to answer many research questions. These questions include but are not limited to the following questions:

- Which pragmatic criteria are most helpful to users in finding the best tool for their task?
- For the ontologies in the repository, how to we develop a benchmark ontology? Does this "gold standard" mapping even exist for many of the ontologies?
- How do we measure how close to the gold standard is the analysis that the tools produce?
- Can we use some of the metric and analysis approaches that are being developed for evaluating ontologies themselves in our evaluation of ontologies resulting from the tool's analyses?

5 Acknowledgments

We would like to thank Monica Crubézy for her thoughtful comments on the paper. We are very grateful to the students at Stanford Medical Informatics who participated in the experiment. The National Cancer Institute provided the funding for our work on ontology merging and management.

References

1. J.C. Arpírez, O. Corcho, M. Fernández-López, and A. Gómez-Pérez. WebODE: a scalable workbench for ontological engineering. In *KCAP-01*, Victoria, Canada, 2001.
2. S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. OILEd: a reason-able ontology editor for the semantic web. In *KI2001, Joint German/Austrian conference on Artificial Intelligence*, volume LNAI Vol. 2174, pages 396–408, Vienna, 2001. Springer-Verlag LNAI Vol. 2174.
3. P. A. Bernstein, A. Y. Halevy, and R. A. Pottinger. Model management: Managing complex information structures. *SIGMOD Record*, 29(4):55–63, 2000.
4. DAML. DAML ontology library, 2001.
5. A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *The Eleventh International WWW Conference*, Hawaii, US, 2002.
6. A. J. Duineveld, R. Stoter, M. R. Weiden, B. Kenepa, and V. R. Benjamins. Wondertools? a comparative study of ontological engineering tools. *International Journal of Human-Computer Studies*, 52(6):1111–1133, 2000.
7. A. Farquhar, R. Fikes, and J. Rice. The Ontolingua server: a tool for collaborative ontology construction. In *Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada, 1996.

8. N. Guarino and C. Welty. Ontological analysis of taxonomic relationships. In A. Laender and V. Storey, editors, *ER-2000: The 19th International Conference on Conceptual Modeling*. Springer-Verlag, 2000.
9. M. Klein. Combining and relating ontologies: an analysis of problems and solutions. In *IJCAI-2001 Workshop on Ontologies and Information Sharing*, pages 53–62, Seattle, WA, 2001.
10. J. Madhavan, P. A. Bernstein, P. Domingos, and A. Halevy. Representing and reasoning about mappings between domain models. In *Eighteenth National Conference on Artificial Intelligence (AAAI'2002)*, Edmonton, Canada., 2002.
11. D. L. McGuinness, R. Fikes, J. Rice, and S. Wilder. An environment for merging and testing large ontologies. In A. G. Cohn, F. Giunchiglia, and B. Selman, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference (KR2000)*. Morgan Kaufmann Publishers, San Francisco, CA, 2000.
12. E. Mena, A. Illarramendi, V. Kashyap, and A. Sheth. OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. *Distributed and Parallel Databases—An International Journal*, 8(2), 2000.
13. P. Mitra, G. Wiederhold, and M. Kersten. A graph-oriented model for articulation of ontology interdependencies. In *Proceedings Conference on Extending Database Technology 2000 (EDBT'2000)*, Konstanz, Germany, 2000.
14. N. F. Noy and M. A. Musen. Anchor-PROMPT: Using non-local context for semantic matching. In *Workshop on Ontologies and Information Sharing at the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-2001)*, Seattle, WA, 2001.
15. N. F. Noy and M. A. Musen. PromptDiff: A fixed-point algorithm for comparing ontology versions. In *Eighteenth National Conference on Artificial Intelligence (AAAI-2002)*, Edmonton, Alberta, 2002.
16. N.F. Noy and M.A. Musen. PROMPT: Algorithm and tool for automated ontology merging and alignment. In *Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, Austin, TX, 2000.
17. Protege. The Protégé project, http://protege.stanford.edu, 2002.
18. G. Stumme and A. Mädche. FCA-Merge: Bottom-up merging of ontologies. In *7th Intl. Conf. on Artificial Intelligence (IJCAI '01)*, pages 225–230, Seattle, WA, 2001.
19. Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke. OntoEdit: Collaborative ontology engineering for the semantic web. In *International Semantic Web Conference 2002 (ISWC 2002)*, Sardinia, Italia, 2002.