Anchor-PROMPT: Using Non-Local Context for Semantic Matching

Natalya F. Noy and Mark A. Musen

Stanford Medical Informatics, Stanford University, Stanford, CA 94305-5479
{noy, musen}@smi.stanford.edu

Abstract

Researchers in the ontology-design field have developed the content for ontologies in many domain areas. Recently, ontologies have become increasingly common on the World-Wide Web where they provide semantics for annotations in Web pages. This distributed nature of ontology development has led to a large number of ontologies covering overlapping domains, which researchers now need to merge or align to one another. The processes of ontology alignment and merging are usually handled manually and often constitute a large and tedious portion of the sharing process. We have developed and implemented Anchor-PROMPT—an algorithm that finds semantically similar terms automatically. Anchor-PROMPT takes as input a set of anchors—pairs of related terms defined by the user or automatically identified by lexical matching. Anchor-PROMPT treats an ontology as a graph with classes as nodes and slots as links. The algorithm analyzes the paths in the subgraph limited by the anchors and determines which classes frequently appear in similar positions on similar paths. These classes are likely to represent semantically similar concepts. Our experiments show that when we use Anchor-PROMPT with ontologies developed independently by different groups of researchers, 75% of its results are correct.

1 Ontology Merging and Anchor-PROMPT

Researchers have pursued development of ontologies—explicit formal specifications of domains of discourse—on the premise that ontologies facilitate knowledge sharing and reuse (Musen 1992; Gruber 1993). Today, ontology development is moving from academic knowledge-representation projects to the world of e-commerce. Companies use ontologies to share information and to guide customers through their Web sites. The ontologies on the World-Wide Web range from large taxonomies categorizing Web sites (such as on Yahoo!) to categorizations of products for sale and their features (such as on Amazon.com). The WWW Consortium is developing the Resource Description Framework (Brickley and Guha 1999), a language for encoding semantic information on Web pages in machine-readable form. Such encoding makes it possible for electronic agents searching for information to share the common understanding of the semantics of the data represented on the Web. Many disciplines now develop standardized ontologies that domain experts can use to share and annotate information in their fields. Medicine, for example, has produced large, standardized, structured vocabularies such as SNOMED (Price and Spackman 2000) and the semantic network of the Unified Medical Language System (Humphreys and Lindberg 1993).

With this widespread distributed use of ontologies, different parties inevitably develop ontologies with overlapping content. For example, both Yahoo! and the DMOZ Open Directory (Netscape 1999) categorize information available on the Web. The two resulting directories are similar, but also have many differences.

Currently, there are extremely few theories or methods that facilitate or automate the process of reconciling disparate ontologies. Ontology management today is mostly a manual process. A domain expert who wants to determine a correlation between two ontologies must find all the concepts in the two source ontologies that are similar to one another, determine what the similarities are, and either change the source ontologies to remove the overlaps or record a mapping between the sources for future reference. This process is both labor-intensive and error-prone.

The semi-automated approaches to ontology merging that do exist today (Section 2) such as PROMPT and Chimaera analyze only **local context** in ontology structure: given two similar classes, the algorithms consider classes and slots that are *directly* related to the classes in question. The algorithm that we present here, Anchor-PROMPT, uses a set of heuristics to analyze **non-local context**.

The goal of Anchor-PROMPT is not to provide a complete solution to automated ontology merging but rather to augment existing methods, like PROMPT and Chimaera, by determining additional possible points of similarity between ontologies.

Anchor-PROMPT takes as input a set of pairs of related terms—**anchors**—from the source ontologies. Either the user identifies the anchors manually or the system generates them automatically. From this set of previously identified anchors, Anchor-PROMPT produces a set of new pairs of semantically close terms. To do that, Anchor-PROMPT traverses the paths between the anchors in the corresponding ontologies. A path follows the links between classes defined by the hierarchical relations or by slots and their domains and ranges. Anchor-PROMPT then compares the terms along these paths to find similar terms.

For example, suppose we identify two pairs of anchors: classes A and B and classes H and G (Figure 1). That is, a class A from one ontology is similar to a class B in the other ontology; and a class H from the first ontology is similar to a class G from the second one. Figure 1 shows

one path from A to H in the first ontology and one path from B to G in the second ontology. We traverse the two paths in parallel, incrementing the similarity score between each two classes that we reach in the same step. For example, after traversing the paths in Figure 1, we increment the similarity score between the classes C and D and between the classes E and F. We repeat the process for all the existing paths that originate and terminate in the anchor points, cumulatively aggregating the similarity score.
The central observation behind Anchor-PROMPT is that if two pairs of terms from the source ontologies are similar and there are paths connecting the terms, then the elements in those paths are often similar as well. Therefore, from a small set of previously identified related terms, Anchor-PROMPT is able to suggest a large number of terms that are likely to be semantically similar as well.

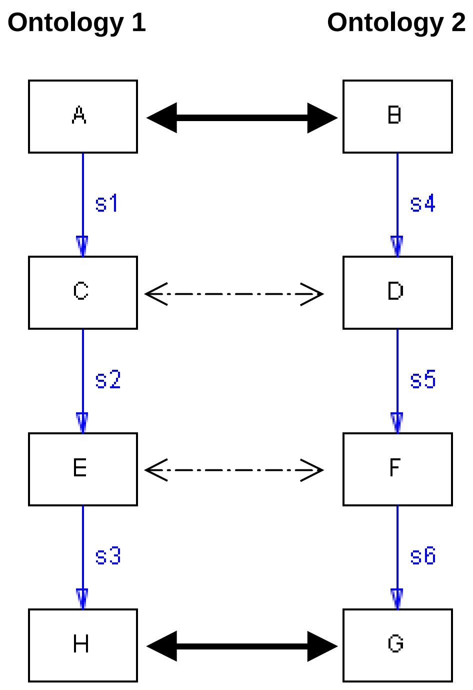


Figure 1. Traversing the paths between anchors. The rectangles represent classes and labeled edges represent slots that relate classes to one another. The left part of the figure represents classes and slots from one ontology; the right part represents classes and slots from the other. Solid arrows connect pairs of anchors; dashed arrows connect pairs of related terms.

2 Related Work

To date, researchers working on tools for ontology merging have expended their greatest effort finding mostly lexical matches among concepts in the source ontologies. Such systems usually rely on dictionaries to determine synonyms, evaluate common substrings, consider concepts whose documentation shares many uncommon words, and so on (Chapulsky et al. 1997; Wiederhold and Jannink 1999). These approaches, however, do not take into account the internal structure of concept representation, the structure of an ontology itself, or the steps users take during merging.

Researchers in the database community have addressed the problem of finding semantically similar terms in automating the process of matching database schemas. A number of schema-matching approaches use not only syntactic information (the similarity of the term names) but also the types of relations among terms. For example, the Artemis system (Castano and De Antonellis 1999) uses thesauri to determine lexical affinity between terms and combines uses domain types of schema elements with user input to determine structural affinity. The TransScm system (Milo and Zohar 1998) traverses the graph representation of two schemas performing a node-by-node comparison. However, the TransScm approach works well only if the input schemas have an extremely high degree of similarity.

The Chimaera ontology-merging environment (McGuinness et al. 2000), an interactive merging tool based on the Ontolingua ontology editor (Farquhar et al. 1996), considers limited ontology structure in suggesting merging steps. However, the only relations that Chimaera currently considers is the subclass–superclass relation and slot attachment.

In our earlier work, we developed PROMPT—a tool for semi-automatic guided ontology merging (Noy and Musen 2000). PROMPT identifies candidates for merging as pairs of **matching terms**—terms from different source ontologies representing similar concepts. It determines not only syntactic but also semantic match based on (1) the content and structure of the source ontologies (e.g., names of classes and slots, subclasses, superclasses, domains and ranges of slot values) and (2) the user's actions (i.e., incorporating in its analysis the knowledge about similarities and differences that the user has already identified).

To summarize, those automatic approaches to semantic matching that do consider the structural relations among terms, base their analysis on studying only the terms that are *directly* related to one another. Both PROMPT and Chimaera consider subclasses and superclasses in question and slots directly attached to a class. PROMPT also considers classes that are referenced by the slots attached to the class in question.

Anchor-PROMPT, which we present here, complements these approaches by analyzing non-local context, by "looking further," and by providing additional suggestions for possible matching terms.

3 The Problem

To illustrate how Anchor-PROMPT works, we will consider two ontologies for representing clinical trials, their protocols, applications, and results. The first ontology, the Design-a-Trial (DaT) ontology (Modgil et al. 2000), underlies a knowledge-based system that helps doctors produce protocols for randomized controlled trials. The second ontology, the randomized clinical-trial (RCT) ontology (Sim 1997), is used in creating electronic trial banks that store the results of clinical trials and allow researchers to find, appraise, and apply the results. Both ontologies represent clinical trials, but one of them, DaT, concentrates on defining a trial protocol itself, and the other, RCT, on representing the results of the trial. The two groups developed their respective ontologies completely independent from each other. Therefore there is no intensional correlation between them. As part of the work on representing clinical guidelines in our laboratory, we needed to merge the two ontologies.

We implemented Anchor-PROMPT based on the knowledge model defined by the Open Knowledge-Base Connectivity (OKBC) protocol (Chaudhri et al. 1998). An ontology in OKBC consists of classes organized in a hierarchy, instances of classes, and slots representing relations between classes and between instances of classes.

In Anchor-PROMPT, we represent classes, slots, and their relations in the ontologies as directed labeled graphs. Figure 2 shows a part of the graph representing the RCT ontology. Classes are **nodes** in the graph. Slots are **edges** in the graph. A slot S connects two classes, A and B, in the graph, if both of the following conditions are true:

(1) The slot S is attached to class A (either as template slot or as an own slot), and
(2) The class B is either a value of slot S for the class A, or B is the range of allowed values for slot S at class A.

For example, the edge representing the slot `latest-protocol` in the RCT ontology links the class `TRIAL` to the class `PROTOCOL` (Figure 2). The slot `latest-protocol` at class `TRIAL` can have as its values instances of the class `PROTOCOL`.

Two nodes connected by an edge in a graph are **adjacent**. There is a **path** between two nodes of a graph, A

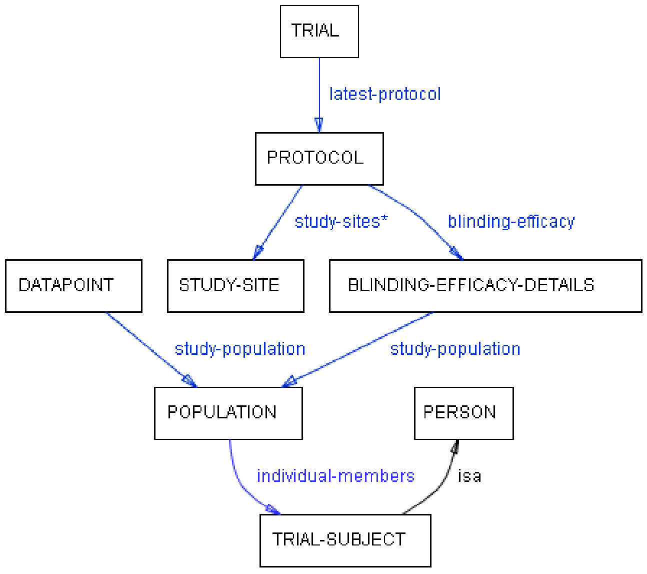


Figure 2. A graph representing a part of the RCT ontology.

and B, if, starting at node A, it is possible to follow a sequence of adjacent edges to reach node B. The **length** of the path is the number of edges in the path.

The goal of the Anchor-PROMPT algorithm is to produce automatically a set of semantically related concepts from the source ontologies using a set of anchor matches identified earlier (manually or automatically) as its input.

4 The Anchor-PROMPT Algorithm

Anchor-PROMPT takes as input a set of **anchors**—pairs of related terms in the two ontologies. We can use any of the existing approaches to term matching to identify the anchors (Section 2). A user can identify the anchors manually. An automated system can identify them by comparing the names of the terms. For example, we can assume that if the source ontologies cover the same domain, the terms with the same names are likely to represent the same concepts. We can also use a combination of system-determined and user-defined anchors. We can use pairs of related terms that Anchor-PROMPT has identified in an earlier iteration after the user has validated them.

For the example in this section, we will consider the following two pairs of anchors for the two clinical-trial ontologies (the first class in the pair is in the RCT ontology; the second is in the DaT ontology[1]):

`TRIAL, Trial`
`PERSON, Person`

Using these two pairs as input, the algorithm must determine pairs of other related terms in the RCT and DaT ontologies. It generates a set of all the paths between `PERSON` and `TRIAL` in the RCT ontology and between `Person` and `Trial` in DaT ontology (Figure 3 shows some of these paths[2]). It considers only the paths that are shorter than a pre-defined parameter *length*. Now consider a pair of paths in this set that have the same length. For example:

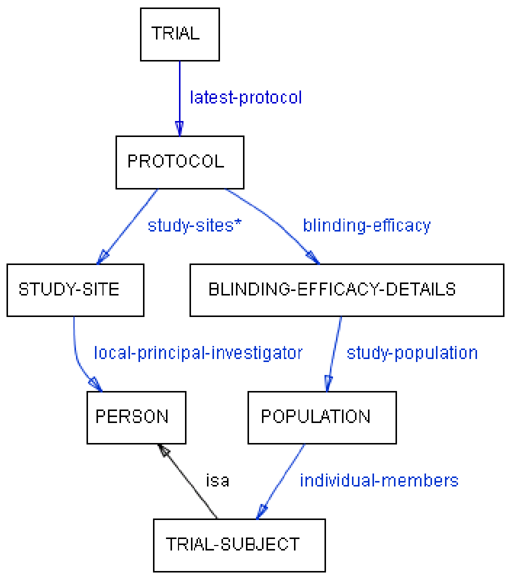
Path 1 (in the RCT ontology):
`TRIAL→PROTOCOL→STUDY-SITE→PERSON`
Path 2 (in the DaT ontology):
`Trial→Design→Blinding→Person`

As it traverses the two paths, Anchor-PROMPT increases the **similarity score**—a coefficient that indicates how closely two terms are related—for the pairs of terms in the same positions in the paths. For the two paths in our example, it will increase the similarity score for the following two pairs of terms:

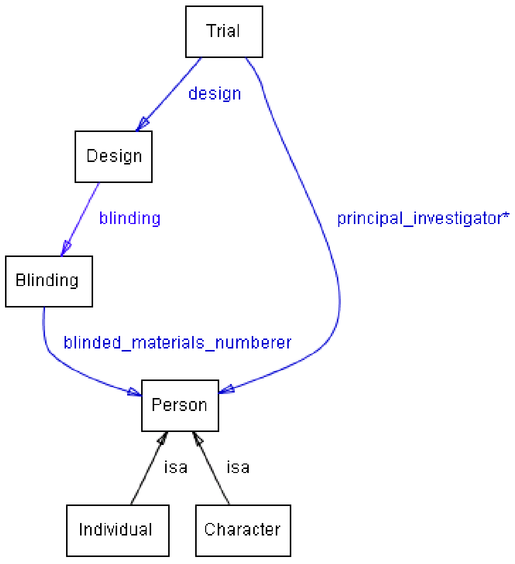
`PROTOCOL`, `Design`
`STUDY-SITE`, `Blinding`

[1]The RCT ontology uses all `UPPER-CASE` letters for class names. The DaT ontology `Capitalizes` the class names. Therefore, it is easy to distinguish which class names come from which ontology, and we will sometimes omit the source information.

[2] We have changed the original RCT ontology slightly to simplify this example.



(a)



(b)

Figure 3. (a) The paths between the classes `TRIAL` and `PERSON` in the RCT ontology; (b) the paths between the classes `Trial` and `Person` in the DaT ontology

Anchor-PROMPT repeats the process for each pair of paths of the same lengths that have one pair of anchors as their originating points (e.g., `TRIAL` and `Trial`) and another pair of anchors as terminating points (e.g., `PERSON` and `Person`). During this process it increases the similarity scores for the pairs of terms that it encounters. It aggregates the similarity score from all the traversals to generate the final similarity score. Consequently, the terms that often appear in the same positions on the paths going from one pair of anchors to another will get the highest score.

4.1 Equivalence groups

In traversing the graph representing the ontology and generating the paths between classes Anchor-PROMPT treats the subclass–superclass links differently from links representing other slots. Consider for example the path from `TRIAL` to `CROSSOVER` in Figure 4.

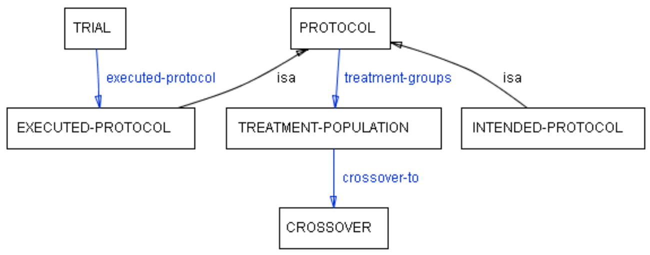


Figure 4. A path from `TRIAL` to `CROSSOVER`: The classes `EXECUTED-PROTOCOL` and `PROTOCOL` form an equivalence group

We could treat the `is-a` link in exactly the same way we treat other slots. However, this approach would disregard the distinct semantics associated with `is-a` links. Instead we can employ the difference in meaning between the `is-a` link and regular slots to improve the algorithm. An `is-a` link connects the terms that are already *similar* (e.g., `PROTOCOL` and `EXECUTED-PROTOCOL`); in fact one describes a *subset* of the other. Other slots link terms that are arbitrarily related to each other.

Anchor-PROMPT joins the terms linked by the subclass–superclass relation in **equivalence groups**. In the example in Figure 4, the classes `PROTOCOL` and `EXECUTED-PROTOCOL` constitute an equivalence group. Here is one of the paths from `TRIAL` to `CROSSOVER` in Figure 4 that goes through `EXECUTED-PROTOCOL`. We identify the equivalence group by brackets.

```
TRIAL→
  [EXECUTED-PROTOCOL, PROTOCOL]→
  TREATMENT-POPULATION→CROSSOVER.
```

Anchor-PROMPT treats an equivalence group as a single node in the path. The set of incoming edges for an equivalence-group node is the union of the sets of incoming edges for each of the group elements. Similarly, the set of outgoing edges for an equivalence-group node is the union of the sets of outgoing edges for each of its elements.

The [`EXECUTED-PROTOCOL`, `PROTOCOL`] equivalence group in Figure 4 has one incoming edge, `executed-protocol`, and one outgoing edge, `treatment-groups`.

4.2 Similarity score

We use the following process to compute the similarity score $S(C_1, C_2)$ between two terms C_1 and C_2 (where C_1 is a class from the source ontology O_1 and C_2 is a class from the source ontology O_2).

1. Generate a set of all paths of length less than a parameter L that connect input anchors in O_1 and O_2.
2. From the set of paths generated in step 1, generate a set of all possible pairs of paths of equal length such that one path in the pair comes from O_1 and the other path comes from O_2.
3. For each pair of paths in the set generated in step 2 and for each pair of nodes N_1 and N_2 located in the identical positions in the paths, increment the similarity score between each pair of classes C_1 and C_2 in N_1 and N_2 respectively be a constant X. (Recall that N_1 and N_2 can be either single classes or equivalence groups that include several classes).

Therefore the similarity score $S(C_1, C_2)$ is a **cumulative** score reflecting how often C_1 and C_2 appear in identical positions along the paths considering all the possible paths between anchors (of length less than L).

We change the constant by which we increment the similarity score when the matching nodes along the paths include not single classes but equivalence groups. Suppose we have the following two nodes at the same position on two paths between anchors: A_1 and $[B_2, C_2]$, a single class A_1 on one side, and an equivalence group with two classes B_2 and C_2 on the other side. Do we give the same score to both pairs of classes A_1, B_2 and A_1, C_2? Is this score the same as the one we would have given the pair A_1, B_2 had B_2 been the only class at the node? Do we give the pairs A_1, B_2 and A_1, C_2 any similarity score at all? We analyze the results for different values of these constants in Section 5.3.2.

4.3 Revisiting the example

We provided Anchor-PROMPT with the following set of three pairs of anchors from the RCT and DaT ontologies correspondingly:

```
TRIAL, Trial
PERSON, Person
CROSSOVER, Crossover
```

We allowed the paths of length less than or equal to 5 and limited the equivalence-group size to 2. Here are the output results in the order of the descending similarity score.

```
PROTOCOL, Design
TRIAL-SUBJECT, Person
INVESTIGATORS, Person
POPULATION, Action_Spec
PERSON, Character
TREATMENT-POPULATION, Crossover_arm
```

In fact, all but one of these results represents a pair of concepts that either are similar or one is a specialization (subclass) of the other. The only exception is the pair `POPULATION, Action_Spec`. Note that many of these pairings are specific to the domain of clinical trials (e.g., `PROTOCOL`, `Design` and `TRIAL-SUBJECT`, `Person`). The pair `PERSON`, `Character` indeed identifies the correct sense in which `Character` is used in the DaT ontology.

5 Evaluation

We perform a formative evaluation of Anchor-PROMPT by testing it on a pair of ontologies that were also developed independently by different groups of researchers. In our experiments, we varied the set of anchor pairs that was the input to the algorithm and various parameters, such as maximum path length, maximum size of equivalence groups, and constants in the similarity score computation. We then analyzed which fraction of the results produced by Anchor-PROMPT were indeed correct results and which parameter settings produced optimal performance.

5.1 Source ontologies

In order to evaluate Anchor-PROMPT formally, we chose a set of ontologies that was different from the two ontologies we used to develop and illustrate the algorithm. We imported two ontologies from the DAML ontology library (DAML 2001):

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).[1]

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 each other and therefore there was no intensional correlation among terms in the ontologies.

5.2 Experiment setup

Input:

The UMD and the CMU ontologies;
Four anchor pairs.

Parameters:

1. A set of *anchor pairs* (we generated all possible sets of anchor pairs from the four input pairs)
2. The maximum number of elements allowed in an *equivalence group* (0, 1, or 2)
3. Similarity score for equivalence-group members along the path given that the score for single elements is 1 (1 and 3)
4. *Length* of path to consider (2, 3, or 4)

Output:

[1] Both ontologies consisted of several smaller ontologies which we merged into a single ontology for the experiment.

For each set of parameters, a set of related terms as determined by Anchor-PROMPT.

Process:

Run Anchor-PROMPT for all the possible combinations of parameters.

For each set of results, compute the *median* similarity score M and discard from the results set all pairs of terms with a similarity score less than M.

We then analyzed the results determining how many of the results were pairs of concepts that were either equivalent or were in a subclass–superclass relationship.

5.3 Evaluation results

5.3.1 Equivalence-group size

If the maximum equivalence-group size is 0 (do not consider subclass–superclass relationships at all) or 1 (allow equivalence groups of size 1), 87% of the experiments produce empty result sets. If the maximum equivalence-group size is 2, only 12% of the result sets are empty.

For the rest of the experiments we fix the maximum equivalence-group size at 2.

5.3.2 Similarity score for equivalence-group members

We have conducted two sets of experiments: in the first set all classes in the same position along the path got the same score N and in the second experiment classes that shared their position with other members of an equivalence group received only 1/3 of the score.[1]

Differentiating the score improved the correctness of results by 14%.

For the rest of the experiments we reduced the scores for members of equivalence groups.

5.3.3 Number of anchor pairs and maximum length of path

Table 1 presents results for various values for the two remaining parameters: the number of anchor pairs that were input to an experiment and the maximum allowed length of the path. For these experiments (as well as for a set of other experiments with different source ontologies), we received the best result precision (the highest ratio of correct results to all the returned results) with the maximum length of path equal to 2. When we limit the maximum path length to 3, we achieve the average precision of 61%. The precision goes up slightly (to 65%) with maximum path length of 4.

[1] In fact, varying the fraction of the score that we assigned to equivalence-group members has not changed the result: The results were identical for equivalence-group scores that were 1/3 or 1/2 of the score for single classes.

Max path length	Number of anchors	Result precision
4	4	67%
4	3	67%
4	2	61%
3	4	67%
3	3	61%
3	2	56%
2	4	100%
2	3	100%
2	2	100%

Table 1. Result precision with respect to maximum path length and the number of anchors.

6 Discussion

To understand the intuition behind Anchor-PROMPT, consider paths of length one (Figure 5a). Recall that the length of a path is the number of edges in the path. If class A is similar to class A’ and class B is similar to class B’, it is plausible to assume that the slots connecting them, s and s’, are similar as well. In Figure 5b, we introduce an additional class, C and C’ correspondingly, on the path. We get the paths of length 2). We continue the analogy by assuming that there is an increased likelihood that C and C’ are similar. In addition, slots s and s’ and p and p’ are similar (Anchor-PROMPT does not currently record the similarity among slots).

The algorithm is based on the assumption that developers link the terms in the ontology in a similar manner even if they do not call the terms with the same names. Therefore, very long paths are unlikely to produce accurate results. As the path that we traverse becomes longer, it becomes less likely that they represent the same series of terms and relations.

Very short paths, however, consistently produced extremely small (but also extremely precise results sets). For the maximum path length of 2, Anchor-PROMPT produced result sets that contained only one pair of terms (with a similarity score above the median for that set) but this pair was always a correct one.

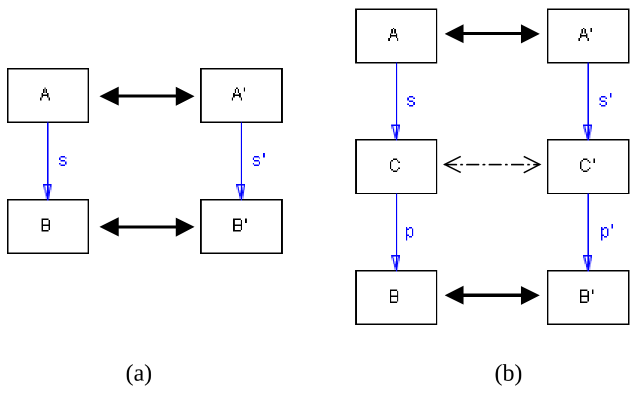


Figure 5. The simple case: the paths of length 1 and 2.

Setting maximum path length to 0 will produce the results that are equivalent to Chimaera's results. Limiting the path length by 1 will produce the results that are equivalent to PROMPT's results (Section 2).

6.1 Reducing the effect of negative results

The similarity score between concepts is a *cumulative* similarity score: Anchor-PROMPT combines the score along all the paths. As a result, we reduce the effect of false matches: Two unrelated terms could certainly appear in identical positions in one pair of paths (and usually do). However, the same two unrelated terms are less likely to appear in identical positions on a different pair of paths.

To remove these incidental matches, we determine the median similarity score in each experiment and discard the pairs of terms with a similarity score less than the median. Therefore, Anchor-PROMPT will discard most of the incidental pairs of terms because they would have appeared only once in the identical positions and would have a low similarity score.

6.2 Performing ontology mapping

Throughout our discussion we have referred to the process of ontology merging, the process in which we start with two source ontologies and generate a new ontology that includes and reconciles all the information from the two source ontologies.

However, the approach that we have presented can be used directly for creating a mapping between terms in ontologies, as well as in matching database schemas. The result of the Anchor-PROMPT algorithm is a set of pairs of similar terms ranked by how close to each other the terms are. This result can be used either to trigger merging of the closely related terms or to establish a mapping between the terms.

6.3 Limitations

Anchor-PROMPT produced highly promising results with two sets of ontologies that were developed entirely independently from each other.

Our approach does not work equally well for all ontologies however. The approach does not work well when the source ontologies are constructed differently. For example, we used Anchor-PROMPT to find related terms in two ontologies of problem-solving methods: (1) the ontology for the unified problem-solving method (UPML) development language (Fensel et al. 1999) and (2) the ontology for the method-description language (MDL) (Gennari et al. 1998). Both ontologies describe reusable problem-solving methods, however, their designers used different approaches to knowledge modeling. The UPML ontology has a large number of classes with slots attached to and referring to classes at many different levels of hierarchy. The MDL ontology has a lot fewer classes with the hierarchy which is only two levels deep. If we think of the ontologies in terms of a graph, many of the nodes from the UPML ontology were "collapsed" in a single node in the MDL ontology. As a result, no two pairs of anchors had paths with the same length between them and the output of Anchor-PROMPT was empty.

In general, Anchor-PROMPT does not work well when one of the source ontologies is a deep one with many inter-linked classes and the other ontology is a shallow one where the hierarchy has only a few levels and most of the slots are associated with the concepts at the top of the hierarchy, and very few notions are reified. If this is the case, the results produced by the algorithm are no different from the results produced by the approaches that consider only very local context.

7 Conclusions

The Anchor-PROMPT algorithm that we have presented uses relations among the terms in an ontology and a set of anchors—pairs of similar terms—to determine which other terms in the ontology are similar.

We conducted experiments using unrelated source ontologies developed by different research groups. We have achieved the results that could not have been achieved using just the terms names (e.g., determine that `TRIAL-SUBJECT` and `Person` are very similar terms in the ontology of trial protocols).

Our experiments show that we can achieve result precision between 61% and 100% depending on the size of the initial anchor set and the maximum length of the path that we traverse.

The algorithm relies on *limited* input from the user. The user does not need to analyze the structure of the ontology deeply, just to determine some pairs of terms that "look similar".

Based on our results, we believe that Anchor-PROMPT can significantly improve the sets of suggestions that other tools identify by producing sets of semantically similar terms using a small set of previously determined similar terms.

Acknowledgments

We have implemented Anchor-PROMPT as a plugin to the Protégé-2000 ontology-editing and knowledge-acquisition tool developed at Stanford Medical Informatics (http://protege.stanford.edu). We generated all the graphs in this paper automatically using OntoViz, a Protégé-2000 plugin, developed by Michael Sintek. Samson Tu and John Nguyen helped us to understand the clinical-trial ontologies. Whatever clarity and readability this paper has, it owes it all to the detailed and thoughtful comments of Monica Crubézy and Ray Fergerson. We are very grateful to anonymous reviewers for suggestions on improving the paper. This work was supported in part by a grant from Spawar and by a grant from FastTrack Systems, Inc

References

Brickley, D. and Guha, R.V. (1999). Resource Description Framework (RDF) Schema Specification. Proposed

Recommendation, World Wide Web Consortium**:** http://www.w3.org/TR/PR-rdf-schema.

Castano, S. and De Antonellis, V. (1999). A Schema Analysis amd Reconciliation Tool Environment. In: *Proceedings of the International Database Engineering and Applications Symposium (IDEAS'99)*, IEEE.

Chapulsky, H., Hovy, E. and Russ, T. (1997). Progress on an Automatic Ontology Alignment Methodology.

Chaudhri, V.K., Farquhar, A., Fikes, R., Karp, P.D. and Rice, J.P. (1998). OKBC: A programmatic foundation for knowledge base interoperability. In: *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, Madison, Wisconsin, AAAI Press/The MIT Press.

DAML (2001). DAML ontology library. http://www.daml.org/ontologies/

Farquhar, A., Fikes, R. and Rice, J. (1996). The Ontolingua Server: a Tool for Collaborative Ontology Construction. In: *Proceedings of the Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada.

Fensel, D., Benjamins, V.R., Motta, E. and Wielinga, R. (1999). UPML: A Framework for knowledge system reuse. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-99)*, Stockholm, Sweden.

Gennari, J.H., Grosso, W. and Musen, M.A. (1998). A method-description language: An initial ontology with examples. In: *Proceedings of the Eleventh Banff Knowledge Acquisition for Knowledge-Bases Systems Workshop*, Banff, Canada.

Gruber, T.R. (1993). A Translation Approach to Portable Ontology Specification. *Knowledge Acquisition* **5**: 199-220.

Humphreys, B.L. and Lindberg, D.A.B. (1993). The UMLS project: making the conceptual connection between users and the information they need. *Bulletin of the Medical Library Association* **81**(2): 170.

McGuinness, D.L., Fikes, R., Rice, J. and Wilder, S. (2000). An Environment for Merging and Testing Large Ontologies. *Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference (KR2000)*. A. G. Cohn, F. Giunchiglia and B. Selman, editors. San Francisco, CA, Morgan Kaufmann Publishers.

Milo, T. and Zohar, S. (1998). Using Schema Matching to Simplify Heterogeneous Data Translation. In: *Proceedings of the 24th International Conference on Very Large Data Bases*, New York City, Morgan Kaufmann.

Modgil, S., Hammond, P., Wyatt, J. and Potts, H. (2000). The Design-A-Trial Project: Developing A Knowledge-Based Tool for Authoring Clinical Trial Protocols. In: *Proceedings of the First European Workshop on Computer-based Support for Clinical Guidelines and Protocols (EWGLP 2000)*, Leipzig, Germany, IOS Press, Amsterdam.

Musen, M.A. (1992). Dimensions of knowledge sharing and reuse. *Computers and Biomedical Research* **25**: 435-467.

Netscape (1999). DMOZ Open Directory. http://www.dmoz.org/

Noy, N.F. and Musen, M.A. (2000). PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In: *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, Austin, TX.

Price, C. and Spackman, K. (2000). SNOMED clinical terms. *BJHC&IM-British Journal of Healthcare Computing & Information Management* **17**(3): 27-31.

Sim, I. (1997). Trial Banks: An Informatics Foundation for Evidence-Based Medicine. PhD Dissertation, Stanford University**:** SMI-97-0701/STAN-CS-TR-97-1599.

Wiederhold, G. and Jannink, J. (1999). Composing Diverse Ontologies. In: *Proceedings of the IFIP Working Group on Database, 8th Working Conference on Database Semantics (DS-8)*, Rotorua, New Zealand.